Prediction and Parameter Coding for Non-rectangular Block Partitioning

Von der Fakultät für Elektrotechnik und Informationstechnik der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Ingenieur

Max Bläser

aus Oberhausen

Berichter:

Univ.-Prof. Dr.-Ing. Jens-Rainer Ohm

Prof. Dr.-Ing. Heiko Schwarz

Tag der mündlichen Prüfung: 29.04.2020

Vorwort

Die vorliegende Doktorarbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Nachrichtentechnik der Rheinisch-Westfälischen Technischen Hochschule Aachen. Die fast sechs Jahre, die ich am Institut mit Forschung, Lehre, Konferenzreisen und all den nicht-fachlichen Aktivitäten verbringen durfte, waren eine wunderbare Zeit, an die ich mich immer gerne zurückerinnern werde. An dieser Stelle möchte ich mich herzlichst bei allen Menschen bedanken, die auf unterschiedlichste Art und Weise zum Gelingen dieser Arbeit beigetragen haben: An erster Stelle danke ich meinem Doktorvater Prof. Dr.-Ing. Jens-Rainer Ohm für die Betreuung der Arbeit, den stets offenen und freundlichen Austausch und die vielen hilfreichen Anmerkungen und Ratschläge. Die fachliche Unterstützung und das freie, wissenschaftliche Arbeiten, bei dem auch das soziale Institutsleben niemals zu kurz kam, bleiben mir in guter Erinnerung. Des Weiteren danke ich Prof. Dr.-Ing. Heiko Schwarz für die Übernahme des Zweitgutachtens und die detaillierten Kommentare zur Arbeit.

Mein besonderer Dank gilt außerdem Priv. Doz. Dr.-Ing. habil. Mathias Wien und Dr.-Ing. Christian Rohlfing für die enge Zusammenarbeit und dafür, dass sie nicht nur viel organisatorischen Projekt-Ballast von mir ferngehalten haben, sondern auch stets mit neuen Ideen und Impulsen diese Arbeit vorangebracht haben. Meinen engsten Kollegen im Video Coding Team, Jens Schneider und Johannes Sauer danke ich ebenfalls für ihre Hilfe, Expertise, und den Gemeinschaftssinn. Diese Art der Zusammenarbeit werde ich sicherlich sehr vermissen! Zuletzt gebührt mein spezieller Dank Clemens Jansen für die Bereitstellung, Wartung und Pflege der technischen Infrastruktur.

Ich danke allen Kollegen und Studenten für die großartige Arbeitsatmosphäre. Wenn aus Kollegen Freunde werden, spricht dies sicherlich für sich. Ich freue mich auf die nächsten Sommerfeste, IENT Movie Nights und LAN Partys mit Euch!

Zu guter Letzt danke ich meiner gesamten Familie, insbesondere meinen Eltern, die mir diesen Lebensweg erst ermöglicht haben. Mein größter Dank gilt meiner Frau Franziska, die mich während der Erstellung dieser Arbeit für einige Wochenenden entbehren musste. Ich danke Dir für die Motivation, die Geduld und die liebevolle Unterstützung während dieser Zeit.

Aachen, im Mai 2020

Contents

Abbreviations

AD	Angle-distance
ALF	Adaptive loop filter
AMP	Asymmetric motion partitioning
AMVP	Adaptive motion vector prediction
AMT	Adaptive multi-core transform
AMVR	Adaptive motion vector resolution
AR	Autoregressive
ATMVP	Alternative temporal motion vector prediction
AVC	Advanced Video Coding
BCW	Bi-prediction with CU-level weight
BD	Bjøntegaard-delta
BDOF	Bi-directional optical flow
BI	Block boundary-intercept
CABAC	Context-adaptive binary arithmetic coding
CB	Coding block
CU	Coding unit
CBF	Coded block flag(s)
CCLM	Cross-component linear model
CE	Core Experiment
CIIP	Combined inter and intra-prediction
CSG	Coefficient sub-group
CTB	Coding tree block
CTC	Common test conditions
CTU	Coding tree unit

DCT	Discrete cosine transform
DF	Deblocking filter
DFT	Discrete Fourier transform
DMVR	Decoder-side motion vector refinement
DST	Discrete sine transform
EMT	Explicit multiple-core transform
FIR	Finite impulse response
GEO	Geometric Block Partitioning
GMP	Geometric Motion Partitioning
GIP	Geometric Inter-Prediction
GOP	Group of Pictures
GOPs	Groups of Pictures
HD	High definition
HEVC	High Efficiency Video Coding
HMVP	History-based MVP
IBC	Intra block copy
ISP	Intra sub-partitions
ITU	International Telecommunication Union
JEM	Joint exploration model
JVET	Joint Video Experts Team
KLT	Karhunen-Loève transform
LBT	Large block transforms
LDB	Low-delay B
LDP	Low-delay P
LFNST	Low-frequency non-separable transform
LMCS	Luma mapping and chroma scaling
LTI	Linear time-invariant
M4V	MPEG-4 Part 2: Visual

MB Macroblock

MCP Motion-compensated prediction

MDDT Mode dependent directional transform

MDNSST Mode dependent non-separable secondary transform

MIP Matrix weighted intra-prediction

MMVD Merge mode with motion vector difference

MPEG Moving picture experts group

MRL Multiple reference line intra prediction

MSE Mean squared error

MTS Multiple transform selection

MTT Multi-type tree

MV-HEVC Multiview High Efficiency Video Coding

MVP Motion vector predictor

MVD Motion vector difference

OBMC Overlapped block motion compensation

OMP Orthogonal matching pursuit

PB Prediction block

PDF Probability density function

PDPC Position dependent (intra) prediction combination

PMF Probability mass function

POC Picture order count

PSNR Peak Signal-To-Noise Ratio

PU Prediction unit

QP Quantization parameter

QTBT Quadtree binary tree

QTBTTT Quadtree binary tree ternary tree

QT Quadtree

BT Binary tree

TT	Ternary tree
RA	Random access
RD	Rate-distortion
RDO	Rate-distortion optimization
RDOQ	Rate-distortion optimized quantization
RDPCM	Residual differential pulse coded modulation
RExt	HEVC Format Range Extension
RQT	Residual quadtree
SAD	Sum of absolute differences
SADCT	Shape-adaptive DCT
SAO	Sample-adaptive offset
SATD	Sum of absolute transform differences
SBP	Segmentation-based block partitioning
SBT	Sub-block transform
SbTMVP	Sub-block temporal motion vector prediction
SCC	Screen Content Coding
SHVC	Scalable High Efficiency Video Coding
SIMD	Single instruction, multiple data
SMVD	Symmetric motion vector difference
SPS	Sequence parameter set
SSE	Streaming SIMD Extensions
SSD	Sum of squared differences
SSIM	Structural Similarity
TB	Transform block
TGM	Text and graphics with motion
TMVP	Temporal motion vector prediction
TPM	Triangular prediction mode
TS	Transform-skip

TSQ	Transform and quantization bypass
TU	Transform unit
UHD	Ultra high definition
URQ	Uniform reconstruction quantizer
VCEG	Visual coding experts group
VMAF	Video Multi-Method Assessment Fusion
VTM	VVC Test Model
VVC	Versatile Video Coding
WSS	Wide-sense stationary
3D-HEVC	3D High Efficiency Video Coding

Notation

Mathematical Definitions and Operators

$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$	Set of natural, integer, real, and complex numbers, respectively
$[\cdot]^{\mathrm{T}}$	Matrix/vector transpose
$[\cdot]^{*}$	Complex conjugate
$\boldsymbol{a} = [a_0, \ldots, a_{n-1}]^{\mathrm{T}}$	n-vector
$\boldsymbol{A} = \left[A_{mn}\right]$	$m \times n$ matrix
$\boldsymbol{A} \circ \boldsymbol{B}$	Hadamard product / element-wise product
$\boldsymbol{A} \oplus \boldsymbol{B}$	Hadamard sum / direct sum
$a(x)$	1D signal or function
$a(x, y)$	2D signal or function
$\boldsymbol{x} = (x_0, x_1)^{\mathrm{T}} \in \mathbb{Z}^2$	Pixel position 2-vector
$\boldsymbol{f} = (f_0, f_1)^{\mathrm{T}} \in \mathbb{R}^2$	Fourier domain 2-vector
$\mathrm{j} = \sqrt{-1}$	Complex unit
$e^{(\cdot)}$, $\exp(\cdot)$	Element-wise natural exponentiation, s.t. $e^{\boldsymbol{x}} = (e^{x_0}, e^{x_1})^{\mathrm{T}}$
$\ln(\cdot)$	Element-wise natural logarithm
$\arg(\cdot)$	Argument (phase) of a complex number or 2-vector
$s * h$	2D convolution
$\delta(\boldsymbol{x}) = \delta(x_0)\delta(x_1)$	Kronecker delta
$\mathrm{P}(\cdot)$	Probability
$\mathrm{p}(\cdot)$	Probability density or probability mass
$\mathrm{p}(\cdot \mid \cdot)$	Conditional probability density
$\mathrm{E}\{\cdot\}$	Expectation
$\mathrm{Re}\{\cdot\}$	Real part
$\mathrm{Im}\{\cdot\}$	Imaginary part
$\lvert\mathbf{a}\rvert = \sqrt{\mathbf{a}^{\mathrm{T}}\mathbf{a}^{*}}$	Real/complex/vector magnitude
$\lVert a \rVert_p$	L^p-norm
$\lvert\cdot\rvert$	Absolute value
$\lfloor\cdot\rfloor$	Round to lower integer value
$\lceil\cdot\rceil$	Round to upper integer value
$\mathrm{sgn}(\cdot)$	Signum function
%	Modulo operator
&	Bit-wise and
|	Bit-wise or
»	Bit-shift to the right
«	Bit-shift to the left

1 Introduction

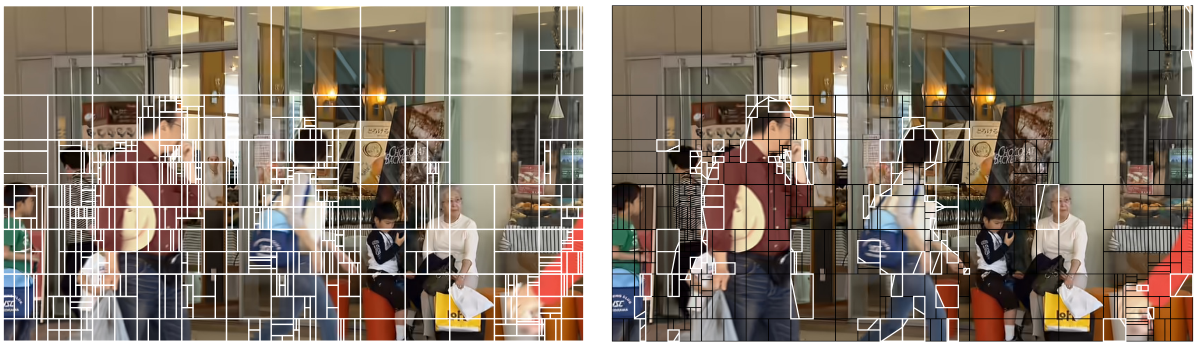
In the first half of 2019, digital video data was reported to make up 60% of the total downstream volume of traffic on the internet [San]. With new, bandwidth-intensive video applications on the rise, such as immersive video or cloud-based gaming, and established applications with increased spatial and temporal resolution to be expected, the need to compress the data with higher efficiency is evident.

The principle technology used for video compression since the last 30 years is based on the hybrid video coding scheme: Using a combination of prediction, transform coding, and quantization of the data, the spatial and temporal redundancy of the video is exploited, and less information is needed for transmission from the encoder to the receiving decoder. Typically, a lossy representation of the original video is reconstructed by the decoder. The performance – or coding efficiency – of a video coding scheme can be assessed by measuring the objective distortion of the reconstructed video for a given data rate.

The objective of this thesis is to improve the inter-picture prediction and coding process for a hybrid video coding scheme such that higher coding efficiency can be obtained. A fundamental principle in modern video coding is the segmentation of every picture into rectangular blocks of pixels. This is visualized in Figure 1.1a. The available methods of prediction and coding, also termed the coding tools, of a video coding scheme are then applied to these blocks individually. Static areas or areas with homogeneous motion of the video can be efficiently coded using large blocks and motion compensated prediction, whereas areas of high spatio-temporal activity typically require a fine block partitioning granularity, prediction error coding, and hence, more data rate. Especially moving natural objects, such as people, trees, or cars can be difficult to approximate using rectangular block partitioning, due to their smooth and curved object boundaries. At low data rates, the rectangular block partitioning can further result in block artifacts, subjectively perceived as poor video quality by human viewers.

This thesis proposes non-rectangular block partitioning as an additional coding tool, to better adapt to the signal characteristics. A specific variant of non-rectangular block partitioning is called geometric block partitioning. In this scheme, a rectangular block is partitioned by a straight line into two segments. This method is visualized in Figure 1.1b. The pixels associated with each segment are then predicted using motion compensation techniques. This idea is not new but has not been adopted into established video coding standards, due to the algorithmic complexity of the involved processes, which is a relevant factor for software or hardware implementations of encoders and decoders.

The main contribution is a fully developed, geometric block partitioning coding tool that provides up to 1% of improved bitrate saving on top of the state-of-the-art without significantly increasing the decoding run time. This is achieved through novel methods of prediction and coding of the inter-prediction side information of geometrically partitioned blocks. Furthermore, methods are demonstrated on how an encoder can estimate the optimal coding parameters of the proposed tool with 6% increased encoding run time. This work has been



(a) Rectangular block partitioning (b) Geometric block partitioning

Figure 1.1 Visualization of block partitioning methods in hybrid video coding schemes.

presented to the Joint Video Experts Team (JVET) of ISO/IEC JTC1/SC29/WG11 (MPEG) and ITU-T SG 16 (VCEG), the leading standardization group for video coding technology, and has been adopted into the working draft of the newest video coding standard in development, Versatile Video Coding (VVC).

First, fundamental topics of signal processing and video coding are briefly introduced in Chapter 2. This is followed by a review of prior art in the field of non-rectangular block partitioning in Chapter 3.

The main body of the thesis is structured into four chapters, covering the topics of representation, quantization, prediction, and coding of geometric partitioning parameters, required modifications to the inter-prediction process, algorithmic simplifications relevant for standardization, and lastly an exploration into methods of transform coding for geometric partitions.

In Chapter 4, two different mathematical representations for geometric block partitioning are studied. Since a block can be partitioned by a straight line in theoretically infinite ways using continuous modeling, quantization is required to be applied to the model parameters that describe the partitioning for discrete blocks. These parameters are signaled to the decoder as additional side-information. A finer quantization may lead to better adaptivity to the video characteristics but also requires more side-information to be signaled. Furthermore, the parameter space that needs to be searched by an encoder is increased. A novel quantization scheme is therefore experimentally derived that maximizes coding efficiency and balances this trade-off for the application of geometric partitioning to blocks resulting from a Multi-type tree (MTT) partitioning. In the second part of the chapter, entropy coding methods to signal the side-information are evaluated. Known methods of spatial prediction of the geometric partitioning parameters are systematically investigated and an improved method of temporal prediction is presented, which can further increase coding efficiency.

In Chapter 5, all adaptations to the inter-prediction scheme for geometric block partitions are presented and optimized. The two predicted geometric segments are combined into a rectangular block for further processing. This is performed using a blending filter, which provides a smooth transition between the two segments. The size of the transition zone affects the coding efficiency and is experimentally optimized. The new insights gained from these contributions lead to the development of a variant of geometric partitioning which is specifically suited for the efficient coding of screen content. Furthermore, differences between uni- and bi-directional motion compensation and the effects on coding efficiency and memory bandwidth are analyzed. Lastly, motion vector coding and storage approaches based on existing methods in VVC are adapted for geometric partitioning.

In Chapter 6, simplifications of the previous developed methods are presented. These simplifications are a necessity for the practical application and standardization of such a coding tool and are in many cases mandated by the JVET standardization group. Methods are derived on how the proposed scheme can be performed with low-complexity integer arithmetic. An encoder strategy is presented that demonstrates how the optimal partitioning parameters can be determined for the first time with reasonable additional encoding time while still performing a nearly exhaustive search.

In Chapter 7, alternative methods to linear block transforms for transform coding of the prediction error are briefly explored. The Shape-adaptive DCT is adapted for VVC and other potential methods, such as transform-skipping and symmetric extension of the prediction error in combination with regular DCT coding are investigated.

In the last Chapter 8, the main findings of the thesis are summarized and an outlook for potential future research is given.

2 Fundamentals

In the following sections, fundamentals topics are introduced that are the foundation of subsequent chapters. First, mathematical concepts are introduced in Section 2.1, followed by the basics of video coding relevant for this thesis in Section 2.2. Since much of the terminology used is coming from High Efficiency Video Coding (HEVC) and also used for its successor VVC, fundamental aspects of video coding are explained in close alignment with these two standards. The experimental setup used for the evaluation of developed algorithms is detailed in Section 2.3. At last, an introduction into the specific inter-prediction coding tools of VVC is given in Section 2.4. Where applicable, the origin and relation of such coding tools to HEVC and the interim Joint exploration model (JEM) is pointed out.

2.1 Mathematical Fundamentals

In this section, mathematical fundamentals are briefly introduced that help in the understanding of this thesis. Since video coding is a field where numerous concepts from signal and image processing, modeling, prediction and entropy coding are being combined, this brief overview cannot claim to be comprehensive. For more details, the reader is referred to [Ohm15], [GW08] or [OS09] for in-depth fundamentals in discrete-time signal processing.

2.1.1 Signal Representation

The predominant types of signals considered in this thesis are two-dimensional digital images and image sequences, i.e. videos. An image or picture in the context of this thesis is a sampled, discrete, digital representation, captured by a camera or computer-generated. It may be defined by a two-dimensional function $s(x, y)$, where x and y are spatial coordinates and s being the amplitude of the image at that point. An element of this two-dimensional field is referred to as a sample or a picture element, in short, a pel. The picture of size $M \times N$, in terms of width and height, can be represented as a matrix $\boldsymbol{S}$:

$$\boldsymbol{S} = \begin{bmatrix} s(0,0) & \cdots & s(M-1,0) \\ \vdots & \ddots & \vdots \\ s(0,N-1) & \cdots & s(M-1,N-1) \end{bmatrix} \tag{2.1}$$

It is sometimes beneficial to interpret the picture as a single row vector $\boldsymbol{s}_\mathrm{r}$ or column vector $\boldsymbol{s}_\mathrm{c} = \boldsymbol{s}_\mathrm{r}^\mathrm{T}$. These representations can be generated by row- or column-wise scanning of the image $\boldsymbol{S}$, obtained by the concatenation of rows of columns into a single vector.

A picture is typically composed of multiple color-channels unless it is a monochromatic picture or video. Depending on the given color model, the color components span a color space. The most relevant color spaces for video coding are the $YC_\mathrm{B}C_\mathrm{R}$ and RGB color spaces that can be easily converted into one another. The $YC_\mathrm{B}C_\mathrm{R}$ color space consists of one luma component Y and two chroma components C_B and C_R, which are the gamma-encoded versions of the luminance and chrominance. An image $\boldsymbol{S}_\mathrm{YCbCr}$ given in this color space could also be thought of as a tensor composed of three planes, e.g. $\boldsymbol{S}_\mathrm{YCbCr} = [\ \boldsymbol{S}_\mathrm{Y},\ \ \boldsymbol{S}_\mathrm{Cb},\ \ \boldsymbol{S}_\mathrm{Cr}]$, making the image effectively a three-dimensional signal. In video coding however, it is much more common to regard *time* as the third signal dimension. Therefore, a video can be thought of as a signal $S(x, y, t)$ with t being the discrete time variable. Unless otherwise stated, S often implicitly refers to the luminance component only. An entire video $\boldsymbol{S}_\mathrm{V}$ with T pictures can now be expressed as the concatenation of T luminance pictures, e.g. $\boldsymbol{S}_\mathrm{V} = [\ \boldsymbol{S}_0,\ \ \boldsymbol{S}_1,\ \ \cdots,\ \ \boldsymbol{S}_t,\ \ \cdots,\ \ \boldsymbol{S}_{T-1}]$.

2.1.2 Random Variables

In signal processing and its applications, it is often the case to model a specific source signal as a stochastic process using random variables. For image and video sources this is necessitated by the fact that these signals most often do not fulfill ideal properties of stationarity, meaning that their statistical properties can vary greatly over time and space. However, methods of statistical analysis can be applied to local groups of samples for which stationarity is assumed. In the field of video coding, statistical analysis plays a crucial role in the design of compression tools: After all, it is the high spatial and temporal redundancy of a video source, quantifiable through statistical properties, that is exploited for compression. These statistics are often measured over a given test set, for specific pictures, or individual blocks.

Sample statistics can be measured for continuous or discrete signals. An important statistical measure for a continuous random process x is the distribution of amplitudes. This is described by the probability density function (PDF) $p(x)$, fulfilling the properties of being a real-valued function for a real-valued process, i.e. $p : \mathbb{R} \to \mathbb{R}$, non-negativity for all x, i.e. $p(x) \geq 0$, and being normalized in the sense of having unit area, i.e. $\int_{-\infty}^{\infty} p(x)\mathrm{d}x = 1$. The PDF for an interval $[a, b]$ over x defines a probability measure $P(x)$, expressing that x takes on values in the range between a and b with the given probability:

$$P([a, b]) = \int_a^b p(x)\mathrm{d}x \tag{2.2}$$

The PDF of a signal or process can be used to derive further statistical properties that characterize it. This is possible in the most general terms using the expected value operator $E\{\cdot\}$, defined as:

$$E\{f(x)\} = \int\limits_{-\infty}^{\infty} f(x)p(x)\mathrm{d}x$$

Intuitively, this can be thought of as a dense summation of probabilities using a weighting function $f(x)$. If $f(x) = x$, the expected value is the *mean value*, also denoted as μ_x and is given by:

$$\mu_x = E\{x\} = \int\limits_{-\infty}^{\infty} xp(x)\mathrm{d}x \tag{2.3}$$

In general terminology, this is denoted as the *first order moment* of the PDF. Higher, n-th order moments of the PDF can be computed accordingly:

$$E\{x^n\} = \int_{-\infty}^{\infty} x^n p(x) \mathrm{d}x \tag{2.4}$$

By subtracting the mean from the weighting function, the so-called *central moments* can be computed. The n-th order central moment is given by:

$$E\{(x-\mu_x)^n\} = \int_{-\infty}^{\infty} (x-\mu_x)^n p(x) \mathrm{d}x \tag{2.5}$$

Most importantly, the second order central moment is given by the variance of the signal, denoted as σ_x^2.

Since all signal sources considered in this thesis are discrete in both space and time, only the discrete statistical equivalents are relevant. If $x(n)$ is now a discrete random process, then the equivalents to the above equations are given by:

$$\mu_x = E\{x(n)\} = \sum_n x(n) \cdot p\left(x(n)\right) \tag{2.6}$$

$$\sigma_x^2 = E\{(x(n)-\mu_x)^2\} = \sum_n (x(n)-\mu_x)^2 \cdot p\left(x(n)\right) \tag{2.7}$$

Since the PDF is now a function of the underlying discrete process x, it is also discrete in nature and therefore termed the probability mass function (PMF) or the discrete density function. The true PDF of a process x remains unknown, but from the observed, discrete realizations of x, the PMF can be estimated by counting the amplitude observations. This results in a so-called histogram of x. A parameter that controls the resolution and therefore visual appearance of the histogram, is the binning of the range of values. It is often unpractical to count every single possible amplitude occurrence of x. Therefore, the entire range of values is subdivided into a series of intervals $\Delta x_k = [x_{a,k}, x_{b,k}]$. Typically, these intervals are non-overlapping and of equal size, e.g. $x_{b,0} - x_{a,0} = x_{b,1} - x_{a,1} = \ldots = x_{b,k} - x_{a,k}$, but this is not a strict requirement. Using the following notation for counting if the amplitude of x falls into the interval Δx,

$$c_{\Delta x}(x) := \begin{cases} 1 & \text{if } x \in \Delta x \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

the histogram p_{H} as a function of the bins Δx_k is given by:

$$p_{\mathrm{H}}(\Delta x_k) = \sum_n c_{\Delta x_k}\left(x(n)\right) \tag{2.9}$$

For simplicity and if equal binning is used, this can also be shortened to $p_{\mathrm{H}}(k)$,where k denotes a bin index.

Another important statistical measure for video compression applications is the covariance. The covariance measures the joint variability of two random variables. Specifically, it measures the linear relationship between these two variables.

The covariance of two discrete random processes $x(n)$ and $y(n)$ is given by:

$$\sigma^2_{x,y} = E\{(x-\mu_x)(y-\mu_y)\} \tag{2.10}$$

This can be extended to define the covariance between any number of sequences or processes. It is specifically interesting to apply the covariance for analysis of the inter-sample relationship that exist within multiple images or a single image. For this purpose, defining a covariance function $\mu_{x,y}(k)$ is helpful:

$$\mu_{x,y}(k) = E\{(x(n)-\mu_x)(y(n+k)-\mu_y)\} \tag{2.11}$$

For the special case $x(n) = y(n)$, Equation 2.11 becomes the autocovariance function. Considering a maximum of K sequences that were extracted from the same sequence $x(n)$, the resulting covariances can be arranged into a symmetric autocovariance matrix $\boldsymbol{C}_{xx}$:

$$\begin{aligned}\boldsymbol{C}_{xx} &= \begin{bmatrix} \mu_{x,x}(0) & \mu_{x,x}(1) & \cdots & \mu_{x,x}(K-1) \\ \mu_{x,x}(1) & \mu_{x,x}(0) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mu_{x,x}(1) \\ \mu_{x,x}(K-1) & \cdots & \mu_{x,x}(1) & \mu_{x,x}(0) \end{bmatrix} \\ &= \sigma^2_x \begin{bmatrix} 1 & \rho_{x,x}(1) & \cdots & \rho_{x,x}(K-1) \\ \rho_{x,x}(1) & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho_{x,x}(1) \\ \rho_{x,x}(K-1) & \cdots & \rho_{x,x}(1) & 1 \end{bmatrix}\end{aligned} \tag{2.12}$$

Since covariances and autocovariances can be difficult to interpret in absolute numbers, due to the dependence on the underlying processes and signal amplitudes, a normalized version of the covariance is often used, termed the correlation coefficient. Here, in Equation (2.12), $\rho_{xx}(k) = \mu_{xx}(k)/\mu_{xx}(0) = \mu_{xx}(k)/\sigma^2_x$ are the autocorrelation coefficients of $x(n)$. The covariance matrix for 2D images, e.g. $x(m,n)$ can be generated by concatenating rows or columns into a one-dimensional column vector. It is clear from this representation that the autocovariance matrix for large values of K quickly becomes unwieldy and difficult to interpret visually. A compact representation for the 2D case is derived in [Cla85].

2.1.3 Markov Chains and Autoregressive Modeling

Markov chains are discrete-time and discrete valued random processes in which the current value depends on the entire past values only through the most recent value. In a K-th order Markov chain, the current value depends on the past values only through the most recent K values. This is called the Markov property. Markov models play an important role in signal processing and are frequently encountered in the context of video coding, for example in binary arithmetic coding (BAC), intra-prediction or in quantization with quantizer state modeling (dependent quantization or trellis-coded quantization, see Section 2.2.7).

An example of a process that fulfills the Markov property is an Autoregressive (AR) process, with $v(n)$ being a zero-mean, independent and identically distributed (iid) white noise process with variance σ_v^2:

$$x(n) = \sum_{i=1}^{K} a_i x(n-i) + v(n) \tag{2.13}$$

AR(1) processes are often used to model images. Following the naming convention $a_1 = \rho$, an AR(1) process has the autocovariance function

$$\mu_{x,x}(k) = \sigma_x^2 \rho^{|k|} \tag{2.14}$$

with a variance

$$\sigma_x^2 = \frac{\sigma_v^2}{1-\rho^2}. \tag{2.15}$$

Typical values for natural images are between $\rho = 0.85$ and $\rho = 0.99$. Such AR(1) processes can for example be used to assess the decorrelation and energy-compaction efficiency of linear block transforms.

2.1.4 Entropy

In information theory, entropy is a quantitative measure of information, originated by Shannon in 1948. For a discrete random process $x(n)$, the entropy is defined as the average self-information. If all possible outcomes of a random process $x(n)$ are defined by a source alphabet $S = \{x_0, x_1, ..., x_{N_s-1}\}$, then the first order entropy $H(S)$ is defined as:

$$H(S) = -\sum_{i=0}^{N_s-1} p(x_i) \log_2 (p(x_i)) \tag{2.16}$$

The unit of entropy is given in bit/symbol. The first order entropy is used for example to design fixed-length and variable-length codes of a given source alphabet. Another important concept in the context of video coding is the *conditional entropy* $H(S_2|S_1)$:

$$H(S_2|S_1) = -\sum_{i_1}^{N_{s,1}-1} \sum_{i_2}^{N_{s,2}-1} p(x_{i_1}, x_{i_2}) \log_2 \left(p(x_{i_2}|x_{i_1})\right) \tag{2.17}$$

The conditional entropy is a measure of the information content of S_2, given the knowledge of the states of S_1. Since $H(S_2|S_1) \leq H(S_2)$, the conditional entropy is the property which is being exploited in predictive coding and context-dependent entropy coding. In this thesis, the latter case is especially relevant, since the Context-adaptive binary arithmetic coding (CABAC) engine of VVC is used for the final stage of coding.

2.1.5 Transforms

In the most general terms, a transform $\boldsymbol{T}$ applied to a 2D input signal $\boldsymbol{X}$ of size $N \times N$ can be expressed as $\boldsymbol{c}_\mathrm{c} = \boldsymbol{T} \cdot \boldsymbol{x}_\mathrm{c}$, where $\boldsymbol{x}_\mathrm{c}$ is the column-scanned vector representation of $\boldsymbol{X}$. Accordingly, the transform $\boldsymbol{T}$ must be of size $N^2 \times N^2$. For practical applications, it is well known that such a *non-separable* transform $\boldsymbol{T}$ quickly becomes difficult to handle for increasing values of N, due to the computational complexity and large memory requirements of storing the non-separable transform matrices. Therefore, special attention is given to *separable* 2D transforms. In this case, the coefficient matrix $\boldsymbol{C}$ containing coefficients $c(x, y)$ resulting from the 2D separable transform of a real-valued block of data $\boldsymbol{X}$ of size $N \times N$ is given by the matrix multiplication $\boldsymbol{C} = \boldsymbol{T}_\mathrm{v} \cdot \boldsymbol{X} \cdot \boldsymbol{T}_\mathrm{h}^\mathrm{T}$, where $\boldsymbol{T}_\mathrm{v}$ and $\boldsymbol{T}_\mathrm{h}$ are the vertical and horizontal transforms, respectively. This overall process is denoted as the forward transform. The corresponding backward transform is given by $\boldsymbol{X} = \boldsymbol{T}_\mathrm{v}^{-1} \cdot \boldsymbol{C} \cdot \left[\boldsymbol{T}_\mathrm{h}^{-1}\right]^\mathrm{T}$. A complex, orthonormal transform furthermore fulfills the property that $\boldsymbol{T}^{-1} = [\boldsymbol{T}^*]^\mathrm{T} = \boldsymbol{T}^\mathrm{H}$,where $\boldsymbol{T}^\mathrm{H}$ is the Hermitian matrix (conjugate transpose) of $\boldsymbol{T}$, and $\boldsymbol{T} \cdot \boldsymbol{T}^{-1} = \boldsymbol{J}$.

Linear block transforms are an essential part of a video coding scheme. They have three important properties useful for video coding: First, they decorrelate the data and provide a frequency-related distribution of energy such that low energy coefficients can be discarded. Secondly, the retained coefficients can be quantized using a scalar quantizer. Here, perceptual considerations relating to the human visual system can also be exploited. Lastly, the sparse matrix of remaining quantized coefficients exhibits symbol redundancies that can be exploited using variable length and CABAC-coding. In video coding, the separability feature of an orthonormal linear block transform is often desired, since it simplifies the computation process. Further considerations are made regarding the internal structure, e.g. symmetry and number of unique values of a transform matrix $\boldsymbol{T}$ [Wie14]. In recent video coding developments, it is also common to apply a secondary, non-separable transform to the coefficients $\boldsymbol{C}$ or a sub-matrix of $\boldsymbol{C}$ in order to further decorrelate them in the transform domain.

An $N \times N$ transform matrix $\boldsymbol{T}$ is composed of rows $\boldsymbol{t}_n$, $n \in \{0, ..., N-1\}$ of basis vectors. The elements $t_n(m)$ of each vector $\boldsymbol{t}_n$ are the coefficients of the transform matrix. Sinusoidal transforms based on the cosine and sine function and their discrete approximations are of particular importance. In total, 8 different discrete cosine (DCT type) and 8 different discrete sine (DST type) transforms are being distinguished. Each of these two types have different even or odd symmetry properties. Table 2.1 lists a selection of some of the more common basis functions in their orthonormal variant [BYR07][Zha+16]. Figure 2.1 visualizes the second basis function for each of the DCT and DST types given in Table 2.1 for length $N = 32$.

The effectiveness of a transform can be assessed using the following criteria:

- The energy packing efficiency η_e, which is the ratio of energy contained within the first T out of U diagonal coefficients:

$$\eta_\mathrm{e}(T) = \frac{\sum_{l=0}^{T-1} E\{c^2(l,l)\}}{\sum_{k=0}^{U-1} E\{c^2(k,k)\}} \tag{2.18}$$

- The decorrelation efficiency η_c, which measures the amount of diagonalization provided by the transform. This is computed by the sum of non-diagonal covariances $\mu_{c,c}$ in the transform domain divided by the sum of non-diagonal covariances $\mu_{x,x}$ in the

Transform type	Basis function $t_n(m)$, $n,m \in \{0,1,\dots,N-1\}$
DCT-II	$t_n(m) = c_n\sqrt{\frac{2}{N}}\cos\left(\frac{n(2m+1)\pi}{2N}\right)$, $c_n = \begin{cases}\sqrt{\frac{1}{2}} & \text{if } n=0\\ 1 & \text{otherwise}\end{cases}$
DCT-V	$t_n(m) = c_n c_m \frac{2}{\sqrt{2(N-1)+1}}\cos\left(\frac{2nm\pi}{2N-1}\right)$, $c_{\{n,m\}} = \begin{cases}\sqrt{\frac{1}{2}} & \text{if } n,m=0\\ 1 & \text{otherwise}\end{cases}$
DCT-VIII	$t_n(m) = \frac{2}{\sqrt{2N+1}}\cos\left(\frac{(2n+1)(2m+1)\pi}{4N+2}\right)$
DST-I	$t_n(m) = \sqrt{\frac{2}{N+1}}\sin\left(\frac{(n+1)(m+1)\pi}{N+1}\right)$
DST-VII	$t_n(m) = \sqrt{\frac{4}{2N+1}}\sin\left(\frac{(2n+1)(m+1)\pi}{2N+1}\right)$

Table 2.1 Transform basis functions.

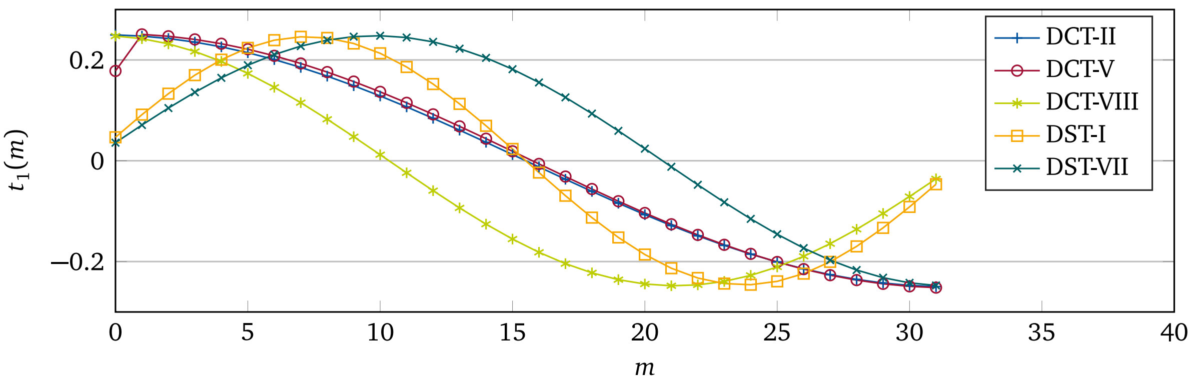


Figure 2.1 Visualization of the second basis function $t_1(m)$ for different DCT and DST types with $N = 32$.

spatial domain.

$$\eta_c = 1 - \frac{\sum_{x\neq y}\left|\mu_{c,c}(x,y)\right|}{\sum_{x\neq y}\left|\mu_{x,x}(x,y)\right|} \tag{2.19}$$

- The coding gain G for Gaussian processes under high-rate assumptions is given by the ratio of the arithmetic and geometric mean values of the squared expected values over the discrete set of U coefficients:

$$G = \frac{\frac{1}{U}\sum_{k,l} E\{c(k,l)^2\}}{\sqrt[U]{\prod_{k,l} E\{c(k,l)^2\}}} \tag{2.20}$$

2.1.6 Quantization

In most general terms, a quantizer maps an input value to an output value according to a specified rule. If just a single input value x is quantized using $Q(\cdot)$ to an output value y, this is termed scalar quantization. If the input $\boldsymbol{x}$ is a vector, this is termed vector quantization. The most common type of scalar quantizers relevant for this thesis are uniform quantizers using a defined step size Δ:

$$y_{\mathrm{T}} = Q_{\mathrm{T}}(x) = \operatorname{sgn}(x) \cdot \left\lfloor \frac{x}{\Delta} + \frac{1}{2} \right\rfloor \tag{2.21}$$

$$y_{\mathrm{T}} = Q_{\mathrm{R}}(x) = \operatorname{sgn}(x) \cdot \left(\left\lfloor \frac{x}{\Delta} \right\rfloor + \frac{1}{2} \right) \tag{2.22}$$

Equation 2.21 is called a mid-tread quantizer, due to the fact that small values $|x| < \Delta/2$ are mapped to zero. Equation 2.22 is called a mid-rise quantizer, mapping small values around zero to $\pm\frac{1}{2}$. Typically, if the output quantization levels of y are odd in number and symmetric about the origin, this corresponds to a mid-tread quantizer. Accordingly, if the number of output quantization levels is even, this corresponds to a mid-rise quantizer.

Quantizers with non-uniform step size can be beneficial in many cases, for example if the Probability density function (PDF) or Probability mass function (PMF) of the signal x is not uniform, if certain signal levels (e.g. $x \approx 0$) contain excessive noise or if it is beneficial to map a larger signal range to zero (e.g. in transform coding). The most common type of non-uniform quantization is the so-called deadzone quantizer. Here, the quantization step size is larger for input values around zero. In image and video coding, quantizers are specified from a decoder point of view. The most common are called Uniform reconstruction quantizers (URQs). In this case, the reconstructed value $\hat{x}$ is given by:

$$\hat{x} = \Delta \cdot y \tag{2.23}$$

The task of finding the optimal y is handled by the encoder and could be determined by simple application of Equations (2.21) or (2.22), or a more complex process, such as Rate-distortion optimized quantization (RDOQ).

2.1.7 Filters

Finite impulse response (FIR) filters are encountered at many stages in a video coding framework. A distinction must be made between linear and non-linear filters, which are both common. If a unit impulse $\delta(n)$ is given as an input to a Linear time-invariant (LTI) system, $h(n)$ is termed the impulse response of the system. For a general input signal $x(n)$, the output

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{2.24}$$

is observed. This is also written using the convolution operator $*$ as $y(n) = x(n)*h(n)$. The effect of filtering is frequently analyzed in the transform domain using an N-point Discrete Fourier transform (DFT). Examples for linear filters in the video coding framework are the sub-sample interpolation filters for motion compensation, FIR Wiener filters or the blending filter applied to the prediction signals resulting from geometric partitioning.

Rank-order, median filters and morphological filters are typical examples of non-linear filters. The de-blocking filters of HEVC and VVC are also examples of non-linear filters. These filters are not based on linear relationships between input and output. Unlike linear filters, they do not have an explicit frequency domain transfer function.

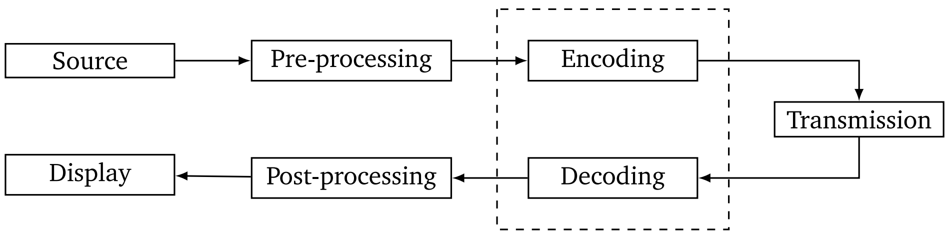


Figure 2.2 General concept of a video transmission scheme. The video coding part is indicated by the dashed box.

2.2 Video Compression Fundamentals

The main idea of video coding is to exploit the high temporal and spatial redundancy of the signal for compression. Two general distinctions can be made: In *lossless* compression schemes, only the statistical redundancies are exploited for coding and no information is lost in the encoding process. In other words, the lossless decoder is able to reconstruct a video from the bitstream which is identical to the source video. This is an important feature in many applications, e.g. in medical imaging where compression artifacts are to be avoided. The achievable compression ratios, measured by dividing the original video file size by the file size of the compressed bitstream, is typically in the range from 2:1 to 5:1 for natural video, heavily depending on the characteristics of the source material. For screen content for example, higher compression ratios may be achieved.

If the condition that all information must be preserved is relaxed, this opens the field of *lossy* compression, which is the predominant type of video coding used for broadcast, streaming, and consumer video on DVD and Blu-Ray discs. Lossy compression exploits not only the statistical redundancy but also the psychovisual redundancy of the video signal by means of color sub-sampling, frequency transforming, or quantization of the prediction error signal. Information about the original video source is discarded that is considered to be irrelevant for the human observer. The more information is discarded – thereby lowering the bitrate – the lower the quality of the compressed video. Naturally, this opens up the question of how the quality of the compressed video could be assessed. This important aspect is further detailed in Section 2.3. An important operating point of a lossy video coder is the bitrate at which the compressed video can be considered *visually lossless,* meaning that most human observers could not tell a difference between the compressed and original video. Considering this specific operating point, compression ratios of 1000:1 and more can be achieved by today's video coding schemes.

The basic concept of a video transmission scheme is visualized in Figure 2.2. A video source, which might be a camera or any other device that outputs a digital video representation, provides a sequence of pictures at a given frame rate. This is followed by an optional pre-processing step, such as a color conversion or denoising process. Then, the encoding is performed, compressing the video and transforming it into a bitstream. In the transmission stage, the data is prepared and sent to the receiver. This stage may involve many sub-steps such as packetization, encryption, channel coding and all of the inverse processes at the receiver side. The decoder, assuming its input is the error-free bitstream, reconstructs the compressed video. Subsequent operations may be applied on the decoded video in a post-processing stage, such as the addition of film grain or a frame interpolation. Lastly, the video is displayed for viewing.

2.2.1 Hybrid Video Coding

The predominant type of video codecs deployed today apply the hybrid video coding scheme. The term *hybrid* is used to emphasize that two different approaches are being combined: Predictive coding and transform coding. The most basic structure of a hybrid video coder is visualized from an encoder perspective in Figure 2.3. Essentially, the hybrid video coder is a DPCM loop: From the input video – more specifically input blocks – a prediction is subtracted. The prediction error or *residual signal* is then transformed, and the resulting transform coefficients are quantized. These are fed to an entropy coder. Since encoder and decoder are synchronized, an inverse transform is also part of coding loop. Therefore, the prediction process only utilizes information that has already been coded and is identically available at a decoder side. The prediction itself might relate to intra-picture prediction (in short intra-prediction), meaning that only already coded samples of the current picture are used, inter-picture prediction (in short inter-prediction), meaning that already coded past pictures are used for prediction, or a combination of both of these approaches.

Before a picture is being output for display or put into the picture buffer, one or more loop filters are applied to remove visible compression artifacts and to improve the dependent prediction process.

Pictures can be distinguished by the type of prediction that is allowed to be applied for the blocks that compose them:

- Intra-coded pictures or *I-pictures* only allow intra-prediction to be applied to the coding blocks. Therefore, every I-picture can be decoded independently.
- Inter-coded pictures with one motion hypothesis per block, e.g. uni-directional or uni-prediction, are denoted as *P-pictures*. This generally does not exclude the possibility that individual blocks are intra-coded.
- Inter-coded pictures with up to two motion hypothesis per block, e.g. bi-directional or bi-prediction, are denoted as *B-pictures*. This generally does not exclude the possibility that individual blocks are intra-coded or using inter-prediction with just one motion hypothesis.

The following sections give brief introductions into the relevant sub-blocks of a hybrid video coding scheme, closely aligned with recent video coding standards, such as HEVC and VVC. More specific information on concrete methods of inter-prediction used in VVC is detailed in Section 2.4.

2.2.2 Source Formats

The most common type of video source format relevant for video coding and also this thesis is called a YUV format. YUV is an integer, byte-aligned, and packed representation of the $YC_\mathrm{B}C_\mathrm{R}$ color space . Historically, YUV was also used for a specific analog encoding of color in television systems, while $YC_\mathrm{B}C_\mathrm{R}$ always related to the digital encoding for video and image compression. Nowadays, both terms YUV and $YC_\mathrm{B}C_\mathrm{R}$ are often used interchangeably.

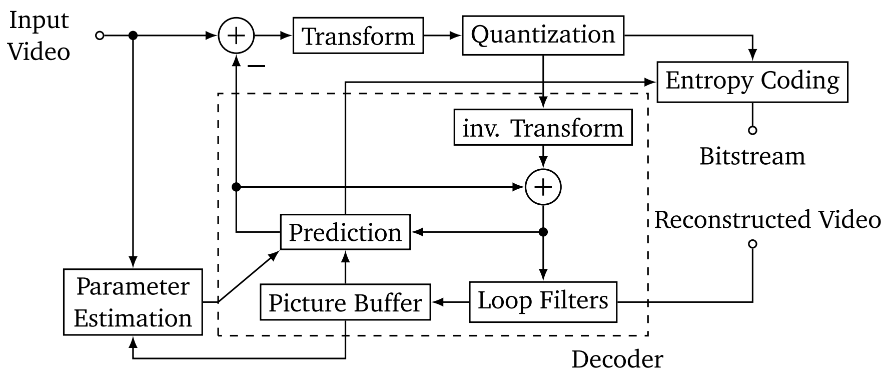


Figure 2.3 General concept of a hybrid video coding scheme.

The luma and chroma components of the $YC_\mathrm{B}C_\mathrm{R}$ color space are encoded with a given bit-depth B_d. For the video sequences included in the JVET test set for example, 8 bit and 10 bit sequences are defined, but higher bit-depths up to 16 bit are encountered in professional applications. The theoretical signal range of each component is therefore given by $[0,\, 2^{B_\mathrm{d}}-1]$. International Telecommunication Union (ITU) recommendations however, which specify common properties and parameters for video standards such as ITU-R BT.709 (HDTV) and ITU-R BT.2020 (UHD), may give different quantization level assignments. BT.709 for example specifies an allowed value range for video data of $[1, 254]$ for 8 bit and $[4, 1019]$ for 10 bit video. In this thesis, the video encoder scales every input source to a 10 bit representation by a factor $2^{B_\mathrm{d}-8}$. The color components in this integer representation are also denoted as the *luma* and *chroma* components.

Since the human visual system is less sensitive to color than it is to structure, sub-sampling of the chroma components is commonly applied in consumer applications. Many sub-sampling schemes exist but three are of special interest for video coding. The sub-sampling scheme is expressed by the ratio between the number of luma and chroma samples:

$$Y : X_1 : X_2 \tag{2.25}$$

The first value Y denotes the number of luma samples considered and the two other values X_1 and X_2 indicate the number of chroma samples in horizontal and vertical direction in relation to the Y value. The three very common types of sub-sampling are:

- 4:2:0, where $X_1 = 2$ specifies that the horizontal chroma resolution is halved. The value $X_2 = 0$ indicates that the same sub-sampling is also applied in the vertical direction. If the luma component of a picture has a resolution of $w \times h$ in terms of width and height, the chroma component has a resolution of $\frac{w}{2} \times \frac{h}{2}$. All JVET test sequences are given in 4:2:0 chroma sub-sampling.

- 4:2:2, where $X_1 = 2$ again specifies that the horizontal chroma resolution is halved. The value $X_2 = X_1$ however indicates that no sub-sampling is applied in the vertical direction. The chroma component would have a resolution of $\frac{w}{2} \times h$.

- 4:4:4, where $X_1 = 4$ specifies that no horizontal sub-sampling is applied and $X_2 = X_1$ specifies that also no sub-sampling is applied in the vertical direction. All components of a picture would have a resolution of $w \times h$. This sub-sampling scheme is often encountered in professional video applications and in captured screen content.

2.2.3 Picture Partitioning

Every picture is partitioned into an array of non-overlapping blocks that are processed in raster-scan order. The blocks in all relevant recent video coding schemes are square-shaped and have a defined maximum size. In AVC for example, a fixed block size of 16×16 luma samples is specified, denoted as a Macroblock (MB). In HEVC, this is increased to a maximum of 64×64 luma samples to cope with higher resolutions and denoted as a Coding tree block (CTB). The same terminology is kept for VVC, but the maximum block size is again increased to 128×128 luma samples. If all three color components are considered as a whole, this is also termed a Coding tree unit (CTU). CTUs may be further grouped into structures that fulfill certain requirements, such as independent decodability or parallel processing capability.

In HEVC and VVC, a CTU may be further split into multiple Coding units (CUs), comprising the Coding blocks (CBs) of each component. In HEVC, the splitting is governed by a quad-tree, always resulting in square shaped CUs while in VVC a quad-tree with nested binary and ternary trees is used. Therefore, a CU in VVC can be square-shaped or rectangular.

Further sub-division of a CU is also possible: Depending on the processing stage that is being considered in the hybrid video coding scheme, the distinction is made between Prediction units (PUs) and Transform units (TUs) with their associated Prediction blocks (PBs) and Transform blocks (TBs). In HEVC, a CU might be partitioned into two rectangular or four square PUs, meaning that prediction is performed for these blocks separately. In VVC however, no sub-partitioning of a CU into PUs is possible. This is a fundamental difference which is substantially modified by the geometric block partitioning scheme presented in this thesis.

Lastly, the pixels of a CU considered in the transform stage of the hybrid video coder may be partitioned into TUs, meaning that separate linear block transforms of smaller size are applied to the samples of each TB. In HEVC, this segmentation is performed by a separate quad-tree while in VVC only a single quad-split may be employed[1].

2.2.4 Intra-Prediction

Intra-prediction tries to predict the samples of the current block by using only already decoded samples of the current picture. Numerous methods are practically and theoretically discussed. Most of these methods try to exploit the high spatial correlation between neighboring samples by simple linear modeling. If the coding scheme supports more than one method, different methods are typically utilized, depending on the properties of the local picture region. The choice which method shall be employed is made by the encoder and signaled as additional side-information. The following common methods are distinguished: Planar prediction is a type of bilinear interpolation of samples using reference samples from the top and left neighborhood of the current block and is particularly suited for the prediction of smooth content with gradual changes in signal amplitude. DC intra-prediction assumes

[1]VVC Draft > 3 / VTM $> 3.2.$, see Intra sub-partitions (ISP) and Sub-block transform (SBT)

that the texture of current block is completely flat and therefore assigns an average DC value, computed from the reference samples, to all samples in the prediction block. Directional or angular intra-prediction methods on the other hand are particularly suited for the modeling of directional structures containing edges. For these methods, the interpolation is performed along an indicated direction [Sul+12].

Novel methods of intra-prediction try to predict the chroma samples from reconstructed luma samples, combine or enhance the result of planar, DC, or directional prediction method with additional information from reference samples or perform further sub-division and intermediate reconstruction of the prediction blocks. Although typically classified as a prediction method specifically for screen-content, the copying of entire blocks of samples can also be considered an intra-prediction method. Such a scheme can even be enhanced by not signaling the location which is to be copied, but rather determining this at the decoder using, for example, template matching.[BCL18; CYK19]

Modeling of the prediction using 2-D non-separable Markov processes is also employed. In this scheme, a block is predicted by recursively sliding a filter over the reference and the newly predicted samples [Che+18].

A new class of algorithms tries to predict the current samples using non-linear modeling, moving away from the aforementioned hand-designed signal models. Advances in Machine Learning in recent years have opened up the field to prediction schemes based on Neural Networks (NN) which were trained on huge data sets. Although truly Deep Neural Networks (DNNs) are still algorithmically too complex for real world video coding applications, a greatly simplified, linear version of NN-based intra-prediction is currently being included in VVC.

2.2.5 Inter-Prediction

Inter-prediction exploits the temporal redundancies of the video source for coding. Unless a scene change is occurring or novel content is being uncovered, the individual pictures of a video sequence are often very similar to each other – except for the motion of any objects present in the video. By estimating this motion and compensating it, a prediction for the current picture based on already coded pictures can be determined. This is accordingly termed Motion-compensated prediction (MCP) and is the key tool for the efficient compression of video.

The classic approach to MCP is to signal a *displacement* or *motion vector* for a given block. Based on the collocated position of the current block in the reference picture, the displacement vector locates a block of samples in the reference picture that is being copied into the current picture. Since the true motion of an object is rarely discrete and coinciding with the pixel grid, interpolation filters are used to approximate the sub-pixel motion.

Novel methods of inter-prediction extend the translational motion model to motion vectors of higher order, such as an affine motion model. In this scheme, the motion of a block is estimated using an affine transformation, which can also capture effects such as zoom, rotation, or shearing. Other current methods of inter-prediction try to improve the prediction of a motion compensated block by capturing irregular sub-block motion, illumination changes or by an adaptive weighting of multiple prediction hypothesis [BCL18; CYK19].

Reference pictures used in motion compensation are also not always required to be pictures used for actual displaying. Techniques are proposed to generate special reference pictures, composited from past and future reference pictures solely for the purpose of prediction [Che+18; RH19].

A big aspect of inter-prediction is the efficient coding of translational motion vectors. Not only the samples of a picture are correlated but also the motion withing a single or across multiple pictures changes only slowly.

2.2.6 Other Compression Methods

Inter- and intra-prediction are the two predominant types of prediction, but other schemes are also deployed in recent video coding standards or those in development. In particular, the combination of two or more prediction hypothesis, merging inter- and intra-prediction signals together, are established methods [BCL18; CYK19][Che+18; RH19].

Another compression approach modifies the dynamic range of prediction samples using linear mapping functions. Effectively, this assigns new code words to the sample values.

In the context of screen content coding, it is common that a source video only makes use of a limited range of colors. This property can similarly be exploited by designing code books that encode the color directly, rather than using the prediction- or transform-based representation [Pen+16].

Another type of compression scheme adaptively changes the resolution of pictures or individual blocks using down-sampling and up-sampling filters in the coding loop [Jos+19]. This makes use of a well-known result from rate-distortion theory that for low rates and specific properties, it is better to perform down-sampling and coding with a smaller quantization step size than encoding at full resolution and discarding many transform coefficients.

2.2.7 Transform Coding and Quantization

Next to prediction, transform coding is the other essential part of the hybrid video coding scheme. Depending on the success of the prediction stage, the prediction error may still contain significant correlation. Video coding schemes such as HEVC and VVC therefore offer different options on how the prediction error is handled and quantized. These choices are signaled as transform coding modes. For HEVC and VVC, the following choices can be distinguished:

- In HEVC, a Residual quadtree (RQT) can be signaled to further segment the residual block into smaller TBs. These smaller blocks can then be coded using separate block transforms. In VVC, only one such split is available, denoted as SBT.

- If the prediction, e.g. through motion compensation, achieves such high accuracy that no residual is required or it is determined through Rate-distortion optimization (RDO) that the residual can be quantized to zero, this is indicated by the *Skip-mode* in HEVC and VVC. The prediction block is therefore used *as is*.

- A sinusoidal block transform is applied to the residual. The main transforms as introduced in Section 2.1.5 used for video coding are the DCT-II and the DCT-III, which is the inverse of the DCT-II and considered the *core* transform, and the DST-VI / DST-VII pair. In the context of image and video coding, the designation *the DCT* implicitly relates to the DCT-II / DCT-III and the designation *the DST* to the DST-VI / DST-VII pair.

In HEVC these transforms can be of size $2^N \times 2^N$ with $N \in \{2,3,4,5\}$. However, the DST is only available for 4×4 intra-predicted blocks. In VVC, the transforms are of size $2^N \times 2^M$ with $N, M \in \{2,3,4,5,6\}$. Here, the DST-VII and the DCT-VIII are added as additional core transforms for variable block sizes. Due to the separability of these transform, it is possible in VVC to apply four different combinations of DCT-VIII and DST-VII in vertical and horizontal direction. The transforms defined in Section 2.1.5 are approximated using integer matrices such that all transform and quantization operations can be performed using fixed-point integer arithmetic, e.g. no intermediate or output values are exceeding a certain bit-depth. In HEVC and VVC, this is specified to be 16 bit.

- Another coding option is to skip the transform and perform quantization directly on the residual signal. This is denoted as *Transform-skip*. Since no decorrelation is applied, this method is mostly suitable for high bitrates, very small prediction errors or prediction errors whose properties are otherwise not suitable for transform coding.
- Extending on the previous method, both transform and quantization can be skipped, therefore allowing perfect reconstruction and a lossless representation of the coded block. This is denoted as *Transquant-bypass* in HEVC.
- Lastly, direct PCM coding of the sample levels is possible. In HEVC, this is performed without applying prediction or quantization to the samples beforehand. In VVC, both prediction and / or quantization of the residual before PCM coding is possible.

HEVC and VVC use similar scalar quantizers. The quantizer step size is determined by the *QP* (Quantization parameter) which has been given to the encoder as a configuration parameter and is therefore also encoded in the compressed bitstream. The quantization step size follows a logarithmic structure such that the step size doubles between the given QP and QP+6. Specifically for HEVC, the quantizer step sizes Δ_Q for QP > 5 are given by [Wie14]:

$$\Delta_\text{Q}(\text{QP}) = \Delta_{\text{Q},0}(\text{QP}\%6) \cdot 2^{\lfloor \frac{\text{QP}}{6} \rfloor} \tag{2.26}$$

$$\Delta_{\text{Q},0}(k) = [\ 2^{-\frac{4}{6}} \quad 2^{-\frac{3}{6}} \quad 2^{-\frac{2}{6}} \quad 2^{-\frac{1}{6}} \quad 1 \quad 2^{\frac{1}{6}}\ ] \tag{2.27}$$

In VVC, the concept of a single scalar quantizer is replaced by a Trellis-coded quantizer (TCQ) (or dependent quantizer), which is composed of two scalar quantizers, mapping to even and odd reconstruction values. A state machine switches between these two quantizers using defined transition rules [Sch+19].

2.2.8 Entropy Coding

Entropy coding is performed as the final step after transform and quantization. In the entropy coding stage, all syntax elements relating to the CUs are written into the bitstream. Unlike previous stages of the hybrid video coder, this is a lossless mapping of information given by syntax elements (or *symbols*) to a coded representation. Two types of entropy coding can be distinguished that are applied in this thesis and in video coding in general: The first type approaches the first order entropy of the given set of symbols. This is performed by

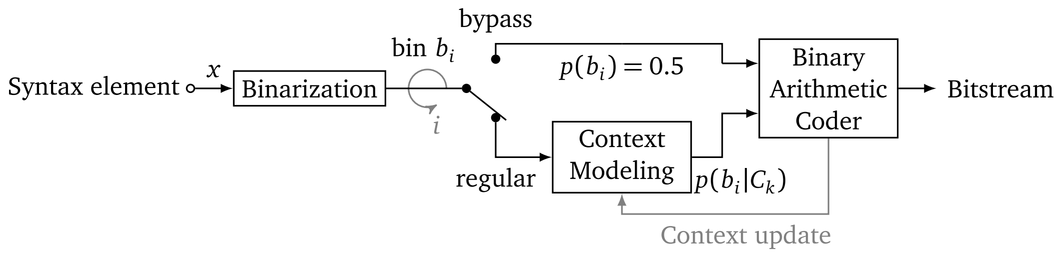


Figure 2.4 High-level view of the CABAC coding engine.

assigning variable length codes to the symbols according to their probabilities. The second type approaches the conditional entropy of the source. This is performed using CABAC. Both approaches can also be combined to achieve highest compression efficiency.

In the following sub-sections, introductions into these two coding types are given. The overall entropy coding scheme is visualized in Figure 2.4. First, syntax elements are binarized. Then, every *bin* b_i of a *bin-string* b is given to the CABAC-coding engine. Here, the bin can be coded with a fixed probability of $p(b_i = 0) = p(b_i = 1) = 0.5$, called the bypass-mode, or with a probability that is derived from a context model, called the regular-mode. For a given bin, many different contexts may be available that are selected based on additional criteria. In every available context model, the probability of the bin having a value of 0 (or 1) can be different from 0.5 and is updated after every coding step. Hence, the context model adapts to the probability of actual coded 0's and 1's for the specific bin.

2.2.8.1 Binarization

The entropy coding in HEVC and VVC is performed using a *binary* arithmetic coder. Therefore, every non-binary syntax element needs to be turned into a bin-string of 1's and 0's before being given to the coding engine. The binarization is the process of converting a non-binary value x into a binary code $b = C(x)$. Conversely, the inverse process $x = C^{-1}(b)$ is called de-binarization. In the context of this thesis, different binarizations are applied to syntax elements, which are selected based on assumptions or measurements made about the distribution of $p(x)$.

Although the binarization could be an arbitrary mapping of binary code words b to values x, systematic approaches play a special role in video coding due to their simplicity. The following overview summarizes the different relevant binarization approaches.

A source alphabet $S = \{s_0, s_1, ..., s_x, ..., s_{N_S-1}\}$ shall have a maximum of N_S entries. The entries s_x, $x \in \{0, ..., N_S-1\}$ in the given alphabet can have arbitrary integer or floating-point values. Therefore, the integer value x only relates to the index of the alphabet entry s_x. In actual coding examples, it is however often the case that no such remapping is required, e.g. an alphabet S may be given by $S = \{c, c+1, ..., x, ..., N_S + c - 1\}$ with some $c \geq 0$, and therefore directly encoding a syntax element.

The length of a code word shall be measured in bit, e.g. the code word $b = [\ 1\ \ 1\ \ 0\ ]$ has a length of $N_C(b) = 3\,\text{bit}$. Each element of b is called a *bin* b_i, which can always take the value of 0 or 1. The entire code word b is thus denoted as a *bin-string*.

2.2.8.2 Fixed-length Codes

The most trivial fixed-length (FL) code is a flag, which can be coded with 1 bit. If the source alphabet contains N_S entries, a fixed length code of length $N_\mathrm{C} = \lceil \log_2 N_\mathrm{S} \rceil$ is required:

$$C_{\mathrm{FL},N_\mathrm{C}}(x) = [\ b_0 \quad b_1 \quad \cdots \quad b_{N_\mathrm{C}-1}] \tag{2.28}$$

2.2.8.3 Systematic Variable-length Codes

Systematic variable length codes are constructed using defined rules. The most common codes are unary codes (U), truncated-unary codes (TU), truncated-binary codes (TB), and Exponential Golomb codes of order k (EG,k).

Unary Code The unary code, or thermometer code, represents a value x by x bits of one parity and a stop bit of the other parity. The length of a code word b is therefore $N_\mathrm{C}(b) = x + 1$.

$$C_\mathrm{U}(x) = [\underbrace{\ 1 \quad 1 \quad \cdots \quad 1\ }_{x \text{ times}} \ 0\ ] \tag{2.29}$$

Truncated-Unary Code The truncated-unary code corresponds to the unary code with a defined maximum value N_S. Thus, for the last valid code word, the terminal bit can be omitted:

$$C_{\mathrm{TU},N_\mathrm{S}}(x) = [\underbrace{\ 1 \quad 1 \quad \cdots \quad 1\ }_{x<N_s-1 \text{ times}} \ b_{N_\mathrm{S}-1}\ ],\ b_{N_\mathrm{S}-1} = \begin{cases} 1 & \text{if } x = N_\mathrm{S} - 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.30}$$

Truncated-Binary Code The truncated-binary code is an extension of the fixed-length code for maximum values which are not a power of two, e.g. $N_\mathrm{S} \neq 2^k$, $k \in \mathbb{N}_0$. If $l = \lfloor \log 2(N_\mathrm{S}) \rfloor$, then truncated binary coding assigns the first M symbols to code words of length l and the remaining symbols $N_\mathrm{S} - M$ to code words of length $l + 1$, where $M = 2^{l+1} - N_\mathrm{S}$. For these remaining code words, the input value x is offset by M. The last bin b_l can therefore be omitted for all $x < M$.

$$C_{\mathrm{TB},N_\mathrm{S}}(x) = [\underbrace{\ b_0 \quad \cdots \quad b_{l-1}\ }_{x<M} \underbrace{\ b_l\ }_{x \geq M}] \tag{2.31}$$

Exponential-Golomb Code The k-th order Exponential Golomb code is constructed by using a unary prefix code and a suffix of configurable length. The number of prefix bins l is determined by the value x and the order k as follows:

$$2^k(2^{l-1} - 1) \leq x < 2^k(2^l - 1) \tag{2.32}$$

The number of bins m in the suffix is determined by $m = k + l - 1$. The suffix encodes the value $x - 1 - 2^k(2^{l-1} - 1)$ with a fixed-length binary code.

x	$C_{\mathrm{FL},3}(x)$	$C_{\mathrm{U}}(x)$	$C_{\mathrm{TU},5}(x)$	$C_{\mathrm{TB},5}(x)$	$C_{\mathrm{EG},0}(x)$	$C_{\mathrm{EG},1}(x)$	$C_{\mathrm{EG},2}(x)$
0	000	0	0	00	0	0 0	0 00
1	001	10	10	01	10 0	0 1	0 01
2	010	110	110	10	10 1	10 00	0 10
3	011	1110	1110	110	110 00	10 01	0 11
4	100	11110	1111	111	110 01	10 10	10 000

Table 2.2 Example binarizations using the systematic codes detailed in Section 2.2.8.1.

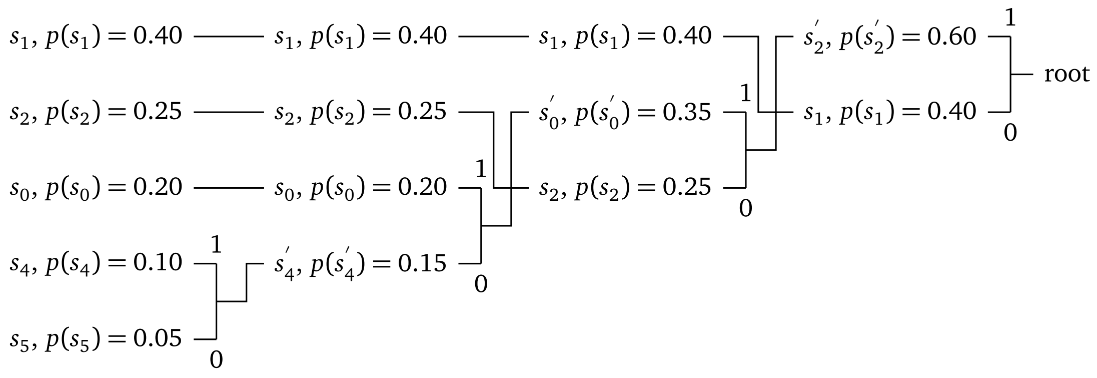


Figure 2.5 Visualization of Huffman code tree construction.

$$C_{\mathrm{EG},k}(x) = [\ \underbrace{1\ \ 1\ \ \cdots\ \ 0}_{l\text{ prefix bins}}\ \underbrace{b_0\ \ b_1\ \ \cdots\ \ b_{m-1}}_{m\text{ suffix bins}}\ ] \tag{2.33}$$

An example for these different binarizations is shown in Table 2.2.

2.2.8.4 Basic Huffman Coding

Huffman coding is a type of variable length coding that produces prefix-free codes, approaching the first order entropy $H(S)$ of the given source alphabet S. If the probability mass function $p(s_i)$ for all source symbols $S = \{s_0, s_1, ..., s_i, ..., s_{N_S-1}\}$ is known or estimated, the code is constructed using a so-called Huffman code tree. The construction starts by combining the two-least probable symbols and assigning them the codes 0 and 1 respectively. These two symbols are removed from S but combined to a new entry which has the sum probability of both symbols. This combined symbol is re-added to S. Then, again the two least probable symbols are determined which are assigned the codes 0 and 1. The process continues until S contains only one entry. This will be the root node of the Huffman code tree and the construction of the code has finished. Figure 2.5 visualizes the process for a set of five symbols with given probabilities.

2.2.8.5 Context Modeling and Arithmetic Coding

In HEVC and VVC, all syntax elements on the block level are coded with CABAC. As visualized in Figure 2.4, every bin b_i is fed into the arithmetic coding engine. Here, the pre-determined decision is made, whether the bin is coded in bypass-mode, assuming equiprobability of the bin values, or in regular-mode where an adaptive probability based on context modeling is employed. A context model C_k with context index k can be seen as the conditional probability $p(b_i|C_k)$ that a bin takes a specific value $b_i = 0$ or $b_i = 1$, given the context C_k. The context index k per bin indicates that more than one context may be available. This necessitates a context selection process. The most commonly used types of context modeling and selection types are:

- Spatial context selection: The applicable context model depends on the existence and value of previously coded syntax elements in the local neighborhood. For example, the selection of a context for a specific coding mode flag in the current coding unit may depend on the value of the coding mode flag coded for the left and top neighboring coding units. If the value of the flag for the left or top coding unit is true, a context C_1 is chosen. If the value of the flag is true for both neighbors, a context C_2 is chosen. Otherwise, a default model C_0 is chosen.

- Slice type dependent selection: Different context models are chosen based on the current slice type (I, P, or B), assuming, for example, that the coding statistics vary with different picture types and different context initialization values are given.

- Component dependent selection: Syntax elements relating to luma or chroma samples can be coded with different contexts.

- Bin index dependent selection: A context C_k is chosen according to the bin index i. In the simplest case, $k = i < M$ is specified, where M denotes a maximum number of regular coded bins. If the bin-string is longer than M, the remaining bins are coded using bypass mode. This is frequently done in video coding standards to limit the complexity of CABAC coding.

The aforementioned approaches can also be combined. The probability provided by a context model is directly used for the coding interval subdivision process used in arithmetic coding. For more information on arithmetic coding, the reader is referred to [WNC87] and [MNW98]. Once a context C_k is selected and the bin value of b_i has been coded, the probability of $p(b_i|C_k)$ is updated. In HEVC and VVC, the probability update is modeled using finite state machines. The state of a context is represented by the probability of the least probable symbol $0 < p_{\text{LPS}} \leq 0.5$ and by the value of the most probable symbol $b_{\text{MPS}} \in \{0, 1\}$. If the most probable symbol is coded, p_{LPS} decreases. If the least probable symbol is coded, p_{LPS} increases. The step size by which the probability increases or decrease is dependent on the current state, often following a logarithmic function. If p_{LPS} reaches the equiprobability threshold of $p_{\text{LPS}} = 0.5$, the polarity of b_{MPS} is flipped. More specific information on CABAC can be found in [MSW03] and for HEVC in [SB12]. In contrast to HEVC, VVC uses two state machines as probability estimators, having different adaptation rates.

2.2.9 Loop Filters

The last step in the processing flow of the hybrid video coding scheme and before a picture can be displayed is the in-loop filtering. The goal of the in-loop filtering is to reduce coding noise and edge-artifacts. This improves both the objective coding efficiency as well as the subjective visual quality of the reconstructed video. Two types of filters can be distinguished: Non-linear and linear filters. The first type comprises filters such as the *de-blocking* filter (DF) and the Sample-adaptive offset (SAO) filter, which are both included in HEVC. The second type of filter is based on Wiener-filtering approaches. One particular method is called the Adaptive loop filter (ALF), which is included in VVC. Both SAO and ALF require the signaling of filter parameters. For more information, the reader is referred to [Fu+12] and [CYK19].

2.2.10 Rate-Distortion Optimization

The final bitrate of the compressed video depends on a number of factors. The video content and its intrinsic characteristics (resolution, frame rate, spatial and temporal activity, natural or screen content, etc.) are obvious aspects that influence by how much a video can be compressed for a given quality and is considered to be outside of reach in this thesis. The other factors however, which are freely choosable, are the coding tools and the dependent coding parameters for these methods. Considering a single video sequence and a fixed set of coding tools, the rate R versus the distortion D characteristic for a given set of coding parameters $\boldsymbol{P}$ provides a means to assess the quality of the choice of parameters. If all operating points $\boldsymbol{P}$, e.g. sets of coding parameters, are visualized in the RD-plane, then the best choice of coding parameters is given by the convex hull of these points. For a fixed rate R_T, this can also be formalized as the minimization problem:

$$\min_{\boldsymbol{P}} D(\boldsymbol{P}) \text{ subject to } R(\boldsymbol{P}) \leq R_\mathrm{T} \tag{2.34}$$

For a practical coder, this set of parameters can be found through RDO. As can be seen from the introductions given above, a wide range of coding parameters (block sizes, prediction modes, transform coding modes, loop filter settings) are available for recent video codecs. The process of selecting the best set of parameters is the task of the encoder and out of the scope of a video coding standard. In the reference encoders for HEVC and VVC however, the method of choice for performing RDO is Lagrangian optimization. The Lagrangian cost function for a coding unit is given by:

$$J(\boldsymbol{P}|\lambda) = D(\boldsymbol{P}) + \lambda R(\boldsymbol{P}) \tag{2.35}$$

The Lagrange multiplier λ controls the slope along a line of operating points in the RD-plane. Minimizing J for $\lambda = 0$ would only minimize the distortion and $\lambda \rightarrow \infty$ would conversely only minimize the rate. The constrained problem given in Equation 2.34 can now be represented as an unconstrained problem using the Lagrangian cost function:

$$\boldsymbol{P}_\mathrm{opt} = \arg\min_{\boldsymbol{P}} (D(\boldsymbol{P}) + \lambda R(\boldsymbol{P})) \tag{2.36}$$

In the reference encoders for HEVC and VVC, the minimum cost J is determined by a nearly exhaustive search of the best coding tools for a given coding unit in a one-pass encoding, although certain speed-ups and short-cuts are utilized to find individual coding parameters. Clearly, this method does not cater to the fact that spatial or temporal dependencies between coding parameters of consecutive coding units may exist. For example, the selection of a precise but more costly (in terms of J) motion vector for the current coding unit could result in lower RD-costs for subsequent coding units due to improved prediction. This is not considered in the reference encoders. Theoretically, Lagrangian optimization should be performed over all coding units jointly, which undoubtedly would make the encoding process more complex.

2.3 Evaluation and Metrics

The algorithms and methods developed in the context of this thesis have been implemented into the reference software for VVC, termed the VVC Test Model (VTM) [Joi20]. For most of the experiments conducted in Chapters 4, 5, and 7, VTM-3.2 has been used. For the simplifications proposed to JVET, detailed in Chapter 6, VTM-5.0 has been used. The coding performance of the proposed algorithms is analyzed using objective metrics. For this purpose, coding simulations are performed in which a set of specified video sequences is encoded and decoded according to the JVET Common test conditions (CTC). For determining the rate-distortion performance, the decoding process is technically not required, since all metrics can be gathered at the encoder side. However, next to the rate-distortion performance, complexity measurements in terms of encoder and decoder run time are also recorded, requiring the decoding step. In this thesis, the rate and distortion measurement are determined always from the bitstream and the decoded video, not from measurements output by the reference software. This provides an additional safety against possible software issues at the encoder. For every simulation performed, a perfect match of the reconstructed video at the encoder side and at the decoder side is asserted.

The coding performance of the proposed algorithms and methods is assessed in terms of Bjøntegaard-delta (BD)-rate changes. Therefore, the coding results of one simulation acts as an *anchor* or reference against the *tested* simulation. Except for Chapter 7, the common anchor is always the unmodified respective software version. In addition to the evaluation according to the JVET CTC, a visual assessment is performed using Video Multi-Method Assessment Fusion (VMAF) [Net20]. Although VMAF is reported to have high correlation with visual test results performed by human test subjects and is used extensively in industry, it is not a metric per se and has not yet achieved widespread academic or standardization activity adoption. Since VMAF cannot replace actual visual testing using human viewers, which is an elaborate and time-consuming process, these results can be seen as supplementary and indicators at most.

2.3.1 JVET Common Testing Conditions

If not otherwise stated, the experiments conducted in this thesis follow the JVET CTC, specifically [Bos+18a] for experiments with VTM-3.2, and [Bos+18b] for experiments with VTM-5.0. The two test specifications do not differ in terms of the sequences to be tested but only in the applicable configuration files supplied to the encoder.

There are four encoding scenarios described in the JVET CTC:

- Random-access (RA): This scheme provides the highest compression performance. The coding order of pictures differs from the output order, as the pictures are arranged in a so-called *hierarchical-B* coding structure. I-picture are inserted in this coding structure at roughly every second, allowing the decoding process to be started at these locations independently. The pictures composing such an independently decodable structure are also called a Group of Pictures (GOP). RA is the typical configuration for streaming and broadcasting applications.
- Low-delay B (LDB): In this scheme the coding order is equal to the output order, therefore no structural delay is introduced. Pictures in this configuration may use bi-prediction. The typical application for LDB is video conferencing, where no random-access property is required.
- Low-delay P (LDP): This scheme is identical to the LDB case, except that pictures may only use bi-prediction.
- All-Intra (AI): In this scheme all pictures are coded using intra-prediction and can therefore be decoded independently. The typical application are video editing or video encoding with extremely low delay.

Due to the importance of RA, all experiments in this thesis are conducted in this configuration. In Chapter 6, additional LDB coding results are provided. Since this thesis deals with the improvement of inter-prediction, no AI simulations are conducted.

The JVET CTC specify 26 video sequences of different resolutions and characteristics. Video sequences of same resolution are grouped into classes. For natural video content, the following classes are specified:

- Class A1 and A2: 6 UHD sequences of size 3840×2160 luma samples.
- Class B: 5 HD sequences of size 1920×1080 luma samples.
- Class C: 4 WVGA sequences of size 832×480 luma samples.
- Class D: 4 WQVGA sequences of size 416×240 luma samples.
- Class E: 3 WXGA sequences of size 1280×720 luma samples.

Class A1 and A2 are not mandatory for LDB simulations, while class E is only mandatory for LDB simulations. For screen content, the JVET CTC only specify one additional class. To augment this, coding experiments in Chapter 6 are performed using additional screen content sequences containing text with graphics and motion (TGM), specified in the CTC for screen content coding of HEVC [Yu+15]:

- Class F: 4 sequences of mixed resolution and content (computer generated, screen content, natural video with screen overlay).
- Class TGM: 4 sequences of size 1920×1080 luma samples, exclusively containing screen content.

The overall coding efficiency in terms of BD-rate change is always reported according to the JVET CTC, specifying an average in terms of BD-rate change to be calculated over classes A1, A2, B, C, and E, if applicable. All simulations are run with quantizer settings using base QP={22, 27, 32, 37}.

2.3.2 Bjøntegaard Delta Measurements

The Bjøntegaard-delta (BD) [Bjø01] measurement provides a single number to easily asses the coding performance of two different coding schemes. The basic concept is to interpolate a curve between the rate-distortion operating points of the two coding schemes under consideration. The rate of each operating point is given in kbps, whereas the distortion is measured by the Peak Signal-To-Noise Ratio (PSNR). Then, the difference between these two curves is integrated. Only the overlapping interval of both curves is used for the calculation. Here, it is distinguished between BD-rate, where the overlapping rate interval is chosen, and BD-PSNR, where the overlapping PSNR interval is chosen. According to the JVET CTC, a piece-wise cubic interpolation method for calculation of the BD measurements is used. The interpolation is performed in the logarithmic domain both for the rate and the distortion, already given by the PSNR.

For consistency and ease of language in this thesis, a negative percentage in terms of BD-rate always reflects a bitrate saving and hence, higher compression efficiency.

2.3.3 VMAF

It is a well-known fact that the PSNR does not consistently reflect the human perception of image or video quality. Especially at low bitrates, humans can judge the quality of two compressed videos with equal PSNR but otherwise distinct coding parameters very differently, due to the perception of spatial and temporal compression artifacts. The Video Multi-Method Assessment Fusion (VMAF) measurement provides a prediction of subjective visual quality. Although the authors term VMAF as a "metric", it does not fulfill all the required mathematical conditions of a metric, such as symmetry and triangle inequality. For a given coded video and the uncompressed reference, VMAF provides a so-called *VMAF score*, which is a number between 0 and 100. The score of 100 corresponds to the quality of the reference, thereby indicating excellent quality if the reference is the uncompressed video source. A score of 0 corresponds to poor quality. Therefore, VMAF can also be seen as a prediction of the Differential Mean Opinion Score (DMOS), frequently reported in visual testing for video and image coding.

At its core, VMAF computes two objective image measures, Visual Information Fidelity (VIF) [SB06] and Detail Loss Metric (DLM) [Li+11], and the temporal difference in terms of absolute difference between adjacent frames. These measures are fused to a single feature using a support vector machine regressor, assigning weights to each elementary measure. This machine learning model is trained and tested using the opinion scores obtained through real subjective visual testing.

In analogy to BD-measurements, a bitrate vs. VMAF-score plot for two coding schemes provides a comparison of the visual performance. Since the VMAF-scores over rate shows a similar logarithmic behavior as the PSNR, it is proposed in this thesis to compute a "BD-VMAF" using the same piece-wise cubic interpolation method as for BD-rate and BD-PSNR. These BD-VMAF values are reported for all experiments in the appendix.

2.4 Overview of Inter-Prediction in Versatile Video Coding

After the finalization of HEVC in 2013, the HEVC Format Range Extension (RExt), Scalable High Efficiency Video Coding (SHVC) and Multiview High Efficiency Video Coding (MV-HEVC) in 2014, 3D High Efficiency Video Coding (3D-HEVC) in 2015, and the Screen Content Coding (SCC) extension in 2016, the focus of the joint standardization activity between Visual coding experts group (VCEG) of ITU-T and Moving picture experts group (MPEG) of ISO quickly shifted towards the development of a new video coding standard. The Joint Video Exploration Team (JVET, later changed to Joint Video *Experts* Team) was founded between these two entities and started working on video coding technology which surpasses the coding efficiency of HEVC. Over the course of several joint meetings, the Joint exploration model (JEM) was developed, which consists of additional coding tools and modified existing methods based on HEVC. JEM in its final version 7.0 achieved −28.5 % luma BD-rate change for RA over HEVC, coded with HM-16.6, at roughly 10 times encoding run time and 7 times decoding run time [Che+19b]. Therefore, it was concluded that significant evidence exists that coding efficiency beyond HEVC can be obtained, justifying a new standardization cycle.

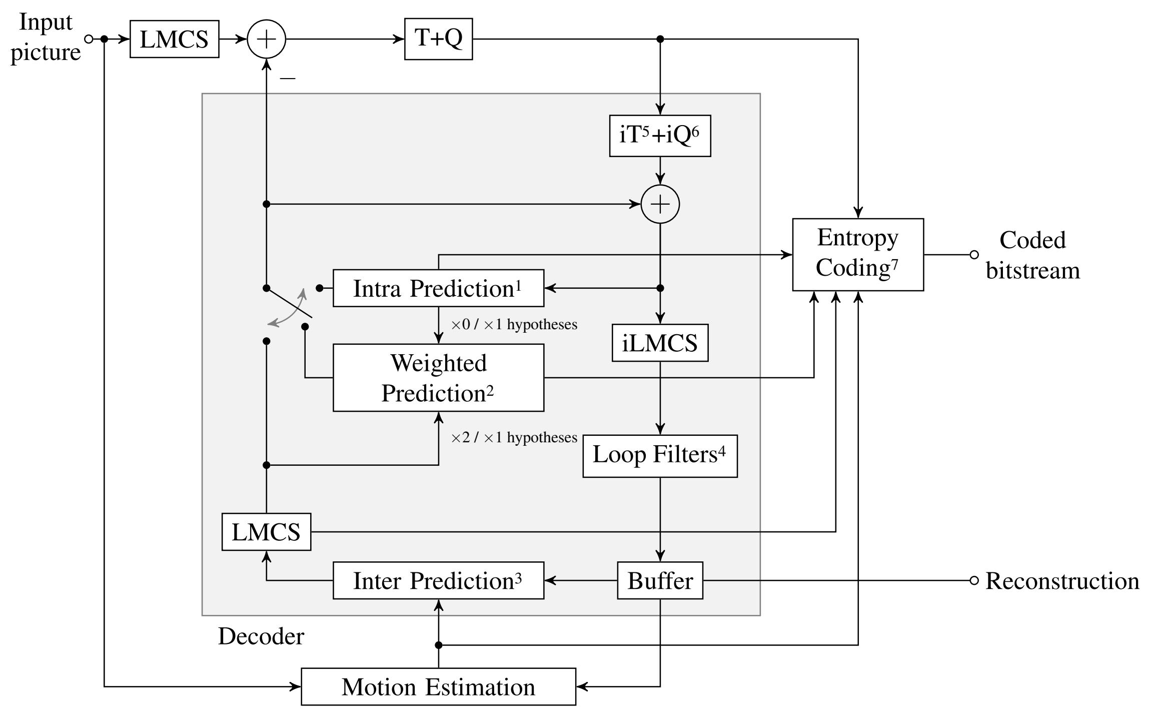
This was followed by a *Joint Call for Proposals on Video Compression with Capability beyond HEVC* [Seg+17], which was answered by 32 organizations from industry and academia with technology proposals. The majority of these proposals were based on JEM. The most promising new coding tools and proven coding tools from HEVC were selected for a first test model. The new standardization project was given the name Versatile Video Coding, to stress its broad range of targeted applications, such as 360° and immersive video – next to classical 2D video. Screen content coding is also a component of VVC from the start.

In the following, a brief overview into specific aspects of VVC is given. Since the proposed geometric partitioning tool is an inter-prediction coding tool, no further details are given on novel intra-coding tools in VVC or novel loop-filters. Instead, the reader is referred to [BCL18; CYK19] and subsequent standardization documents, due to the ongoing nature of the VVC standardization. Therefore, certain aspects detailed might be subject to change in the future.

Figure 2.6 shows an updated version of the basic hybrid video coding block diagram shown before in Figure 2.3, including all coding tools for VVC (Draft 5).

First and foremost, but not visible in the figure, the block partitioning structure of VVC is different compared to HEVC, as already introduced in Section 2.2.3. The MTT (or Quadtree binary tree ternary tree (QTBTTT)) structure provides means to partition a picture with higher flexibility. This effectively allows a more efficient segmentation of the content, according to its spatial and temporal properties, providing a large amount of the overall improved coding gain over HEVC. Consequently, coding tools that operate on the block level are implicitly influenced by the superordinate block partitioning. For a reference encoder, which shall demonstrate the coding performance of the scheme, this means that more coding parameters can be tested during RDO. This is an important aspect when a new coding tool such as the proposed geometric partitioning tool is added to the codec.

VVC has significantly more inter-prediction coding tools compared to HEVC. Many of these tools build upon the concept of weighted prediction, where multiple prediction hypotheses are combined using a weighted average. Where HEVC only distinguished between Adaptive motion vector prediction (AMVP) and Merge-mode motion vector coding for inter-predicted



Intra Prediction[1]: 67 Intra Modes, CCLM, PDPC, MRL, ISP, MIP

Weighted Prediction[2]: TPM, CIIP, BCW

Inter Prediction[3]: AMVP, Ext. Merge, MMVD, Affine MC, SbTMVP, AMVR, BDOF, DMVR, SMVD

Loop Filters[4]: DF, SAO, ALF

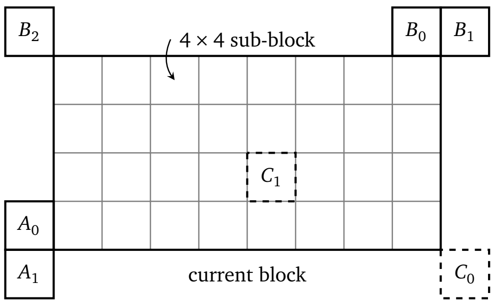
Transform[5]: LBT, MTS, LFNST, SBT, TS, TQB

Quantization[6]: Dep. Scalar Quantization, Joint Chroma Coding

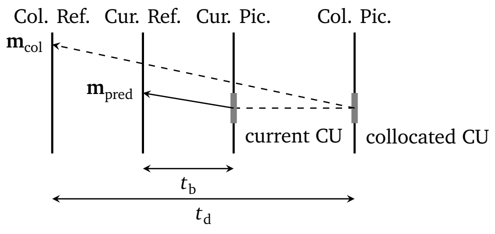
Entropy Coding[7]: Multi-Hypothesis CABAC

Screen Content Coding: IBC, RDPCM

Figure 2.6 High level block diagram of VVC (Draft 5).



(a) Spatial and temporal merge candidates.



(b) Temporal scaling process.

Figure 2.7 Visualization of merge candidate positions and temporal motion vector scaling for VVC.

blocks [Wie14], VVC has the following prediction and coding tools:

- The Merge-mode is a coding tool known from HEVC which allows the signaling of a motion vector (uni- or bi-directional) for the current coding unit from a list of unique merge candidates. No motion vector difference is added. Therefore, Merge-mode allows the efficient representation of highly correlated motion vector fields. The merge candidate list is constructed from the following type of candidates $\boldsymbol{m}_{\text{pred},j}$, $j \in \{0, ..., 5\}$ in the given order:

  1. Spatial merge candidates from neighboring CUs
  2. Temporal merge candidates from collocated CUs
  3. History-based merge candidates from a FIFO table
  4. Pairwise averaged merge candidates
  5. Zero motion vectors

  The spatial candidate positions $A_{\{0,1\}}$ and $B_{\{0,1,2\}}$ as well as the temporal positions $C_{\{0,1\}}$ are indicated in Figure 2.7a. For the predicted temporal motion vector $\boldsymbol{m}_{\text{pred}}$, an additional scaling process is performed according to the POC distances t_{b} and t_{d} of the involved pictures. This is indicated in Figure 2.7b. The history-based merge candidate is a novelty compared to HEVC. For each CTU row, a FIFO table of 6 unique motion vectors is constructed and updated after coding of each CU.

- The Merge mode with MVD (MMVD) is an additional merge-based tool. Here, the first or second merge candidate from the regular merge candidate list, as detailed above, is further refined using a motion vector difference (MVD), e.g. $\boldsymbol{m}_{\text{mmvd}} = \boldsymbol{m}_{\text{pred},\{0,1\}} + \boldsymbol{m}_{\text{mvd}}$. The motion vector difference is signaled using a type of non-linear vector quantization using a distance index for the magnitude and a direction index, e.g. $\boldsymbol{m}_{\text{mvd}} = m_{\text{mag},k} \cdot \boldsymbol{n}_l$. The mapping is shown in Table 2.3. In total, 64 different motion vectors can be signaled using MMVD.

- The Triangular Prediction Mode (TPM) can also be considered as a weighted prediction coding tool, using merge-based motion vector prediction. TPM performs a leaf partitioning of the current coding unit into two triangles. The samples $p_{\text{TPM},k}(x, y)$, $k \in \{0, 1\}$ contained in each triangle are inter-predicted using a spatial merge candidate list, containing only uni-directional motion vectors. The final prediction block

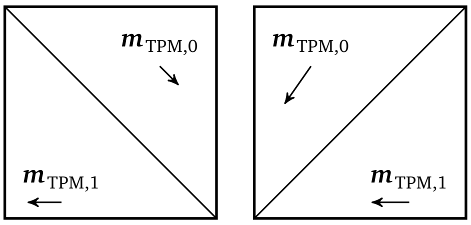
Dist. index k	0	1	2	3
$m_{\text{mag},k}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2
Dist. index k	4	5	6	7
$m_{\text{mag},k}$	4	8	16	32

(a) Distance quantization.

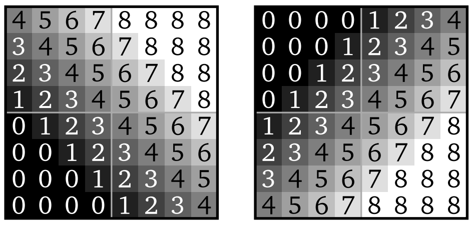
Dir. index l	0	1	2	3
$\boldsymbol{n}_l$	$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}-1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}0\\-1\end{bmatrix}$

(b) Direction quantization.

Table 2.3 Mapping for distance and direction index for MMVD.



(a) Triangular partitioning concept.



(b) Blending weights $w_{\text{TPM}}(x, y)$.

Figure 2.8 Visualization of the triangular prediction mode for VVC.

$p_{\text{Block}}(x, y)$ is composited from the prediction samples of each triangle using a blending filter. This blending filter defines per-sample weights $w_{\text{TPM}}(x, y)$ and computes a weighted average based on the distance of each sample to the diagonal or anti-diagonal partitioning line.

$$p_{\text{Block}} = \frac{w_{\text{TPM}} \cdot p_{\text{TPM},0} + (8 - w_{\text{TPM}}) \cdot p_{\text{TPM},1}}{8} \text{ for all } x, y \tag{2.37}$$

Effectively, this provides a smooth transition at the boundary between the two samples. These concepts are visualized in Figure 2.8.

- Affine motion compensated prediction extends the translational motion model to a 4 or 6 parameter affine motion model. The affine motion field of a block is fully described using two control point (4 parameter model) or three control point (6 parameter model) motion vectors. Affine motion compensation can be performed using AMVP or Merge-based signaling of the control point motion vectors. To simplify the motion compensation process, the affine transform is performed on a block basis. For each 4×4 luma sub-block, a motion vector relating to the sub-block center is derived from the control point motion vectors. Unlike regular motion compensation in $1/4$ sample accuracy, this is performed using $1/16$ sample accuracy. Hence, higher accuracy interpolation filters for affine motion compensation are defined.

- Sub-block temporal motion vector prediction (SbTMVP) is a special type of temporal motion vector prediction at the sub-block level. Unlike regular Temporal motion vector prediction (TMVP), SbTMVP applies a motion shift before fetching the temporal motion information from the collocated picture, where the motion shift is obtained from the motion vector from one of the spatial neighboring blocks of the current CU. While TMVP fetches a single motion vector at the collocated position, SbTMVP fetches the entire motion vector field of same size as the current CU at the shifted collocated position. In essence, this could also be viewed as a motion compensated prediction of the motion vector field. SbTMVP is signaled as an additional merge candidate.

- Adaptive motion vector prediction (AMVP) is augmented by Adaptive motion vector resolution (AMVR), which allows the motion vector difference resolution to be adapted at the CU level. Next to $1/4$ luma sample resolution, $1/2$, 1, and 4 sample resolution can be signaled.

- While bi-prediction in HEVC is generated by averaging the two prediction signals, a weighted average can be signaled in VVC. This is denoted as Bi-prediction with CU-level weight (BCW). 5 different weights w_{BCW} can be signaled such that the final prediction p_{Block} is generated from the two prediction blocks p_0 and p_1 in a similar way as for TPM:
$$p_{\text{Block}} = \frac{w_{\text{BCW}} \cdot p_0 + (8 - w_{\text{BCW}}) \cdot p_1}{8} \tag{2.38}$$

- The bi-prediction signal of a CU can be further refined using Bi-directional optical flow (BDOF). Optical flow is calculated from the coded reference pictures using spatial and temporal gradients of 4×4 sub-blocks, assuming a smooth motion vector field. A sample refinement is then derived based on the estimated motion vectors and applied to the bi-prediction signal.

- The well-known method of Decoder-side motion vector refinement (DMVR) is now a part of VVC. A signaled motion vector can be further refined at the decoder-side by performing a bilateral matching.

- In another weighted prediction scheme, inter- and intra-prediction signals can also be averaged in the Combined inter and intra-prediction (CIIP) mode, using prediction samples p_{Inter} generated by the regular inter-prediction process of the Merge-mode and p_{Intra} of the regular intra-prediction process. Here, a sample weight $w_{\text{CIIP}} \in \{1, 2, 3\}$ is derived depending on the intra-prediction coding modes of the left and top neighbor.
$$p_{\text{Block}} = \frac{(4 - w_{\text{CCIP}}) \cdot p_{\text{Inter}} + w_{\text{CIIP}} \cdot p_{\text{Intra}} + 2}{4} \tag{2.39}$$

3 Prior Art of Non-rectangular Partitioning

Non-rectangular prediction and transform coding of digital video data has been the subject of much work in academia, industry, and standardization, evident through a large body of publications, patents, and standardization documents. This chapter aims at giving an overview over the historic developments, arriving at the state-of-the-art in the field.

The application of non-rectangular partitioning to video is motivated by the fact that natural objects and their motion can be spatially better approximated by such a segmentation, than by rectangular trees. This is a commonality among the different realizations of geometric partitioning in AVC, HEVC, and VVC, as proposed in this thesis. In all these cases, GEO is used as an additional coding tool among other tools. This also means that object boundaries are not *required* to be coded with this coding tool. The application is merely determined by the encoder through RDO. In many cases though, the better approximation of object boundaries may result in higher coding efficiency.

A different and in terms of partitioning much more flexible approach has been taken earlier in MPEG-4 Part 2: Visual (M4V) with the concept of *video objects*. In M4V, a video object is an arbitrary shaped area of the video scene that may exist for an arbitrary duration of time. This concept and the resulting coding tools are not specifically introduced for higher coding efficiency, but for specific applications, such as the browsing and manipulation (cutting, pasting, compositing) of these video objects.

These two different views of non-rectangular partitioning are reviewed in Sections 3.1 and 3.2. Lastly, an approach is reviewed in Section 3.3 that tries to combine the flexibility of arbitrarily partitioned blocks with the target application of using it as a coding tool.

3.1 Shape-adaptive Coding in MPEG-4

Prior to the development of H.264/AVC, the focus of the standardization activity of MPEG was put on the MPEG-4 Part 2: Visual (M4V) coding standard. M4V was published in its first edition in 1999. Different to H.264/AVC, whose main focus was put on the efficient compression of video frames, M4V provides a very diverse functionality for the coding of rectangular frames, video objects, still images, and hybrids of synthetic and natural video content [Ric03]. For this thesis, the object-based coding profile is of particular interest. The shape of an object in M4V is defined by a bitmap and explicitly coded into the bitstream. This can be either a *binary* valued bitmap or a *grey-level* / *greyscale* bitmap. Binary bitmaps have certain visual drawbacks, since it is often not possible to neatly separate natural objects from their background. Grey-scale shape coding gives a more flexible control of object transparency, for example it is possible that the edges of an object “fade out”, and therefore the foreground samples are transitioning into the background.

Both types of bitmaps nevertheless utilize a binary shape coding called context-based arithmetic encoding (CAE), a method which is known from the JBIG standard for binary image compression [93]. In contrast to contour-based coding methods, like polygon or vertex-based coding, B-spline coding, chain coding, and base line coding, all of which were intensively investigated during the standardization of M4V [Kat+98], CAE directly operates on bitmaps. Contour-based approaches require an additional step to transform the contour into a binary shape, typically through a filling algorithm. CAE has been shown to outperform vertex-based coding when lossless shape coding is required, whereas vertex-based coding is better in terms of coding efficiency for lossy shape coding [Kat+98].

A video object is segmented into binary alpha blocks (BABs) of size 16×16 luma samples, identical to the macroblock (MB) grid. In case of binary shape coding, only those BABs are required to be shape-coded that are overlapping with the object boundaries. Thus, a *bab_type* syntax element indicates for every BAB whether the BAB is transparent (not part of the current video object), opaque (internal to the current video object with motion and texture coded as usual), or a boundary BAB.

In CAE, the binary-valued samples of a boundary BAB are scanned in a given scan-order. Then, for every scanned sample position, a context is calculated. A context-template is used to define a region of neighboring samples that have been previously coded. Similar to texture coding methods, it is distinguished between intra-coded BABs and inter-coded BABs. For intra-coded BABs, the context template is formed from 10 spatial neighbors c_n. Then, the context index is calculated by arranging the previously coded values into a 10 bit word $c_9c_8c_7c_6c_5c_4c_3c_2c_1c_0$, defining a probability value for the current sample being zero. Each of the possible 1024 context probabilities are stored in a table. Inter-coded BABs use a template that consists of nine sample positions, four in the current video object plane (c_0 to c_3) and five in a reference video object plane (c_4 to c_8). The position of the central context sample in the reference video object may be offset from the collocated position by an integer-valued vector. This effectively allows motion compensated shape-coding using a “shape motion vector”.

For motion compensated prediction across video object planes, the transparent samples in transparent MBs and boundary MBs are padded. This is performed in boundary MBs by first extrapolating from opaque samples horizontally and then vertically. Fully transparent MBs with a single neighboring boundary MB are filled by horizontal or vertical extrapolation of the boundary pixels of that MB. Transparent MBs without boundary MBs as neighbors are simply filled with the value $2^{B_\mathrm{d}-1}$,where B_d is the bit-depth of the video, e.g. $B_\mathrm{d} = 8$.

The prediction error of a boundary MB is coded in the core profile using four 8×8 DCTs followed by quantization, run-level coding, and entropy coding as usual. In the advanced coding efficiency profile, the Shape-adaptive DCT (SADCT) provides a more efficient method of coding the boundary texture. The SADCT is introduced in greater detail in Section 7.2.

3.2 Geometry-based Partitioning Proposals for AVC and HEVC

Unlike the shape coding using bitmaps as discussed in M4V, geometry-based partitioning can be considered a parametric block partitioning approach.

The first scientific mentioning of a partitioning method for inter-prediction blocks using a straight line has been made by Kondo and Sasai in 2005 [KS05]. They propose a “sliced block” motion compensation method that claims a 5 % bitrate reduction for P-picture coding over H.26L JM 1. In their method, MBs and sub-MBs are partitioned by a line parametrized using two points. Recognizing the potential complexity and bitrate overhead when the line is allowed to pass through any two points, they impose the limitation that all points shall exist on a fixed, one-pixel interval on the boundary of the macroblock. It is claimed by the authors that the slice line for the current block is encoded using prediction from the shape information of neighboring MBs. No technical and experimental information is however given how this is accomplished and how effective the spatial prediction is performed. The method is evaluated on three sequences, *Mobile & Calendar*, *Stefan*, and *Foreman*.

In a parallel work in 2006, Hung, de Queiroz, and Mukherjee [HDM06] propose using a straight line parametrized by an angle and a distance value for MB and sub-MB partitioning. Angle and distance values are quantized, resulting in two wedge dictionaries for 16×16 and 8×8 blocks, with 2012 and 340 partitioning entries. The wedge partitioning mode is coded with an entropy coding scheme assuming equiprobable symbols, except for those canonical (halves) partitions. The encoder performs an edge detection and a linear regression to determine a seed partitioning that is being further refined by varying the angle and distance parameters. Two different encoder strategies are tested: The first tests all available partitions exhaustively. The second encoder strategy– called *fast wedge partitioning* – tests a sub-set of 77 partitions for both block sizes around the seed partitioning. For the exhaustive strategy, a bitrate reduction of −4.7 % is reported and −3.1 % for the fast wedge partitioning technique in terms of BD-rate change. The method seems to be evaluated only on the *Foreman* sequence without giving details on the used reference, testing condition, or complexity impact.

Divorra, Yin, Dai, and Li expanded on the previous works in 2007 and proposed *geometry-adaptive block partitioning (GEO)* for AVC [Esc+07], hence providing the name and acronym to the method. They also use a parametrization using an angle θ and a distance ρ, with the exact quantization specifications given in Section 4.2.3. Using the equation $f(x, y) = x \cos\theta + y \sin\theta - \rho$, every sample of the geometrically partitioned block is classified into three categories. Samples at position (x, y) with $f(x, y) > 0$ are assigned to the first partition, samples with $f(x, y) < 0$ are assigned to the second partition, and samples fulfilling $f(x, y) = 0$ are classified as boundary samples. For these boundary samples, the final prediction value is computed as a linear combination of each corresponding value, if they were fully assigned to the first and second partition. The partitioning is signaled using index values for the quantized angle and distance parameters. Also, spatial prediction is applied: If a neighboring block is geometrically partitioned, the parameter differences are signaled using Exp-Golomb codes. Otherwise, a fixed-length coding for the angle and a variable-length coding for the distance, therefore assuming an exponential distribution, is used. In terms of prediction, they propose using either inter- or intra-prediction in both partitions. For simplicity, only DC or directional intra-prediction with the same direction as the partitioning line is considered. They evaluate their method on two sequences, *Tiger* and *Foreman*, against JSVM 6. For intra-coding with GEO, −8.69 % and −5.65 % bitrate reductions are reported for the first 10 pictures of Foreman and Tiger, respectively. For inter-coding, −7.6 % and −4.46 % bitrate reduction are reported for the first 60 pictures. They also show a coded example of Foreman at QP36, claiming a sharper look and higher detail around the contours. The authors present a follow-up paper in [Dai+07], improving the geometric intra-prediction method, and also proposed GEO to VCEG for the next video coding standard [DYG07].

In 2009, Ferreira, Hung, de Queiroz, and Mukherjee proposed additional improvements for motion vector prediction and complexity reductions of a GEO video coder [Fer+09]. For motion vector prediction, they adapt the search positions of spatial predictors (compare Figure 2.7a) according to the angle of the current GEO partition mode. 7 different cases are defined. For complexity reduction, they measure the distribution of angles and distances in a training set and only utilize the $N = \{32, 64, 128, 256\}$ most likely combinations, coded with fixed-length binarization. Against JSVM 9.2 and using the VCEG M33 testing recommendation, they measured consistent coding efficiency improvements with a mean BD-rate change of −6.92 % for $N = 256$, decreasing to −5.56 % for $N = 32$. Interestingly, the corresponding implications on relative encoding complexity according to N are not discussed.

Conceptually similar complexity reductions were proposed by Guo et. al. in 2010 [Guo+10]. They also study the distribution of GEO modes and define *most valuable partitions (MVPs)* that are a subset of all available GEO partitions for a given block size. They measure the distributions independently for the angle and distance parameter and conclude that more balanced partitions with equal number of samples in both segments are more probable to be used by the encoder. This is explained by the assumption that block matching likely fails for very small GEO segments. The MVP set is constructed by sampling the distance space ρ with two different quantizers, having step sizes $\Delta\rho_0$ and $\Delta\rho_1$. The application of each quantizer is determined by a threshold T_0. If $\rho \leq T_0$, the first quantizer with smaller step size $\Delta\rho_0$ is used. Otherwise, the coarser quantizer $\Delta\rho_1$ is used. The number of GEO partitions is reduced from the full set of 58, 274, and 1110 partitions to the simplified set of 10, 30, and 40 for 8 × 8, 16 × 16, and 32 × 32 blocks, respectively. The coding efficiency against JSVM 6 for the full set is measured at −7.97 % for CIF and −3.02 % for 720p content, without using 32 × 32 blocks. For the simplified set, coding efficiency drops slightly to −5.37 % for CIF and −2.31 % for 720p content. When 32 × 32 GEO blocks are allowed for 720p content, the situation is reversed: Coding efficiency increases more for the simplified set to −6.16 % and to −5.86 % for the full set of GEO partitions. The encoder runtime for the simplified set is reported to be 400 % that of the H.264 reference encoder JSVM.

GEO and a similar Flexible Macroblock Partition (FMP) scheme were subsequently proposed as new coding tools among the responses to the Call for Proposals for HEVC [Yan+10][Kar+10b]. A description of the video coding technology of [Kar+10b] in a journal publication can be found in [Kar+10a]. Here, GEO is available for 16 × 16, 32 × 32, and 64 × 64 blocks with 256, 512, and 1024 possible geometric partitions, respectively. The blending of samples in the region of the partitioning line is denoted as Overlapped block motion compensation (OBMC) for geometric partitions.

In consequence of the proposals, GEO was selected to be incorporated into the Test Model under Consideration (TMuC) software and investigated in Subtest 2: Flexible Motion Partitioning of Tool Experiment 3: Inter Prediction in HEVC [Kru+10]. The first version of GEO provided −3.12 % bitrate savings. The similar FMP approach provided −3.48 % of bitrate savings [KS10a]. Additional proposals were made, focusing on simplifications, specifically reductions of GEO and FMP split modes [ZY10; Che+10; GYF10]. Over several meeting cycles, GEO and FMP were investigated with further focus put on encoding run time [KS10b; FZC11; Zhe+11a; Zhe+11b; ZCK11]. For GEO, this meant a proposed reduction in the number of splits. First, optimizations were proposed in [Fra+11], showing different encoder complexity vs. coding gain trade-offs. The full complexity GEO performance, estimated over a reduced number of frames, is reported as giving −5 % in terms of BD-rate change for a Random access (RA) coding configuration. 238 modes for 16 × 16 blocks and 494 modes for 32×32 with algorithmic simplifications were reported to provide −2.8 % for an increase in encoding time to 190 % against TMuC-0.9. A reduction down to 127 and 122 modes still provided −1.9 % for 130 % encoding time. A further reduction down to 26 partition modes for 16×16 and 32×32 and disabling for 64×64 was tested, providing −1.8 % BD-rate change over the TMuC, but with similar complexity. An explanation of this method outside of the standardization context can be found in [BFT11].

Simplifications to the FMP scheme resulted in partitions which are now known as Asymmetric motion partitioning (AMP) in HEVC. A unified solution to flexible motion partitioning was proposed, consisting of asymmetric rectangular splits, two non-rectangular splits, given by the diagonal and anti-diagonal line through the block and OBMC [Bor+11]. The method provided −1.4 % coding gain in terms of luma BD-rate change, with 174 % encoding time and 117 % decoding time compared to HM-2.0. Lastly, a non-rectangular partitioning scheme consisting of the two diagonal partitions and four wedge partitions for 8×8 blocks was proposed [Zhe+11c], providing −0.7 % of BD-rate change for 119 % encoding time increase over HM-3.0. By the 7th JCT meeting however, all work on non-rectangular partitions has stopped.

Outside of the standardization activity, the complexity impact of GEO at the encoder side was recognized by others and possible solutions were proposed. An extensive wedge-based and object-based partitioning framework was proposed by Wang, Sun, and Sullivan [Wan+12; Wan+13], using a texture-difference partition line selection at the encoder side to speed up the RDO for finding the optimal partitioning. In this scheme, the luma and chroma mean values are computed for each segment. Then, a texture difference is computed using a component-weighted average of the absolute differences between the segment mean values. The partitioning mode that maximizes this texture difference function is selected as the optimal partitioning. The scheme, implemented in JM-16.2, provides −8.1 % BD-rate change for a wedge-based coding scheme and −5.6 % for an object-based coding scheme, with 125 % and 123 % encoding time increase over the H.264 reference encoder, respectively. Additionally, the method is tested on 3D video sequences that have high-quality depth-maps available. For these sequences, the partitioning of texture blocks is derived from segmenting the associated depth blocks.

Muhit, Pickering, and Frater [MPF09a] proposed a similar texture-based fast search strategy using a Sobel edge detection algorithm, applied to the original, uncompressed block, followed by a Hough transform to find dominant edges or lines. This initializes a second refinement step, in which the angle and distance parameters of the partitioning line are further optimized. They later also published an improved motion compensation method, in which the regular block-based motion compensation of each geometric segment is augmented by an elastic motion model to capture sub-block motion [MPF09b].

The suitability of geometric or wedge-based partitioning for coding of depth maps has been recognized. Various methods have been proposed that use geometric partitioning in combination with intra-prediction to adapt to the unique, flat textured and high-frequency characteristics of depth maps [Kan+10; KH12; Mer+16]. In consequence, 3D-HEVC offers the possibility of intra-coding depth maps using wedges. An index signaled at the PB level indicates a binary wedge pattern from a set of wedgelets [Tec+16].

After the finalization of HEVC, interest in non-rectangular block partitioning declined. A joint inter-intra prediction scheme using geometric partitioning was proposed in [Che+14] by Chen, Mukherjee, Han, and Rose. In this proposal, two different coding modes are added to the VP9 codec:

- An inter-intra multi-hypothesis prediction scheme, where inter- and intra-prediction signals are combined following a 1D weighting function along the intra-prediction angle.
- A geometric partitioning scheme using partitioning code books, segment-based inter- or intra-prediction and soft masking at the partitioning boundary.

Parts of this method were incorporated into the emerging open-source video compression format AV1, finalized in 2018 by the Alliance for Open Media (AOMedia) industry consortium. AV1 specifies a prediction mode named *Advanced Compound Prediction* and further distinguishes several sub-modes [Che+18; RH19]. One of the inter-prediction sub-modes is termed *Compound Wedge Prediction*, which defines a code book of 16 possible wedge partitions. The code book contains wedges that are oriented either horizontally, vertically, or oblique, with slopes of ± 2 or ± 0.5 samples for rectangular and square-shaped blocks. Similar to TPM and the proposed method in this thesis, two predictions are averaged using per-sample weights to generate a combined predictor.

Besides block partitioning models using a line, some research has been conducted into extending this approach to parametric models with higher degrees of freedom. Zhang, Wang, Zheng, and Wu proposed polyline block partitioning in which a block is partitioned using two line segments [Zha+09]. Two points A and B of the polyline are located on the boundary of the current macroblock, just as it is the case for regular geometric block partitioning. A third point C can now be located within the macroblock and the line segments AC and BC constitute the polyline. The method claims an additional $-1.08\,\%$ to $-1.45\,\%$ of BD-rate change for low and high bitrates compared to H.264/AVC using geometric block partitioning and P-picture coding.

Besides block-based approaches for motion compensated prediction, much research has been conducted in the area of mesh-based motion compensation [BK97; Ost97; HLC97; VTP97; HWZ11]. In most of these schemes, a triangular tessellation of the picture is performed, and affine motion models are applied for prediction. In the most recent proposal by Huang, Woods, and Zhao, this approach is combined with a quad-tree: Each leaf of the quad-tree is partitioned into two triangles. The motion of each triangle is represented using motion vectors at control points, located at the triangle vertices. The encoder can signal the number of control points per block, one, two, or four. This allows a flexible motion representation, which can be pure translational motion for the entire block, horizontal bilinear (motion field is uniform within each column), vertical bilinear (motion field is uniform within each row), and affine for both triangles. The method is however not compared against state-of-the-art hybrid video coders and only shown to outperform the Motion Compensated Embedded Zero Block Coding (MC-EZBC) coder.

In summary, prior art in geometric block partitioning demonstrates that higher coding efficiency can be consistently obtained. However, previous methods only consider the case of further partitioning square coding blocks. A state-of-the-art video coding scheme such as VVC offers a variety of rectangular coding blocks, with different sizes and aspect ratios. This necessitates a quantization of partitioning parameters shown in Chapter 4 that reflects these new requirements. Prior art also lacks an in-depth investigation into the suitability of spatial prediction and coding of geometric partitioning parameters, applicable to an ample set of video sequences. No previous prior art on temporal prediction through projection as proposed by the author of this thesis has been found.

The implications of performing blending at the geometric segment boundary have not been studied extensively. Chapter 5 shows that optimal blending is largely resolution and content-dependent. Furthermore, prior art in AVC and HEVC only considers differential motion vector signaling using motion vector differences. This approach can be considered being costly in terms of signaling bits when the coding scheme already offers a flexible rectangular block partitioning and an extensive set of motion vector prediction tools. Therefore, more efficient signaling approaches of the motion information are also detailed in Chapter 5. This is also a key step that enables the development of a low-complexity encoder which does not perform a computationally intensive block-matching for geometric partitions.

3.3 Segmentation-based Partitioning

Block partitioning approaches that do not explicitly signal the partitioning boundary using a parametric model or binary shape coding typically rely on a method to derive the partitioning information automatically. This avoids the costly signaling of arbitrarily shaped boundaries. The main approach, which has been proposed by multiple authors, is based on segmentation algorithms [Che+07; Kim+08; MC11; Ahm+13b; Ahm+13a]. These methods are denoted as segmentation-based partitioning (SBP), implicit block partitioning, or partitioning using motion hints. In these proposals, it is assumed that moving objects can be clearly separated from the background and that the object boundaries coincide with motion vector field discontinuities. Chen et. al. [Che+07] first proposed to segment a local region of a reference picture into a binary valued shape. This binary shape can be used as a partitioning mask for the current block, assuming that the shape is only subject to translational motion and not being deformed. The estimated translational motion coincides with the motion of the object. Hence, it can be predicted from one of the two motion vectors that are associated with the partitioned block. An additional motion vector difference is nevertheless signaled for refinement, in cases where the object boundary cannot be reliably located in the segmented reference picture. Note that due to the prediction of the shape motion from one of the block motion vectors, it is actually irrelevant, whether the binary shape describes the foreground or background. The method achieves a reported −6.79 % BD-rate change for P-picture coding over H.264/AVC.

A more recent approach in the 3D extension of HEVC (3D-HEVC) is made by Jäger with Depth Based Block Partitioning (DBBP) [Jäg13]. DBBP is a coding tool that utilizes the depth map as side-information in order to perform a segment-wise prediction of the texture. Explicit coding of the shape of the partition boundary is avoided by thresholding of the collocated depth map into a binary segmentation mask. Average bitrate reductions of −0.5 % are reported for the texture.

Inspired by DBBP, the author of this thesis also investigated SBP as an additional coding tool in a more recent, post-HEVC hybrid video coder (HM-14.0-KTA-1.0) [BHW16]. While DBBP and the proposals using the macroblock of AVC used simple thresholding to derive the partitioning, the larger block sizes of HEVC required the usage of more complex segmentation, pre- and post-processing methods. It is proposed to median filter the reference picture, segment local areas of the reference pictures using k-means clustering of the $C_B C_R$ compo-

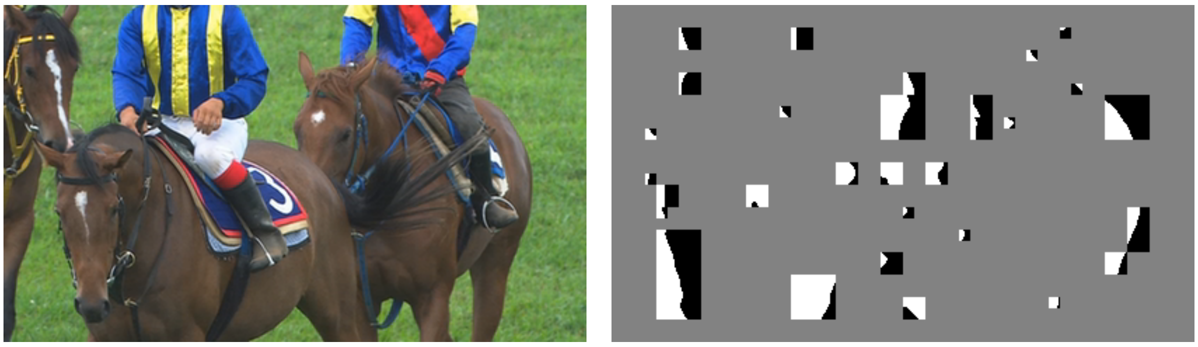

(a) Original (b) SBP blocks

Figure 3.1 Original picture and partitioning masks for SBP blocks chosen by the encoder. *RaceHorses* sequence, POC 193, QP27.

nents and apply morphological opening and closing filters. Through experimentation, it was determined that five iterations of k-means clustering suffice to achieve stable cluster centers. Figure 3.1 shows an example of the coded texture and the corresponding partitioning masks for blocks using SBP. It can be seen in the given example blocks that a color-based segmentation succeeds if the foreground objects can be clearly separated. For video sequences with highly textured background, as in the *Kimono* sequence, the segmentation based on color alone fails and therefore no meaningful partitioning can be obtained. The method is reported to provide an overall luma BD-rate change of −0.39 % for Low-delay B (LDB) and −0.62 % for Low-delay P (LDP), using 12 selected sequences of the JCTVC and JCTVC-3V test set. It is further investigated how a hypothetically optimal segmentation could improve the coding efficiency. This is simulated by using depth maps for generation of the block partitioning. Coding efficiency with an optimal segmentation increases, measured by a luma BD-rate change of −2.21 % and −2.14 % for LDB and LDP, respectively.

4 Prediction and Parameter Coding for Non-rectangular Block Partitions

This chapter discusses the main concepts for performing non-rectangular partitioning, prediction, and coding in state-of-the-art video coding. First, general aspects of non-rectangular block partitioning are introduced in Section 4.1 with the main focus on geometry-based partitioning (GEO). Then, different parametrization variants of non-rectangular partitions are discussed in Section 4.2. In this context, a baseline algorithm in terms of parametrization and quantization of the geometric partitioning side-information is established that can be used for comparison of further developments. In the following Section 4.3, entropy coding and prediction methods for the additional side-information are presented and local optima of parameter settings are empirically determined. In particular, Section 4.2.2 investigates which explicit signaling strategy is suitable and providing the highest coding gain for a given parametrization and quantization. Section 4.3.2.1 investigates the possibility of spatially predicting the geometric partitioning, while Section 4.3.2.2 applies the same methodology for the concept of predicting the geometric partitioning over time.

Multiple aspects essential for a working geometry-based partitioning coding mode are assumed to be disposable in this chapter, such as the blending process, motion vector prediction and the overall mode control. These are discussed in great detail in the subsequent Chapter 5. Based on the findings in this chapter and Chapter 5, Chapter 6 presents simplifications for the geometric partitioning coding mode which were developed in the context of the VVC standardization activity.

Although the order of the specific chapters and sections presented may suggest a successive approach in the development of the final geometric block partitioning coding tool, the author recognizes that the typical development process of such a coding tool is often cyclical and iterative in nature. Different components of a coder are highly intertwined and – due to the presence of the prediction feedback loop – recursive in nature. Therefore, the optimization of one specific aspect of the presented method, for example the partitioning granularity, may require the re-tuning of other aspects that are dependent, such as the blending strength for overlapped-wedge motion compensation or the signaling and vice versa.

Figure 4.1 shows a high-level block diagram of the necessary steps for non-rectangular,

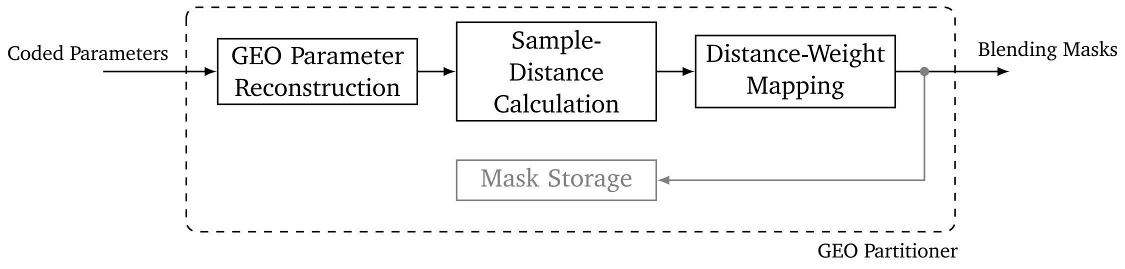


Figure 4.1 Main components of the geometric block partitioner.

specifically geometry-based block partitioning as presented in this thesis. From a decoder perspective, input to the process is the coded representation of the partitioning information relating to a given coding block. Successively, this partitioning information is decompressed through sub-steps such as parameter prediction, look-up tables and inverse quantization. Based on the underlying geometric model and the decompressed parameters, per-sample properties can then be calculated that define to which partition a sample can be assigned. In the proposed method, a sample distance to an idealized partitioning line, dividing the two partitions, is calculated. In a third step, the distance criterion is used to derive per-sample weighting masks which are used in the core prediction and coding steps of the encoder and decoder.

4.1 General Aspects of Prediction using Non-rectangular Block Partitions

Many building blocks of a modern video coder directly operate on rectangular blocks of samples. Motion compensation as introduced in Section 2.2.5 is typically performed by fetching a contiguous, two-dimensional array of samples from a reference picture. If bi-prediction is employed, two blocks of samples are fetched and the samples are averaged. This can be expressed as a weighting process:

$$\boldsymbol{P}_{\mathrm{PB}} = w_0 \cdot \boldsymbol{P}_{\mathrm{L0}} + w_1 \cdot \boldsymbol{P}_{\mathrm{L1}}, \tag{4.1}$$

where $\boldsymbol{P}_{\mathrm{PB}}$ is the resulting prediction block of the current coding block, $\boldsymbol{P}_{\mathrm{L0}}$ and $\boldsymbol{P}_{\mathrm{L1}}$ are intermediate, motion compensated prediction blocks and w_0 and w_1 are weights which fulfill the condition $w_0 + w_1 = 1$. As w_0 and w_1 have identical values for all samples in $\boldsymbol{P}_{\mathrm{L0}}$ and $\boldsymbol{P}_{\mathrm{L1}}$, respectively, this operation can be performed efficiently in hardware, for example with Single instruction, multiple data (SIMD) vectorization technology such as Intel's Streaming SIMD Extensions (SSE) on general purpose CPUs.

In HEVC, a coding block may be composed of two prediction sub-blocks $\boldsymbol{P}_{\mathrm{PB},0}$ and $\boldsymbol{P}_{\mathrm{PB},1}$ for the case of AMP:

$$\boldsymbol{P}_{\mathrm{PB}} = \boldsymbol{P}_{\mathrm{PB},0} + \boldsymbol{P}_{\mathrm{PB},1} \tag{4.2}$$

$$\boldsymbol{P}_{\mathrm{PB},0} = \begin{bmatrix} \boldsymbol{P}_{\mathrm{PB},0}^{w_0 \times h_0} & \boldsymbol{0}^{(w-w_0)\times h_0} \end{bmatrix} \tag{4.3}$$

$$\boldsymbol{P}_{\mathrm{PB},1} = \begin{bmatrix} \boldsymbol{0}^{(w-w_1)\times h_1} & \boldsymbol{P}_{\mathrm{PB},1}^{w_1 \times h_1} \end{bmatrix} \tag{4.4}$$

For simplicity, $\boldsymbol{P}_{\mathrm{PB},0}$ and $\boldsymbol{P}_{\mathrm{PB},1}$ in Equation (4.2) have identical dimensions $w \times h$ and could be the result of uni- or bi-directional motion compensation. Thus, they can also be seen as the zero-extended versions of the actual prediction sub-blocks $\boldsymbol{P}_{\mathrm{PB},0}^{w_0 \times h_0}$ and $\boldsymbol{P}_{\mathrm{PB},1}^{w_1 \times h_1}$ which are of size $w_0 \times h_0$ and $w_1 \times h_1$, respectively. This extension to identical dimensions is exemplified by Equations (4.3) and (4.4) for a vertical sub-block partitioning, e.g. $h_0 = h_1 = h$.

In the case of non-rectangular block partitioning, the block-based processing of Equation (4.2) is modified to introduce a sample-wise assignment to the first or second partition:

$$\boldsymbol{P}_{\mathrm{PB}} = \boldsymbol{M}_0 \circ \boldsymbol{P}_{\mathrm{PB},0} + \boldsymbol{M}_1 \circ \boldsymbol{P}_{\mathrm{PB},1} \tag{4.5}$$

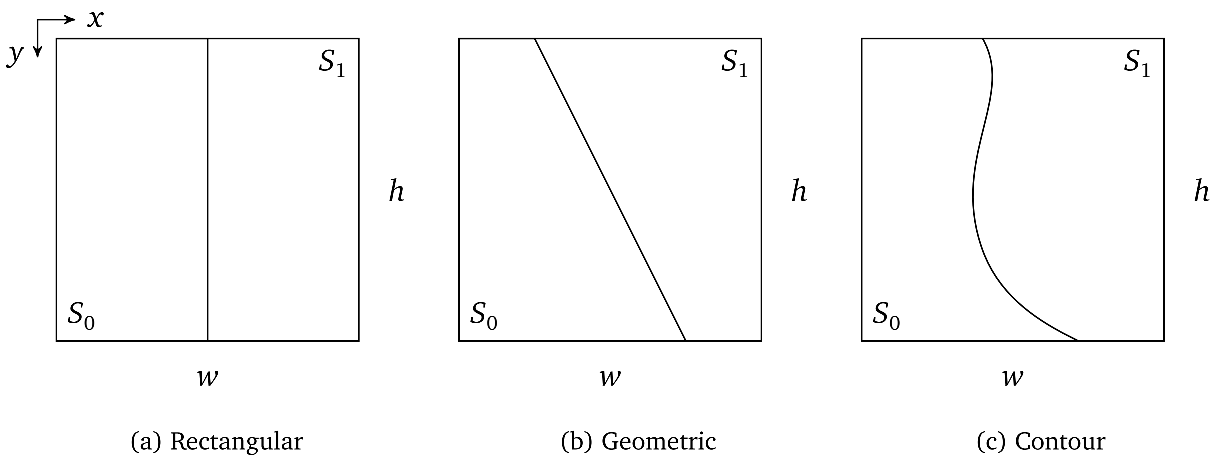


Figure 4.2 Parametric block partitioning models.

Similar to weighted prediction, $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$ now define per-sample weights. In the context of this thesis, the weighting matrices shall fulfill the property $\boldsymbol{M}_0 + \boldsymbol{M}_1 = c\boldsymbol{J}$, where $\boldsymbol{J}$ is a matrix of ones and $c \in \{2^k\}$, $k \in \mathbb{N}_0$. If all weights of the weighting matrices fulfill $\boldsymbol{M}_\text{n} \in \{0, 1\}$, $n = 0, 1$, they can also be thought of as binary masks or *hard* masks. Such a hard masked block of samples is now denoted as a segment or a partition $\boldsymbol{S}$, e.g. $\boldsymbol{S}_n = \boldsymbol{M}_n \circ \boldsymbol{P}_{\text{PB},n}$, $n = 0,\ 1$. In principle, the masks $\boldsymbol{M}_n$ could have arbitrary binary shapes. Examples for such shapes are shown in Figure 4.2. Rectangular shapes as visualized in Figure 4.2a are used in all modern video codecs. As mentioned above, the AMP modes of HEVC belong to this type of partitioning, where a square coding block is partitioned into two prediction sub-blocks. The rectangular leaf splits that are the result of a Quadtree binary tree (QTBT) partitioning as part of JEM or the MTT partitioning of VVC can also be considered to belong in this category.

If the partitioning is determined by a geometric model, e.g. by a straight line as shown in Figure 4.2b, this is termed Geometric Block Partitioning (GEO) , Geometric Motion Partitioning (GMP) or Geometric Inter-Prediction (GIP) if strictly only inter-prediction is considered. Symmetric and asymmetric rectangular partitions can therefore be considered a subset of geometric partitioning. Also, triangular block partitioning, where a rectangular block is partitioned into two triangles, can be considered a special subset of GEO.

If the partitioning is determined by a contour that is not explicitly parametrized using a simple model (e.g. using splines), this is termed object-based partitioning. One particular method reviewed in Chapter 3 employs segmentation to obtain the contour information and is therefore termed Segmentation-based block partitioning (SBP).

It is apparent that using a hard mask with $\boldsymbol{M}_\text{n} \in \{0, 1\}$ may lead to visible edge artifacts in the prediction signal along the partitioning boundary, which is detrimental for natural video content. These edge artifacts can be mitigated by a de-blocking filter applied to the samples in the region of the partitioning boundary, similar to the de-blocking process applied to the samples along the coding block edges. Alternatively, as proposed in the thesis at hand, a *blending* process is used, where samples of $\boldsymbol{P}_{\text{PB0}}$ and $\boldsymbol{P}_{\text{PB1}}$ in the region of the partitioning boundary are weighted based on their distance to the partition boundary. Instead of a hard transition from one partition $\boldsymbol{S}_0$ to the other partition $\boldsymbol{S}_1$ and vice versa, this results in a

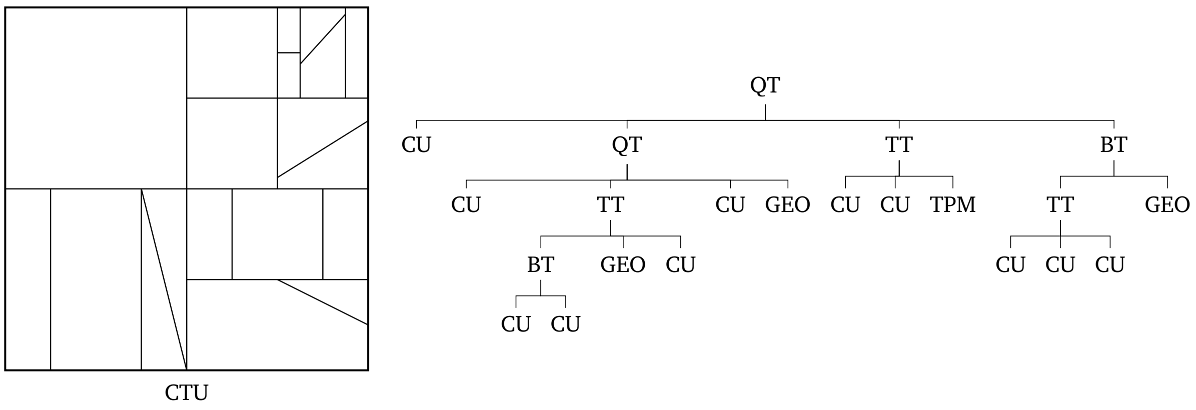


Figure 4.3 Visualization of a partitioned CTU using the MTT, TPM and GEO. Note that each leaf CU could be coded using intra- or inter-prediction coding tools, whereas TPM and GEO leaf CUs are inter-predicted.

smooth transition of the two contents. This can be easily achieved by specifying non-binary weights for $\boldsymbol{M}_\mathrm{n}$. The blending process bears similarities to OBMC and is detailed further in Section 5.1.1 since it is an essential part of the algorithm.

The following sections deal with fundamental aspects of non-rectangular, specifically geometric block partition-based prediction and try to answer questions that arise when designing such an inter-prediction coding tool:

- What is the optimal parametric representation of a geometry-based partitioning scheme? How does the parametric representation affect the coding efficiency and processing?
- How many different geometric partition shapes are optimal in terms of coding efficiency and also providing a reasonable algorithmic complexity trade-off?
- How are the geometric splits distributed among different block sizes?
- How can geometric partitioning parameters be predicted and coded?

4.2 Parametrization of Non-rectangular Block Partitions

The parametrization of a non-rectangular partitioning scheme is concerned with the mathematical representation and modeling of the partitioning itself. As summarized in Chapter 2, for the case of rectangular partitions in modern video codecs, tree-based structures are employed, such as quad-trees, binary-trees, ternary-trees, or combinations thereof. Geometry-based partitions are typically used in combination with rectangular partitions on the last hierarchy level, e.g. on the leaf nodes of existing tree structures. Although in principle it would be possible to further subdivide a geometric segment recursively into geometric sub-segments – a well-known tessellation technique in computer graphics – such a scheme is not considered in the context of video coding, due to the obvious complexity involved in finding an optimal segmentation at the encoder.

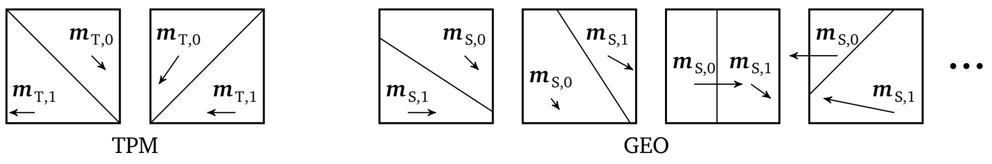

$\boldsymbol{m}_{\mathrm{T},0}$
$\boldsymbol{m}_{\mathrm{T},1}$
$\boldsymbol{m}_{\mathrm{S},0}$
$\boldsymbol{m}_{\mathrm{S},1}$
TPM
GEO
...

Figure 4.4 Visualization of the general concept of triangular partitioning (TPM) and geometric partitioning (GEO) for inter-prediction. A coding block is segmented into two partitions. Each resulting triangle or segment is the predicted by motion compensation using the triangle motion vectors $\boldsymbol{m}_{\mathrm{T},0}$ and $\boldsymbol{m}_{\mathrm{T},1}$ or the segment motion vectors $\boldsymbol{m}_{\mathrm{S},0}$ and $\boldsymbol{m}_{\mathrm{S},1}$. In TPM, the motion vectors are predicted using the *Merge*-mode of VVC.

An example of a non-rectangular leaf coding mode is Triangular prediction mode (TPM) of VVC as reviewed in Section 2.4. TPM is applied on the last hierarchical level of the partitioning tree. The partitioning in VVC is based on the MTT, which is the combination of a Quadtree (QT) with nested Binary trees (BTs) and Ternary trees (TTs). Each node of the initial QT can be further partitioned using a nested QT, BT or TT. Resulting BT and TT nodes can also be further partitioned using BTs and TTs. For BT and TT splits, the direction of the split (horizontal or vertical) is further indicated. The leaf nodes at the end of the partitioning tree, also termed CUs, are coded with the specified inter- or intra-prediction coding block tools. Non-rectangular partitioning tools such as TPM and GEO are applied at this level and partition the leaf-CU further into two segments, terminating the tree-structure at this point. An example of a fully partitioned CTU using TPM and GEO is shown in Figure 4.3. The direct consequence of combining GEO with the MTT of VVC is the availability of variable block sizes: Whereas GEO for AVC was only considering 16×16 blocks and GEO for HEVC square blocks ranging from 8×8 to 64×64 luma samples, GEO for VVC can be applied to 25 different square and non-square block sizes that are the result of the MTT. This requires a careful design of the number and spatial distribution of geometric splits for each block size. It is clear that too many, similar geometrical splits may be an over-representation and therefore do not provide any significant coding gain due to the expensive signaling involved. Too few geometric splits on the other hand likely do not provide significant benefit over the available TPM and might still require an extensive MTT signaling. The third consideration that must be reflected in this trade-off is algorithmic complexity: For a practical encoder, fewer geometric splits to be tested are preferable, requiring less encoder optimization. One of the main contributions of this thesis is providing a practical solution to this problem and offering a sensible trade-off between achievable coding gain and algorithmic complexity for geometric block partitions.

TPM provides the comparable algorithmic framework of the experiments in the following sections, since the overall processing steps are very similar for GEO and the same motion vector coding approach using merge vectors is used initially. In TPM, two triangular partitions are defined, which are the result of splitting the coding block along the diagonal or anti-diagonal line. This is visualized in Figure 4.4. In geometric partitioning, this concept is extended to allow significantly more partitioning shapes, such as those exemplified in Figure 4.4.

In the subsequent section, the following coding experiments are conducted:

- Experiment 4.1 investigates the basic performance of an angle-distance-based quantization scheme that was already used or Advanced Video Coding (AVC), extended to the MTT of VVC.

- Experiment 4.2 improves upon this model by addition of an angle-dependent quantization of the distance. A rate-distortion optimal distribution of geometrical splits is experimentally determined.
- Experiments 4.3 through 4.5 investigate the coding performance of an alternative representation for geometric partitioning parameters using block intercept coordinates.

These experiments will provide a baseline for the subsequent optimizations and coding approaches. Furthermore, they introduce the two different representations considered in this thesis.

4.2.1 Geometry-based Partitioning Models

Per definition, geometry-based partitioning relates to a partitioning scheme where a rectangular-shaped block is partitioned into two disjunct sub-partitions by a straight line. Unlike for TPM, this line can conceptually split the block in every possible way and is therefore not limited to the diagonal or anti-diagonal line. For simplicity and future references, this straight line shall be defined as a simple linear curve (e.g. a curve with zero curvature) in Euclidean geometry. As it is well known, a straight line can have many different parametrizations. For the case of a Cartesian coordinate system with (x, y)-axis definitions, a straight line may, for example, be defined by

- two points $P_0 = [x_0, y_0]^\mathrm{T}$ and $P_1 = [x_1, y_1]^\mathrm{T}$ on the x,y-plane,
- the slope-intercept form $y = mx + b$,
- a vector equation $\boldsymbol{r} = P_0 + \lambda \boldsymbol{v}$ with P_0 being a position and $\boldsymbol{v}$ the direction of the line,
- the Hesse normal form $\boldsymbol{r} \cdot \boldsymbol{n}_0 = d$ with $\boldsymbol{n}_0$ being the normal vector and d the distance to the coordinate origin,
- a line given in polar coordinates.

For the last case of a polar coordinate system (θ, r) on the Euclidean plane, the straight line can be parametrized by an angle φ and a distance ρ which fulfills:

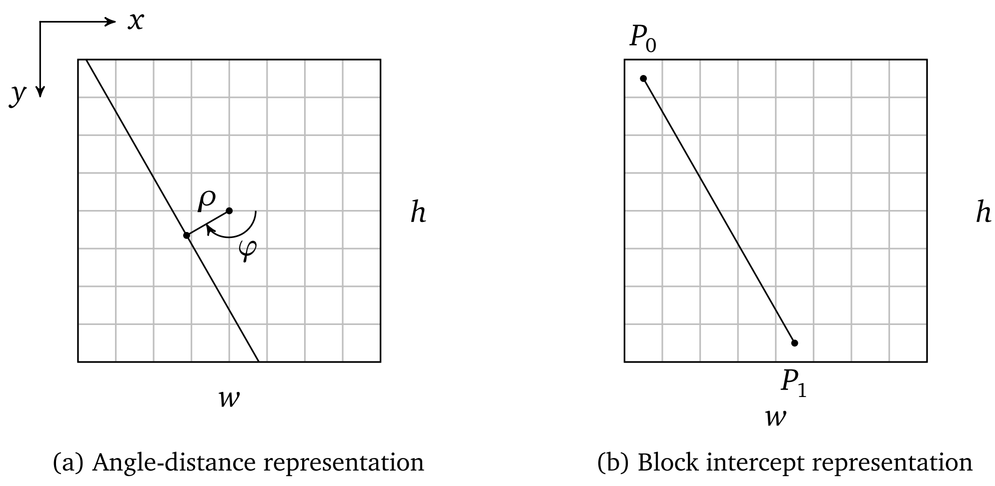
$$r = \frac{\rho}{\cos(\theta - \varphi)} \tag{4.6}$$

By applying the polar and Cartesian coordinate transform relations

$$\begin{aligned} y &= r \sin\theta \\ x &= r \cos\theta \end{aligned} \tag{4.7}$$

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \theta &= \arctan(\tfrac{y}{x}) \end{aligned} \tag{4.8}$$

the different line representations in Cartesian and polar coordinates can be easily converted into one another. Mathematically, the choice of the line parametrization is irrelevant,



(a) Angle-distance representation (b) Block intercept representation

Figure 4.5 Parametrizations for geometric block partitioning

however for application in video coding, which requires a software- and / or hardware implementation of the underlying algorithms, the following considerations have to be made with regard to different representations:

- Quantization effects
- Hardware-friendliness, e.g. number and type of operations required to perform various functions
- Implementability in integer, fixed-point or floating-point arithmetic
- Compactness of representation
- Simplicity

The two parametrizations based on so called Block boundary-intercept (BI) points P_0, P_1 and the Angle-distance (AD) representation φ, ρ are analyzed in great detail in this thesis and coding experiments will be conducted for both representations. The partitioning parameters for these two variants can be represented by a parameter matrix $\boldsymbol{A}$ as follows:

$$\boldsymbol{A}_{\mathrm{BI}} = \begin{bmatrix} P_0 & P_1 \end{bmatrix} \tag{4.9}$$

$$\boldsymbol{A}_{\mathrm{AD}} = \begin{bmatrix} \varphi & \rho \end{bmatrix} \tag{4.10}$$

The two representations also differ in regard to the quantization applied to the parameters in $\mathbf{A}_{\mathrm{BI}}$and $\mathbf{A}_{\mathrm{AD}}$. Quantization of the parameters is an important aspect as it controls the number and shapes of possible geometric splits. The number of geometric partitions on the other hand has direct influence on other aspects, such as the achievable coding gain and the encoder and decoder complexity in terms of run time. These topics and its implications are discussed in greater detail in the following sub-sections 4.2.2 through 4.2.4.

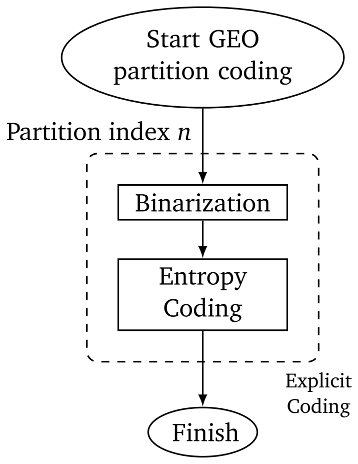


Figure 4.6 Explicit coding of the partitioning information using an index value n. The index value is binarized using a fixed length code $C_{\mathrm{FL}}(\cdot)$. The bins are subsequently bypass-coded using the binary arithmetic coding engine of VVC.

4.2.2 Explicit Coding of Geometric Partitioning Parameters

For the first coding experiments conducted in sub-sections 4.2.3, 4.2.4 and 5.1.1, a simple, explicit signaling of the geometric partition parameters is chosen. For each block size $w \times h = 2^i \times 2^j,\ i, j \in \{3...7\}$ supported by GEO, a code book of geometric partitioning parameters is available at the encoder and decoder side. Each code book $\mathscr{C}_{w,h}$ contains an ordered list of geometric partitioning parameters $\boldsymbol{A}$.

$$\mathscr{C}_{w,h} = \{\boldsymbol{A}_0, \boldsymbol{A}_1, ..., \boldsymbol{A}_k, ..., \boldsymbol{A}_{N_{w,h}-1}\} \tag{4.11}$$

For reconstruction of the partitioning, it is therefore sufficient to signal an index value k to the decoder. Since a binary arithmetic coder with switchable context modeling is used in VVC, a binarization step is required to translate the decimal index k into a binary representation as discussed in Section 2.2.8.1. For the most basic GEO implementation, an equiprobable distribution of geometric partitioning parameters is assumed first. Therefore, a fixed length binarization $C_{\mathrm{FL,l}}(k)$ is chosen for the index value k, using l bit as it is the simplest way of signaling and can be considered a baseline for further experiments. The length l of the code is determined by the maximum number of partitions in the code book:

$$l = \lceil \log_2 N_{w,h} \rceil \tag{4.12}$$

All bins of the resulting bin-strings shall be coded using the bypass mode of the CABAC engine. Further, no encoder simplifications will be considered and a full RDO is performed.

4.2.3 Angle-distance Representation

The angle-distance representation has been used in previous implementations of GEO as reviewed in Chapter 3. It has been proposed first by Divorra et. al. in the context of AVC [Esc+07]. The partitioning line in this scheme is described by a pair of angle and distance parameters $\boldsymbol{A}_{\mathrm{AD}} = [\ \varphi \quad \rho\ ]$, which specify a point in polar coordinates relative to the center of the current block given by its width w and height h in luma samples. Thus for simplicity, a local coordinate system is constructed with its origin O located at the center of the current block, as shown in Figure 4.5a. The partitioning line is now defined by the line perpendicular to the vector $\boldsymbol{p} = \boldsymbol{A}_{\mathrm{AD}} - O$ and passing through $\boldsymbol{A}_{\mathrm{AD}}$.

In order to obtain a finite set of geometric block partitions $\mathscr{C}_{w,h}^{\mathrm{AD}}$, quantization is applied to the parameters φ and ρ. The quantization is performed in a straightforward manner, using a linear quantizer. The authors in [Esc+07] proposed to use a quantization according to the following scheme:

$$\varphi = m \cdot \Delta\varphi, \quad m \in \{0, 1, ..., N_\varphi(\rho)\} \tag{4.13}$$

$$\rho = n \cdot \Delta\rho, \quad n \in \{0, 1, ..., N_\rho\} \tag{4.14}$$

$$N_\varphi(\rho) = \begin{cases} \left\lfloor \frac{180^\circ}{\Delta\varphi} \right\rfloor, & \text{if} \quad \rho = 0 \\ \left\lfloor \frac{360^\circ}{\Delta\varphi} \right\rfloor, & \text{otherwise} \end{cases} \tag{4.15}$$

where $\Delta\varphi$ and $\Delta\rho$ are the angle and distance quantization step sizes, respectively. The maximum angle integer quantization value N_φ is dependent on the distance ρ. For all partitioning lines which are passing through the origin (e.g. $\rho = 0$), only those angles that are contained in the first half of the unit circle need to be considered. For the original proposal of GEO in the context of AVC, only square shaped blocks of size 16×16 luma samples – the so called MB – or of size 8×8 were considered. Therefore, the authors proposed a maximum, quantized integer distance values of $N_\rho = \sqrt{2}\mathrm{MB}_{\mathrm{Size}}/\ 2$, which is equal to half the length of the diagonal slicing the macroblock. It is to be noted at this point that such a simplistic definition of N_ρ, independent of the angle, may result in partitioning lines which lie entirely out of the current block, which can be easily seen for $\varphi = 0^\circ$ or $\varphi = 90^\circ$ and $\rho = \sqrt{2}\mathrm{MB}_{\mathrm{Size}}/2$ for example. It can be assumed that the authors included an additional, not further specified or published method of checking whether a geometric partition lies entirely within the current block. An example of such a quantized (φ,ρ)-space using $\Delta\varphi = 22.5^\circ$ and $\Delta\rho = 1$ is shown in Figure 4.7. For an 8×8 block Divorra et. al. proposed quantization steps sizes of $\Delta\varphi = 11.25^\circ$ and $\Delta\rho = 1$. This would lead to a maximum of 140 partitions for an 8×8 block, assuming that all quantized (φ, ρ)-points must lie entirely within the block.

As non-square block sizes resulting from the MTT of VVC are also considered in the proposed codec described in this thesis, N_ρ shall be more precisely defined by deriving a maximum distance $\rho_{\max}$ depending on φ, w and h in the following way:

$$\rho_{\max}(\varphi) = \cos(\varphi) \cdot \left[\frac{w-1}{2} + \frac{h-1}{2\tan(90^\circ - \varphi)} \right], \quad 0 \leq \varphi \leq 90^\circ \tag{4.16}$$

The reasoning behind this derivation is visualized in Figure 4.8: The maximum distance $\rho_{\max}$ is equal to the distance of the origin to the line which slices the upper-right luma sample. This ensures that at least a single luma sample is contained in one of the two geometric partitions. By utilizing the block symmetry and re-mapping the angle φ of the three remaining quadrants to the first quadrant, $\rho_{\max}$ can also be computed for the angles in the range of

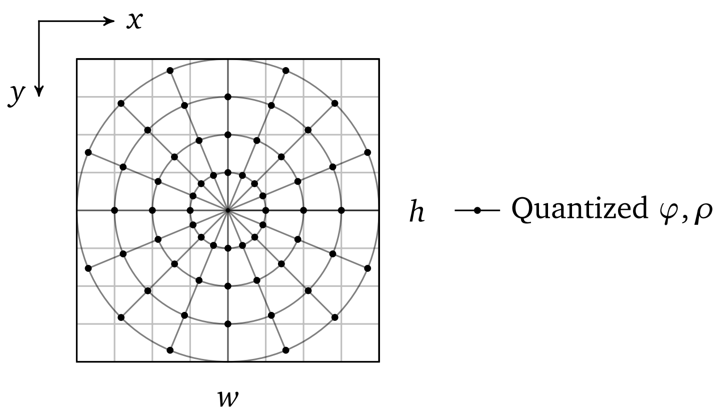


Figure 4.7 Example of a quantized (φ, ρ)-space for given step sizes of $\Delta\varphi = 22.5°$ and $\Delta\rho = 1$.

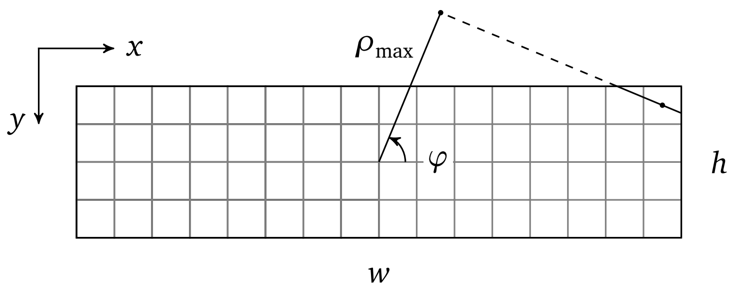


Figure 4.8 The maximum distance ρ_{max} for which a valid partitioning is generated.

$90° < \varphi < 360°$. Now, $N_\rho(\varphi)$ can be expressed depending on ρ_{max} as follows:

$$N_\rho(\varphi) = \left\lfloor \frac{\rho_{\text{max}}}{\Delta\rho} \right\rfloor \tag{4.17}$$

According to Equation (4.17), the distance quantization step size $\Delta\rho$ is independent of the angle φ as well as the block width w and height h. It can be easily seen that a fixed quantization of the distance step size, e.g. $\Delta\rho = 1$ would lead to a prohibitively large number of partitions for the largest supported block size in VVC, which is 128×128 luma samples. Therefore, in the following, two different distance quantization approaches shall be analyzed in more detail. The first method assumes a uniform distance step size $\Delta\rho$ for the current block, which is adjusted according to the block width and height. A second quantization scheme varies the distance step size $\Delta\rho$ according to the angle φ and the width and height, thereby resulting in an angle-dependent step size $\Delta\rho(\varphi)$. Both quantization schemes assume a uniform angle quantization step size $\Delta\varphi$.

As mentioned before, for the following experiments that are conducted, no specialized coding of the GEO partitioning is employed. Unless otherwise stated, the GEO partition is encoded by using an index value k. The index value is binarized using a fixed length code of size $l = \left\lceil \log_2 N^{\text{AD}}_{w,h} \right\rceil$, where $N^{\text{AD}}_{w,h}$ is the maximum number of available geometric partitions for a given block size $w \times h$. This naive coding of the partitioning ensures that the encoder chooses a GEO partition purely based on minimizing the achievable distortion of the current block. More advanced methods of coding the partitioning itself will be investigated in Section 4.3. Furthermore, the *blending* process, detailed in Sections 5.1.1 and Sections

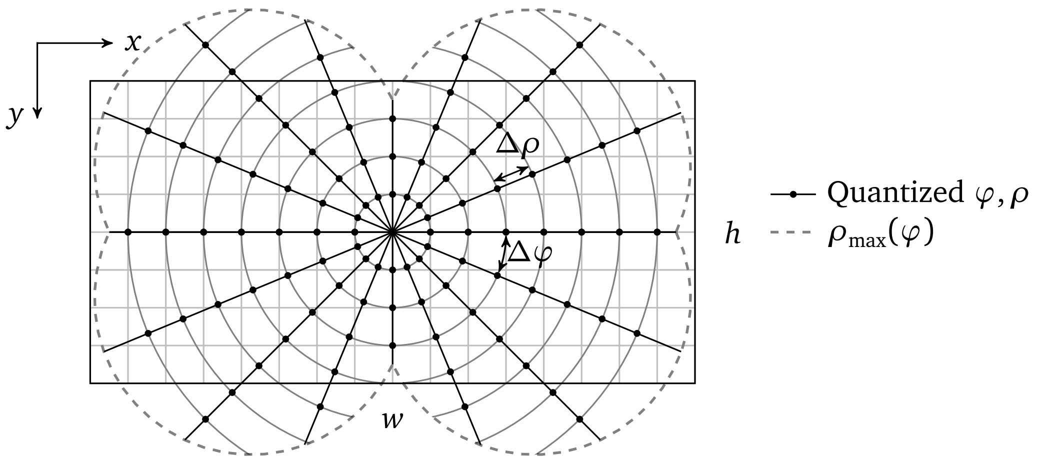


Figure 4.9 Example of a quantized (φ, ρ)-space for given quantization step sizes of $\Delta\varphi = 22.5°$ and $\Delta\rho = 1$. The maximum distance $\rho_{\max}$ was computed according to Equation (4.17) and is indicated by the dashed line. Note that the quantization points, which would lie directly on the $\rho_{\max}$ curve are not included in the set of partitions as they would provide very unequally distributed GEO partitions.

6.3, is using a fixed blending filter length of $d_{\mathrm{m}} = 3$. For motion vector coding, Merge-mode based techniques from TPM are adapted.

4.2.3.1 Block-size dependent quantization

In the previous section, Equation (4.16) was specified to provide the maximum distance $\rho_{\max}$ that still results in a valid partition. In a straightforward approach, the given maximum distance $\rho_{\max}$ shall now be quantized with a fixed step size $\Delta\rho$. The resulting (φ, ρ)-space therefore consists of (φ, ρ) points that are distributed along concentric circles at angles being multiples of $\Delta\varphi$. Each point represents a geometric parameter combination $\boldsymbol{A}$. In order to avoid partitions that are very unequally distributed, meaning that one GEO partition is significantly smaller than the other partition, quantization points close to the maximum distance $\rho_{\max}$ are avoided. This can be achieved by setting a threshold according to:

$$n \cdot \Delta\rho \leq \rho_{\max} - \rho_{\mathrm{th}} \tag{4.18}$$

A smaller value of ρ_{th} would allow the quantization points to be located closer to the maximum distance $\rho_{\max}$, whereas a larger value of ρ_{th} will cause the block to be sliced by a line closer to the origin of the current block. A reasonable, predetermined threshold can be set at $\rho_{\mathrm{th}} = 1.5$ luma samples, preventing the generation of very small partitions consisting only of a single pixel but still allowing a large number of partitions.

Figure 4.9 shows an example of a quantized (φ, ρ)-space for a 16×8 block and quantization step sizes of $\Delta\varphi = 22.5°$ and $\Delta\rho = 1$. For the shown example, this results in a total of 96 potential partitions. Using Equations (4.15) and (4.17), the maximum number of geometric partitions $N_{w,h}^{\mathrm{AD}}$ in the given quantized (φ, ρ)-space can now be estimated based on N_{φ} and $N_{\rho}(\varphi)$, which in turn depend on the quantization step sizes $\Delta\varphi$ and $\Delta\rho$ and the luma block sizes w and h.

Block size	Number of Partitions $N_{w,h}^{\mathrm{AD}}$	
$\{w,h\} \times \{h,w\}$	$\Delta\rho = 1$	$\Delta\rho = \max(w, h)/8$
8×8	96	96
8×16	172	88
8×32	336	78
8×64	660	78
8×128	1312	74
16×16	256	128
16×32	416	96
16×64	744	86
16×128	1396	82
32×32	572	140
32×64	900	112
32×128	1556	86
64×64	1232	148
64×128	1872	116
128×128	2532	156

Table 4.1 Number of geometric partitions using the distance-angle representation, depending on the block size for two different quantization schemes. The angle quantization step size is fixed to $\Delta\varphi = 11.25°$.

$$N_{w,h}^{\mathrm{AD}} = \sum_{m=0}^{N_\varphi - 1} (N_\rho(m \cdot \Delta\varphi) - 1) + N_\varphi / 2 \tag{4.19}$$

Table 4.1 lists the maximum number of partitions for $\Delta\varphi = 11.25°$ (e.g. $N_\varphi = 32$ angles) and two different quantization schemes for $\Delta\rho$. The second column shows the number of partitions when using the fixed quantization scheme $\Delta\rho = 1$, regardless of the block size. As mentioned before, it can be argued that $\Delta\rho = 1$ would result in a very large number of partitions, especially for block sizes larger than 16×16 luma samples, such that performing a full search of all partitions at the encoder side is a task of high complexity.

The third column shows the number of partitions using an alternative quantization where the distance step size $\Delta\rho$ is adjusted according to the width and height of the current block as $\Delta\rho = \max(w, h)/8$. This quantization results in a fairly even distributed number of partitions across all block sizes and is used for the subsequent first Experiment 4.1. On average, a block can have approximately 104 different geometric partitions using this scheme. For this first experiment, the coding of the respective index k of the chosen partitioning is performed using a binary code of length $l = \left\lceil \log_2 N_{w,h}^{\mathrm{AD}} \right\rceil$. Based on numbers provided in Table 4.1, this results in fixed length codes of 7 or 8 bit, depending on the block size. Detailed coding results with sequence specific BD-rate changes for a RA configuration are shown in Table A.1. The

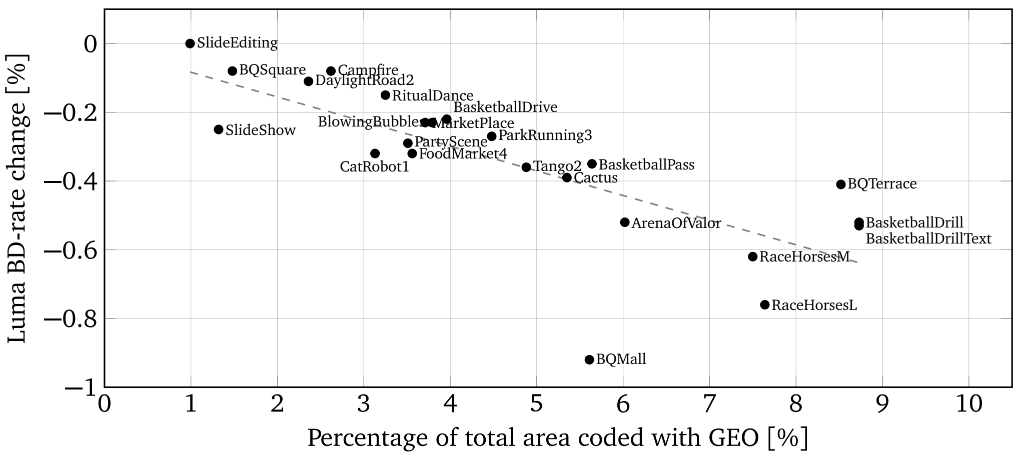


Figure 4.10 Visualization of the relationship between coded area of the GEO mode and the achievable coding gain for Experiment 4.1. The linear trend line is indicated in gray.

overall coding gain that can be achieved is −0.36 % in terms of luma BD-rate change against the VTM-3.2 anchor. It is notable that the coding gain is qualitatively evenly distributed among the different resolution classes. The largest coding gain can be achieved for class C sequences with −0.62 %. In particular, for the BQMall sequence −0.92 % can be achieved, as well as −0.76 % for the RaceHorses sequence. Table A.2 lists the corresponding mode usage of GEO for each class in percentage of total area that is coded using the GEO mode.

The usage statistic indicates that on average, 4.64 % percent of all pixels are coded with GEO. For lower QPs, the usage generally tends to increase. By analyzing the dependency between mode usage in terms of area coded and the resulting coding gain, it can be seen that a roughly linear relationship between the usage of GEO and the achievable coding gain exists, as indicated in Figure 4.10. Here, *BQMall* is the strongest outlier, as it provides the most coding gain with less coded area than expected from the linear trend. It can therefore be assumed that *BQMall* fulfills the geometric partitioning model quite well. Similarly, the two different resolutions of *RaceHorses,* the *BasketballDrill* and the *ArenaOfValor* sequence have higher GEO modes usages which are explainable by the characteristics of these sequences. *BQTerrace* is another interesting deviation, as GEO is utilized in roughly 8.52 % of the entire sequence, although less coding gain at −0.41 % is realized as indicated by the linear trend. A closer examination of the results reveals that for BQ*Terrace* 15.97 % of the entire sequence is coded with GEO at QP22, which is the most for any sequence and Quantization parameter (QP). An explanation of this behavior might be given due to the noisy characteristics of the *BQTerrace* sequence, which influences the encoder more strongly at lower QPs.

Figure 4.11 shows the relative distribution of different block sizes using GEO over the entire CTC test set. As it is to be expected with any coding block tool, with increasing QPs, larger blocks will be utilized more frequently, although the absolute usage of GEO decreases as shown in Table A.2. Further, it is noteworthy that blocks with disproportionate ratios between width and height are used far less frequently compared to those blocks that have identical area, e.g. 32×32 blocks coded with GEO are occurring more frequently compared to 64×16 and 16×64 blocks.

Next to the overall usage, a more in-depth analysis can also be made regarding the distribution of φ and ρ for this particular experiment. By analyzing the distribution of angle and

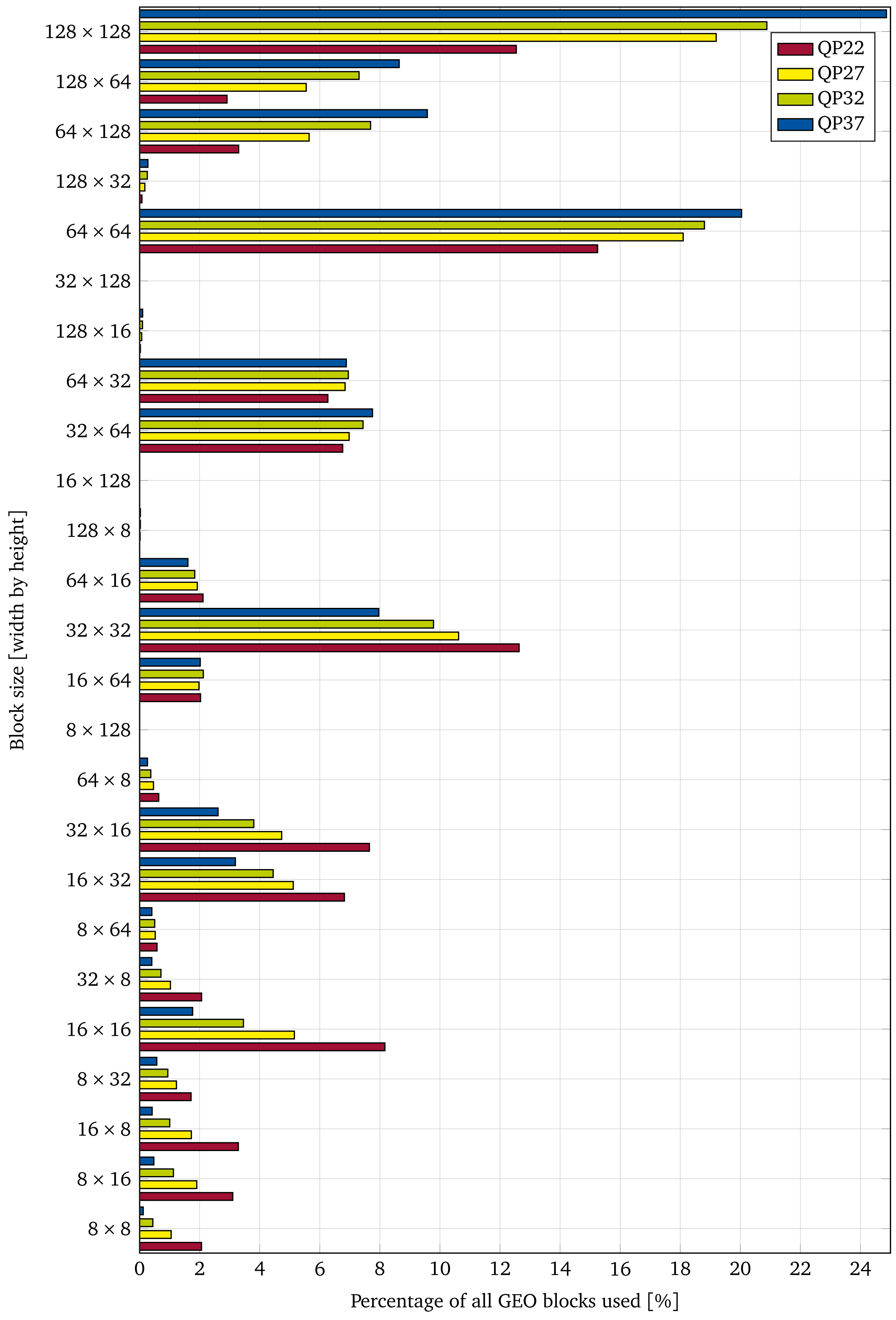


Figure 4.11 Relative block size utilization of GEO for Experiment 4.1.

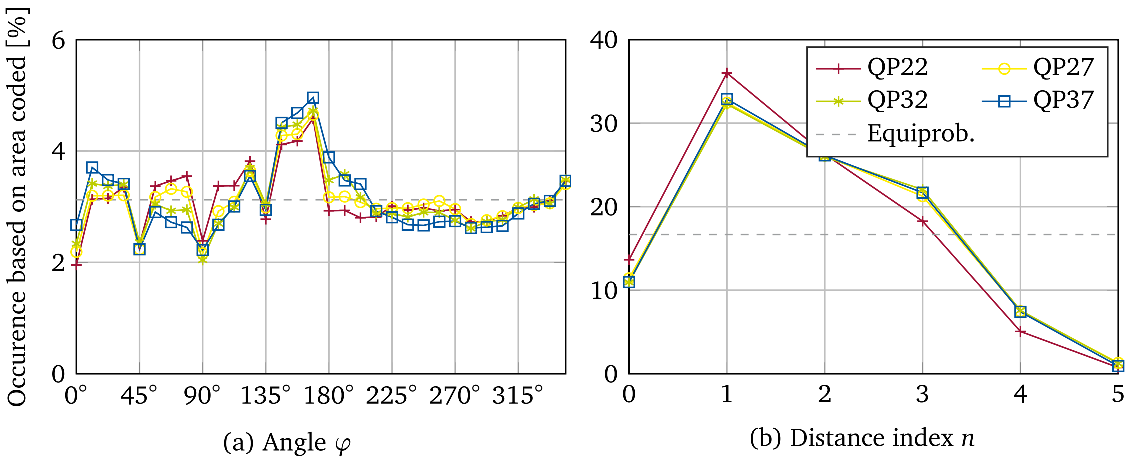


Figure 4.12 Utilization of different angles and distances (as quantized distance indices) for Experiment 4.1.

distance components, it can be assessed whether certain angles and distances are more relevant. This knowledge can be utilized in the subsequent design of a more advanced coding method.

As can be seen in Figure 4.12a, the relative distribution of angles in terms of area coded is rather uniform, albeit with a preference for certain angles. Especially angles in the range between 135° and 180° seem to be slightly favored. These angles typically result in blocks which are split by a line starting from the left or top boundary and ending at the bottom boundary of the current block. One of the resulting GEO partitions will therefore be located close to the top-left corner of the block.

Figure 4.12b shows the corresponding distribution of distances for the given experiment. Unlike the nearly equiprobable distribution of angles, it is noteworthy that a distinctly non-uniform distribution can be observed. The highest probability can be measured for distances with a quantization index of $n = 1$. These are geometric partitions where the partitioning line is slightly off-centered, e.g. the line does not slice the block into segments of equal area. The more off-centered the geometric partitions become with increasing distance index n, the lower the probability that such a partitioning will be used by the coder.

A possible explanation for this behavior might be given by the fact that some GEO partitions for which the partitioning lines pass through the block center, e.g. for $n = 0$, can also be reached by a combination of MTT splits and a TPM split, needing less signaling bits.

A potential problem with the applied quantization scheme in Experiment 4.1 is the independence between the angle and the distance step size $\Delta\rho$ that becomes apparent for blocks with disproportionate ratios of width and height, e.g. $^{w}/_{h} \neq 1$. For example, considering a block of a larger width and smaller height, e.g. 128×16, it can be easily seen that a fixed quantization for the distance $\Delta\rho$ would heavily favor vertical splits over horizontal splits.

4.2.3.2 Angle-dependent quantization

In the following, a quantization scheme is analyzed that still divides the (φ, ρ)-space into equi-angular segments but applies an angle-dependent quantization to the distance component. This can be expressed as:

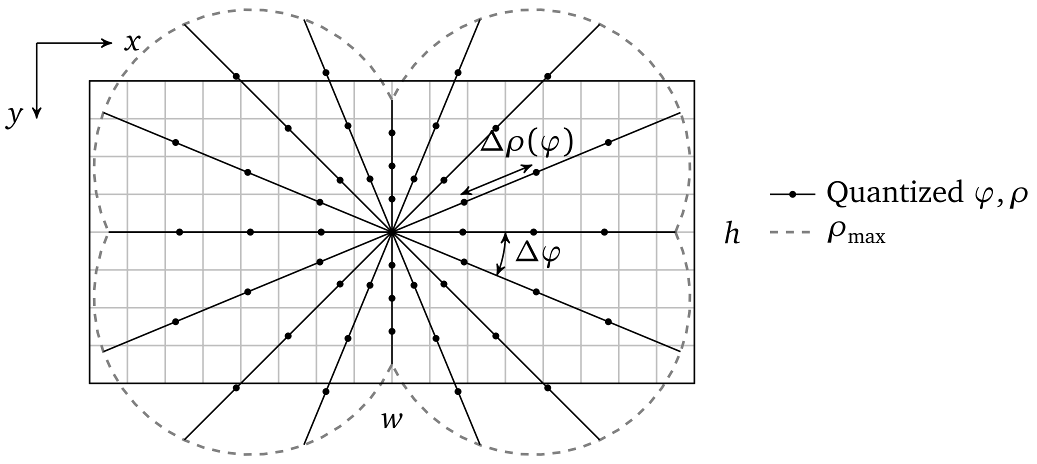


Figure 4.13 Example of a quantized (φ, ρ)-space for given step sizes of $\Delta\varphi = 22.5°$,e.g. $N_\varphi = 16$ and $N_\rho = 4$. The maximum distance $\rho_{\max}$ was computed according to Equation (4.17) and is indicated by the dashed marking. Note that the quantization points which would lie directly on the $\rho_{\max}$ curve are not included in the set of partitions as they would provide very unequally distributed GEO partitions.

$$\Delta\varphi = \frac{360.0°}{N_\varphi} \tag{4.20}$$

$$\Delta\rho\left(\varphi\right) = \frac{\rho_{\max}(\varphi) - \rho_{\mathrm{th}}}{N_\rho} \tag{4.21}$$

Essentially, this quantization scheme keeps the number of distances per angle fixed to the value of N_ρ. From a conceptual point of view, this allows a better coverage of the (φ, ρ)-space of possible geometric partitions and also allows for an easier way of limiting or expanding the total number of geometric partitions by varying the parameter N_ρ. In Figure 4.13, an example of such a quantization is visualized. Here, a threshold of $\rho_{\mathrm{th}} = 1.5$ luma samples is set, to prevent the generation of extremely small partitions.

The total number of partitions using the angle-dependent quantization can now be computed as:

$$N_{w,h}^{\mathrm{AD}} = N_\varphi/2 + N_\varphi \cdot (N_\rho - 1) \tag{4.22}$$

In Table 4.2, the total number of geometric partitions per block size is listed for reasonable values of N_φ and N_ρ. For comparison with Experiment 4.1, the angle-dependent quantization scheme is tested for these combinations of N_φ and N_ρ. First, the coding results for $N_\varphi = 32$, e.g. $\Delta\varphi = 11.25°$ and $N_\rho = 4$ are presented in more detail in Experiment 4.2. The corresponding detailed coding results are given in Table A.3 and the mode usage statistics are listed in Table A.4. This combination of angles and distances results in a varying number of partitions, ranging from 96 partitions up to 172 partitions which can be coded using a 7 or 8 bit fixed length binary code.

Overall, a slight coding gain can be achieved with an overall luma BD-rate change of −0.45 % compared to the −0.36 % that were achieved with the block-size dependent quantization. Also, the GEO mode usage increased from 4.64 % to 5.11 %. In terms of individual sequences, the largest coding gain improvements are achieved for the *BQMall* sequence and

	Number of Partitions $N_{w,h}$								
Block size	$N_\varphi = 32$						$N_\rho = 4$		
$\{w,h\} \times \{h,w\}$	$N_\rho = 1$,	2,	3,	4,	5,	6	$N_\varphi = 8$,	16,	64
8×8	44	96	96	96	96	96	32	56	176
8×16	44	80	118	172	172	172	52	92	332
8×32	44	76	110	156	194	238	44	88	292
8×64	44	76	110	136	186	212	40	72	272
8×128	44	76	106	136	174	200	40	72	264
16×16	44	76	124	172	256	256	52	84	340
16×32	44	76	112	160	202	250	48	88	316
16×64	44	76	108	144	178	220	48	80	272
16×128	44	76	108	144	178	208	48	80	272
32×32	44	76	108	140	196	224	44	76	276
32×64	44	76	108	140	184	210	44	76	268
32×128	44	76	108	140	176	206	44	76	268
64×64	44	76	108	140	172	212	44	76	268
64×128	44	76	108	140	172	204	44	76	268
128×128	44	76	108	140	172	204	44	76	268

Table 4.2 Number of geometric partitions for every block size using the angle-dependent quantization scheme with varying numbers of N_φ and N_ρ.

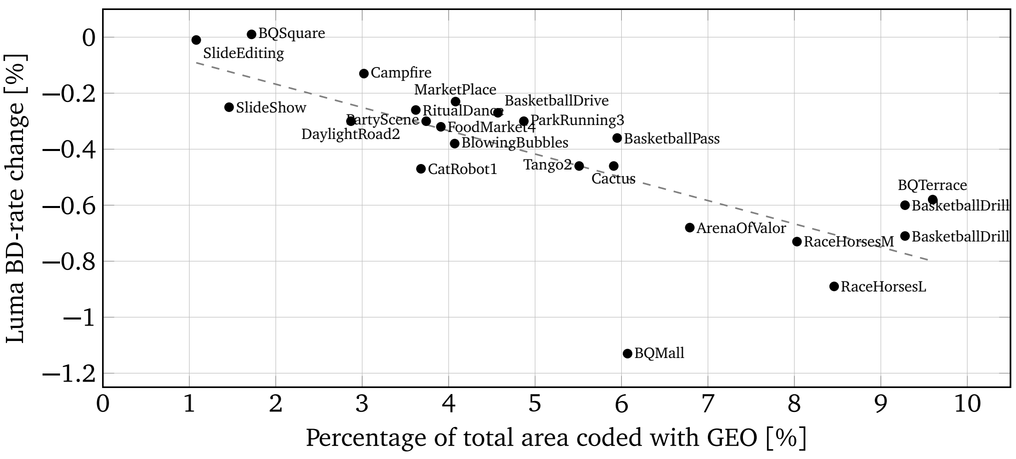


Figure 4.14 Visualization of the relationship between coded area of the GEO mode and the achievable coding gain for Experiment 4.2. The linear trend line is indicated in gray.

	$N_\varphi = 32$					
	$N_\rho = 1$	$N_\rho = 2$	$N_\rho = 3$	$N_\rho = 4$	$N_\rho = 5$	$N_\rho = 6$
Luma BD-rate change	−0.16 %	−0.33 %	−0.38 %	−0.45 %	−0.38 %	−0.43 %

Table 4.3 Coding results for varying number of distances.

the *DaylightRoad2* sequence. An overall loss in coding efficiency compared to the anchor is observed for the *BQSquare* sequence, although very small at 0.01 %. Similar mode usage versus coding gain statistics for Experiment 4.2 can be gathered an visualized in Figure 4.14. However, the two different quantization schemes cannot be directly compared, as in general, more partitioning shapes are available in Experiment 4.2 compared to Experiment 4.1. The benefit of the angle-dependent quantization scheme lies in the fact that quantization is more easily handled and the number of geometric partitions per block size can be controlled precisely.

In the following, a thorough investigation is performed, how many geometric partitions per block provide the most coding gain, still assuming that a fixed length code is applied to signal the partitioning. This ensures that the GEO mode decisions of the encoder for each block are purely based on achieving the minimum distortion.

A full search of the entire (N_φ, N_ρ) parameter space would require an excessive amount of simulation time. Therefore, as already indicated in Table 4.2, a variation of the maximum number of quantized distance parameters N_ρ is performed first, while keeping the number of angles fixed at $N_\varphi = 32$. Quantized distances in the range from $N_\rho = 1$ to $N_\rho = 6$ are evaluated in steps of 1. The actual distance $\Delta\rho$ in terms of luma samples is computed according to Equation (4.21). The overall coding gains for these experiments are listed in Table 4.3.

It can be seen that the largest overall coding gain can be achieved for the setup of Experiment 4.2 using $N_\varphi = 32$ and $N_\rho = 4$, although very similar coding gain is achievable when the number of quantized distances is increased to $N_\rho = 6$.

In the second optimization step, N_ρ is now kept fixed to the value of $N_\rho = 4$, while the

	$N_\rho = 4$			
	$N_\varphi = 8$	$N_\varphi = 16$	$N_\varphi = 32$	$N_\varphi = 64$
Luma BD-rate change	−0.28 %	−0.40 %	−0.45 %	−0.39 %

Table 4.4 Coding results for varying number of angles.

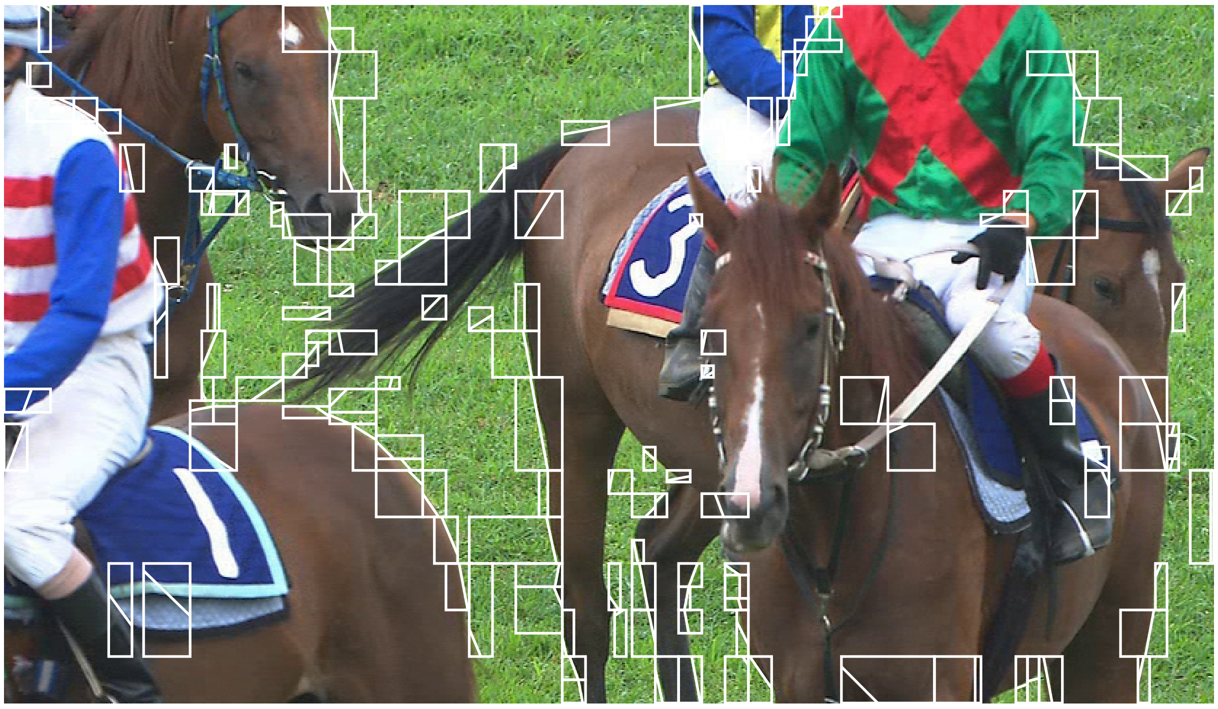


Figure 4.15 Example of a geometrically partitioned picture with the setup described in Experiment 4.2. The relative usage of GEO in terms of pixels coded for this particular picture is 16.06 %. RaceHorses sequence, POC31, RA, QP37.

number of angles N_φ is tested for $N_\varphi = 8$, $N_\varphi = 16$, $N_\varphi = 32$ and $N_\varphi = 64$. The coding results are given in Table 4.4. Again, the combination of parameters used in Experiment 4.2, $N_\varphi = 32$ and $N_\rho = 4$ appears to be optimal if a fixed length code is applied and no methods of prediction are applied to the GEO partitioning information.

Figure 4.15 shows an example of a partitioned picture using GEO for the *RaceHorses* sequence. In this particular example, it can be observed that GEO is frequently used for the coding of object boundaries, such as at the heads, backs and legs of the horses or the shoulders of the horse riders.

4.2.4 Intercept Representation

An alternative representation of the geometric partitioning next to the angle-distance representation is achieved by specifying block intercept points as given in Equation (4.23). Here, the partitioning line is parametrized by two points $P_0 = [x_0\ y_0]^\mathrm{T}$ and $P_1 = [x_1\ y_1]^\mathrm{T}$ which are placed on the boundary of the current block. The idealized partitioning line is then given by all points in the (x, y)-plane that fulfill the condition

$$(y_1 - y_0)x - (x_1 - x_0)y + x_1 y_0 - x_0 y_1 = 0. \tag{4.23}$$

The boundary can be defined as the rectangular border of samples comprising the first and the last row as well as the leftmost and rightmost column. For simplicity, the origin of the coordinate system referring to P_0 and P_1 is moved to the center of the top-left sample, e.g. the location $P = [\ 0 \quad 0\ ]^\mathrm{T}$ denotes the center at the top-left sample. Furthermore, only full-pixel coordinates shall be allowed, thereby all possible coordinates for P_0 and P_1 can be expressed as integers. By taking into account that P_0 and P_1 can never be located in the same row or column, the maximum number $N^{\mathrm{BI}}_{w,h}$ of geometric partitions for a given block of size $w \times h$ can be derived as

$$N^{\mathrm{BI}}_{w,h} = w^2 + h^2 + 4wh + 10 - 8w - 8h. \tag{4.24}$$

It is to be noted that Equation (4.24) also includes the diagonal partitions and the symmetrical vertical and horizontal partitions that are already available in other partitioning modes of VVC, namely TPM. Table 4.5 lists the maximum number of geometric partitions according to Equation (4.24) for typical block sizes. The second column of Table 4.5 gives the number of geometric partitions when no quantization on the granularity of the partitioning is applied. It can be argued that 266 different partitions for an 8 × 8 block and even 96266 partitions for a 128 × 128 block are providing a level of precision which unlikely results in significant coding gain. The burden on an encoder to search so a large partitioning space can also be considered too high.

A straightforward method of quantization to combat this issue using the intercept representation can be achieved by distributing the coordinate points P_0 and P_1 on the rows and columns using variable steps sizes Δw and Δh, respectively. For symmetry and ease of calculation, the block corners shall be included, regardless of the quantization step sizes. Figure 4.16 shows an exemplified quantization for an 8×8 block using $\Delta w = 2$ and $\Delta h = 2$. By connecting all valid points, a total of 80 different partitions are generated for this given example. This is visualized in Figure 4.16.

Effectively, introducing variable step sizes Δw and Δh will quantize the width w and height h to:

$$w_\mathrm{q} = \frac{w}{\Delta w}, \quad \Delta w = 1, 2^k, \dots\ w \tag{4.25}$$

$$h_\mathrm{q} = \frac{h}{\Delta h}, \quad \Delta h = 1, 2^k, \dots\ h \tag{4.26}$$

Using Equation (4.24) and taking into account that the top-left, top-right, bottom-right and bottom-left corner are always valid coordinate points, the total number of partitions for the case $\Delta w \neq 1$ and $\Delta h \neq 1$ can be derived as:

$$N^{\mathrm{BI}}_{w_\mathrm{q},h_\mathrm{q}} = (w_\mathrm{q}+1)^2 + (h_\mathrm{q}+1)^2 + 4(w_\mathrm{q}+1)(h_\mathrm{q}+1) + 10 - 8(w_\mathrm{q}+1) - 8(h_\mathrm{q}+1) \tag{4.27}$$

Similar as for the quantization process detailed in Subsection 4.2.3.1, the step sizes Δw and Δh can now be made block-size dependent by scaling the step sizes Δw and Δh according

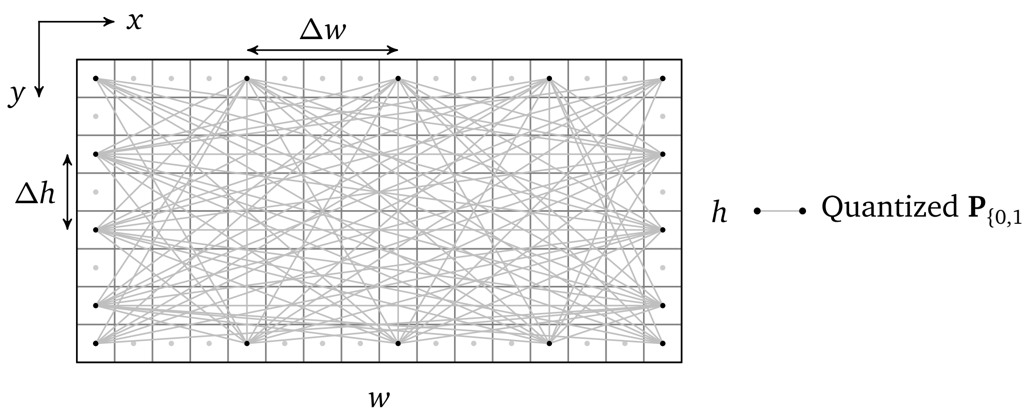


Figure 4.16 Example of all valid geometric partitions using the quantized intercept representation for a block of 16 × 8 luma samples. The example shows a total of 80 partitions using step sizes of $\Delta w = 4$ and $\Delta h = 2$.

to the width and height of the current block. Table 4.5 lists the total number of geometric partitions per block size for different quantization schemes in the third to fifth column. In contrast to the angle-distance representation detailed in Section 4.2.3, the number of geometric partitions appears to be more difficult to control, if non-base-2 step sizes Δw and Δh are avoided.

The following Experiments 4.3, 4.4 and 4.5 evaluate how the coding performance is influenced by different step sizes $\Delta w = w/4$ and $\Delta h = h/4$, $\Delta w = w/8$ and $\Delta h = h/8$, or $\Delta w = w/16$ and $\Delta h = h/16$. The overall coding results for these three experiments are shown in Table 4.6. Detailed results are given in Tables A.5, A.6 and A.7 of the appendix. The three corresponding quantization schemes are listed in Table 4.5 and show how many different geometric split modes are available per block.

Identical to the previous Experiments 4.1 and 4.2, the coding of GEO partitions is performed by signaling an index $k \in \{0 \ldots N^{\mathrm{BI}}_{w_\mathrm{q},h_\mathrm{q}} - 1\}$ using a fixed length binary code of length $l = \left\lceil \log_2 N^{\mathrm{BI}}_{w_\mathrm{q},h_\mathrm{q}} \right\rceil$ per block.

The results show that the best achievable coding efficiency in Experiment 4.3 with −0.42 % is lower compared to the best achievable result using the angle-distance representation. However, in terms of number of GEO partitions available, the setup of Experiment 4.3 using 80 partitions is comparable to the angle-distance combination $N_\varphi = 32$ and $N_\rho = 2$ with a median of 76 partitions. Here, the intercept representation achieves a BD-rate change of −0.38 % compared to −0.33 % for the angle-distance representation. This could indicate that the partitioning shapes resulting from the quantization performed using the intercept representation provide a better distribution and are therefore more useful to the coder. The number of possible GEO partitions for the next quantization step size already greatly increases for Experiment 4.4 to 266-352 partitions, depending on the block size. Overall coding efficiency improves with more partitions to −0.42 % compared to the VTM-3.2 anchor. An even higher number of partitions is available in Experiment 4.5 with 266-1472 possible splits, depending on the block size. This causes a relative drop in coding efficiency to −0.37 %, indicating that the excessive number of partitions needing more signaling bits does

Block size	Number of Partitions $N_{w,h}^{\mathrm{BI}}$			
$\{w,h\} \times \{h,w\}$	$\Delta w = 1$ $\Delta h = 1$	$\Delta w = w/4$ $\Delta h = h/4$	$\Delta w = \lceil w/8 \rceil$ $\Delta h = \lceil h/8 \rceil$	$\Delta w = \lceil w/16 \rceil$ $\Delta h = \lceil h/16 \rceil$
8×8	266	80	266	266
8×16	650	80	307	650
8×32	1802	80	307	707
8×64	5642	80	307	707
8×128	19466	80	307	707
16×16	1290	80	352	1290
16×32	2954	80	352	1379
16×64	7818	80	352	1379
16×128	23690	80	352	1379
32×32	5642	80	352	1472
32×64	12554	80	352	1472
32×128	32522	80	352	1472
64×64	23562	80	352	1472
64×128	51722	80	352	1472
128×128	96266	80	352	1472

Table 4.5 Number of geometric partitions using the intercept representation, depending on the block size for two different quantization schemes.

	Step size quantization		
	$\Delta w = w/4$ $\Delta h = h/4$	$\Delta w = w/8$ $\Delta h = h/8$	$\Delta w = w/16$ $\Delta h = h/16$
Luma BD-rate change	−0.38 %	−0.42 %	−0.37 %

Table 4.6 Coding results for different block intercept step size quantizations. More detailed results are shown in Table A.5, Table A.7 and Table A.5.

not provide a better Rate-distortion (RD) trade-off.

4.2.5 Summary

In summary it can be stated that the angle-distance and intercept-based representation of GEO partitions provide similar coding gains if the number of geometric splits is comparable. The number and shape of the partitions is considered to be more precisely controllable from an implementation point of view using the angle-distance representation. Here, the highest coding gain was achieved with a luma BD-rate change of −0.45 % over VTM-3.2 for a setting using $N_\varphi = 32$ angles with equiangular spacing and an angle-dependent quantization of the distance using $N_\rho = 4$ steps. The best parameter settings for each representation will be used for further experiments.

4.3 Prediction and Coding of Geometric Partitioning Side-Information

This section details different approaches to the prediction and coding of geometric partitioning parameters. In the previous section, these two aspects of geometric partitioning were performed only in the most simplistic manner: the partitioning is signaled by means of an index value k which maps to the respective representation of geometric partitioning parameters, e.g. angle-distance values or (P_0, P_1)-coordinate pairs. The index value was coded with a fixed length binary code and thereby neglecting the inherent source entropy or conditional entropy.

It has already been observed in Experiment 4.1 that certain geometric partitions are more likely to occur than others. By tendency, those geometric partitioning shapes seem to less likely occur that produce a more unbalanced segmentation of the given block. This observation will be analyzed in greater detail. Further, for the case of using an angle-distance representation, it has been observed that the angle and distance information have different distributions, which can also be potentially exploited for coding.

Another aspect which this section tries to answer is the open question how geometric partitioning parameters can be predicted. As object boundaries are often smooth and not subject to sudden changes in direction, it is reasonable to assume that this property can be exploited for prediction and coding.

First, coding experiments are conducted to analyze the RD-performance of different explicit signaling schemes in Section 4.3.1. Then, possibilities of spatial and temporal prediction of the partitioning are investigated in Section 4.3.2.

4.3.1 Entropy Coding for Geometric Partitioning Parameters

An improved coding without prediction of the partitioning parameters can be achieved by variable length coding according to the source statistics. In the context of modern video coding schemes using a binary arithmetic coder as the back-end of the codec, this can be achieved through two different approaches:

- Variable-length binarization of the source alphabet according to the source entropy.

- Context-adaptive coding of the binarized bins.

Both approaches can be performed independently of each other (e.g. a variable length binarization with equiprobable- or bypass-coding of the bins or context-adaptive coding of fixed length binarized bins) and also in combination for highest compression performance. Since fixed length coding of the partitioning index was used in the previous coding experiments, other entropy coding methods will be tested in the following subsections, namely Huffman coding of the partitioning index, truncated binary coding of the partitioning index and separate coding of the angle and distance parameters:

- Experiment 4.6 analyzes the coding performance of Huffman coding for explicit signaling, exploiting the knowledge that certain geometric partitions are more likely to occur than others.

- Experiment 4.7 analyzes the coding performance using truncated binary coding.

- Experiment 4.8 analyzes the coding performance when angle and distance parameters are coded separately.

4.3.1.1 Huffman Coding

An optimal binarization for a discrete source with known symbol statistics can be obtained through Huffman coding, see Section 2.2.8.1. The Huffman code is considered as it can be used for coding the partitioning information without knowledge of the chosen representation of geometric partitioning parameters. Each partitioning mode per block size is associated with the index value k within a known range 0 to $N_{w,h} - 1$. In comparison to the previous approach, this index value k is now binarized using Huffman trees $C_{\text{H}}(\cdot)$ which have been pre-computed for each block size. Each bin of the resulting bin-string is bypass-coded using the CABAC engine of VVC. The statistics necessary for pre-computing the Huffman trees have been collected from the previous Experiment 4.2 using the identical sequences. For evaluation, a new Experiment 4.6 is conducted according to the JVET CTC.

The coding results shown in Table A.8 indicate that Huffman coding can increase the overall coding efficiency from −0.45% using fixed length binary coding in Experiment 4.2 to −0.50 %. The improvement in coding efficiency is very similar for different sequence classes, which validates that separate Huffman coding for individual block sizes improves the coding for different resolutions. In particular, the coding efficiency for class C sequences was improved from −0.73 % to −0.83 %, which marks the biggest increase per class.

Detailed results on the Huffman code associated with the respective geometric partitioning for a block size of 32×32 luma samples are given in Table 4.7. The table lists the source statistics for the five most-probable geometric partitions and the five least-probable geometric partitions along with the Huffman code, e.g. the binarized partitioning index value $C_{\text{H}}(k)$ and the measured entropies.

In summary, the entropy of the resulting code words $H(C_{\text{H}}) = 6.87$ bit is close to the entropy of the source $H(S) = 6.83$ bit measured by the occurrences of the partitioning indices. Further, Table 4.7 reveals which partitioning shapes are utilized more frequently and are accordingly coded with fewer bits. In Figures 4.12a and 4.12b, the independent probability distributions of different angles and distances were already studied and it was observed that

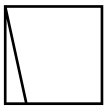
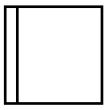
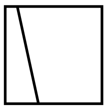
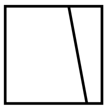
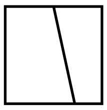
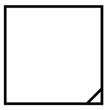
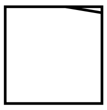
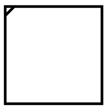
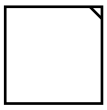
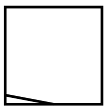
Entry k	Partition Index	Source Statistics S		Code words $C_H(k)$	Partition
		Count	Probability		
0	64	2696	0.0143	011101	
1	68	2664	0.0141	011011	
2	63	2484	0.0131	010100	
3	117	2473	0.0131	010011	
4	116	2453	0.0130	010010	
...					
115	19	597	0.0032	01011000	
116	99	584	0.0031	00110001	
117	82	578	0.0031	00110000	
118	109	548	0.0029	111101111	
119	40	495	0.0026	111101110	
Summary		188908	$H(S)$=6.83 bit	$H(C)$=6.87 bit	

Table 4.7 Huffman coding of the partition index information for a block of size 32 × 32 luma samples.

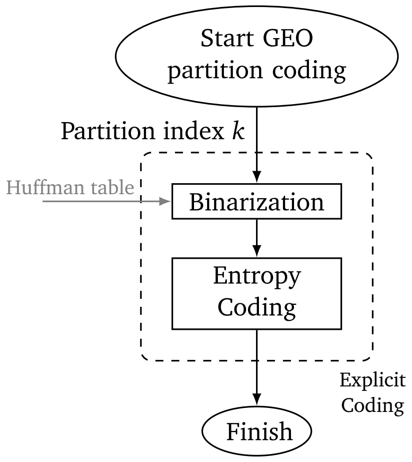


Figure 4.17 Huffman coding of the partitioning information. The index value k is translated into a variable length bin string using Huffman tables $C_{\mathrm{H}}(\cdot)$. The bins are subsequently bypass coded using the binary arithmetic coding engine of VVC.

geometric splits occurring in the left side of the block are qualitatively more probable. This observation is now confirmed, as the three most probable partitions show the same general shape. Of course, it must be pointed out that the variable-length Huffman coding is immediately reflected by changing encoder rate-distortion optimization decisions. As the rate required for signaling the partitioning information in Experiment 4.6 is now varying from roughly 6 bit to 9 bit, the selection of the most probable geometric splits is automatically enforced by the encoder.

4.3.1.2 Truncated Binary Coding of Geometric Partitioning Parameters

Truncated binary coding is a simple modification to fixed length coding detailed in Section 2.2.8.1, used to model a uniformly distributed finite alphabet of symbols, where the total number of symbols N is not a power of two. If $l = \lfloor \log2(N) \rfloor$, then truncated binary coding assigned the first M symbols to code words of length l and the remaining symbols $N - M$ to code words of length $l + 1$, where $M = 2^{l+1} - N$. As it is used for fixed length coding and Huffman coding, the bins of the resulting bin-string are bypass-coded.

In Experiment 4.7, the coding performance for truncated binary coding is assessed. Coding results for this setup are given Table A.9. The overall coding efficiency improves slightly from −0.45 % for fixed length coding in Experiment 4.2 to −0.47 % with truncated binary coding. Considering the very low complexity of truncated binary coding compared to Huffman coding, this method can be seen as a sensible alternative to fixed length coding.

4.3.1.3 Separate Coding of Geometric Partitioning Parameters

The analysis of the distribution of angle and distance parameters performed in Section 4.2.3.1 and displayed in Figure 4.12 shows that a nearly uniform distribution of angles can be expected and a distribution of distances that favors blocks which are sliced slightly off-centered. Distances that produce more unbalanced GEO partitions become less likely to be chosen by the encoder. This characteristic can be utilized for a coding scheme of the partitioning parameters, specifically adapted to needing less bits for more likely parameter combinations. A coding scheme that models the measured parameter more closely is given by:

1. Fixed length bypass coding of the angle φ by signaling the quantization index m, e.g. as in $\varphi = m \cdot \Delta\varphi$. For the 32 different angles, a 5 bit fixed length code $C_{\mathrm{FL},5}(\cdot)$ is chosen.

2. Single context CABAC-coded non-zero flag combined with truncated-unary bypass coding of the distance quantization index n, e.g. as in $\rho = n \cdot \Delta\rho(\varphi)$. The non-zero CABAC-coded flag adapts to the probability $P(n = 0) \approx 0.12$, while the remainder is coded by the TU code $C_{\mathrm{TU},\mathrm{N}_\rho-1}(n-1)$ with maximum symbol $N_\rho - 1$. The same context model is used for all available block sizes.

The detailed coding results for this Experiment 4.8 are shown in Table A.10. It can be seen that the coding efficiency compared to fixed length coding is nearly identical overall, measured at −0.46 %. These results also verify clearly that the distributions of angle and distance parameters cannot be considered to be statistically independent of each other, as almost no additional coding gain can be reported.

4.3.2 Predictive Coding for Geometric Partitioning Parameters

Predictive coding of side-information is widely used in different parts of a video coding scheme. For the case of inter-prediction in VVC for example, this is realized by prediction of motion vectors for translational and affine motion. In natural video content, motion is frequently observed to be spatially and temporally coherent, meaning it is only slowly changing locally and over time. Examples for such video content are rigid objects in motion, such as cars moving along a road, people walking or general camera motion. This property of the motion vector field is exploited extensively through predictive coding: the motion of already coded picture regions is searched by an encoder for a good predictor that provides the best rate-distortion performance according to the selected operating point, e.g. the given λ value. For very dynamic content and especially dynamic textures with random or quasi-random motion however, it is well known that motion vector prediction may fail. Examples of such video content are the leaves of a tree in motion or the motion of water, smoke, and fire. In such cases, other prediction modes may be favorable and chosen by an encoder.

For rigid objects, the question arises whether the properties of the video content can also be exploited for coding the partitioning, specifically the geometric partitioning parameters of a block. The visual coding example of Figure 4.15 given in Section 4.2.3.2 already seems to indicate that for higher QPs, the geometric block partitioning is roughly aligned with object boundaries. In analogy to motion vector prediction, the simplest approach to predicting the geometric partitioning would assume that the partitioning itself changes slowly and can therefore be predicted using a simple linear model. Ultimately, it would be beneficial to exploit this postulated property and use it for improved coding of the partitioning data.

The predictive coding of the geometric partitioning parameters can be integrated into the existing codec using a separate GEO coding mode as indicated in Figure 4.18. The GEO coding mode, signaled by a flag, indicates whether predictive coding or explicit coding as introduced in Section 4.2.2 is used for the current block. This ensures that in cases where predictive coding fails or where it is not available, the partitioning information can be signaled

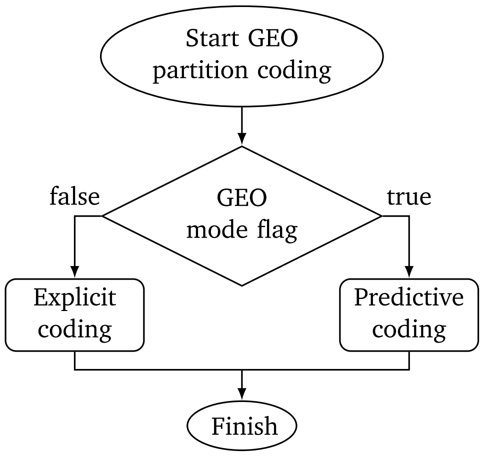


Figure 4.18 Block diagram showing the addition of a predictive coding mode next to the explicit coding mode for the partitioning data.

using the established method. Based on the coded partitioning of a spatially or temporally neighboring block given by parameters $\boldsymbol{A}_{\mathrm{N}}$, the goal of the prediction process is to obtain a prediction of the current block given by parameters $\boldsymbol{A}_{\mathrm{Pred}}$. It is noted that the parameters $\boldsymbol{A}_{\mathrm{N}}$ and $\boldsymbol{A}_{\mathrm{Pred}}$ can relate to the angle-distance or intercept representation, as they can be easily converted into one another. Since it is unlikely that the predicted partitioning $\boldsymbol{A}_{\mathrm{Pred}}$ will be identical to the RD-optimal partitioning determined by an encoder for the current block, the prediction can be updated using prediction error parameters given by $\boldsymbol{A}_{\mathrm{E}}$:

$$\boldsymbol{A}_{\mathrm{C}} = \boldsymbol{A}_{\mathrm{Pred}} + \boldsymbol{A}_{\mathrm{E}} \tag{4.28}$$

In the following sub-sections, methods will be detailed, how $\boldsymbol{A}_{\mathrm{Pred}}$ can be generated from the spatial or temporal neighborhood of a coding block and how this side-information can be signaled to the decoder. Further, coding approaches for the prediction error $\boldsymbol{A}_{\mathrm{E}}$ are discussed with specific adaptations based on the chosen representation of the geometric partitioning parameters.

4.3.2.1 Spatial Prediction

Geometric partitioning parameters can be predicted from the spatial neighborhood of a coding block by different approaches. In general, three different sources of side-information that could be utilized for this purpose come to mind:

- The partitioning parameters of the local neighborhood. This may include geometric partitioning parameters but also conventional quad- or binary-split information could be utilized.

- The prediction mode information of the local neighborhood. This could for example encompass the motion vector field, where a sudden local change of the motion vector direction may indicate the presence of an object boundary. Further, directional intra-prediction modes could indicate the presence of structures parallel or perpendicular to a partitioning boundary.

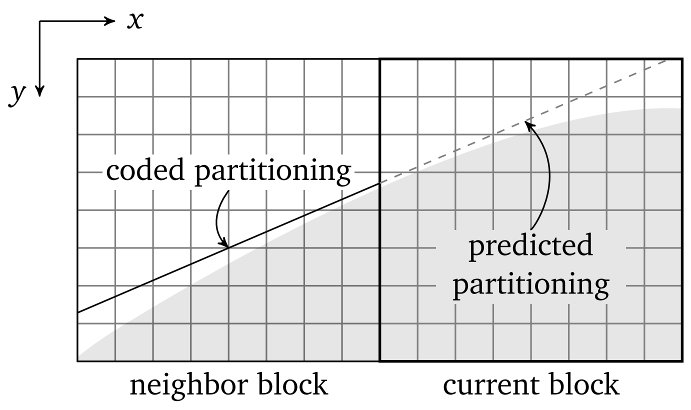


Figure 4.19 Example of spatial prediction of the geometric partitioning from the local neighborhood of a coding block. The prediction, which is a linear continuation of the partitioning boundary of the neighboring block into the current block, is indicated by the dashed line. The hypothetical segmented object is shaded in gray.

- Reconstructed samples which, for example, could be segmented and used to interpolate the object boundary. Other approaches could utilize edge- or corner detectors in combination with Hough-transform based feature extraction.

As the prediction process needs to be performed at the decoder, all considered examples for prediction sources listed above need to be assessed in terms of their technical feasibility. It is often prohibited in an actual decoder to access reconstructed samples for purposes of mode derivation. Such operations may violate the sequential pipelining used in hardware decoders and may introduce significant latency [CLC06]. Therefore, in the following sub-sections the focus is laid on methods utilizing the coded partitioning side-information for prediction purposes. Specifically, only the immediate local neighborhood of the current block may be used for prediction purposes, meaning that only those coding units are considered and accessed which are adjacent to the top and the left block boundary. This limiting constraint is in line with the access constraints applied in other coding modes, such as motion vector prediction or intra-mode prediction. Figure 4.19 exemplifies the process of spatial prediction for geometric partitions, assuming a linear continuation of the neighboring partitioning line.

Analysis of spatial geometric partitioning properties The application of spatial prediction assumes that correlation exists between the partitioning information of adjacent blocks. As a first prerequisite, it is necessary that blocks coded with GEO are occurring in clusters with adjacent block boundaries as prediction is only applied across the immediate local neighborhood. The coding results that have already been generated in Experiment 4.2 are analyzed to obtain further insights whether this property can already be observed in coded sequences. The coding results of Experiment 4.2 are suited for this purpose as no prediction of the geometric partitioning data has been used in the setup. The encoder used for Experiment 4.2 chooses the geometric partitioning mode without knowledge of the local neighborhood, optimizing for lowest possible distortion only.

Secondly, as it is assumed that the partitioning is predictable at object boundaries, the previous coding results can be analyzed for occurrences where the geometric partitioning lines of neighboring blocks are *connected*. Connected partitioning lines, meaning that two

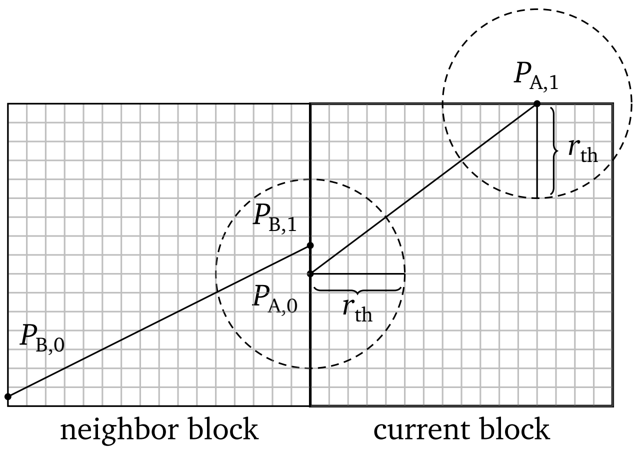


Figure 4.20 Adjacent GEO blocks are labeled as connected, if the Euclidean distance between two line coordinates is smaller than a given threshold r_{th}. The distance is measured for interpolated points, which are located on the boundary of the current block.

end points of two line segments are reasonably close to each other, could indicate the presence of a coherent object boundary. Naturally, an object boundary could also extend over multiple, adjacent coding blocks. Therefore, it is reasonable to not only measure the relative frequency of GEO blocks with connected partition lines but also the number of adjacent GEO blocks that are connected in a single cluster.

For the analysis of the partitioning data, a radial threshold r_{th} is introduced. Two partitioning lines are denoted as being *connected,* if any coordinate pair meets the condition:

$$\left\| P_{A,i} - P_{B,j} \right\|_2 < r_{th} \quad i,j \in \{0,1\} \tag{4.29}$$

Figure 4.20 illustrates this measurement procedure. If the condition given by Equation (4.29) is not met, a cluster size $g_S = 1$ is assigned for the current coding block, indicating that this block is isolated and therefore unlikely to be usable for spatial prediction. If the condition given by Equation (4.29) is met for two adjacent GEO blocks, the cluster size $g_S = 2$ is assigned. Generalizing this for more than two adjacent blocks, the cluster size $g_S = n$ is assigned if n GEO blocks are connected. Accordingly, a large radial threshold r_{th} in terms of luma samples will result in more adjacent GEO blocks being labeled as connected. Thus, r_{th} can also be thought of as the parameter controlling the sensitivity of a detector for connected partitions.

Figure 4.21 shows a visual example of the distribution of cluster sizes g_S for picture 31 of the *RaceHorses* sequence, using a detection threshold of $r_{th} = 3$ luma samples. For this particular example, quite a few larger clusters with $g_S \geq 3$ are observed. Although this single example seems to indicate that smooth, connected object boundaries are existing in the video content and reflected by the partitioning of the coded material, it remains to be seen how this observation generalizes to a larger selection of natural video sequences at different resolutions. Therefore, the measurement is performed for the entire JVET CTC test set, using the RA coding results of Experiment 4.2. The cluster sizes g_S are collected for every sequence and averaged for each QP. To estimate the influence of the detection threshold r_{th},the measurement is repeated for three different thresholds r_{th} =1, 2.25 and 3

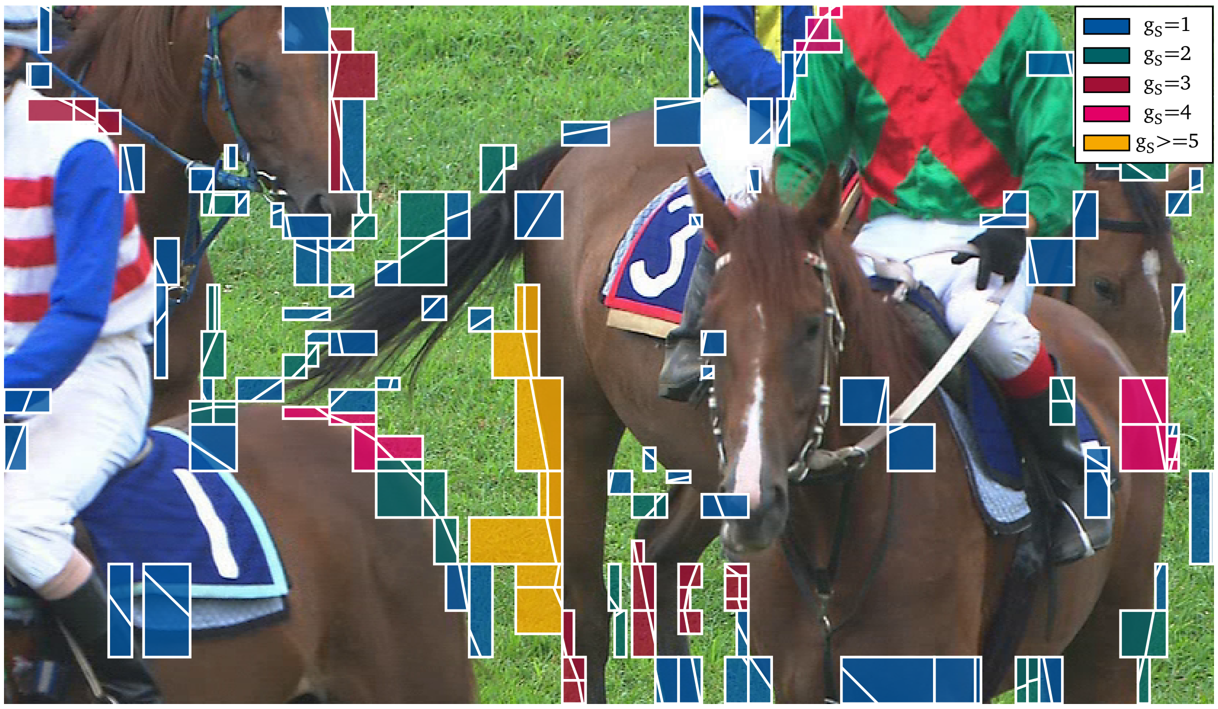


Figure 4.21 Visualization of connected GEO block clusters for a threshold of $r_{\mathrm{th}} = 3$. The number of connected blocks contained in a cluster is given by g_{S} and indicated by color. RaceHorsesL sequence, POC31, RA, QP37.

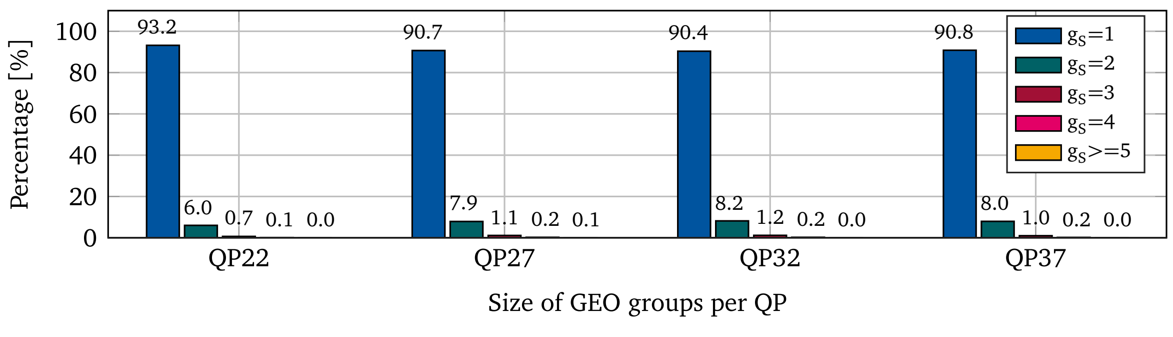
luma samples, which are reasonably small values for which adjacent blocks are labeled as connected.

The measurement results shown in Figure 4.22 indicate that the majority of the coded GEO blocks of Experiment 4.2 are in fact either isolated blocks or not connected and do not form part of a larger object boundary. For a radial threshold of $r_{\mathrm{th}} = 1$ luma sample, more than 90 % of all GEO blocks are labeled as $g_{\mathrm{S}} = 1$ and therefore isolated. However, a higher QP increases the percentage of connected GEO blocks. For a detection threshold of $r_{\mathrm{th}} = 3$ and a QP of 37 for example, 16.8 % of all coded GEO blocks are occurring in connected clusters.

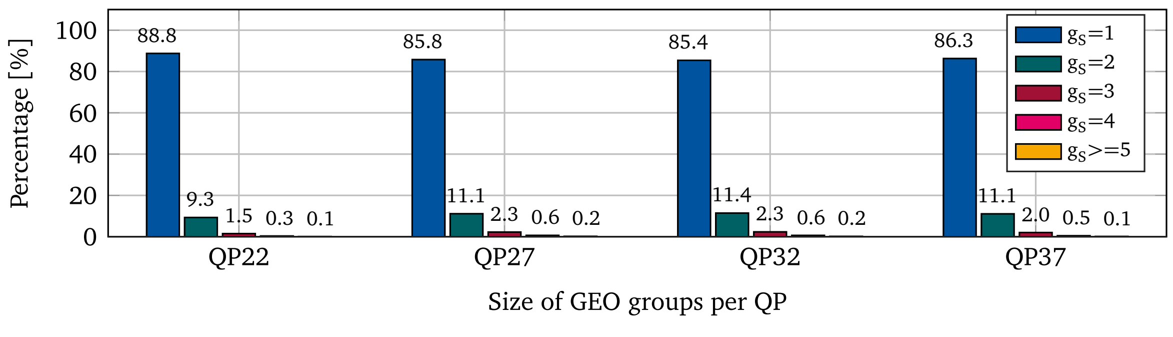
For these connected blocks it can further be measured, if the proposed simple linear continuation of the partitioning line of each neighboring block coincides with the partitioning of the current block, as chosen by the encoder during RDO. For this purpose, the difference angle θ is measured, which is given by the smallest absolute angle between two partitioning lines, parametrized by $\boldsymbol{A}_{\mathrm{A}} = [\varphi_{\mathrm{A}}\, \rho_{\mathrm{A}}]$ and $\boldsymbol{A}_{\mathrm{B}} = [\varphi_{\mathrm{B}}\, \rho_{\mathrm{B}}]$:

$$\theta = \min(\pi - |\varphi_{\mathrm{A}} - \varphi_{\mathrm{B}}|, |\varphi_{\mathrm{A}} - \varphi_{\mathrm{B}}|). \tag{4.30}$$

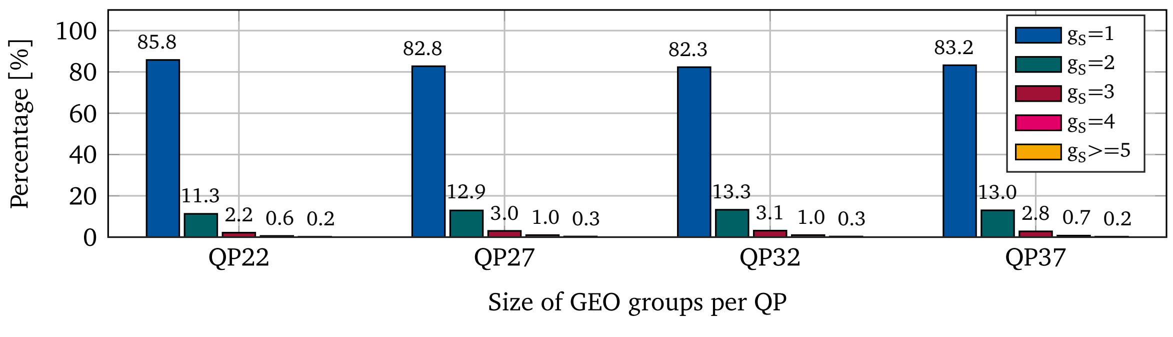
This is illustrated in Figure 4.23. The difference angle θ is equal to zero if the prediction and the coded partitioning line are identical. The maximum value of the difference angle is given by $\theta = 90°$, if the coded partitioning line is perpendicular to the prediction. For all connected blocks with $g_{\mathrm{S}} \geq 2$, the difference angle is measured over the JVET CTC test set and a histogram of the difference angles is computed using a bin size of 11.25°. The results for the four different QPs are shown in Figure 4.24. The relative frequency of difference



(a) $r_{\text{th}} = 1$.



(b) $r_{\text{th}} = 2.25$.



(c) $r_{\text{th}} = 3$.

Figure 4.22 Distribution of connected GEO blocks measured for different thresholds r_{th} and gathered over the JVET CTC test set for RA configuration of Experiment 4.2.

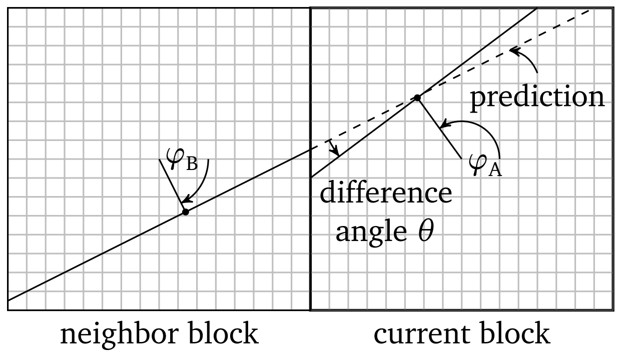


Figure 4.23 Visualization of the analysis methodology regarding the difference angle θ between predicted and coded partitioning for connected blocks.

angles indicates that a slight correlation exists between adjacent GEO blocks with connected partitions. For high QPs, the difference angle is in the range of $0 \leq \theta \leq 11.25°$ for over 40 % of all connected blocks, thus indicating the presence of smooth object boundaries. A perfect continuation of the partitioning with a difference angle of $\theta = 0°$ however, was measured in only 13.5 % of all connected blocks for QP37. The difference angle distribution becomes markedly flatter for the lowest QP.

Summarizing these findings, it can be stated that although the overall percentage of coded GEO blocks which are adjacent to each other and sharing a connected partitioning is comparatively low, prediction of the partitioning by linear extrapolation could be a viable method to improve coding efficiency for sequences where distinct object boundaries are present. Especially for higher QPs, it has been shown that GEO blocks occur in connected clusters more frequently. In those cases, the geometric partitioning for the current block can be modeled by linear extrapolation of the partitioning from a neighbor block.

Further, it must be noted that it is difficult to assess the potential of spatial prediction based on these findings alone. The RDO process and search strategy of an encoder may undoubtedly have significant influence on the global distribution of GEO modes, block size utilization and other local properties in the coded material. The influence of RDO on the GEO distribution in the shown analysis has been limited due to the fixed-length signaling of the GEO mode used in Experiment 4.2. A practical encoder however could also be designed which performs a pre-analysis of the texture or the local neighborhood and selects a GEO mode accordingly. The measurement could be further refined by considering neighboring non-GEO blocks and the rectangular partitioning structure as a source for potential spatial predictors.

Spatial prediction using angle-distance representation In the following, the geometric partitioning of the current block is assumed to be predictable through a linear continuation of the partitioning of a neighboring block, due to the conclusions drawn based on the previous analysis. Given the sizes and positions of the current and adjacent neighbor block, the predicted parameters $\boldsymbol{A}_{\text{Pred}}$ relating to the current block can be derived. The coordinates of the neighboring block with respect to the top-left luma sample shall be denoted as $[x_{\text{N}}, y_{\text{N}}]^{\text{T}}$ and the size of the neighboring block in terms of luma samples as $w_{\text{N}} \times h_{\text{N}}$. Accordingly, the coordinates and size of the current block are given by $[x_{\text{C}}, y_{\text{C}}]^{\text{T}}$ and $w_{\text{C}} \times h_{\text{C}}$.

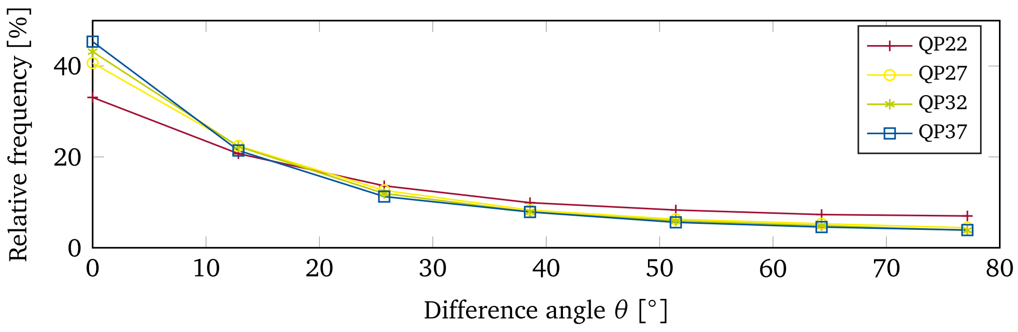


Figure 4.24 Distribution of the difference angle θ between predicted and coded partitioning using the coding results generated in Experiment 4.2. The difference angle distribution was gathered for each QP and accumulated over all sequences.

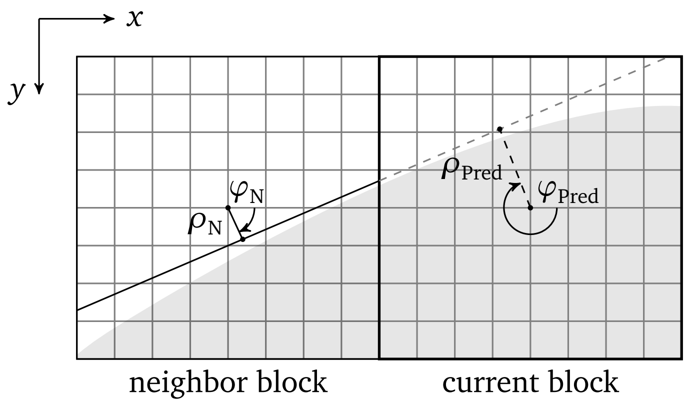


Figure 4.25 Exemplified spatial prediction of the partitioning parameters φ_{Pred} and ρ_{Pred} for the current block, based on the partitioning parameters of the neighbor block given by φ_{N} and ρ_{N}. Note that φ_{pred} and φ_{N} differ only by a 180° shift. The hypothetical object being partitioned is indicated by the shaded shape.

The angle and distance parameters $\boldsymbol{A}_{\mathrm{N}} = [\varphi_{\mathrm{N}}\ \rho_{\mathrm{N}}]$ are given with respect to the center of neighboring block, e.g. $[x_{\mathrm{N,c}}, y_{\mathrm{N,c}}]^{\mathrm{T}} = [x_{\mathrm{N}}, y_{\mathrm{N}}]^{\mathrm{T}} + [{}^{w_{\mathrm{N}}}/2, {}^{h_{\mathrm{N}}}/2]^{\mathrm{T}}$. Accordingly, the center coordinates of the current block are given as $[x_{\mathrm{C,c}}, y_{\mathrm{C,c}}]^{\mathrm{T}} = [x_{\mathrm{C}}, y_{\mathrm{C}}]^{\mathrm{T}} + [{}^{w_{\mathrm{C}}}/2, {}^{h_{\mathrm{C}}}/2]^{\mathrm{T}}$. The predicted distance ρ_{Pred} of the current block can now be easily derived as:

$$\rho_{\mathrm{Pred}} = |d_{\mathrm{Pred}}| \tag{4.31}$$

with d_{Pred} defined by the distance of the center of the current coding block to the partitioning line:

$$d_{\mathrm{pred}} = d_{\mathrm{CN,x}} \cos(\varphi_{\mathrm{N}}) - d_{\mathrm{CN,y}} \sin(\varphi_{\mathrm{N}}) - \rho_{\mathrm{N}} \tag{4.32}$$

$$\begin{bmatrix} d_{\mathrm{CN,x}} \\ d_{\mathrm{CN,y}} \end{bmatrix} = \begin{bmatrix} x_{\mathrm{N,c}} \\ y_{\mathrm{N,c}} \end{bmatrix} - \begin{bmatrix} x_{\mathrm{C,c}} \\ y_{\mathrm{C,c}} \end{bmatrix} \tag{4.33}$$

The distance d_{Pred} can also be utilized to determine, whether the predicted partitioning will slice the current block. This can be achieved by comparing d_{Pred} against the maximum valid distance ρ_{max} as specified in Equation (4.16). Regarding the prediction of the angular partitioning parameter, it can be seen that the predicted angle φ_{Pred} is identical to the angle of the partitioning line of the neighboring block, augmented by a $\pm 180°$ offset, depending on the sign of d_{Pred}. This is indicated in the example given in Figure 4.25.

$$\varphi_{\mathrm{Pred}} = \varphi_{\mathrm{N}} + k\pi, \quad k = \begin{cases} +1 & d_{\mathrm{Pred}} > 0,\ \varphi_{\mathrm{N}} > \pi \\ -1 & d_{\mathrm{Pred}} > 0,\ \varphi_{\mathrm{N}} \leq \pi \end{cases} \tag{4.34}$$

The final predicted parameters are quantized to the nearest values contained in the code book of partitioning parameters for the current block by using a linear quantization:

$$\varphi_{\mathrm{Pred,q}} = \left\lfloor \frac{\varphi_{\mathrm{Pred}}}{\Delta\varphi} + \frac{\Delta\varphi}{2} \right\rfloor \cdot \Delta\varphi \tag{4.35}$$

$$\rho_{\mathrm{Pred,q}} = \left\lfloor \frac{\rho_{\mathrm{Pred}}}{\Delta\rho(\varphi_{\mathrm{Pred,q}}, w_{\mathrm{C}}, h_{\mathrm{C}})} + \frac{\Delta\rho}{2} \right\rfloor \cdot \Delta\rho(\varphi_{\mathrm{Pred,q}}, w_{\mathrm{C}}, h_{\mathrm{C}}) \tag{4.36}$$

Spatial prediction using intercept representation Similarly, for the intercept representation, where the partitioning parameters of the neighboring block are given by the intercept coordinates $\boldsymbol{A}_{\mathrm{N}} = [\ P_{\mathrm{N},0} \quad P_{\mathrm{N},1}\ ]$, the predicted parameters $P_{\mathrm{Pred},0}$ and $P_{\mathrm{Pred},1}$ are calculated by finding the intersection points of the extrapolated partitioning line with the block boundary. The coordinates of $P_{\mathrm{Pred},0} = \begin{bmatrix} x_{\mathrm{Pred},0}, & y_{\mathrm{Pred},0} \end{bmatrix}^{\mathrm{T}}$ and $P_{\mathrm{Pred},1} = \begin{bmatrix} x_{\mathrm{Pred},1}, & y_{\mathrm{Pred},1} \end{bmatrix}^{\mathrm{T}}$ can be calculated using the following Equations (4.37), where a straight segment of the block boundary is given by $\boldsymbol{A}_{\mathrm{Block},i} = \begin{bmatrix} P_{\mathrm{B},0} & P_{\mathrm{B},1} \end{bmatrix}$ with $i \in \{0 \ldots 3\}$:

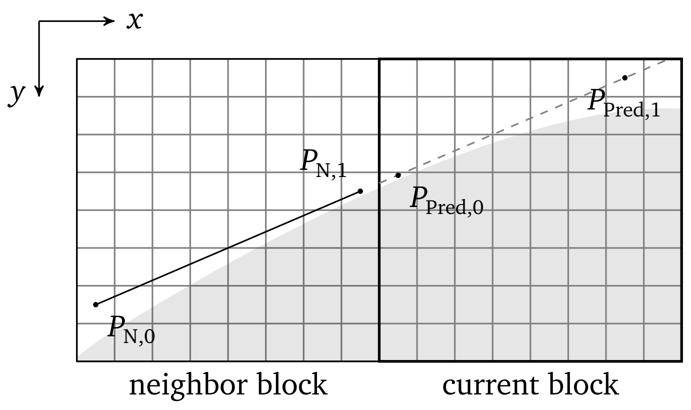


Figure 4.26 Exemplified spatial prediction of the partitioning parameters $\boldsymbol{A}_{\mathrm{C}} = [\ P_{\mathrm{Pred},0} \quad P_{\mathrm{Pred},1}\ ]$ for the current block, based on the partitioning parameters of the neighbor block given by $\boldsymbol{A}_{\mathrm{N}} = [\ P_{\mathrm{N},0} \quad P_{\mathrm{N},1}\ ]$. The partitioning line of the neighbor block is extrapolated and the intersections with the first column and first row of samples are determined. Note that $P_{\mathrm{Pred},0}$ and $P_{\mathrm{Pred},1}$ have not been quantized to the sample-grid.

$$\left.\begin{aligned} x_{\mathrm{Pred},k} &= \frac{\left(x_{\mathrm{N},0}y_{\mathrm{N},1} - x_{\mathrm{N},1}y_{\mathrm{N},0}\right)\left(x_{\mathrm{B},0} - x_{\mathrm{B},1}\right) - \left(x_{\mathrm{B},0}y_{\mathrm{B},1} - x_{\mathrm{B},1}y_{\mathrm{N},0}\right)\left(x_{\mathrm{N},0} - x_{\mathrm{N},1}\right)}{\left(x_{\mathrm{N},0} - x_{\mathrm{N},1}\right)\left(y_{\mathrm{B},0} - y_{\mathrm{B},1}\right) - \left(y_{\mathrm{N},0} - y_{\mathrm{N},1}\right)\left(x_{\mathrm{B},0} - x_{\mathrm{B},1}\right)} \\ y_{\mathrm{Pred},k} &= \frac{\left(x_{\mathrm{N},0}y_{\mathrm{N},1} - x_{\mathrm{N},1}y_{\mathrm{N},0}\right)\left(y_{\mathrm{B},0} - y_{\mathrm{B},1}\right) - \left(x_{\mathrm{B},0}y_{\mathrm{B},1} - x_{\mathrm{B},1}y_{\mathrm{N},0}\right)\left(y_{\mathrm{N},0} - y_{\mathrm{N},1}\right)}{\left(x_{\mathrm{N},0} - x_{\mathrm{N},1}\right)\left(y_{\mathrm{B},0} - y_{\mathrm{B},1}\right) - \left(y_{\mathrm{N},0} - y_{\mathrm{N},1}\right)\left(x_{\mathrm{B},0} - x_{\mathrm{B},1}\right)} \end{aligned}\right\}, \ k \in \{0,1\} \tag{4.37}$$

Since $x_{\mathrm{Pred},k}$ and $y_{\mathrm{Pred},k}$ as calculated by Equation (4.37) can specify non-integer coordinate points or coordinate points not contained in the code book of valid GEO parameters, quantization is again applied to generate the final predicted coordinate pairs:

$$\left.\begin{aligned} x_{\mathrm{Pred,q},k} &= \left\lfloor \frac{x_{\mathrm{Pred},k}}{\Delta w} + \frac{\Delta w}{2} \right\rfloor \cdot \Delta w \\ y_{\mathrm{Pred,q},k} &= \left\lfloor \frac{y_{\mathrm{Pred},k}}{\Delta h} + \frac{\Delta h}{2} \right\rfloor \cdot \Delta h \end{aligned}\right., \ k \in \{0,1\} \tag{4.38}$$

The block boundary is defined by the four vertical and horizontal line segments given in Table 4.8.

Therefore, a total of four intersection points are calculated. Intersection points whose coordinates lie outside of the current block are dismissed. Further, for purely vertical or horizontal geometric partitions of the neighboring block, Equation (4.37) can be avoided and the calculation of intersections is greatly simplified. Conceptually, the order of the coordinates $P_{\mathrm{Pred},0}$ and $P_{\mathrm{Pred},1}$ is irrelevant. For statistical analysis performed in later chapters however, the convention shall be followed that the coordinate $P_{\mathrm{Pred},0}$ always refers to the coordinate closest to the neighboring block that is the source of the prediction.

Partitioning predictor candidate list Similar to spatial motion vector prediction as reviewed in Section 2.4, predetermined spatial locations along the border of the coding block

Block boundary segment	$\boldsymbol{A}_{\text{Block},i} = \begin{bmatrix} P_{\text{B},0} & P_{\text{B},1} \end{bmatrix}$
left	$\boldsymbol{A}_{\text{Block},0} = \begin{bmatrix} 0 & 0 \\ 0 & h-1 \end{bmatrix}$
top	$\boldsymbol{A}_{\text{Block},1} = \begin{bmatrix} 0 & w-1 \\ 0 & 0 \end{bmatrix}$
right	$\boldsymbol{A}_{\text{Block},2} = \begin{bmatrix} w-1 & w-1 \\ 0 & h-1 \end{bmatrix}$
bottom	$\boldsymbol{A}_{\text{Block},3} = \begin{bmatrix} 0 & w-1 \\ h-1 & h-1 \end{bmatrix}$

Table 4.8 Line segment parameters of the block boundary given by coordinates $P_{\text{B},0}$ and $P_{\text{B},1}$ for a block of size $w \times h$.

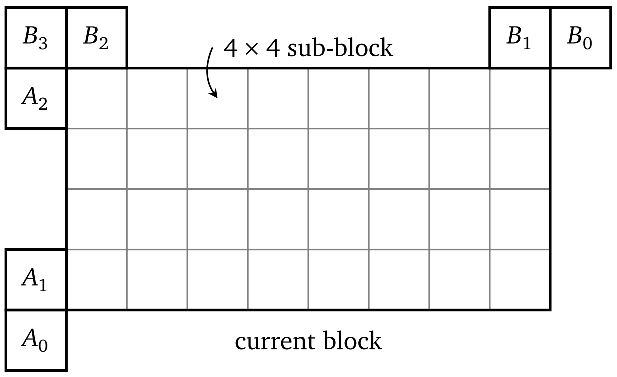


Figure 4.27 Visualization of the spatial positions used to determine the presence of neighboring GEO blocks.

are checked for the occurrence of spatially neighboring GEO blocks. These positions are labeled A_0, A_1 , A_2 for the neighbor positions at the left block boundary, and B_0, B_1, B_2, B_3 for the neighbor positions at the top block boundary. The spatial context, e.g. the meaning of each label is listed in Table 4.9a. Each position as exemplified in Figure 4.27 relates to a 4×4 sub-block, which is the granularity of information storage in VVC.

The predictor candidate list shall be limited to a maximum number of two candidates such that the selected candidate can be signaled using a single flag. It is also considered unlikely that more than two spatial predictors will be available for a block, given the GEO cluster analysis performed on the already coded data.

The spatial positions are checked in the order given by Table 4.9b. First, spatial positions close to the top-left corner of the coding block are checked, followed by spatial positions close to the bottom-left and top-right corner. If the predictor candidate list already contains 2 valid predictors, the process is terminated, and no further candidates are added to the list.

If the list contains less than 2 entries and if a neighboring block at these locations is using the GEO coding mode, the predicted parameters are computed and quantized to the reso-

Spatial position	Label
left-below	A_0
left-bottom	A_1
left-top	A_2
top-right above	B_0
top-right	B_1
top-left	B_2
top-left above	B_3

(a) Spatial context and associated label.

Derivation order	Label
1	A_2
2	B_3
3	B_2
4	A_1
5	B_1
6	A_0
7	B_0

(b) Derivation order for spatial candidates.

Table 4.9 Spatial positions used for the derivation of GEO predictor candidates.

lution of the code book for the given current block as detailed above. If the predicted and quantized parameters are contained within the code book and not yet part of the predictor candidate list, they are added to the predictor candidate list successively.

The selection of the predictor candidate for the current coding block is a task performed by the encoder through RDO. The predictor is signaled by an index $p \in \{0, 1\}$ which is translated to a binary code word. The resulting bin is then context coded. If no predictor candidate has been determined from the neighborhood or if the predictor candidate list contains less than two valid entries, the predictor candidate list is filled with the two most probable GEO splits, as determined for the Huffman coding in Experiment 4.6. This ensures that always 2 prediction options are available and the prediction mode flag (see Figure 4.18) is not wasted.

Partition prediction error coding It has been established in the analysis of the coded data that the extrapolated partitioning from a neighboring block only matches the optimal partitioning in a few cases. It is therefore reasonable to introduce the concept of partition prediction error coding, which – in analogy to motion vector difference coding – allows the coder to signal difference values in order to update the predicted partitioning parameters. This can also be seen as a refinement coding of the predicted partitioning. The coding method for this refinement scheme is visualized in Figure 4.28. First, an index value is signaled, mapping to the partition predictor candidate list. This is followed by the coding of an optional prediction error term that modifies the predicted partitioning.

As stated in Equation (4.28), a prediction error term $\boldsymbol{A}_{\mathrm{E}}$ is added to the predicted partitioning $\boldsymbol{A}_{\mathrm{Pred}}$ to obtain the final partitioning parameters. Depending on the chosen representation, the prediction error term $\boldsymbol{A}_{\mathrm{E}}$ modifies the partitioning in distinct ways:

$$\boldsymbol{A}_{\mathrm{C,AD}} = \begin{bmatrix} \varphi_{\mathrm{Pred,q}} & \rho_{\mathrm{Pred,q}} \end{bmatrix} + \begin{bmatrix} m \cdot \Delta\varphi & n \cdot \Delta\rho(\varphi_{\mathrm{Pred,q}}, w, h) \end{bmatrix} \tag{4.39}$$

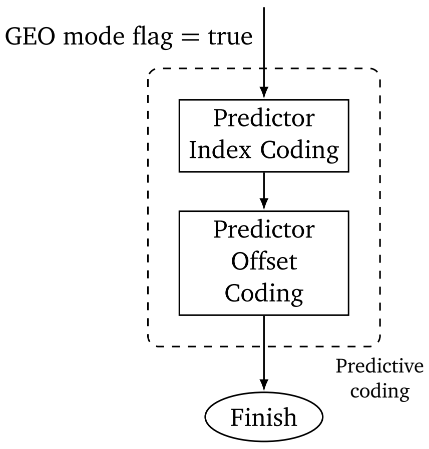


Figure 4.28 Coding method for the prediction mode branch of the geometric partitioning information. First, a predictor (spatial / temporal) is signaled, followed by an (optional) refinement.

$$\boldsymbol{A}_{\mathrm{C,BI}} = \begin{bmatrix} \boldsymbol{P}_{\mathrm{Pred,q,0}} & \boldsymbol{P}_{\mathrm{Pred,q,1}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{D}(m, P_{\mathrm{Pred,q,0}}) & \boldsymbol{D}(n, P_{\mathrm{Pred,q,1}}) \end{bmatrix} \tag{4.40}$$

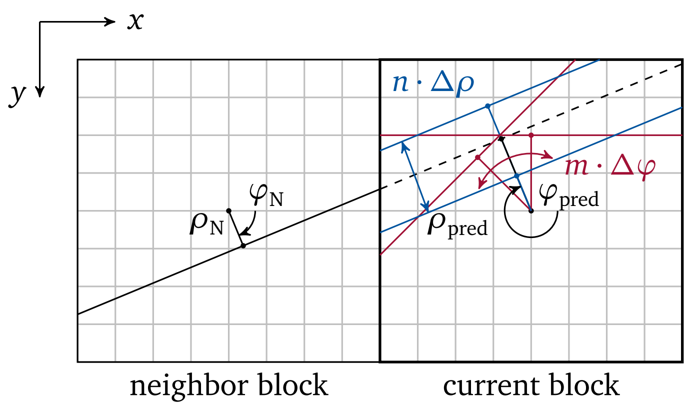
Equation (4.39) describes the prediction update process for the angle-distance representation. In this case, two integer parameters m and n are used to modify the partitioning. The parameters are multiplied with the quantization step sizes $\Delta\varphi$ and $\Delta\rho$ for the angle and distance, respectively. This ensures that the final partitioning parameters $\boldsymbol{A}_{\mathrm{C,AD}}$ are also contained in the partition code book. The effect of this refinement can be seen in Figure 4.29a. The variation of the angle φ causes a change of direction with respect to the predicted partitioning line, whereas the variation of the distance ρ causes a shift parallel to the prediction partitioning line. Further, the condition $\rho_{\mathrm{pred}} + n \cdot \Delta\rho(w, h, \varphi) < 0$ may occur, indicating a 180° flip of the angle φ_{C}. If both parameters m and n are non-zero, the tilting and shifting effects on the partitioning line are combined.

Equation (4.40) describes the prediction update process for the intercept representation. Here, the two integer parameters m and n cause a shift of the intercept points along the block boundary. This is visualized in Figure 4.29b. Comparing this method of prediction error coding with the angle-distance representation, it becomes clear that the intercept approach might be beneficial for approximating a smooth contour, as only a single parameter m or n, relating to the start- or end-point of the partitioning line, needs to be coded while the other is parameter is zero or close to zero. This representation fits the observed cases, where the partitioning is connected across multiple blocks.

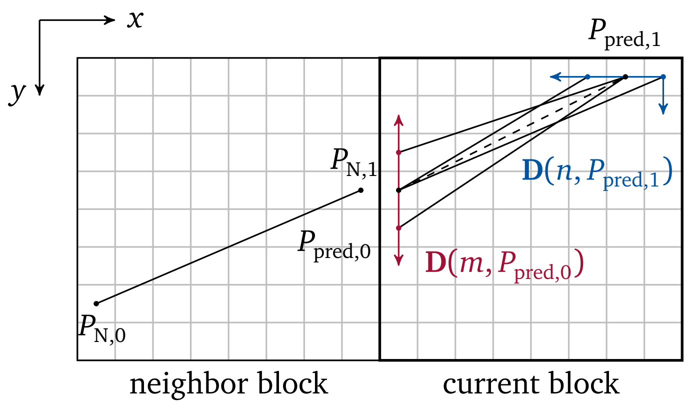
In the following experiments, the refinement parameters m an n are coded using a combination of a context coded flag for m and a context coded flag for n, indicating whether the value of m or n is non-zero, and a separate coding of the remainder value, e.g. the value of $m-1$ and $n-1$. Here, different coding schemes are tested for the remainder:

- Unlimited refinement bypass coding of the remainder using an Exp-Golomb binarization of order $k = 1$, e.g. $C_{\mathrm{EG},1}(|m|-1)$ and $C_{\mathrm{EG},1}(|n|-1)$, respectively.

- Limited refinement bypass coding of the remainder using a fixed length code, indicating $m = \pm 1$ and $n = \pm 1$, e.g. $C_{\mathrm{FL},1}(m > 0)$ and $C_{\mathrm{FL},1}(n > 0)$, respectively.



(a) Effect of partition prediction difference coding for angle-distance representation.



(b) Effect of partition prediction difference coding for intercept representation.

Figure 4.29 Visualization of partition prediction difference coding for different parametrizations. Note that for the same coded offsets m and n, significantly different geometric partitions are generated.

- No refinement coding. One of the two predictors is signaled using the aforementioned candidate flag.

The Exp-Golomb code of order $k = 1$ has been chosen for the unlimited refinement coding experiment since a geometric distribution of m and n is expected, supported by the findings shown in Figure 4.24. Further, no restriction on the range of m or n has been applied. Combined with the flag-based coding of the predictor index, this means that at least 3 bit are necessary to signal the continuation of a neighboring partitioning line. It is to be noted however, that the predictive coding approach is in competition with the explicit coding (see Figure 4.18) and the final coding mode is determined based on the lowest achievable, estimated rate. Therefore, refinement values of $|m| > 6$ or $|n| > 6$ will never be chosen by the encoder, as signaling a value larger or smaller than ± 6 would require 6 bit for the Exp-Golomb code of order $k = 1$, exceeding the bit-budget compared to explicit coding. It is noted that with this given restriction due to the RDO, a binarization of the remainder for finite alphabets, such as Golomb-Rice coding, could further save 1 bit.

The limited refinement and no refinement coding experiments can be seen as a check, whether a simpler approach provides any RD benefit.

Coding performance for angle-distance representation In total, three experiments are conducted in this subsection:

- Experiment 4.9 investigates the coding performance of spatial prediction with full refinement coding.
- Experiment 4.10 investigates the coding performance of spatial prediction with limited refinement coding.
- Experiment 4.11 investigates the coding performance of spatial prediction with no refinement coding. The prediction is used *as is*.

First, Experiment 4.9 investigates the performance of spatial prediction using full refinement coding. The detailed coding results for this setup are shown in Table A.11. Compared to the identical setup in terms of number and distribution of geometric splits used for Experiment 4.2 without prediction, only a slight overall coding gain is observed, improving from −0.45 % to −0.49 %. For the Ultra high definition (UHD) classes A1 and A2, coding efficiency improves comparably more, from −0.30 % to −0.35 % and −0.36 % to −0.45 %, respectively. More insights into this observed low impact on coding efficiency can be gained from analyzing the actual spatial prediction mode usage, listed in Table A.12. For each class, the relative mode usage given by the percentage of GEO pixels that are coded with spatial prediction is measured. The table reveals that only 6.7 % of all coded GEO blocks are using spatial prediction, averaged over all classes and QPs. Further, considering that the rate cost for explicitly coded GEO partitions has been increased by 1 bit due to the signaling of the additional GEO mode flag, it can be stated that not enough blocks are utilizing the spatial prediction mode to significantly improve coding efficiency. It is to be noted that for the screen content class F the highest usage of spatial prediction is measured. This is largely attributed to the sequence *SlideEditing* which contains a computer screen capture with vertical scrolling motion. This scrolling motion is partitioned almost exclusively by horizontal GEO

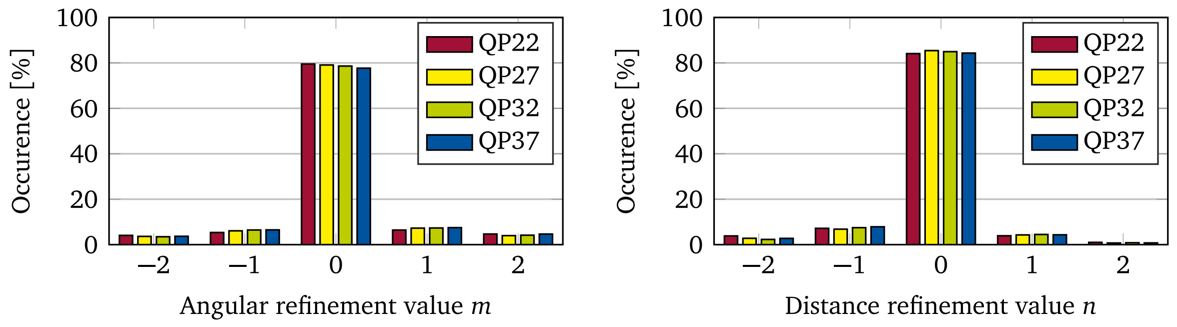


Figure 4.30 Distribution of angular and distance refinement values m and n, measured for Experiment 4.9 using spatial prediction.

modes that can be very well predicted across multiple adjacent blocks. The spatial prediction mode was utilized for 24 % of all pixels coded with GEO in this sequence. For natural video content, the highest spatial prediction mode usages where measured for the *BQMall* and *RaceHorses* sequence at 9 % and 9.21 %. The lowest usage of spatial prediction can be measured for the *Campfire* sequence which contains footage of a campfire next to a group of people in low lighting conditions. Here, the spatial prediction mode usage was measured at 2 %, averaged over all QPs. Non-surprising, dynamic texture content such as fire with its undefined object boundaries is not well suited for spatial prediction.

The distribution of refinement values for m and n is another interesting aspect to analyze, giving further insight in how the coding scheme could be further optimized and how the encoder trades improved segmentation precision against distortion for the given operating point. The distribution for Experiment 4.9, again measured by occurrence in terms of relative area coded, for the angular refinement variable m and the distance refinement variable n is shown in Figure 4.30, grouped by QPs. The statistics reveal that in 80 % of the cases the encoder does not choose to code a refinement value $m \neq 0$ or $n \neq 0$. This statistical behavior is also largely independent of the QPs, with the slight tendency of 2-3 % to code a refinement value for QP37 compared to QP22.

Comparing this result with the statistics provided in Figure 4.22, it can assessed that in the overall few cases (∼15 %) where a *connected* partitioning is observed, 6.7 % of all GEO blocks will be coded in a mode that was designed to handle such specific cases. In summary, the angle-distance representation using spatial prediction with full refinement coding for this representation does not provide significant coding gain, likely due to only very few blocks that have characteristics suiting the model and a signaling scheme that requires too many bits to code an update.

Experiment 4.10 provides coding results for a spatial prediction where a constraint on the maximum and minimum refinement value has been enforced, effectively requiring less bits to signal the refinement values. Detailed coding results for this setup are given in Table A.13. The maximum absolute refinement value for this experiment was set to be $|m| = 1$ and $|n| = 1$, coded by a single flag, depending on the value of the non-zero flag as mentioned before. This means that the predicted partitioning line can be tilted by $\pm 11.25°$ or shifted by $\pm\Delta\rho(\varphi_{\text{Pred,q}}, w, h)$, which for example ranges from 1 luma sample for an 8×8 block to 22 luma samples for a 128×128 block at $\varphi_{\text{Pred}} = 45°$. The overall coding gain slightly drops from −0.49 % for Experiment 4.9 to −0.48 % for Experiment 4.10. Thus, no major impact is observed. This experiment validates that the refinement coding method itself is not the main contributing factor.

Lastly, no refinement coding on top of the spatial predictor is tested in Experiment 4.11. In this setup, one of two different predictor candidates can be signaled using a flag. Therefore, if the GEO mode is selected, only 2 bit are sufficient to signal the geometric split – a first flag to indicate that spatial prediction is used and a second flag to indicate the predictor candidate. The coding efficiency shown in Table A.14 shows that this method does not provide any benefit over unlimited or limited refinement coding. Likely, as it is assumed for Experiments 4.9 and 4.10, there are too few cases where the optimal partitioning of the current block is the straight-line continuation of the partitioning of the neighboring block.

Coding performance for intercept representation The refinement coding experiment is repeated for the intercept representation of geometric partitioning parameters, since the refinement using this representation modifies the partitioning differently compared to the angle-distance representation:

- Experiment 4.12 investigates the coding performance of spatial prediction with full refinement coding.
- Experiment 4.13 investigates the coding performance of spatial prediction with limited refinement coding.
- Experiment 4.14 investigates the coding performance of spatial prediction with no refinement coding. The prediction is used *as is*.

Rather than tilting or shifting the partitioning line around its predicted position, the refinement by moving a line coordinate along the block boundary has a combined effect. This is visualized in Figure 4.29b. Especially if the predicted coordinate closest to the neighboring block is kept constant, this representation seems to be more suitable to approximate connected object boundaries.

In the first Experiment 4.12, again no constraints are imposed on the refinement variables m and n. The basic partitioning setup from Experiment 4.5 is reused, with a varying number of 266-352 partitions per block, requiring 9 bit for explicit signaling. This is the bit budget that can be exploited for spatial prediction. Again, subtracting the 1 bit used to indicate the first or second predictor candidate and the 2 bit required for signaling a non-zero value of m and n, this leaves 6 bit to signal both refinement values. As for Experiment 4.12, an Exp-Golomb code of order $k = 1$ is utilized to binarize m and n. The coding results given in Table A.15 show that the overall coding efficiency in this representation increases very slightly from −0.42 % for Experiment 4.5 to −0.44 % for Experiment 4.12. As no big change in coding efficiency is observed, a similar conclusion as for the angle-distance representation can be drawn. Although the spatial prediction mode utilization is higher, measured at 9 % for the intercept representation as shown in Table A.16, compared to 6.7 % for the angle-distance representation, the higher prediction mode usage does not manifest in significant coding gain.

Interestingly, the distributions of refinement values for m and n, where m relates to the shift of the first coordinate P_0 and n to the shift of the second coordinate P_1, are clearly different from each other. This is shown in Figure 4.31, grouped by QPs and measured over the

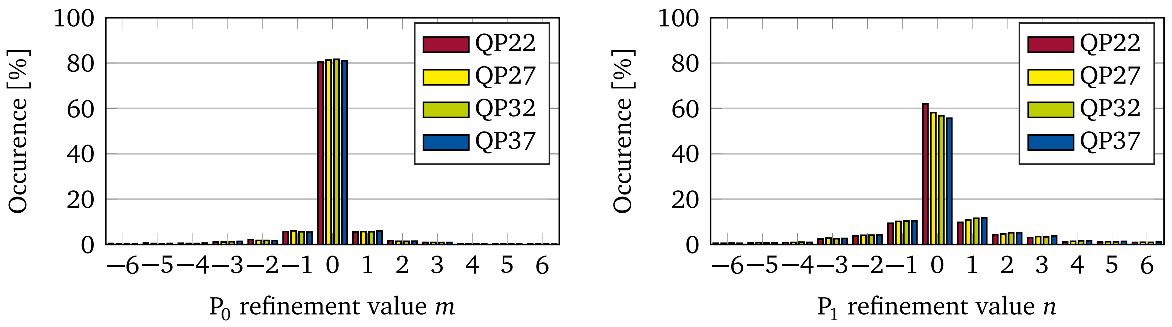


Figure 4.31 Distribution of coordinate refinement values m and n, measured for Exp. 4.12 using spatial prediction.

entire CTC test set. It can be seen that for those blocks that are spatially predicted, in roughly 80% of the cases no refinement is coded for the first coordinate P_0, which per definition is the coordinate closest to the neighboring block. For the second coordinate P_1 however, a non-zero refinement value n is coded in 40 of the cases. This supports the assumptions of smooth and connected object boundaries. However, since the overall usage of spatial prediction is too low, no significant improvement in coding efficiency can be reported.

Experiment 4.13 shows the corresponding coding results for a spatial prediction setup using the intercept representation in Table A.17, where a constraint on the maximum and minimum refinement values has been enforced. These results also do not show any change in coding efficiency using a limited refinement coding.

Lastly, Experiment 4.14 shows the coding results without any refinement coding using the intercept representation in Table A.18. As it is the case for the angle-distance representation, there is no benefit compared to the variants of using an unlimited or limited refinement coding.

4.3.2.2 Temporal Prediction

Significant correlation exists between temporally adjacent pictures in a video sequence. This inter-frame redundancy is extensively exploited by motion compensation and motion prediction to improve coding efficiency. The basic concept of inter-picture prediction using motion compensation in VVC has already been reviewed in Section 2.4. In this sub-section, the question shall be answered, how the temporal correlation can also be exploited for the purpose of improved geometric partition coding. The resulting algorithm is a refined version of the temporal projection developed and published by the author in [BHW17] and [BSW18].

Partitioning data storage In contrast to spatial prediction of the partitioning parameters, temporal prediction requires the availability of the partitioning information for each reference picture. Therefore, the overall process bears strong similarities to TMVP in HEVC or VVC. Table 4.10 lists the syntax elements for HEVC and VVC that are stored in the motion buffer of each reference picture, including the memory requirements for a naive implementation. As the additional memory required for storing the motion data can be significant, motion data compression schemes are employed in both standards. In HEVC, the 4×4 minimum size PU grid is sub-sampled to a size of 16×16 luma samples. In VVC, the constraint on the minimum motion data storage unit size is relaxed to 4 × 4 luma samples. Instead, a rounding operation is applied to the motion vectors to enable a floating-point representation of each motion vector using a 6 bit mantissa and a 4 bit exponent. Additionally, the information is stored whether the motion vector is a classical motion vector, or a displacement vector used for the Intra block copy (IBC) mode of VVC.

In order to enable temporal prediction of the partitioning, the absolute spatial position of each partitioning line must be known for the pictures that were already coded. Therefore, the partitioning information must be stored with each reference picture. As it is the case for temporal motion vector prediction, the memory requirement can be substantial. Next to the coded representation of the partitioning, which – depending on the granularity of the partitioning – may require 7 to 9 bit per GEO coding block, additional partitioning information must be known for the prediction process. This is explained by the fact that the 7 to 9 bit for the GEO parameters map to partitioning line parameters (angle and distance or coordinates) for a given block size. The block partitioning structure however is not stored with each reference picture and there is no information easily available to associate a motion data storage unit with a coding unit. Therefore, in order to fully reconstruct the GEO partitioning from each motion data storage unit, an additional spatial reference for the partitioning parameters is needed. This problem could be solved regardless of the chosen GEO parametrization by storing the coordinates of two points that are traversed by the partitioning line per motion data storage unit, given in absolute coordinates relative to the top-left sample of the coded picture. Assuming that the maximum supported resolution of the coder is beyond 8K UHD, 15 bit per coordinate component are required. This would require 60 bit per motion data storage unit, not utilizing the fact that the geometric partitioning and the block partitioning structure are quantized. Therefore, it is reasonable to take advantage of the already available, quantized representation of the geometric partitioning and store additional information relating to the block structure. Here, two distinctions can be made:

- For the angle-distance representation, the corresponding absolute or relative location of the GEO block coordinate system's origin needs to be made available for each motion data storage unit. Together with an angle and a distance, this sufficiently describes an infinitely extending partitioning line.
- For the intercept representation, the size and position of the corresponding GEO block for each motion data storage unit needs to be known. This allows the reconstruction of absolute line coordinates.

In a naive implementation, this data can be stored within each motion data storage unit next to the motion information. The position of each GEO coding unit can be coded relative to the fixed CTU raster. For a CTU raster of 128 × 128 luma samples in VVC and a 4 × 4 block granularity, this would require 5 bit for the x and 5 bit for the y-component. Therefore, for the angle-distance representation, additional 10 bit are required to store the position of the GEO block coordinate system's origin.

For the intercept representation, the size of the GEO block also needs to be known. Since there are 13 different possible combinations for the width and height of a GEO coding block, ranging from 128 × 128 luma samples to 8 × 8 luma samples in VVC, a 4 bit coded representation is sufficient. Therefore, 14 bit for the partitioning information per motion data storage unit are required for the intercept representation. In summary, if a 7bit coded representation

	HEVC	VVC
Motion data storage unit	16 × 16	4 × 4
Motion data resolution	16 bit	18 bit
Motion data storage per reference picture list	• 16 bit motion vectors • 4 bit reference indices • 1 bit reference picture list utilization flag	• 10 bit motion vectors (6 bit mantissa, 4 bit exponent) • 4 bit reference indices • 1 bit reference picture list utilization flag
Additional data	—	• 1 bit IBC flag

Table 4.10 Motion data storage comparison for HEVC and VVC.

for the GEO partitioning is assumed, the angle-distance representation would require 17 bit of static memory to store the partitioning information per motion data storage unit. The intercept representation would require 21 bit of static memory per motion data storage unit for temporal prediction.

It is noted that alternatively, the entire MTT of each CTU could be efficiently stored. By comparison, a pointer-based implementation using dynamic memory would require $\frac{n}{2}4P + \frac{n}{2}D$ bits for storing a quad tree, where n is the number of nodes of the tree, P is the number of bits required to store the child node pointers and D the number of data bits residing in the leaf node. Assuming the worst case, where a CTU is fully partitioned into 8 × 8 GEO blocks and the geometric partitioning is coded using 7 bit, this would require an equivalent of 32.875 bit dynamic memory per motion data storage unit. With overall larger block partitions, due to the video characteristics or a higher QP for example, this requirement however may become significantly less.

Overall, it can be stated that additional memory is required for performing temporal prediction of the GEO parameters. However, using efficient storage techniques, the memory requirements for a set of GEO parameters are comparable in size to the memory requirements of a single motion vector.

Temporal projection of partitioning data The motion of an object covering multiple pictures can be modeled using a *motion trajectory*. This motion trajectory is an estimate of the three-dimensional function that describes the translational movement of an object. It is clear that from the perspective of the video coder, the true motion trajectory of an object in the world coordinate system remains unknown, since only the apparent motion, which is the projection of the object onto the two-dimensional plane, sampled at regular time instances can be measured and estimated. An example of a motion trajectory is shown in Figure 4.32. The object, visualized by the gray circle, is moving from the left side of the frame to the

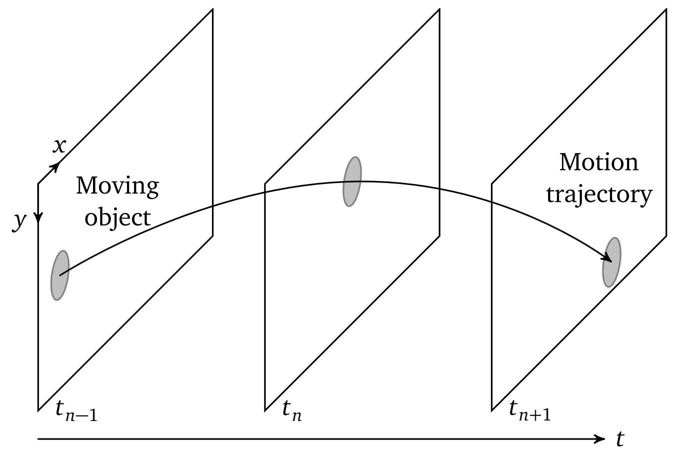


Figure 4.32 Example of spatial prediction of the geometric partitioning from the local neighborhood of a coding block. The prediction, which is a linear continuation of the partitioning boundary of the neighboring block into the current block, is indicated by the dashed lines. The hypothetical object is shaded in gray.

bottom-right of the frame over the duration of three pictures. However, this translational motion may not be a strictly linear motion, as indicated in the Figure. Further, more complex motion could also be superimposed onto the translational motion, such as rotation, zoom or shear. These can be modeled by higher-order motion models, such as an affine motion model.

The estimation of the motion trajectory is therefore a task handled by the underlying encoder. The reference encoders for VVC and HEVC, for example, perform block matching to estimate translational motion vectors under the assumption that the motion is linear. Block matching between two pictures can therefore be seen as a first order, piece-wise approximation of the motion trajectory. This is also reflected by the way temporal motion vector prediction is performed:

1. In HEVC and VVC, regular TMVP is performed by estimating the motion vector of the current block, based on the motion field of coded reference pictures, sampled at collocated spatial positions denoted as C_0 and C_1 (see Figure 2.7a). The motion vector predictor is scaled according to the temporal distances between the current picture and the coded reference pictures. This scaling process can also be viewed as a forward projection of the motion vector encountered at C_0 or C_1 of the reference picture motion field.

2. In VVC, SbTMVP is performed by estimating the motion of multiple sub-blocks based on the motion vector field of coded reference pictures, sampled by a sub-block sized grid at a spatial location that has been derived through an additional motion shift. This motion shift is obtained from spatially neighboring motion vectors. Thus, the overall process can be viewed as a first step of a backward projection using the neighboring motion vector to obtain the shifted position and a second step, which is the identical forward projection step as for regular TMVP.

Under the assumption that moving objects in a video sequence remain rigid, meaning that the object boundaries are not subject to significant deformations, the block partitioning along

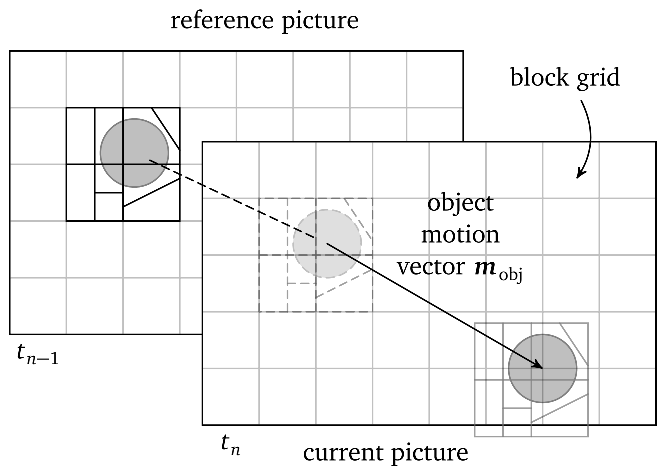


Figure 4.33 Example of temporal prediction of the partitioning using the object motion given by $\boldsymbol{m}_{\text{obj}}$. A very similar concept is used in SbTMVP in VVC.

these object boundaries can be expected to remain static. This allows, in analogy to temporal motion vector prediction, the temporal prediction of the partitioning parameters through a motion compensation step. The partitioning parameters, which are to be stored along with the motion vector field of each reference picture in the decoded picture buffer, can therefore be re-used by applying a motion shift. This motion shift is conceptually identical to the motion of the object. Figure 4.33 visualizes the process of motion compensated partitioning.

The object in this case is segmented by a combination of rectangular block partitioning and geometric block partitioning as indicated. The partitioning that has been coded for the object in the reference picture can be transferred to the current position of the object using the motion vector $\boldsymbol{m}_{\text{obj}}$ that is associated with the object. It can be seen in Figure 4.33 that it is unlikely that the identical, compensated partitioning can be superimposed onto the block (e.g. the CTU) grid of the current picture such that identical sized coding blocks are selected for the current picture. Therefore, an additional process is required to estimate the closest matching block partitioning for the current picture.

At this point, it should be noted that, although temporal prediction of the rectangular block partitioning could also be achieved, only the temporal prediction of geometric block partitioning parameters is considered. Due to the comparably low signaling cost of the rectangular block partitioning using the MTT, no significant coding gain is expected from predicting the rectangular partitioning. Further, the SbTMVP mode already provides a very similar functionality on the prediction block level. SbTMVP transfers the entire sub-block motion of a reference picture to the current picture, which implicitly contains the partitioning information of the reference picture.

The predicted partitioning parameters $\boldsymbol{A}_{\text{Pred,BI}}$ can be easily computed for the intercept representation by addition of the object motion vector m_{obj} to the line coordinates given by $\boldsymbol{A}_{\text{T}} = [P_0\ P_1]$:

$$\boldsymbol{A}_{\text{Pred,BI}} = \boldsymbol{A}_{\text{T}} + [\ \boldsymbol{m}_{\text{obj}} \quad \boldsymbol{m}_{\text{obj}}\ ] \tag{4.41}$$

This shift of the partitioning parameters is also visualized in Figure 4.34a. Given the shifted partitioning parameters $\boldsymbol{A}_{\text{Pred,BI}}$, the final partitioning parameters for the current block can be computed according to Equation (4.37), by determining the intersection points of the partitioning line with the current block boundary.

For the angle-distance representation, a similar approach is taken by shifting the partitioning parameters using the motion vector $\boldsymbol{m}_{\text{obj}}$. For simplicity, the $(\varphi_{\text{T}}, \rho_{\text{T}})$ coordinate is converted to Cartesian coordinates and the motion vector $\boldsymbol{m}_{\text{obj}}$ is added to obtain a point $(x_{\text{shift}}, y_{\text{shift}})$, located on the shifted partitioning line:

$$\begin{bmatrix} x_{\text{shift}} \\ y_{\text{shift}} \end{bmatrix} = \begin{bmatrix} \rho_{\text{T}} \cos(\varphi_{\text{T}}) \\ \rho_{\text{T}} \sin(\varphi_{\text{T}}) \end{bmatrix} + \boldsymbol{m}_{\text{obj}} \tag{4.42}$$

The predicted angle of the partitioning line φ_{Pred} is identical to angle φ_{T} of the partitioning line relating to the collocated block, corrected by a 180° shift if necessary.

$$\varphi_{\text{Pred}} = \varphi_{\text{T}} + k\pi, \quad k \in \{0, 1\} \tag{4.43}$$

Then, the new predicted distance ρ_{Pred} can be determined based on the relative locations of the two block centers, where $\boldsymbol{x}_{\text{T}} = [x_{\text{T}}, y_{\text{T}}]^{\text{T}}$ specifies the center location of the collocated block and $\boldsymbol{x}_{\text{C}} = [x_{\text{C}}, y_{\text{C}}]^{\text{T}}$ the center location of the current block, as follows:

$$\begin{bmatrix} x_{\text{tmp}} \\ y_{\text{tmp}} \end{bmatrix} = \boldsymbol{x}_{\text{T}} + \begin{bmatrix} x_{\text{shift}} \\ y_{\text{shift}} \end{bmatrix} - \boldsymbol{x}_{\text{C}} \tag{4.44}$$

$$\varphi_{\text{tmp}} = \text{atan2}\left(\text{y}_{\text{tmp}}, \text{x}_{\text{tmp}}\right) \tag{4.45}$$

$$d_{\text{Pred}} = \cos\left(\left|\varphi_{\text{tmp}} - \varphi_{\text{Pred}}\right|\right) \sqrt{x_{\text{tmp}}^2 + y_{\text{tmp}}^2} \tag{4.46}$$

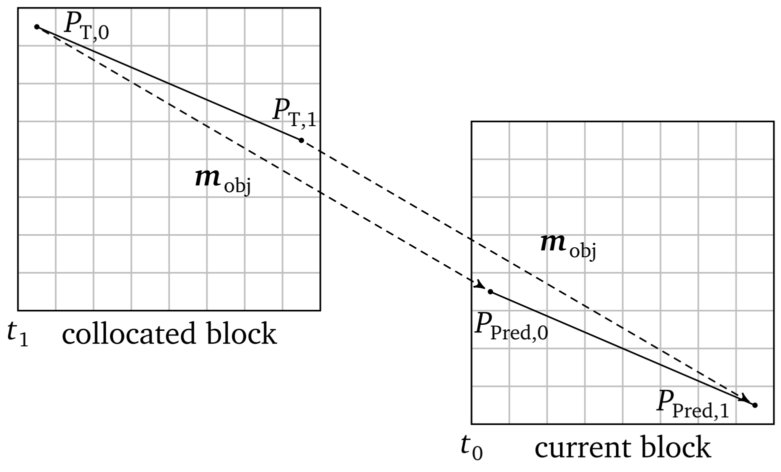
$$\rho_{\text{Pred}} = |d_{\text{Pred}}| \tag{4.47}$$

The sign of d_{Pred} indicates, whether the 180° shift applied to the predicted angle given in Equation (4.43) is necessary:

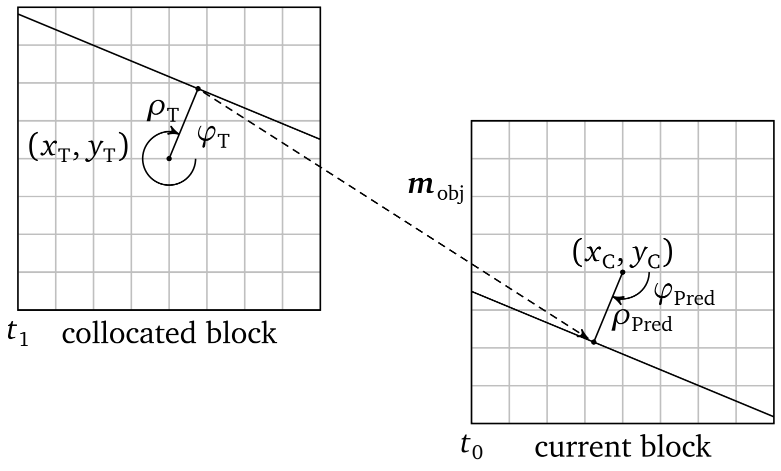
$$k = \begin{cases} 0 & d_{\text{Pred}} \geq 0 \\ 1 & d_{\text{Pred}} < 0 \end{cases} \tag{4.48}$$

For both representations, quantization as given in Equations (4.35), (4.36) and (4.38) is applied to the predicted parameters. From an implementation-based point of view and the derivations given above, it becomes clear that the intercept representation is preferred. Temporal prediction can be achieved through a simple addition of a scaled motion vector, whereas a change to Cartesian coordinates is required for the angle-distance representation.

At this stage, the question arises how the object motion $\boldsymbol{m}_{\text{obj}}$ can be estimated or generated from the motion vector field of the reference picture. Since the true motion of the object boundary is not known, a scanning process is employed that performs the projection for each motion vector encountered within a search range, centered at the collocated position. The projection process is a scaling operation based on the temporal distances of the involved pictures that is applied to each motion vector $\boldsymbol{m}_{\text{mv}}$ of the reference picture motion vector field. In this process, the reference picture shall occur at time t_1, measured by Picture order count (POC). The motion vector $\boldsymbol{m}_{\text{mv}}$ is referencing another reference picture at time t_2.



(a) Temporal prediction visualized for the intercept representation


ρ_{T}
$(x_{\mathrm{T}}, y_{\mathrm{T}})$
φ_{T}
$\boldsymbol{m}_{\mathrm{obj}}$
$(x_{\mathrm{C}}, y_{\mathrm{C}})$
φ_{Pred}
ρ_{Pred}
t_1 collocated block
t_0 current block

(b) Temporal prediction visualized for the angle-distance representation

Figure 4.34 Temporal prediction of GEO parameters by using the object / boundary motion given by the motion vector $\boldsymbol{m}_{\mathrm{obj}}$.

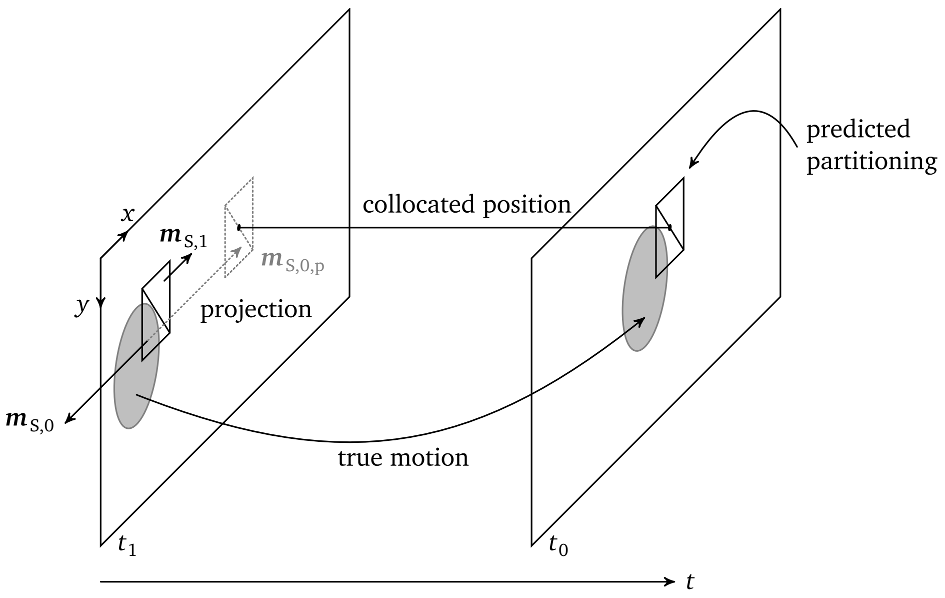


Figure 4.35 Visualization of the temporal prediction process through projection. The partitioning information is made available at the collocated position for the current CU by applying a spatial shift according the motion information.

Note that the picture at t_2 may precede or follow the picture at t_1 in output order, depending on the coding order of the coding scheme. The temporal distance between the two reference pictures shall be denoted t_b. The temporal distance between the current picture at time t_0 and the reference picture at time t_1 shall be denoted as t_d.

$$t_b = t_1 - t_2 \tag{4.49}$$

$$t_d = t_0 - t_1 \tag{4.50}$$

The projection process is now simply an inversion of each motion vector $\boldsymbol{m}_{mv}$ and a scaling according to the ratio of t_b and t_d:

$$\boldsymbol{m}_{obj} = -\frac{t_b}{t_d}\boldsymbol{m}_{mv} \tag{4.51}$$

This process is depicted in Figure 4.35. Based on the collocated position of the current block in the current reference picture, the motion vector field of the current reference picture is tested for the presence of additional geometric partitioning data. As established above, the partitioning data of each GEO coding unit is available for each 4 × 4 motion data storage unit. For each 4 × 4 unit that contains geometric partitioning data, the projection vector $\boldsymbol{m}_{obj}$ is derived according to Equation (4.51) and the compensation processes according to Equation (4.41) for the block-intercept representation or according to Equations (4.42) through (4.47) for the angle-distance representation are invoked. For a faster prediction process, detected duplicate combinations of geometric partitioning data and motion data can be skipped.

Since a geometrically partitioned block contains at least two different motion vectors $\boldsymbol{m}_{S,0}$ and $\boldsymbol{m}_{S,1}$ which are associated with each partition, two projection hypotheses $\boldsymbol{m}_{S,0,p}$ and

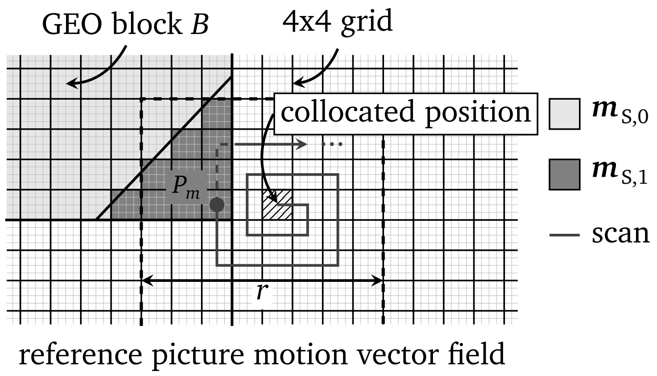

GEO block B
4x4 grid
collocated position
$\boldsymbol{m}_{\mathrm{S},0}$
$\boldsymbol{m}_{\mathrm{S},1}$
scan
P_m
r
reference picture motion vector field

Figure 4.36 Visualization of the scanning process using a spiral pattern. Projection is performed at each unique 4 × 4 storage unit location P_m containing geometric partitioning information.

$\boldsymbol{m}_{\mathrm{S},1,\mathrm{p}}$ are computed per GEO block. If it is assumed that the motion vectors $\boldsymbol{m}_{\mathrm{S},0}$ and $\boldsymbol{m}_{\mathrm{S},1}$ relate to different objects or to the motion of the foreground and the background, only one hypothesis $\boldsymbol{m}_{\mathrm{S},0,\mathrm{p}}$ or $\boldsymbol{m}_{\mathrm{S},1,\mathrm{p}}$ will coincide with the object motion. This can be tested by calculation whether the predicted partitioning parameters $\boldsymbol{A}_{\mathrm{Pred,BI}}$ or $\boldsymbol{A}_{\mathrm{Pred,AD}}$ are actually slicing the current block.

In Alternative temporal motion vector prediction (ATMVP) and SbTMVP only a fixed number of predetermined spatial positions are scanned for computing motion vector predictors. In ATMVP for example, the C_0 and C_1 positions, located at the bottom right and at the center of the current block, are tested for temporal prediction. In SbTMVP, the A_1, B_1, B_0 and A_0 positions are tested first to derive the initial shift vector. This simple scanning process is extended for the temporal prediction of GEO parameters, as it is unlikely to encounter a 4×4 unit containing the partitioning data with such few tested locations. Instead a scanning process is proposed which starts at the center position of the current block in the current reference picture. Then, spiraling outwards as indicated in Figure 4.36, multiple locations P_m are computed and each underlying 4×4 unit is checked for the following conditions:

1. If the scan position at P_m is contained within the current reference picture,
2. if the 4×4 unit is associated with an inter-predicted block,
3. if the 4×4 unit is not using IBC,
4. if the 4×4 unit contains valid GEO or TPM partitioning information,
5. if the motion vectors of the 4×4 unit and the partitioning data have not been tested yet.

If all conditions are met, the projection process is performed and if a valid partitioning has been computed, the predictor candidate is added to the partitioning predictor candidate list. In the following experimental simulations, a scan region of 32 rectangular spiral passes is used, which is equal to a square-shaped scan region size of $r = 256$ luma samples around the collocated position. In VVC and under the JVET CTC, the search area is therefore equal to the area covered by four CTUs when GEO is enabled.

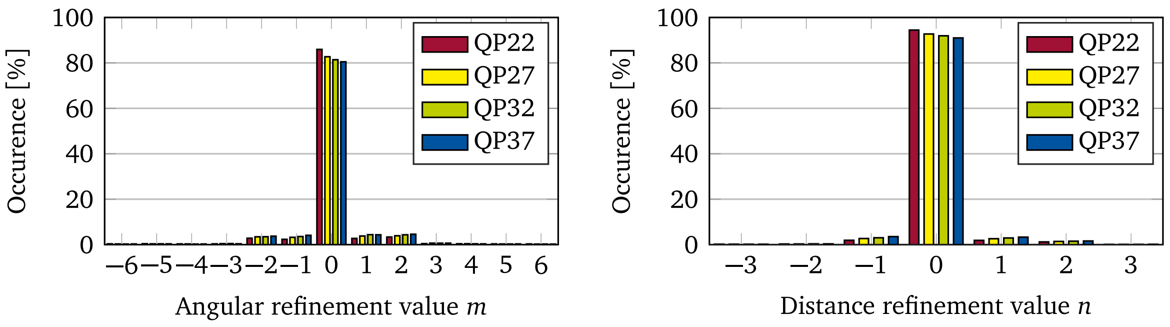


Figure 4.37 Distribution of angular and distance refinement values m and n, measured for Experiment 4.15 using temporal prediction.

Coding performance for angle-distance representation In analogy to the spatial prediction coding experiments, three coding experiments are conducted for temporal prediction that differ with regard to the level of refinement coding optionally applied to the predicted partitioning:

- Experiment 4.15 investigates the coding performance of temporal prediction with full refinement coding.
- Experiment 4.16 investigates the coding performance of temporal prediction with limited refinement coding.
- Experiment 4.17 investigates the coding performance of temporal prediction with no refinement coding. The prediction is used *as is*.

First, the angle-distance representation is tested for its coding performance using temporal prediction in Experiment 4.15. Again, the basic coding setup from Experiment 4.2 is used, where the explicit coding of the partitioning is performed using a fixed length code of 7 or 8 bit. For temporal prediction, one of two predictor candidates can be signaled using a flag. An unlimited refinement coding using an Exp-Golomb code of order $k = 1$ can optionally be signaled. The overall coding efficiency, shown in Table A.19, increases from −0.45 % for the reference Experiment 4.2 without prediction to −0.51 % for Experiment 4.15 with temporal prediction. For classes A1, A2, B and C, a nearly uniform improvement by approximately 0.06 % can be reported. For individual sequences, the highest coding efficiency improvement compared to Experiment 4.2 can be measured for the UHD sequences *CatRobot1, ParkRunning3* at −0.1 % and for the High definition (HD) sequences *Cactus* and *BQTerrace* at −0.1 % and −0.13 %, respectively.

The usage statistic of the temporal prediction mode shown in Table A.20 reveals that overall 27.8 % of all pixels coded with GEO are coded using the temporal prediction mode. This compares to 6.7 % for spatial prediction as measured in Experiment 4.12. Also, a stronger dependency of the QP can be observed. For the highest bitrates at QP22, the temporal prediction mode usage is 20.9 %, whereas for the lowest bitrates at QP37 usage increases to 31.9 %.

Figure 4.37 shows the occurrence of refinement values m and n of Experiment 4.15 that are applied to the predicted angle and distance per block, respectively. Similar as for spatial prediction, a refinement of the angular parameter is coded only for 20 % of the GEO pixels.

For the distance parameter however, a refinement value n is coded in only 5 to 10 % of all GEO pixels.

To test whether a better coding efficiency trade-off exists, limited refinement coding is conducted in Experiment 4.16. As for the case of spatial prediction, only the refinement values $m = \pm 1$ and $n = \pm 1$ can be signaled using a single flag per parameter. The detailed coding results are shown in Table A.21. Overall, coding efficiency for this setup increases marginally to −0.52 %, indicating that the additional bits spent for the unlimited refinement coding in Experiment 4.15 can be omitted.

In analogy to the spatial prediction experiments, the setup of not using any refinement coding on top of the temporal prediction is tested in Experiment 4.17. Coding efficiency, shown in Table A.22, compared to the unlimited refinement scheme increases marginally from −0.51 % to −0.53 %. This increase confirms the findings of the refinement statistics as shown in Figure 4.37. Since the encoder chooses a zero refinement of the angle or distance in most of the cases, these bits can be considered redundant. Combined with the larger usage of temporal prediction compared to spatial prediction, this results in a slight coding gain.

Coding performance for intercept representation

- Experiment 4.18 investigates the coding performance of temporal prediction with full refinement coding.
- Experiment 4.19 investigates the coding performance of temporal prediction with limited refinement coding.
- Experiment 4.20 investigates the coding performance of temporal prediction with no refinement coding. The prediction is used *as is*.

Experiment 4.18 investigates the coding performance of temporal prediction for the intercept representation. The detailed coding results are shown in Table A.23. Here, compared to the case of using no prediction in Experiment 4.4, coding efficiency improves from −0.42 % to −0.49 % as shown in Table A.23. This is a substantial increase in coding efficiency, compared to the relative improvement measured for the angle-distance representation. The temporal prediction mode usage analysis in Table A.24 reveals that for 42.1 % of the area coded with GEO the partitioning has been temporally predicted, averaged over the entire CTC test set, significantly more compared to the usage for the angle-distance representation. In particular, the *CatRobot1* sequence and the *ArenaOfValor* screen content sequence are well suited for temporal prediction with average prediction mode usages of 54.8 % and 53.3 %, respectively. On the lower end of the usage statistics are the *Campfire* and *RitualDance* sequence with 26.6 % and 32.67 %. The *RitualDance* sequence is an interesting example where temporal prediction is not successful, whereas its characteristics otherwise fit the GEO model quite well, showing dancing people. However, the fast speed of the motion may be an explanation why temporal prediction using the projection process is not suitable. The limited search range around the collocated position is likely not able to capture fast moving object boundaries.

The distribution of coordinate refinement values m and n in Figure 4.38 appears overall very similar to those that were previously measured for Experiment 4.15. In roughly 70-80 %

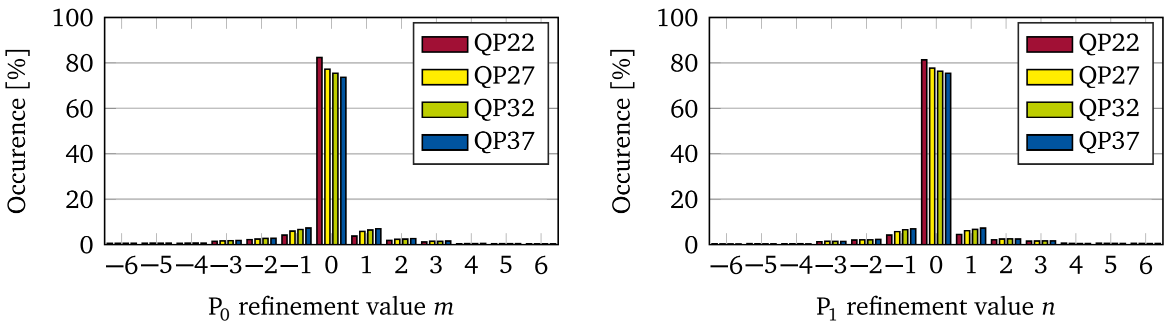


Figure 4.38 Distribution of coordinate refinement values m and n, measured for Experiment 4.18 using temporal prediction.

of the cases for each parameter, no refinement value is coded. In further consequence, the limited refinement coding scheme for the intercept representation tested in Experiment 4.19 also provides a slight coding gain compared to the unlimited refinement coding. The detailed coding results are shown in Table A.25. With this approach, overall coding efficiency improves to −0.51 % in terms of BD-rate change. No additional benefit can be measured when no refinement coding is allowed as shown in Experiment 4.20, as the overall BD-rate change remains at −0.51% . The detailed results for this experiment are given in Table A.26.

4.3.3 Summary

The experiments performed in Sections 4.3.1 and 4.3.2 investigated the two approaches for signaling a given set of geometric partitioning parameters: explicitly, without exploiting any spatial or temporal correlation, and predictively by trying to estimate the partitioning parameters of the current coding block from the spatial and temporal neighborhood. Additionally, it was investigated whether different levels of refinement coding provide any benefit once a predictor has been determined. The coding experiments were performed for the two different representations of geometric partitioning parameters.

In Table 4.11, the explicit coding results that were measured in Experiments 4.2, 4.6, 4.7 and 4.8 for the angle-distance representation are summarized. Huffman coding provides the highest overall coding efficiency with a luma BD-rate change of −0.50 %. These results were achieved by using GEO partition mode usage statistics per block size for the generation of Huffman code trees. Thereby, the joint probabilities of certain angle and distance combinations are mapped to code words with fewer bits. In terms of coding efficiency, this is followed by the less complex truncated binary coding which assumes a uniform distribution of GEO partition modes. With this method, −0.47 % of BD-rate change can be achieved over the VTM-3.2 anchor.

In Table 4.12, the coding results for predictive coding are summarized. In general, temporal prediction improves the coding efficiency more than spatial prediction. Highest overall coding efficiency was achieved by using the angle-distance representation for GEO parameters and not performing any refinement coding on top of the predicted parameters. Using this combination, a BD-rate change of −0.53 % was measured over VTM-3.2 for a RA coding configuration.

In summary, it can be stated that spatial prediction of the partitioning by assuming a lin-

Explicit Coding Method	BD-Rate Change
Fixed length	−0.45 %
Huffman	−0.50 %
Truncated Binary	−0.47 %
Separate Parameters	−0.46 %

Table 4.11 Summary of the explicit coding results using the angle-distance representation for Experiments 4.2, 4.6, 4.7 and 4.8.

Predictive Coding Method	BD-Rate Change			
	Spatial Prediction		Temporal Prediction	
	AD	BI	AD	BI
Full Refinement	−0.49 %	−0.44 %	−0.51 %	−0.49 %
Limited Refinement	−0.48 %	−0.44 %	−0.52 %	−0.51 %
No Refinement	−0.49 %	−0.44 %	−0.53 %	−0.51 %

Table 4.12 Summary of the predictive coding results using the angle-distance and intercept representation with different refinement coding methods. The explicit coding for all simulations was performed using fixed length binarization. The coding efficiency baseline result for the angle-distance results is −0.45 % and −0.42 % for the intercept representation, where no prediction is used.

ear continuation of a neighboring partitioning is not a viable method to improve the coding efficiency of geometric partitioning substantially. Different levels of refinement coding have almost no impact on measured coding efficiency since the usage of such a prediction mode is too low. On the other hand, temporal prediction by projection of the partitioning parameters using the reference picture motion vectors can improve coding efficiency by almost 0.1 %, independent of the chosen representation. For the angle-distance representation, a good rate-distortion trade-off seems to be achievable by not coding any refinement parameters and using the temporally predicted partitioning parameters as they are. In terms of implementation and algorithmic complexity, the intercept representation allows for a simpler projection process. Certainly, temporal prediction adds additional complexity and memory requirements to the coder. However, the additional memory requirements are in the same range as motion vectors. Algorithmically, the presented projection method also offers room for improvement by using a smarter search strategy than the currently employed scan method.

5 Inter-prediction Modifications

The blending process that was already briefly introduced in Section 5.1.1 is one of the main inter-prediction features of non-rectangular partitioning (GEO or TPM), significantly distinguishing it from other motion compensation based block tools. Next to the blending process, additional modifications to the coder are applied for GEO, compared to regular block based inter-prediction. Some of these changes affect only the encoder: The motion estimation process, for example, needs to be adapted to handle non-rectangular partitions. The motion compensation process, on the other hand, is affected at the encoder and decoder side. It is also of interest to analyze the effect of GEO on memory bandwidth for bi-directional motion compensation and to investigate how coding performance changes by restricting the prediction to uni-directional motion compensation. Other adaptations, such as for motion vector prediction and coding may further affect coding efficiency. Minor changes, such as the motion vector storage process are not considered to be critical but are discussed briefly for completeness.

The following sections detail the necessary changes for GEO in close alignment with Versatile Video Coding and the VTM-3.2 reference encoder and decoder. Since VVC inherits many technologies from HEVC, similarities and well-known concepts that apply to both codecs are pointed out. Optimizations and simplification to the processes described in this chapter are highlighted in the subsequent Chapter 6.

5.1 Motion Compensation

The basic principle of motion compensation for GEO has been reviewed in Section 4.1: A composite prediction block $\boldsymbol{P}_{\mathrm{PB}}$ is generated from two intermediate prediction blocks $\boldsymbol{P}_{\mathrm{PB},0}$ and $\boldsymbol{P}_{\mathrm{PB},1}$ through the blending process:

$$\boldsymbol{P}_{\mathrm{PB}} = \boldsymbol{M}_{\mathrm{f},0} \circ \boldsymbol{P}_{\mathrm{PB},0} + \boldsymbol{M}_{\mathrm{f},1} \circ \boldsymbol{P}_{\mathrm{PB},1} \tag{5.1}$$

The prediction blocks $P_{\mathrm{PB},k}$, $k \in \{0, 1\}$ are the result of uni- or bi-directional motion compensation. In contrast to other inter-prediction coding modes of VVC, this requires that at least two block-based motion compensation steps are performed. In the worst-case, four motion compensation steps are required if both GEO segments are utilizing bi-prediction.

Equation (5.1) is a simplified model of the blending process using the filter weights $\boldsymbol{M}_{\mathrm{f}}$ given in floating point accuracy. In the actual specification and reference implementation however, integer arithmetic is used for the entire process of motion compensation. The filter masks $\boldsymbol{M}_{\mathrm{f},0}$ and $\boldsymbol{M}_{\mathrm{f},1}$ then contain per-sample weights $w_0(x, y)$ and $w_1(x, y)$ such that $w_0 + w_1 = 2^{B_{\mathrm{f}}}$ for all x and y.

If a motion vector is having a non-zero, fractional sample part, an interpolation process $f_{\mathrm{Int}}(\cdot)$ is further applied. For a uni-predicted block, using the motion vector $\boldsymbol{m} = [m_x \; m_y]^{\mathrm{T}}$ and the reference pictures $R_{\mathrm{L}i}(x, y)$, $i \in \{0, 1\}$ from reference picture list L0 or L1, respectively, this can be expressed as:

$$\tilde{P}_{\mathrm{L}i} = f_{\mathrm{Int}}(R_{\mathrm{L}i}(x + \lfloor m_x \rfloor, y + \lfloor m_y \rfloor), \boldsymbol{m}) \tag{5.2}$$

In the VTM reference software, fractional sample interpolation is performed using 14 bit accuracy. Therefore, in order to obtain prediction samples in the output bit depth B_d, e.g. $B_\mathrm{d} = 8$ bit or $B_\mathrm{d} = 10$ bit, an additional scaling process is applied after motion compensation. For a motion compensated and interpolated sample $\tilde{p}_{\mathrm{L}i}$, given in 14 bit precision, and a prediction sample p_{PB} in output bit depth B_d, this can be expressed as:

$$p_{\mathrm{PB}} = \left\lfloor \frac{\tilde{p}_{\mathrm{L}i}}{2^{14-B_\mathrm{d}}} + \frac{1}{2} \right\rfloor, \quad i \in \{0,1\} \tag{5.3}$$

Accordingly, if bi-prediction with equal weighting is used and two interpolated samples $\tilde{p}_{\mathrm{L0}}$ and $\tilde{p}_{\mathrm{L1}}$ are given, the prediction sample p_{PB}in output bit depth is derived by

$$p_{\mathrm{PB}} = \left\lfloor \frac{\tilde{p}_{\mathrm{L0}} + \tilde{p}_{\mathrm{L1}}}{2^{15-B_\mathrm{d}}} + \frac{1}{2} \right\rfloor. \tag{5.4}$$

For GEO, Equation (5.1) is implemented such that the weighting by $\boldsymbol{M}_\mathrm{f}$ is applied in the 14 bit domain for higher precision. Up to four interpolated samples are involved, depending on the usage of uni- or bi-prediction for each segment. In the following, $\tilde{p}_{\mathrm{L}i,k}$ shall denote the interpolated sample of segment k, resulting from the segment-wise motion compensation using the motion vector $\boldsymbol{m}_{\mathrm{S,L}i,k}$. The weights w_k of $\boldsymbol{M}_{\mathrm{f},k}$ shall be integer weights in the range of $0 \leq w_k \leq 2^{B_\mathrm{f}}$.

- If both segments are using uni-prediction, the final predicted sample p_{PB} is derived by

$$p_{\mathrm{PB}} = \left\lfloor \frac{w_0 \tilde{p}_{\mathrm{L}i,0} + w_1 \tilde{p}_{\mathrm{L}j,1}}{2^{14+B_\mathrm{f}-B_\mathrm{d}}} + \frac{1}{2} \right\rfloor, \quad i,j \in \{0,1\}. \tag{5.5}$$

- If segment $k = 0$ is using bi-prediction and segment $k = 1$ is using uni-prediction, the final predicted sample p_{PB} is derived by

$$p_{\mathrm{PB}} = \left\lfloor \frac{w_0(\tilde{p}_{\mathrm{L0},0} + \tilde{p}_{\mathrm{L1},0}) + 2 w_1 \tilde{p}_{\mathrm{L}j,1}}{2^{15+B_\mathrm{f}-B_\mathrm{d}}} + \frac{1}{2} \right\rfloor, \quad j \in \{0,1\}. \tag{5.6}$$

- If segment $k = 0$ is using uni-prediction and segment $k = 1$ is using bi-prediction, the final predicted sample p_{PB} is derived by

$$p_{\mathrm{PB}} = \left\lfloor \frac{2 w_0 \tilde{p}_{\mathrm{L}i,0} + w_1(\tilde{p}_{\mathrm{L0},1} + \tilde{p}_{\mathrm{L1},1})}{2^{15+B_\mathrm{f}-B_\mathrm{d}}} + \frac{1}{2} \right\rfloor, \quad i \in \{0,1\}. \tag{5.7}$$

- If both segments are using bi-prediction, the final predicted sample p_{PB} is derived by

$$p_{\mathrm{PB}} = \left\lfloor \frac{w_0(\tilde{p}_{\mathrm{L0},0} + \tilde{p}_{\mathrm{L1},0}) + w_1(\tilde{p}_{\mathrm{L0},1} + \tilde{p}_{\mathrm{L1},1})}{2^{15+B_\mathrm{f}-B_\mathrm{d}}} + \frac{1}{2} \right\rfloor. \tag{5.8}$$

5.1.1 Discretization of Geometric Partitions and Prediction Blending

According to Figure 4.1, the second step in generating a geometric partition is the discretization of the partitioning line. Up to this point, the geometric model only defines a partitioning line in terms of continuous coordinates. The discretization process transforms the geometric model parameters into two-dimensional matrices. This process can also be described as a rasterization. As mentioned before, a weighting mask $\boldsymbol{M}$ is employed to combine the samples of each prediction into a rectangular block for further processing. Two types of weighting masks are being considered:

1. Weighting or blending filter masks $\boldsymbol{M}_\mathrm{f}$, which are used to compose the final prediction block from the two intermediate prediction blocks. In analogy to overlapped-block motion compensation and weighted prediction, the weighting masks $\boldsymbol{M}_\mathrm{f}$ define a transition zone around the partitioning line, where the samples from both predictions are weighted based on their distances to the partitioning line.

2. Binary weighting masks $\boldsymbol{M}_\mathrm{b}$, which are used to unambiguously indicate whether a sample is part of the first S_0 or the second S_1 partition segment. This information is useful for subsequent stages of transform coding of individual segment residuals e.g. using the Shape-adaptive DCT. The binary weighting mask or sub-sampled version may also be used to assign motion vectors to the motion vector buffer.

Since chroma subsampling may be employed for the given video source, two masks of different size can be associated with the blending filter or the binary mask, e.g. $\boldsymbol{M}_\mathrm{f,Luma}$ and $\boldsymbol{M}_\mathrm{f,Chroma}$ or $\boldsymbol{M}_\mathrm{b,Luma}$ and $\boldsymbol{M}_\mathrm{b,Chroma}$. The luma and chroma masks can either be generated from the same discretization process with different input sizes for the width and height as given below or the chroma mask can be generated from the luma mask via subsampling. The latter process is a simplification and detailed in Chapter 6.6.2. For brevity and ease of notation, $\boldsymbol{M}_\mathrm{f}$ and $\boldsymbol{M}_\mathrm{b}$ shall relate to the luma component if not stated otherwise.

Both types of masks can be generated in the same manner, by calculating the distance d of each sample from the partitioning line using the following equations. For the angle-distance representation, the coordinate system is placed in the center of the current block, therefore the shifted coordinates x_c and y_c are used for ease of calculation. Equation 5.9 shows the per-sample distance calculation in polar coordinates, given the partitioning line parameters $\boldsymbol{A}^\mathrm{AD} = [\ \varphi \quad \rho\ ]$. It is noted that a definition with inverted y-coordinates, e.g. $-y_\mathrm{c} \sin\varphi$ is also frequently encountered in the literature, which would align the y-axis direction of the polar coordinate system with the picture coordinate system.

$$d^\mathrm{AD}(\varphi, \rho, x_\mathrm{c}, y_\mathrm{c}) = x_\mathrm{c} \cos\varphi + y_\mathrm{c} \sin\varphi - \rho \tag{5.9}$$

$$\begin{bmatrix} x_\mathrm{c} \\ y_\mathrm{c} \end{bmatrix} = \begin{bmatrix} x - {}^{w}\!/_{2} + {}^{1}\!/_{2} \\ y - {}^{h}\!/_{2} + {}^{1}\!/_{2} \end{bmatrix}, \quad x,\ y \in B \tag{5.10}$$

For the intercept representation, the line coordinates $\boldsymbol{A}^\mathrm{BI} = [\ P_0 \quad P_1\ ]$ are used to determine the distance d to the partitioning line according to Equation (5.11).

$$d^\mathrm{BI}(P_0, P_1, x, y) = \frac{(y_1 - y_0)x_c - (x_1 - x_0)y_c + x_1 y_0 - x_0 y_1}{\sqrt{(y_1 - y_0)^2 + (x_1 - x_0)^2}} \tag{5.11}$$

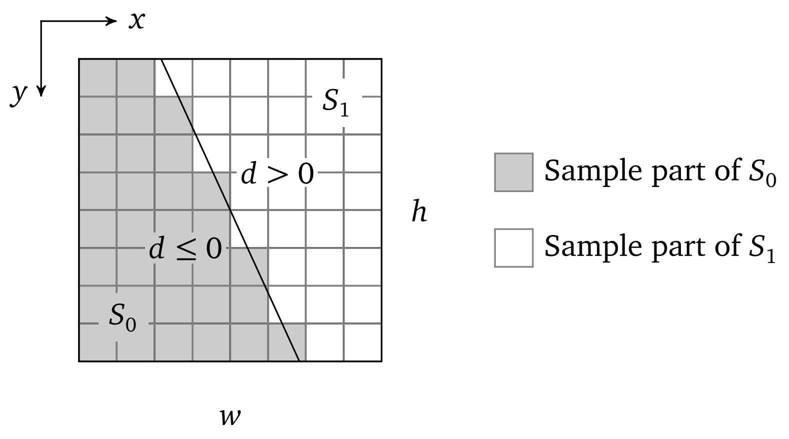


Figure 5.1 Example of a binarized geometric partitioning.

Equation (5.11) is the expanded form of the well-known point-to-plane distance equation using the Hessian normal form for two dimensions, where a normal $\boldsymbol{n}_0$ perpendicular to a given plane and a positive distance d_P from the coordinate origin to the plane is specified. The distance of the plane to a point given by its position vector $\boldsymbol{r}$ can then be calculated as:

$$d(\boldsymbol{r}) = \boldsymbol{r} \cdot \boldsymbol{n}_0 - d_P \tag{5.12}$$

For both representations, the sign of the distance d can be used to identify on which side of the partitioning line a sample (x, y) is located. This information can be stored in the binary mask $\boldsymbol{M}_b$. All samples which fulfill the condition $d \leq 0$ are assigned to segment S_0 and all samples which fulfill $d > 0$ are assigned to segment S_1. This is indicated in Figure 5.1. The determination of the distance of a sample to the partitioning line based on the center of the sample solves the assignment ambiguity for samples that are sliced by the partitioning line. If the center of a sample is located on one side of the partitioning line, then it can be easily seen that the majority of the sample is also located on the same side of the partitioning line.

$$\boldsymbol{M}_b(x, y) = \begin{cases} 0, & \text{if } d(x_c,\ y_c) \leq 0 \\ \\ 1, & \text{if } d(x_c,\ y_c) > 0 \end{cases} \tag{5.13}$$

Further, using a mapping function $f(d)$, the linear weighting masks M_f containing the weights w_{GEO} can be constructed. An example of a mapping function can be seen in Figure 5.2. The mapping function $f(d)$ in this example assigns linear weights in the discretized range from 0...1 to each distance d.

$$w_{GEO}(x, y) = f(d(x_c,\ y_c)) \tag{5.14}$$

$$\boldsymbol{M}_f = \begin{bmatrix} w_{GEO}(0,0) & w_{GEO}(1,0) & \cdots & w_{GEO}(w-1,0) \\ w_{GEO}(0,1) & & & \\ \vdots & & \ddots & \vdots \\ w_{GEO}(0,h-1) & & \cdots & w_{GEO}(w-1,h-1) \end{bmatrix} \tag{5.15}$$

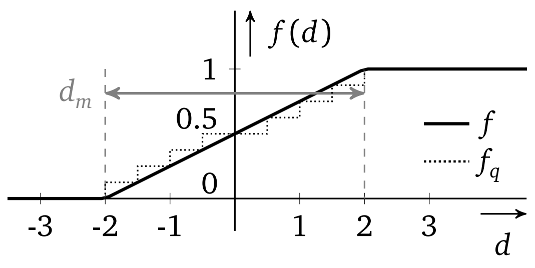


Figure 5.2 Distance-based weighting function with configurable transition region d_m.

$$f(d) = \begin{cases} 0, & d < -d_\mathrm{m} \\ \frac{d}{d_\mathrm{m}} + 0.5, & -d_\mathrm{m} \le d \le d_\mathrm{m} \\ 1, & d > d_\mathrm{m} \end{cases} \tag{5.16}$$

The transition region is denoted as d_m and could be a constant parameter setting or an adaptive parameter, for example depending on the block size or the type of video content. A larger transition region will generally cause a larger blurring or smoothing along the partitioning line, whereas a smaller transition region will likely introduce a visible edge into the composite prediction block. In order to avoid floating point operations when the weighting function $\boldsymbol{M}_\mathrm{f}$ is applied to the samples, $f(d)$ is typically quantized and scaled to integer values in an actual implementation. More details on such implementation specific aspects are given in Chapter 6.

The integer quantization of $f(d)$ is also indicated as f_q in Figure 5.2. Below the minimum or above the maximum range of d, the weight is clipped to the value of 0 or 1. For simplicity and ease of implementation, two blending and binary masks can be defined for each GEO partition mode that relate to each segment, with $\boldsymbol{J}$ being a matrix of all ones:

$$\begin{aligned} \boldsymbol{M}_{\mathrm{f},0} &= \quad \boldsymbol{M}_\mathrm{f} \\ \boldsymbol{M}_{\mathrm{f},1} &= \boldsymbol{J} - \boldsymbol{M}_\mathrm{f} \end{aligned} \tag{5.17}$$

$$\begin{aligned} \boldsymbol{M}_{\mathrm{b},0} &= \quad \boldsymbol{M}_\mathrm{b} \\ \boldsymbol{M}_{\mathrm{b},1} &= \boldsymbol{J} - \boldsymbol{M}_\mathrm{b} \end{aligned} \tag{5.18}$$

TPM in VVC and the proposed GEO coding tool in this thesis are using a quantization step size for f_q of 0.125 that results in 9 different values for $\boldsymbol{M}_\mathrm{f}$, resulting in the closed value interval given by $[0, 0.125, 0.25...1]$. Scaling this interval by 2^{B_f} allows the integer averaging process to be implementable using multiplications, additions, and bit-shifts only. Figure 5.3 visualizes the influence of different transition regions sizes d_m for the case of an 32×32 luma blending filter $\boldsymbol{M}_\mathrm{f}$ with $2^{B_\mathrm{f}} = 8$,e.g. 9 possible weights.

The question arises, how large the transition region d_m shall be defined for best coding efficiency. In the following, a detailed heuristic analysis is presented, how different transition region values d_m impact the coding gain and residual characteristics. An optimal setting is to be empirically determined which can be applied for further experiments. Also, the potential visual impacts are briefly assessed.

In the subsequent Experiments 5.21, 5.22, 5.23 and 5.24, coding results are obtained by encoding the JVET test set using different filter lengths $d_\mathrm{m} = \{1, 3, 5, 7\}$ for all block-sizes.

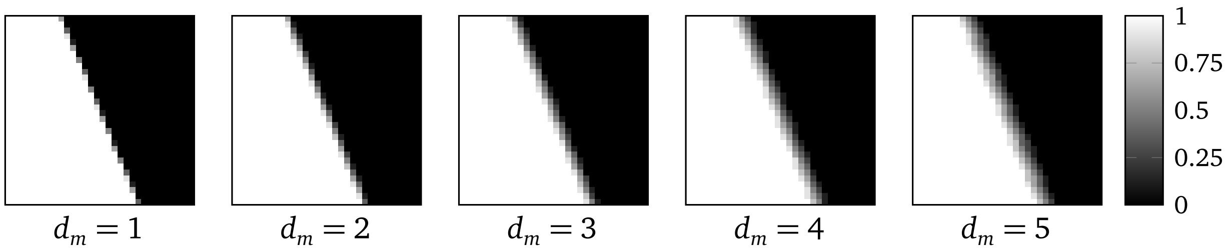


Figure 5.3 Example of 32 × 32 filter masks $\boldsymbol{M}_\mathrm{f}$ for different transition region sizes d_m.

	Luma BD-rate change			
	$d_\mathrm{m} = 1$	$d_\mathrm{m} = 3$	$d_\mathrm{m} = 5$	$d_\mathrm{m} = 7$
JVET Overall	−0.32 %	−0.50 %	−0.48 %	−0.37 %

Table 5.1 Overall coding performance obtained when varying the blending filter length d_m.

As depicted in Figure 5.3, this causes the blending region to vary from a sharp transition to a qualitatively wide transition. To further study the effect of the different blending transition sizes, the unquantized prediction residual is also recorded during the encoding. In order to limit the amount of data and computation time for these simulations, only the first 65 pictures of each sequence were coded for the experiments. This corresponds to one or two Groups of Pictures (GOPs), depending on the frame rate of each video sequence. The detailed results for these four experiments are given in Table A.27 ($d_\mathrm{m} = 1$), Table A.28 ($d_\mathrm{m} = 3$), Table A.29 ($d_\mathrm{m} = 5$) and Table A.30 ($d_\mathrm{m} = 7$) in the appendix. An overview of the corresponding BD-rate change for each blending filter is shown in Table 5.1.

Based on the obtained coding results, the following conclusions can be drawn:

- Overall, a filter length of $d_\mathrm{m} = 3$ provides the best trade-off in terms of coding gain across all classes.
- For UHD sequences, a larger filter length of $d_\mathrm{m} = 5$ appears to be slightly beneficial.
- For screen content, which is contained in class F, the shortest tested filter length of $d_\mathrm{m} = 1$ provides the highest coding gain.

These findings suggest that the optimal size of the blending region may depend on the characteristics of the video content. It is not surprising that applying less blending, which is assumed to have a similar impact on the residual as OBMC, to screen content or Text and graphics with motion (TGM) content is beneficial due to the high spatial-frequency spectrum of such sequences. Surprisingly though, the optimal filter size for natural video content in terms of coding efficiency as measured in Experiments 5.21 - 5.24 is noticeably sharper compared to the blending filter used in TPM. In TPM, the integer filter weight w_TPM is given by the Manhattan distance with an offset of 4 of each sample to the diagonal (or anti-diagonal) partitioning line, clipped to a value range between 0 and 8. For the diagonal from the top-left to the bottom-right, the weight $w_\mathrm{TPM,0}$ is derived as given by Equation (5.19) and for the anti-diagonal from the bottom-left to the top-right the weight $w_\mathrm{TPM,1}$ is derived as given by Equation 5.20:

$$d_{\mathrm{TPM},0} = \frac{x}{a} - \frac{y}{b} + 4 \tag{5.19}$$

$$d_{\mathrm{TPM},1} = \begin{cases} h-1-\frac{x}{a}-\frac{y}{b} & w > h \\ w-1-\frac{x}{a}-\frac{y}{b} & w \leq h \end{cases} \tag{5.20}$$

$$w_{\mathrm{TPM,i}}(x,y) = \mathrm{clip}(0,8,d_{\mathrm{TPM},i}(x,y)) \quad i \in \{0,1\} \tag{5.21}$$

Therefore, it can be seen that the TPM blending filter is 7 samples across, noticeably wider than the optimal GEO blending filter.

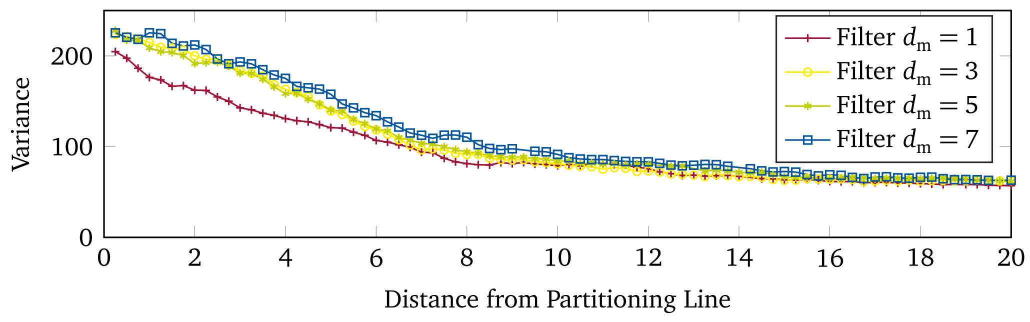
In order to gain more insight how different filter lengths impact the properties of the residual signal, it is of interest to analyze the local energy content of the prediction residual.

It has been theoretically shown by Shishikui [Shi92], Wentao and Zheng [Wen+00; Zhe+01; Wen+02] that the motion-compensated frame difference samples are not wide-sense stationary across a block, meaning that the mean and the covariance are not independent of the spatial position within the block. In fact, the variance of the residual tends to increase towards the edges of the block and is lowest at its center. According to the authors, the space-dependent properties are due to the motion fluctuation and the nature of block-based motion compensation and estimation. Further, OBMC has been proven empirically and theoretically to lower the residual variance at block boundaries and provide coding gain as well as perceptual visual improvements.

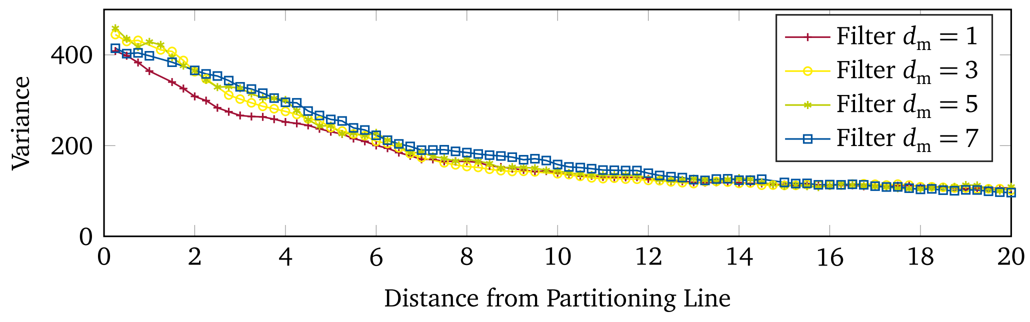
In the following, an empirical analysis of the space-dependent residual variance is extended to geometrically partitioned blocks. As GEO blocks can be partitioned in multiple ways for a range of different block sizes, unlike performed in previous studies, the residual variance is not measured in two dimensions but depending on the distance to the partitioning line. For each sequence of the test set that was encoded in Experiments 5.21 – 5.24, roughly 3000000 residual luma samples drawn from randomly selected geometrically partitioned blocks are analyzed depending on their absolute distance to the partitioning line. To reduce the effect of sequence-dependent outliers, the median across the sample variance per distance bin is measured. The results of this analysis can be seen for each QP in Figure 5.4.

The following observations can be made from Figures 5.4a – 5.4d:

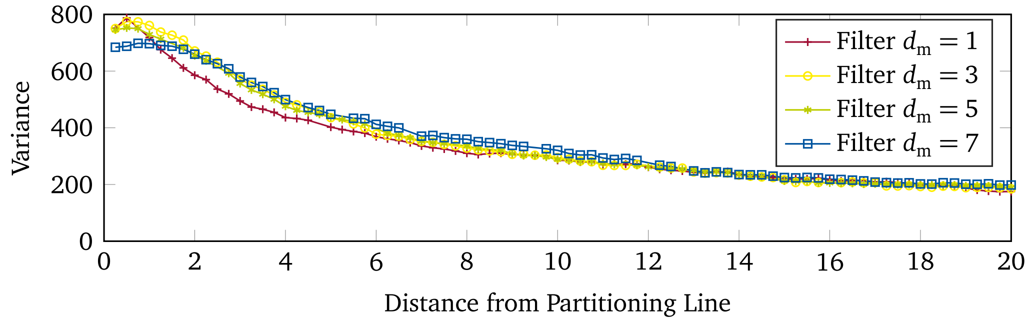
- In general, the observation of a non-stationary variance for the motion compensated prediction error can be confirmed for GEO blocks. As it is known for the block boundaries, the prediction error variance is distinctly higher in the vicinity of the partitioning line.
- For the lower QPs 22 and 27, see Figure 5.4a and Figure 5.4b, the shortest filter length of $d_{\mathrm{m}} = 1$ results in the overall lowest prediction error variance depending on the distance to partitioning line. Although the prediction errors variances for different filter lengths converge to the same value at a distance of roughly $d \geq 10$, the shortest filter length converges faster.
- For higher QPs, especially for QP37, see Figure 5.4d, increasing the blending filter length d_{m} lowers the prediction error variance in the region close to the partitioning line between two segments.



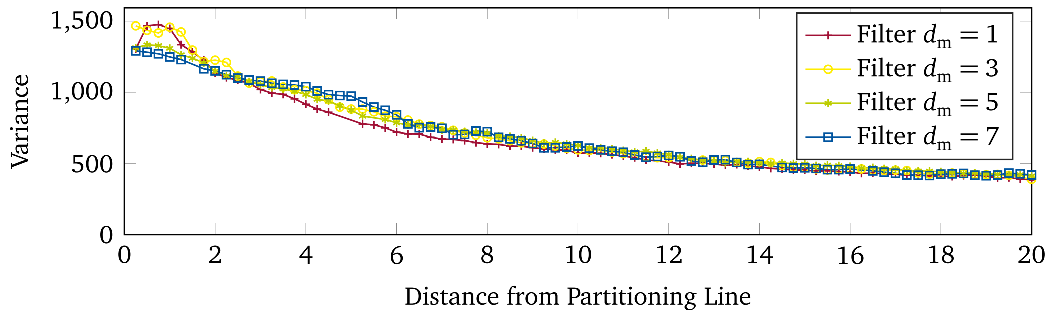
(a) QP22



(b) QP27



(c) QP32



(d) QP37

Figure 5.4 Residual luma variance for GEO blocks depending on the distance to the partitioning line using different filtering lengths d_m.

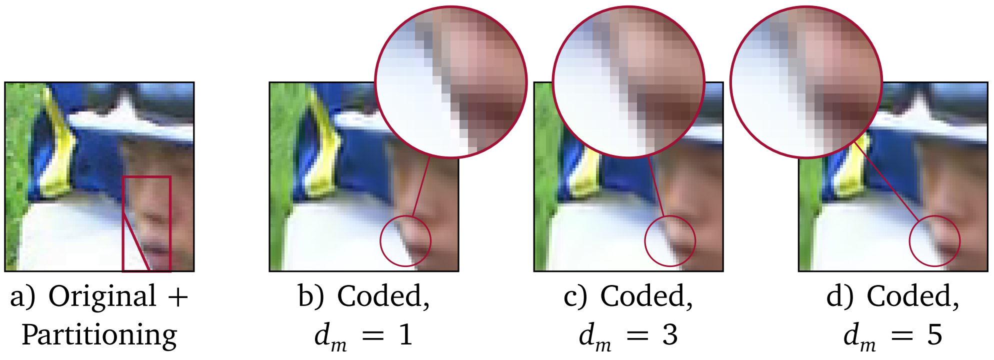


Figure 5.5 Example of 32 × 32 filter masks $\boldsymbol{M}_\mathrm{f}$ for different transition region sizes d_m.

It can be summarized that the known effect of OBMC for regular motion compensated blocks also holds true for geometrically partitioned blocks to a certain degree. At least for higher QPs, averaging the sample values at the boundary between two segments provides the benefit of a lower residual energy. However, the analysis also shows that more blending applied uniformly to all blocks does not result in more coding gain. It can be assumed that also the mode utilization for different block sizes, which is quite QP-dependent as shown in Figure 4.11, is another factor to consider. At lower QPs, typically smaller block sizes in the range from 8 × 8 to 16 × 16 luma samples are used more frequently. At large blending filter lengths, almost all samples are being averaged, and therefore these small blocks may lose their distinct feature of being geometrically partitioned. The GEO mode in this case would operate more like multi-hypothesis prediction with two to four hypothesis per block [Gir98; FWG98].

Lastly, next to the influence on the coding gain, the potential visual impact of different blending parameters must also be considered. It is known from the design of loop-filters that objective measures such as PSNR and Structural Similarity (SSIM) do not always reliably justify certain filtering design choices [Nor+12]. Therefore, objective assessment through extensive visual testing is often performed.

Figure 5.5 exemplifies the potential visual impact of different blending filter lengths for a coded example. It can be seen that the overall differences in sharpness are small but noticeable in a still-picture comparison. For further experiments in this chapter, it was therefore decided to keep the blending filter length at $d_\mathrm{m} = 3$ as this setting provided the most overall coding gain. Further, an even shorter filter might be visually less convincing and may make the object boundaries artificially sharp. In Chapter 6, Section 6.3 the findings in this section are utilized to design an adaptive blending filter for GEO and TPM in VTM-5.0 that can be signaled to switch between two different blending filter lengths depending on the video content.

5.1.2 Uni- and bi-directional Motion Compensation

Using the motion information, e.g. the motion vectors $\boldsymbol{m}_{S,0}$ and $\boldsymbol{m}_{S,1}$ and the associated reference pictures, rectangular arrays of samples are fetched from the Decoded Picture Buffer (DPB). If indicated by the magnitude of the x- and y-component of the motion vectors, fractional sample interpolation may also be performed. Lastly, the two intermediate prediction

Block size	Memory Bandwidth B			
	Uni-prediction		Bi-prediction	
	GEO	Regular	GEO	Regular
8×8	7.03	3.51	14.06	7.03
16×16	4.13	2.07	8.27	4.13
32×32	2.97	1.49	5.94	2.97
64×64	2.46	1.23	4.92	2.46
128×128	2.22	1.11	4.45	2.22

Table 5.2 Theoretical worst-case memory bandwidth requirements for GEO compared to regular predicted blocks using uni- or bi-prediction (both segments bi-predicted for the case of GEO).

blocks $\boldsymbol{P}_{\text{PB},0}$ and $\boldsymbol{P}_{\text{PB},1}$ are multiplied by the respective weighting masks $\boldsymbol{M}_{\text{f},0}$ and $\boldsymbol{M}_{\text{f},1}$, added and scaled, if integer weights are used in the process.

It is noted that each intermediate prediction block $\boldsymbol{P}_{\text{PB},i}$, $i \in \{0, 1\}$ may be the result of uni- or bi-directional motion compensation, depending on the number of motion hypotheses. Thus, in contrast to regular inter-prediction in VVC, GEO would require 4 block-based memory accesses and interpolation steps in a worst-case scenario where both segments are using bi-prediction.

The larger number of motion compensation operations increase the theoretical memory bandwidth requirements. In this context, the theoretical memory bandwidth B is understood as the ratio between the number of fetched reference samples and the number of predicted samples:

$$B = \frac{N_{\text{fetched}}}{N_{\text{predicted}}} \tag{5.22}$$

The memory bandwidth B is higher for smaller blocks, due to the additional samples that have to be fetched around the given block for the interpolation filter. When a K-tap interpolation filter is used, $K/2 - 1$ additional samples to the top and left block boundary and $K/2$ additional samples to the bottom and right block boundary are required. Considering a block of $w \times h$ luma samples, the required memory bandwidth for performing uni-prediction is given as:

$$B = \frac{(w + K - 1) \cdot (h + K - 1)}{w \cdot h} \tag{5.23}$$

As it is the case in HEVC, VVC specifies an 8-tap FIR filter for interpolation the luma component and a 4-tap FIR filter for the chroma component. Table 5.2 lists the theoretical luma memory bandwidth requirements of GEO for different block sizes in comparison to regular uni- or bi-predicted blocks. The analysis has been performed by assuming that GEO requires a full block motion compensation.

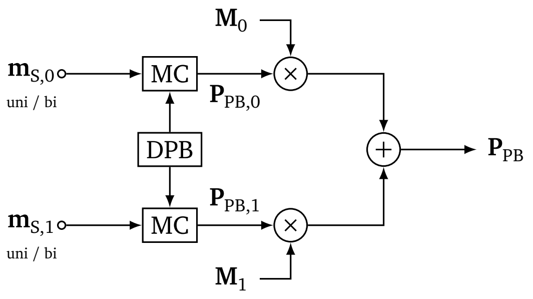


Figure 5.6 Basic principle of motion compensation for GEO using two block-based motion compensation sub-steps (MC) accessing the Decoded Picture Buffer (DPB), followed by the blending process to generate the final, composited prediction block $\boldsymbol{P}_{\mathrm{PB}}$.

The theoretical analysis shows that the worst-case memory bandwidth is effectively doubled. If the proposed GEO mode would utilize the same motion compensation logic twice and perform the intermediate prediction block generation successively for each segment, latency is introduced into the system. It is of interest to further study and quantify the effect of motion compensation for GEO with regard to the required memory bandwidth.

Each motion compensation step requires a read access to external memory where the reference picture data (reconstructed samples, motion vectors) is stored. However, it is not meaningful to measure the required memory bandwidth from external memory alone. In a hardware decoder, a memory access is typically performed using a fast on-chip memory cache[1]. The theoretical analysis in Table 5.2 based on Equation 5.22 is therefore supplemented by a more realistic measurement involving a cache model, as read access to the cache is considerably faster compared to external memory access. This is exemplified in Figure 5.7, showing a high-level block diagram of a video decoder with cached memory access. The on-chip cache is replenished from external memory using a given caching algorithm, such as Pseudo-Least-Recently-Used (PLRU) or Tree-PLRU. The cache is filled on a line-by-line basis from external memory and each cache line, having a pre-defined size, is labeled as an *item*. Further, the cache algorithm keeps track of when an item has been accessed. Items in the cache that are least recently used in an LRU scheme are discarded first and replaced by new items from the external memory. If the decoder now requests an item which is not in the cache, a cache-miss occurs, and an external memory access is triggered to retrieve the missing data. Otherwise, a cache-hit occurs and the decoder can access the data with very low latency. The required overall memory bandwidth for the external memory therefore can be significantly reduced using a cache with few cache-misses. Naturally, the size of the cache has a large impact on the external memory bandwidth. A larger cache will contain more items, leading to less cache-miss events and therefore less external memory access [Li+17; HMI17].

5.1.3 Memory Bandwidth Measurements

The VTM reference software contains an implementation of a virtual cache model and the necessary means to measure the required memory bandwidth for motion compensation of

[1]On modern, general purpose CPUs, this is equivalent to the L2 and / or L3 caches.

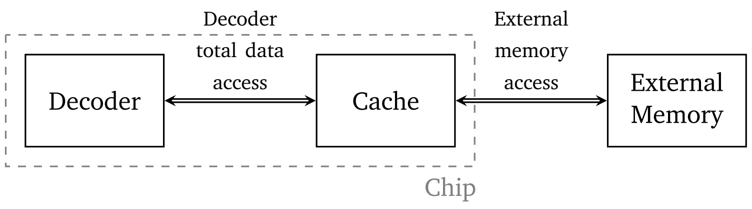


Figure 5.7 High-level block diagram of a data access model for a video decoder.

the decoder [HM18b; HM18a]. The parameters of the cache model are the size of each cache line, the total cache size, and the associativity level. These parameters can be configured by the user. For the standardization activity at JVET, the following parameters are suggested:

- 128 byte cache line size
- 64 cache lines
- 4-way associativity

The memory bandwidth for GEO can now be estimated by measuring the unavoidable data access to the external memory using the virtual cache model. To assess the change in memory bandwidth, the same measurements are made for the VTM-3.2 anchor. For Experiment 4.2, read access statistics measured by the decoder in MB are reported in Table A.31 in the appendix. For each sequence and QP, average memory bandwidth requirements across all pictures are computed. Additionally, the picture with worst-case memory usage is reported. Lastly, the relative changes for average memory usage and worst-case memory usage are computed.

As provided by the measurement results in Table A.31, the average required memory bandwidth for GEO increases by 2.68 % for QP37 and up to 4.24 % for QP22 compared to the VTM-3.2 anchor. Evidently, a higher mode usage of GEO will increase the average bandwidth requirement: For the particular example of a sequence and QP combination with a significant usage of GEO, such as *BQTerrace* coded at QP22, the average bandwidth increases by 12.14 %. On the other hand, the sequences *BQMall* and *RaceHorses*, which fit the geometric partitioning model best, show per-QP increases in memory bandwidth close to the overall average. Table A.31 also lists the worst-case memory bandwidth requirement measured for a single frame per sequence and QP. In most cases, the worst-case memory bandwidth requirement is higher for GEO compared to the anchor. However, instances can be found where the worst-case memory bandwidth actually decreases for GEO compared to the anchor (*SlideEditing* QP27, *BasketballPass* QP37).

The highest worst-case increase on the picture level for all tested sequences was measured at 18.32 % for the MarketPlace sequence coded at QP22. A deeper analysis shows that this worst-case increase occurs at POC 532 in the middle of a scene transition using a dissolve effect. Such a dissolve in combination with dynamic motion and a low QP is a perfect example for the usage of bi-prediction and fine segmentation. This particular picture and the geometric partitioning, since it marks the actual worst-case memory bandwidth increase of GEO, is displayed in Figure 5.8.

In summary, it can be stated that although the theoretical worst-case memory bandwidth for GEO with bi-prediction doubles, the impact for an actual decoder using cached memory

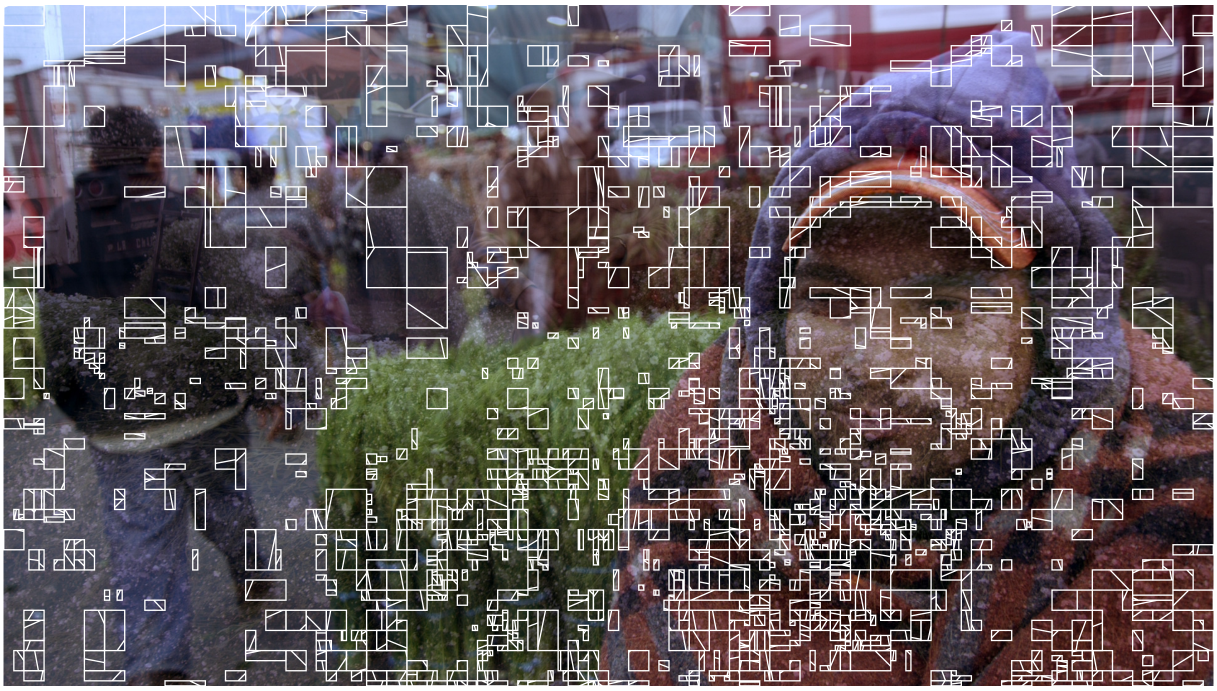


Figure 5.8 Picture with highest maximum memory bandwidth increase measured for the entire test set at 18.32 % compared to the VTM-3.2 anchor. In this particular example of the MarketPlace sequence (POC 532) coded at QP22, a scene transition occurs in combination with object motion (persons on the left and right) and also dynamic texture motion (plants), causing a high usage of bi-prediction and geometric partitioning at the same time. The usage of GEO in this picture is 31.04 % in terms of pixels coded.

is more difficult to predict. Furthermore, alternative motion compensation techniques for GEO, utilizing sub-block motion compensation, could offer a solution to the problem.

5.1.4 Restriction to Uni-prediction

Restricting motion compensation to uni-prediction is an option to limit the worst-case memory bandwidth requirements of GEO. The memory bandwidth requirements for GEO blocks in this setup are identical to those of regular, bi-predicted blocks. The restriction to uni-directional prediction for GEO can be easily enabled by changing the merge candidate list derivation process:

- For GEO with bi-prediction, the regular merge candidate list derivation process is used. A maximum of 6 merge candidates are available.
- For GEO with uni-prediction, the TPM merge candidate list derivation process is used. A maximum of 5 merge candidates are available.

As briefly explained in Section 2.4, uni-directional TPM merge candidates are constructed from spatially and temporally neighboring motion vectors in the following manner.

1. Motion vectors from the five spatial positions A_1, B_1, B_0, A_0 and B_3 (see Figure 4.27) are collected in this order.

2. If the candidate list is not filled yet, temporal motion vectors at the spatial positions C_0 and C_1 are collected in this order for the first reference picture.
3. Collected motion vectors are added to the TPM merge candidate list in the following order:
   a) First, uni-directional motion vectors are added until the list is full, or no more uni-directional motion vectors are available.
   b) If the list is not filled yet, the L0 component of bi-directional motion vectors are added until the list is full, or no more bi-directional motion vectors are available.
   c) If the list is not filled yet, the L1 component of bi-directional motion vectors are added until the list is full, or no more bi-directional motion vectors are available.
   d) If the list is not filled yet, averaged L0 and L1 components of bi-directional motion vectors are added until the list is full, or no more bi-directional motion vectors are available.
4. If the number of TPM merge candidates is less than 5, zero-valued motion vectors are added to the end of the list.

The coding performance for the uni-directional restriction is investigated in Experiment 5.25. The detailed coding results for this experiment are shown in Table A.33. Overall, the coding efficiency drops significantly from −0.45 % to −0.27 % in terms of luma BD-rate change. Especially for UHD sequences, the restriction to uni-prediction causes the coding gain to be halved, from −0.30 % for A1 in Experiment 4.2 with bi-prediction to −0.16 % in this experiment. On the other hand, the drop in coding efficiency improvement is not as high for sequences that fit the geometric partitioning model well. For *BQMall*, for example, luma BD-rate gain drops from −1.13 % to −0.96 % and for the *RaceHorses* sequence from −0.89 % to −0.77 %.

On the one hand, the reason why the coding efficiency improvement drops is clear: Bi-directional motion compensation by linear combination of two prediction signals has been shown theoretically, under the assumptions of precise motion estimation and uncorrelated displacement errors, to lower the bitrate of the residual encoder by at most 0.5 bit/sample [Gir98]. On the other hand, secondary effects also need to be considered but are more difficult to quantify. If the VTM encoder chooses to code the current block using GEO with uni-prediction, the RD-performance of motion-compensated prediction in subsequently coded blocks may suffer. Assuming bi-prediction is the RD-optimal prediction mode for theses blocks, more rate must be spent for the motion information signaling, since motion vector prediction from GEO or TPM blocks will only produce uni-directional motion vectors. This could be even more critical for higher QPs, where Merge-mode is the dominant way of signaling motion vectors.

5.2 Motion Estimation for Geometric Partitions

In the previous Sections 4.2 and 4.3, the motion information for each GEO segment was only viewed from a decoder perspective: Motion information is available for every GEO block, as it was simply decoded from the bitstream. This is necessarily preceded by an estimation process at the encoder side. The encoder estimates the RD-optimal motion information, e.g. the applicable motion model, uni- or bi-directional motion vectors and the best reference pictures for each block. The general process of motion compensation, as it is mentioned in Section 5.1, typically assumes a block-wise processing. Likewise, the motion estimation stage, which can be seen as a motion compensation step followed by a distortion computation step and a decision step, assumes a block-wise processing. In the reference encoders for HEVC and VVC, HM and VTM, respectively, different motion estimation strategies are employed, depending on the coding mode that is evaluated for the current block:

- In *Merge*-based inter-prediction modes, such as TPM, CIIP, or Merge mode with motion vector difference (MMVD) for VVC and the regular Merge-mode for HEVC, a predetermined set of motion vectors, termed merge candidates, is evaluated. Motion compensation is performed for each merge candidate and the resulting prediction error is computed.

- In *AMVP*-based inter-prediction modes which allow signaling of motion vectors and motion vector differences (MVDs), a Block-Matching-Algorithm (BMA) is used to search for an optimal motion vector. In the reference encoders for HEVC and VVC, for example, the Test Zone (TZ) fast search strategy is employed, which is a combination of an adaptive diamond- and raster search around an initial prediction center [Doa+17].

- In *Affine*-based inter-prediction modes with AMVP for the control point motion vectors, iterative gradient-based motion estimation is used. This method relies on the repeated computation of spatial and temporal gradients and solving a linear equation system to obtain higher order motion parameters.

In all of the above cases, a signal distortion D between a compensated block $\boldsymbol{P}_{\text{PB}}$ and the original block $\boldsymbol{P}_{\text{Org}}$ is computed at some point. To extend the block-wise computation for the Sum of squared differences (SSD) and Sum of absolute differences (SAD) metric to arbitrary shaped segments, the weighting masks $\boldsymbol{M}_{\text{f},k}$, $k \in \{0,1\}$ can be easily utilized in the process. This changes the regular block-based motion estimation to a *masked* motion estimation which is performed segment-wise:

$$D_{\text{SSD,M},k} = \sum_{x,y} \boldsymbol{M}_{\text{f},k} \circ (\boldsymbol{P}_{\text{PB}} - \boldsymbol{P}_{\text{Org}})^2, \quad k \in \{0,1\} \tag{5.24}$$

$$D_{\text{SAD,M},k} = \sum_{x,y} \boldsymbol{M}_{\text{f},k} \circ \left|\boldsymbol{P}_{\text{PB}} - \boldsymbol{P}_{\text{Org}}\right|, \quad k \in \{0,1\} \tag{5.25}$$

The weighting masks $\boldsymbol{M}_{\text{f},k}$ in the Equations (5.24) and (5.25) above contain the per-sample weights in a value range from 0 to 1. The weighting masks $\boldsymbol{M}_{\text{f},k}$ are Hadamard-multiplied with the array of squared errors for the SSD or the absolute errors for the SAD case. Note that the squaring operation $(\cdot)^2$ in Equation (5.24) refers to a per-element squaring. Thereby, for a given segment k, samples with a weight of 0 are associated with the other segment and do not influence the distortion $D_{\text{SSE,M},l}$ or $D_{\text{SAD,M},l}$, $l \neq k$ of the other segment. Furthermore, the prediction error occurring in the transition zone along the partitioning line between the two segments is given less weight in the overall distortion. The same rationale is used in motion estimation algorithms for OBMC, such as the Windowed Block Matching Algorithm

(WBMA) [SM00]. However, due to the non-linearity of SSD and SAD, the overall block-wise distortion for a composited GEO block cannot be correctly computed from the addition of segment-wise distortion measurements. This problem is further addressed in Section 5.5 on encoder mode control.

Equations (5.24) and (5.25) can be easily implemented at the encoder side and require very little change for the reference encoders for VVC or HEVC. The overall BMA or Merge-mode estimation strategies can be re-used for GEO and only the calculation of the distortion needs to be adapted from a block-wise to a segment-wise processing. The masked motion estimation can be very efficiently implemented using SIMD, for example using Intel's SSE on general purpose CPUs. An example implementation which computes a masked SAD using only SIMD instructions is given in Listing 5.1. This variant computes the distortion for vectors of 4 samples with single instructions. The algorithm, compared to regular block-based SAD calculation, requires one additional SIMD instruction, `_mm_madd_epi16` requiring an execution time of 1 clock tick per instruction (CPI) on most modern CPUs [Int19]. Extensions of the algorithm for vectors of 8 samples and 16 samples of 16 bit length are given in the appendix in Listing A.1 and Listing A.2.

At this point, it must be noted that no straightforward extension of the Sum of absolute transform differences (SATD) distortion metric can be given for arbitrary shapes. In the reference encoders for VVC and HEVC, SATD is used in AMVP for sub-sample motion estimation refinement, performed after an RD-optimal integer motion vector has been determined. The SATD metric operates by transforming a block of residual values using the Hadamard transform into a frequency domain before a SAD is computed in this domain. The weighting or masking of samples is therefore not directly applicable in the transform domain as the weight matrices $\boldsymbol{M}_{\mathrm{f},k}$ relate to spatial positions. Applying the mask to the residual before computing the Hadamard transform could potentially result in the generation of additional, spurious transform coefficients and thereby even worsen the measured distortion of the masked residual compared to the unmasked residual. The simplest solution to this problem is not to apply the SATD for sub-sample motion estimation of individual segments. However, it is possible to utilize the SATD on the rectangular, composited prediction block generated through the blending process.

5.3 Motion Vector Prediction and Coding

The methods of motion vector prediction and coding that were used in the preceding experiments are closely aligned with the *Merge*-mode of VVC and HEVC. In the VVC draft and its reference implementation VTM-3.2 however, three distinct methods of prediction and coding for block-based, translational motion vectors can be distinguished:

- *Merge*-mode based prediction and coding of motion vectors using a merge candidate list of Motion vector predictors (MVPs). In VVC, this encompasses the HEVC-like spatial, temporal, and zero-valued merge candidates and the additional, extended merge candidates, e.g. History-based MVPs (HMVPs).

- *AMVP*-based prediction and coding using a list of two motion vector predictors and the signaling of a Motion vector difference (MVD). In VVC, the signaling of the MVD can

```
Distortion getMaskedSAD4_SSE(short* pOrg, short* pPred, short* pMask, int
    width, int height, int strideOrg, int stridePred, int strideMask, int
    log2MaxWeight)
{
  __m128i vZero = _mm_setzero_si128();
  __m128i vSum  = vzero;
  for(int y = 0; y < height; y++)
  {
    for(int x = 0; x < width; x+=4)
    {
      __m128i vOrg  = _mm_loadl_epi64((const __m128i*)&pOrg[iX]);
      __m128i vPred = _mm_loadl_epi64((const __m128i*)&pPred[iX]);
      __m128i vmask = _mm_loadl_epi64((const __m128i*)&pMask[iX]);

      vSum32 = _mm_add_epi32(vSum, _mm_madd_epi16(vMask, _mm_abs_epi16(
          _mm_sub_epi16(vOrg, vPred))));
    }
    pOrg  += strideOrg;
    pPred += stridePred;
    pMask += strideMask;
  }
  vSum = _mm_hadd_epi32(vSum, vZero);
  vSum = _mm_hadd_epi32(vSum, vZero);
  unsigned uiSum = _mm_cvtsi128_si32(vSum);
  return uiSum >>= log2MaxWeight;
}
```

Listing 5.1 Code example for SAD-based distortion metric using SIMD instructions for vectors of 4×16 bit samples.

be performed with different resolutions: quarter luma sample, integer luma sample or four luma sample resolution. This method is denoted as AMVR.

- *MMVD*-based prediction and coding for VVC only, using the first two candidates of the regular merge candidate list and a set of 32 pre-defined MVD refinement values in the $\pm x$-axis or $\pm y$-axis direction.

In terms of precision, e.g. how closely the actual translational motion of a given block can be approximated by each coding mode, the Merge-mode offers the least precision since no MVD refinement is possible. Only a single index is coded in this case. This is followed by MMVD that allows the signaling of a refinement value for either the x- or y-component on top of the candidate vector. The eight available MVD magnitudes are non-uniformly distributed and signaled by two indices, specifying the direction and magnitude. Therefore, MMVD, can also be seen as a type of non-uniform vector quantization of the MVD.

The highest precision can be obtained by AMVP in combination with AMVR, where – within the allowed range of 16 bit for HEVC and 18 bit for VVC – arbitrary-valued MVDs can be signaled on top of a motion vector predictor. Each MVD component is signaled through a combination of a first flag, indicating a non-zero value, a second greater-than-two flag and an $C_{\mathrm{EG},1}(\cdot)$ coded remainder. Through AMVR, the MVD can be given in quarter-, full- or 4-sample precision.

It is of interest to investigate, which motion vector prediction and coding scheme can be combined with GEO to obtain further coding gain. All previous results show that the Merge-mode performs favorably due to its simplicity in terms of coding and estimation at the encoder side. In the Core Experiments (CEs) on Combined and Multi-Hypothesis Prediction during the early development of VVC, non-rectangular partitioning for inter-prediction in combination with AMVP was proposed and evaluated:

- Triangular (or *Diagonal* as termed by the authors) partitioning in combination with AMVP was tested in [AS18b; AS18a]. The usage of AMVP in the reference encoder is coupled to motion estimation using block matching. Therefore, encoder runtime increased to 175 % for a luma coding gain of -0.47 % against VTM-2.0. No further experiments were conducted in light of these results.

- Geometric partitioning in a different implementation in combination with AMVP and Merge-mode signaling was tested in [BS18b; BS18a] by the author of this thesis. An overall coding gain over VTM-1.0 of -0.80 % for 250 % encoder runtime increase using bi-prediction was reported. Introducing significant encoder skipping to lower runtime and the memory bandwidth limitation to uni-prediction caused a significant drop in coding gain to -0.10 % for a reported encoder runtime increase of 162 % against VTM-2.0. Also, no further experiments were conducted in light of these results.

Due to these two separate findings, combining non-rectangular partitioning with AMVP is deemed as not providing a favorable coding gain vs. encoder complexity trade off and will therefore not be investigated further in this thesis.

MMVD however has been shown to give significant coding gain at 0.86 % for block-based prediction at the expense of 12 % encoder runtime increase and negligible decoder runtime increase [Chi+19]. An encoder-side benefit of MMVD lies in the fact that no extensive BMA

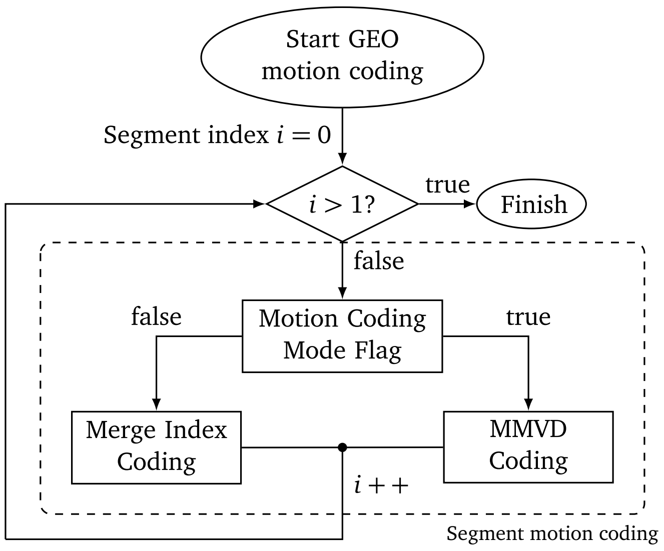


Figure 5.9 Signaling of the motion information for the combination of MMVD and GEO. The coder can signal, whether Mere-mode based coding or MMVD-mode based coding is used for each segment.

must be performed. Therefore, in the following, the possibility of combining MMVD and GEO is investigated.

For the combination of both tools, the motion information signaling is adapted according to Figure 5.9, to allow the switching between regular Merge-mode and MMVD-mode for GEO segments. For this purpose, a motion coding mode flag is introduced. If the value of the flag is false, the regular Merge-mode coding is invoked. Otherwise, the MMVD-mode coding is performed. In detail, the two coding modes operate as follows:

1. For the Merge-mode, an index is signaled for each segment of the GEO block. In accordance with the VVC draft for regular inter-predicted blocks using the Merge mode, a truncated-unary binarization $C_{\mathrm{TU},K}(\cdot)$ is used for the merge index. The first resulting bin is CABAC-coded while the remaining bins are bypass-coded. For the first segment, K is set to the maximum number of merge candidates as configured in the Sequence parameter set (SPS). In a RA coding configuration according to the JVET CTC for VTM-3.2, a maximum of $K = 6$ merge candidates are available. For the second segment, K is decreased by one, as both segments are not allowed to use the identical motion vector.

2. For the MMVD coding, the signaling is conceptually performed identically as for regular MMVD coded blocks. For each GEO segment, a base predictor is signaled by a flag, followed by a truncated-unary coding $C_{\mathrm{TU},8}(\cdot)$ of the magnitude. Finally, the direction of the MVD vector is signaled by a 2 bit fixed-length code, e.g. $C_{\mathrm{FL},2}(\cdot)$. In total, 64 different MVD vectors can be signaled per GEO segment.

The coding results for this Experiment 5.26 are detailed in Table A.32. As it is the case for the baseline Experiment 4.2, the partitioning is again explicitly signaled using a fixed-length code. Compared to the baseline experiment, the results demonstrate that the overall coding efficiency improves from −0.45 % to −0.56 % in terms of luma BD-rate change over VTM-3.2. Especially for the UHD class A2, coding gain improves considerably from −0.36 % to

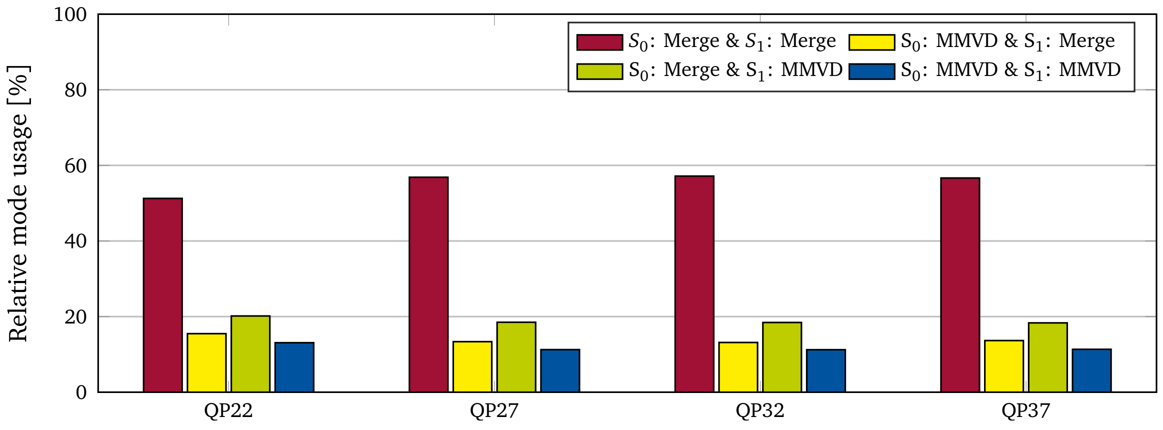


Figure 5.10 Relative motion vector prediction mode usage for Experiment 4.2, grouped by QPs.

−0.55 %. Also, for the WQVGA class D sequences, luma BD-rate coding gain improves from −0.36 % to −0.58 %, indicating that MMVD can be effectively combined with GEO across different resolutions. The relative motion coding mode usage (Merge-mode or MMVD) for this experiment has been collected for all sequences and is shown in Figure 5.10, grouped by QP. The statistics show that – independent of the chosen QP – roughly 50-55 % of all pixels coded with GEO are still predicted using the Merge-mode for both partitions S_0 and S_1. Approximately 10 % of GEO coded pixels are using MMVD for motion prediction of both segments. In the remaining cases, either the first segment S_0 or the second segment S_1 is using MMVD while the other is coded using Merge-mode. Interestingly, the usage of MMVD is slightly more frequent for the second segment S_1. As a reminder, the convention is made that the first segment S_0 always relates to the partition that includes the top-left sample of the block. Except for a few corner-cases, this generally positions the first segment closer to the top or left block boundary than the second segment. This may provide a reason for the unequal usage of MMVD as shown in Figure 5.10: The correlation between samples and the correlation between vectors of the motion vector field typically decreases when moving away from the top and left block boundary. Therefore, it appears sensible that the first segment, being qualitatively closer to the already coded samples and motion vectors, is predicted using Merge-mode. Although Merge-mode signaling may be less precise, it is cheaper in terms of the signaling cost. The prediction of the second segment, however, can benefit from MMVD, since more precision through signaling of a motion vector difference is available for motion compensation.

The encoder for this experiment was configured to run a full search of all available pairings of GEO partitions and MMVD or Merge-mode motion vectors. In total this relates to an upper bound of $(6+64)^2 \cdot N_{w,h}^{\text{AD}}$ different combinations of GEO partitions and motion vectors to be tested. Assuming a median number of $\tilde{N}_{w,h}^{\text{AD}} = 140$ GEO partitions across all block sizes, this gives approximately 686000 possible combinations. This compares to an upper bound of $6 \cdot 5 \cdot N_{w,h}^{\text{AD}}$ combinations to be tested without MMVD, e.g. 4200 for the same median number of $\tilde{N}_{w,h}^{\text{AD}} = 140$ GEO partitions. As will be shown in Section 5.5.1 on encoder mode control and the developed simplification strategies for VVC standardization in Section 6, this requires a staged estimation strategy to manage the additional encoder complexity.

5.4 Motion Vector Storage

Once a block has been coded in an inter-prediction mode at the encoder or decoder side, it is necessary to store the associated motion information in the motion buffer such that the motion information is available for subsequent spatial and temporal motion vector prediction. While this task is trivial for regular blocks resulting from a rectangular partitioning, additional steps are required for geometrically partitioned blocks. GEO coding units always encompass two different motion vectors that may be uni- or bi-directional in nature, as it is the case for conventional coding units. In a video coding scheme such as HEVC or VVC, the motion information is typically stored in the motion buffer with lower spatial resolution compared to the pixel resolution. This is a design choice made to decrease the memory requirements:

- The motion information in HEVC is stored on a 16×16 luma sample grid, e.g. each motion data storage unit can be associated with a corresponding block of 16×16 luma samples. Since the smallest possible prediction unit (PU) in HEVC for inter-predicted blocks has a size of 8×8 luma samples, a sub-sampling of the motion field is performed after a picture has been coded. For prediction blocks smaller than 16×16 samples, only the motion information from the top-left sub-block on the 16×16 grid is stored. This process is also termed motion data storage reduction [Wie14].
- The motion information in VVC is stored on a 4×4 luma sample grid. Since this size coincides with the smallest possible coding and prediction unit size, no sub-sampling-based motion data storage reduction process is required.

For GEO, the motion information of each segment shall be stored in the corresponding 4×4 sub-blocks of the motion buffer. This requires a mapping process that decides whether the motion information of the first segment $\boldsymbol{m}_{\mathrm{S},0}$ or the motion information of the second segment $\boldsymbol{m}_{\mathrm{S},1}$ is stored for a given sub-block. Multiple possible solutions exist for this given task, for example:

1. The decision can be made based on the blending filter mask $\boldsymbol{M}_\mathrm{f}$, see Equation (5.14), or the binary mask $\boldsymbol{M}_\mathrm{b}$,see Equation (5.13). For each 4×4 sub-block of the blending filter or binary mask, the coefficients can be summed and compared to a given threshold:

$$q_f(i,j) = \sum_{x=4i}^{4i+3} \sum_{y=4j}^{4j+3} M_\mathrm{f}(x,y), \quad 0 \le i < \frac{w}{4}, 0 \le j < \frac{h}{w} \tag{5.26}$$

   If $q_\mathrm{f}(i,j)$ is smaller than a predetermined threshold γ_f, the sub-block at position (i,j) is associated with the first segment and stores the motion information $\boldsymbol{m}_{\mathrm{S},0}$. Otherwise, the sub-block stores the motion information $\boldsymbol{m}_{\mathrm{S},1}$. If $\boldsymbol{M}_\mathrm{f}$ contains only integer weights with a maximum value of w_max, the threshold can be set at $\gamma_\mathrm{f} = 8w_\mathrm{max}$. This process is exemplified in Figure 5.11. Further simplifications can be applied, such as limiting the summation given in Equation (5.26) to the corner weights of each 4×4 sub-block.

2. Alternatively, the distance of each 4×4 sub-block to the partitioning line can be computed using Equations (5.9) or (5.11), depending on the representation of the partitioning line.

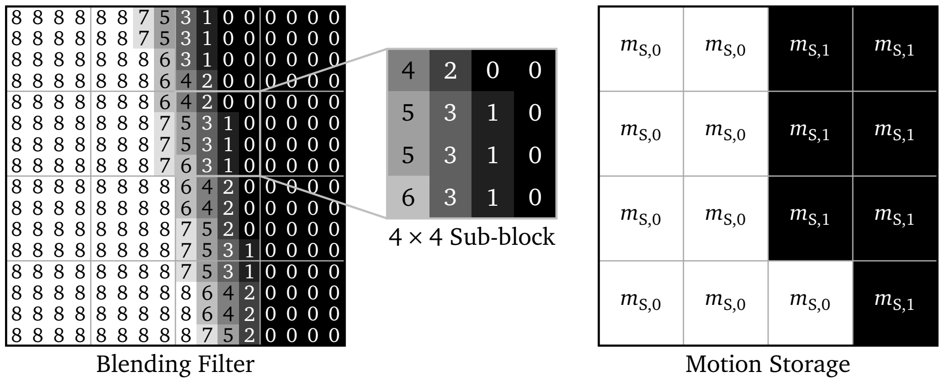


Figure 5.11 Visualization of the motion vector storage for GEO using the blending filter. The decision to store motion vector $\boldsymbol{m}_{\mathrm{S},0}$ or $\boldsymbol{m}_{\mathrm{S},1}$ is made based on the sum of the filter weights of each 4×4 sub-block.

The two approaches may generate slightly different mappings and have different algorithmic complexities. While the first approach only requires several additions, the second approach needs more computations per sub-block, depending on the implementation of the sample-distance equations. However, the second approach may be better parallelizable since the computation is not dependent on the generation of the filter or binary mask. However, these differences are only relevant for actual hardware implementations and no significant variation in terms of coding efficiency is expected.

5.5 Encoder Mode Control

A critical part of the GEO coding block tool is the design of the encoder. The side-information that needs to be transmitted for reconstruction at the decoder side is determined by means of rate-distortion optimization. In detail, this means that different coding parameters of the block tool are evaluated in terms of the respective rate that is required for signaling the particular choice of coding parameters and in terms of the distortion, e.g. the residual error that is achievable. For a coding tool it is a desirable feature to be able to exhaustively test all available coding parameters in terms of their actual RD-performance in reasonable time. In the HEVC and VTM reference encoders, the final mode decision is based on the distortion measured after transform coding using SSD and the rate in terms of fractional bits required by the arithmetic coder. However, since evaluating every coding mode in this manner would require the actual transform coding stage to be applied to the residual with Discrete cosine transform (DCT), Discrete sine transform (DST) or even Explicit multiple-core transform (EMT) computation and CABAC test-encoding to be performed, it becomes prohibitively expensive in terms of encoder runtime with a growing coding parameter space. Therefore, simplifications and heuristics are applied such that only parts of the entire parameter space are evaluated using the residual transform coding stage of the coder. For inter-prediction tools in the VTM reference encoder, the typical scheme that is applied for Merge-mode and AMVP-based tools can be broken down into three parts:

1. Deriving a first set of candidate coding parameters (motion vectors, merge index, AMVR settings, etc.) based on the luma SAD distortion of each motion compensated

prediction block and a simple lookup-table based model for the rate estimate $\tilde{R}_{\text{Motion}}$ of the side-information . The RD-cost is estimated as:

$$J_{\text{SAD}} = D_{\text{SAD}} + \sqrt{\lambda} \cdot \tilde{R}_{\text{Motion}} \tag{5.27}$$

2. From the candidate coding parameters with lowest RD-cost of the first set, deriving a second set of coding parameters using the luma SATD metric. The effect of transform coding is approximated by applying the Hadamard transform to the prediction residual. The RD-cost is estimated as:

$$J_{\text{SATD}} = D_{\text{SATD}} + \sqrt{\lambda} \cdot \tilde{R}_{\text{Motion}} \tag{5.28}$$

3. For those candidate coding parameters performing best from the second set, applying the full transform- and CABAC-coding to choose the final coding parameters. In this stage, the actual squared prediction error is measured across all luminance and chrominance components using the SSD metric. The final RD cost is calculated as:

$$J_{\text{Final}} = \frac{1}{\lambda} D_{\text{SSD}} + R_{\text{Motion}} + R_{\text{Transform}} \tag{5.29}$$

The goal of this approach is to find the global minimum in terms of actual RD-cost without exhaustively testing all available coding options. Consequently, this requires the application of heuristics to the estimation process, for example the maximum number of candidates to be evaluated in each step or RD-based thresholds controlling whether a candidate is skipped or is further tested. These heuristics have been carefully tuned and – in the context of the standardization activity – are subject to frequent change and further optimization.

The λ in the VTM reference encoder is determined mainly by the chosen QP and an intricate set of QP offsets and factors, with values depending on the coding structure, the slice type and further user-defined settings [And+16]. The lambda model for VTM-3.2 (and also HM versions 16.9 and later) can be summarized as follows:

$$\lambda = \lambda_{\text{mod}} \cdot 0.57 \cdot 2^{\frac{\text{QP}-\text{QP}_{\text{Offset}}}{3}} \cdot 2^{\frac{1}{12}} \tag{5.30}$$

with λ_{mod} being a user-defined lambda modifier with a default value of 1. The offset parameter $\text{QP}_{\text{Offset}}$ is set to a fixed value for I-slices (e.g. $\text{QP}_{\text{Offset}} = 3$ for RA) and for P- and B-slices determined from a linear model depending on the QP and a temporal distance:

$$\text{QP}_{\text{Offset}}(t_{\text{GOP}}, \text{QP}) = \Delta\text{QP}(t_{\text{GOP}}) + \text{QP} \cdot \text{QP}_{\text{ModelScale}} + \text{QP}_{\text{ModelOffset}} + 0.5 \tag{5.31}$$

with ΔQP,$\text{QP}_{\text{ModelScale}}$ and $\text{QP}_{\text{ModelOffset}}$ being configuration-dependent parameters and t_{GOP} an index relating to the hierarchical location of the current picture in the GOP. The correction term $2^{\frac{1}{12}}$ in Equation (5.30) is applied only for VTM to compensate for a slightly different quantizer slope due to dependent quantization.

Precise measurements for the overall encoding complexity for the GEO baseline experiment show an increase of about 25 % in terms of encoder runtime compared to the VTM-3.2 anchor without GEO. Figure 5.12 shows the relative computing time spent by different block tools. The data has been gathered by encoding a single CTU at QP32 for a duration of 33 pictures in a RA configuration and by measuring the execution time of each block tool using

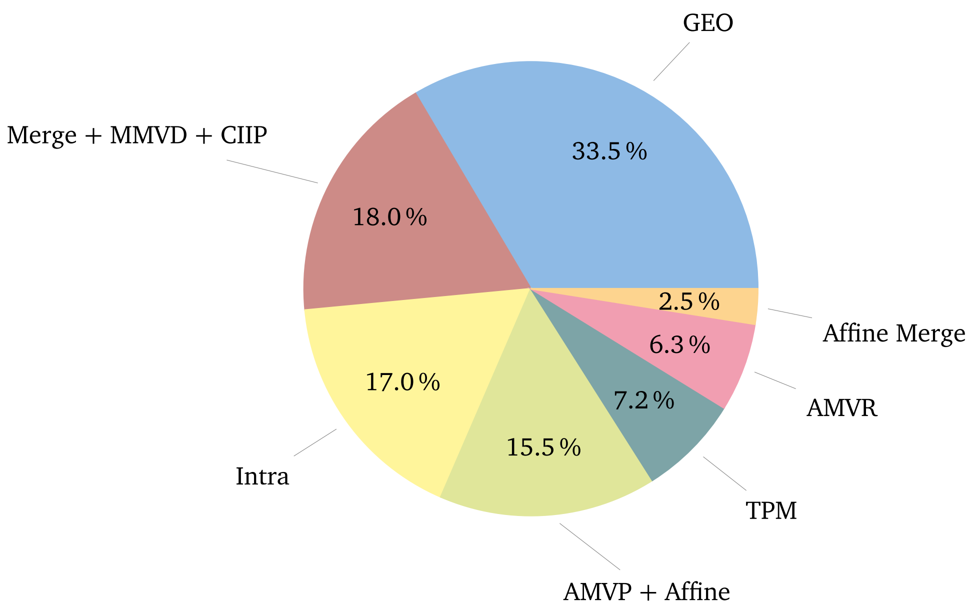


Figure 5.12 Relative encoding runtime distribution for individual coding modes of the VTM-3.2+GEO encoder. The GEO mode is tested by performing a full-search over all available GEO partitions. Statistics were gathered by encoding a single CTU at QP32 in a RA configuration for 33 pictures.

high precision CPU timers. Figure 5.12 reveals that the introduction of GEO significantly impacts the relative distribution of encoding complexity. One third of the entire runtime is attributed to the estimation of optimal GEO coding parameters.

5.5.1 Partitioning Mode Selection

The application of GEO and Merge-mode based signaling of motion vectors results in a coding parameter space with a total of $N_{\text{Merge}} \cdot (N_{\text{Merge}} - 1) \cdot N_{\text{GEO}}$ possible combinations, where N_{Merge} is the number of merge candidates and N_{GEO} the number of geometric partitioning modes per block. Assuming $N_{\text{Merge}} = 6$ and $N_{\text{GEO}} \approx 140$, it can be easily seen that a full search evaluating 4200 possible predictions after blending using transform coding, CABAC test-encoding and SSD is unfeasible. Therefore, a staged estimation strategy similar to those used in other coding tools of the VTM encoder is used in all experiments conducted in Chapters 4 and 5. The underlying idea behind the scheme is to generate only a small set of prediction candidates that are fed to the transform coding stage. Since determining the best prediction and coding methods for GEO is the main focus of the previous experiments, a scheme is proposed with very few manually tuned settings and short-cuts. This non-optimized encoding can be broken down into the following steps:

1. Perform block-based motion compensation for all available N_{Merge} merge candidates, e.g. $N_{\text{Merge}} = 6$. The result is a prediction block $\boldsymbol{P}_l$, $l \in \{0 \ldots N_{\text{Merge}} - 1\}$.

2. Calculate a segment-based SAD distortion of the luma component for each of the available N_{GEO} GEO partition modes and N_{Merge} motion compensated predictions from Step 1. This gives a total of $2 \cdot N_{\text{Merge}} \cdot N_{\text{GEO}}$ distortion estimates $D_{k,l,m}$ with $k \in \{0, 1\}$,

$l \in \{0 \dots N_{\text{Merge}} - 1\}$ and $m \in \{0 \dots N_{\text{GEO}} - 1\}$. The original picture is denoted as $\boldsymbol{P}_{\text{Org}}$.

$$D^{\text{SAD}}_{k,l,m} = \sum_{x,y} \left| \boldsymbol{M}_{\text{f},k,m} \circ (\boldsymbol{P}_l - \boldsymbol{P}_{\text{Org}}) \right| \tag{5.32}$$

3. Approximate the RD-cost $J_{l_1,l_2,m}$ with $l_1, l_2 \in l$ and $l_1 \neq l_2$ for each of the $N_{\text{Merge}} \cdot (N_{\text{Merge}} - 1) \cdot N_{\text{GEO}}$ possible combinations of merge candidates and GEO partition modes using the results from Step 2. Here, R_{l_1} and R_{l_2} are the bits required for signaling the motion information of each segment and R_m are the bits required to signal the GEO partition.

$$\tilde{J}_{l_1,l_2,m} \approx D^{\text{SAD}}_{0,l_1,m} + D^{\text{SAD}}_{1,l_2,m} + \sqrt{\lambda} \cdot (R_{l_1} + R_{l_2} + R_m) \tag{5.33}$$

   Then, determine the set of (l_1, l_2, m) tuples that minimize Equation (5.33):

$$\arg \min_{(l_1,l_2,m)} J_{l_1,l_2,m} \tag{5.34}$$

4. For the $0 \leq k < N_{\text{Blending}}$ tuples from Step 3 with lowest RD-cost, perform the blending process for the luma component and compute the SATD for the resulting block-based predictions. Determine the RD-cost for each candidate as:

$$J_k = D^{\text{SATD}}_{l_1,l_2,m} + \sqrt{\lambda} \cdot (R_{l_1} + R_{l_2} + R_m), \quad 0 \leq k < N_{\text{Blending}} \tag{5.35}$$

5. For the $N_{\text{Transform}} \leq N_{\text{Blending}}$ candidates from Step 4 with lowest RD-cost, perform the blending of all remaining components and apply residual coding and CABAC test-encoding. Compute the actual RD-cost using SSD.

6. Choose the best candidate from Step 5 as the final GEO prediction mode.

In all preceding experiments, the parameters N_{Blending} for SATD calculation and $N_{\text{Transform}}$ for transform coding were set to the value of $N_{\text{Blending}} = N_{\text{Transform}} = 20$. This conservative setting increases the probability that the GEO mode providing the absolute minimum in terms of RD-cost is selected. In Equation (5.33) of Step 3, a critical simplification is made by approximating the combined, block-based SAD using the summation of the segment-based SADs $D_{0,l_1,m}$ and $D_{1,l_2,m}$. Due to the overlapping segment predictions that are combined using the blending process, the absolute value in the form of the SAD metric used in Equation (5.32) is not an LTI system. The maximum error per sample in the transition zone that can occur compared to the block-based SAD computed after blending is equal to $2^{\text{B}_\text{d}-1}$, where B_d is the bit-depth of the video. Although the relative amount of samples in the transition zone is rather small for large blocks – about 4% for a 128×128 block and transition zone size $d_\text{m} = 3$,see Section 5.1.1 – it increases significantly for smaller blocks to about 50% for a 8×8 block and transition zone size $d_\text{m} = 3$. Therefore, Step 4 is introduced which performs the blending process and calculates an SATD for the composited prediction block. This step is critical and has been shown to improve BR-rate coding gain by up to 0.1% on the entire test-set compared to an encoder using the best candidates from Step 3 for the subsequent transform coding stage.

Figure 5.13 provides an analysis of the relative complexity in terms of encoder runtime spent in the individual steps of the proposed scheme (Step 1: Motion Compensation, Step 2:

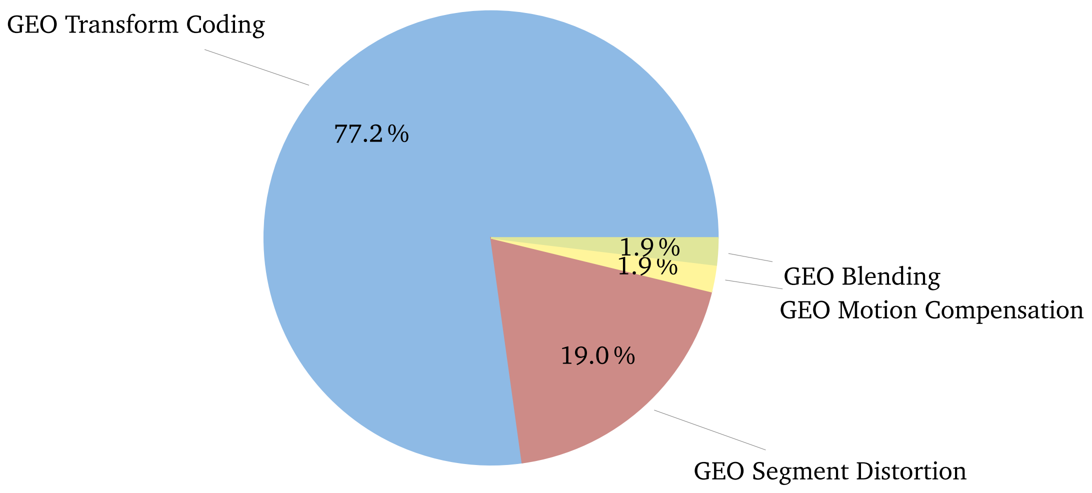


Figure 5.13 Relative encoding runtime distribution for the separate stages of the VTM-3.2+GEO encoder. Note that most of the encoding time is spent in the transform coding stage, which is identical for all inter-prediction modes.

Segment Distortion, Step 4: Blending and Step 5: Transform Coding). The data for this analysis was gathered by encoding one CTU at QP32 for 33 pictures in a RA configuration and measuring the execution time of each step using high precision CPU timers. It can be seen that 77.2 % of the entire encoding time for GEO is spent in the transform coding stage that is unaltered by GEO and identical for all inter-prediction coding tools in the VTM-3.2 encoder. The second largest percentage is taken up by the segment-based SAD computation with 19 % of the encoding time spent. Lastly, the motion compensation and the blending processes can be largely neglected in terms of encoding complexity since both steps only make up 1.9 % of the encoding time each.

5.6 Summary

Inter-prediction using geometric partitioning requires additional modifications to the overall hybrid video coding scheme. Most notably, the two prediction segments are combined using a blending filter, which performs OBMC along the partition boundary. The length of the overlapping region – the size of the transition zone – has been experimentally optimized to provide highest coding efficiency for a wide variety of video content. It is also recognized that the optimal length depends on the resolution and video content type.

Motion compensation with bi-directional prediction provides highest compression efficiency but the theoretical worst-case memory bandwidth requirement doubles, due to the potential computation of 4 inter-prediction hypothesis. The actual bandwidth increase might be lower, considering that a cached memory access is typically used by hardware decoders. Limiting the motion compensation process to uni-prediction as a solution comes at the cost of halving the overall coding efficiency of GEO.

Merge-based motion vector prediction and coding methods are suited for GEO, due to the efficient signaling and small parameter space that needs to be searched. Overall coding efficiency can be increased by an additional −0.1 % using MMVD for motion vector prediction, with higher gains reported for UHD sequences. Encoder mode control is an important aspect, since the additional partitioning options can increase the encoder run time substantially, as reviewed in Chapter (3). As a solution, a staged estimation strategy is presented that selects the RD-optimal partitioning without using heuristics of skipping certain geometric shapes.

Minor changes are further required for motion estimation as well as motion vector storage. The findings and conclusions of this chapter are the basis for optimizations that were conducted in the context of the JVET standardization activity. These are detailed in the subsequent Chapter (6).

6 Optimizations for VVC Standardization

The previous experiments were conducted using a non-optimized software, such as an encoder that exhaustively tests for the best overall combination of GEO partitioning mode and inter-prediction side-information, e.g. motion vectors. This deliberate choice was made since the preceding chapters focused on the development and analysis of techniques and algorithms that maximize coding efficiency while regarding issues such as algorithm complexity, ease of implementation and memory bandwidth requirements only as subsidiary.

For the standardization of a block-based coding tool like GEO however, these aspects are of great interest and require careful revision. Decoding complexity for example, which is assessed in terms of relative decoder runtime, is typically considered to be more critical than encoding complexity. A coding tool operating on the block level must be designed such that hardware implementations with very high throughput are achievable. Often, this requires parts of the algorithm to be simplified to such an extent that they only approximate the original model. Another important aspect for video coding applications is the ability to perform all required decoder-side computations with integer arithmetic. This ensures platform- and architecture independence compared to floating-point variants.

Lastly, the goal of the standardization process for a video coding scheme is to develop an unambiguous, written description of a decoder that correctly decodes a valid bitstream. A clean and simple design of an algorithm is therefore a desirable feature that more easily translates into brief and precise specification text.

In this chapter, the aforementioned aspects are reviewed for the GEO coding tool and simplifications and heuristic approaches are presented that were developed during the JVET standardization activity for VVC. The following constraints are imposed on GEO compared to Chapters 4 and 5 while trying to maintain as much coding gain as possible:

- Decoder and encoder complexity shall be comparable or lower than TPM.
- Motion compensation shall be restricted to uni-prediction.
- A uniform number of GEO partitions across different block sizes shall be specified.
- The overall operation of GEO and TPM shall be harmonized.

The result of these constraints is a design of the GEO coding tool that largely resembles the TPM coding tool in terms of its general operation. Motion vector prediction is performed through the Merge-mode as it is the case for TPM. Further, only explicit signaling of the GEO partitioning is performed using truncated binary coding. To limit the memory bandwidth of the coding tool, only uni-prediction is allowed for GEO which is the same constraint imposed for TPM.

The additional constraint to specify a uniform number of additional GEO splits for different block sizes is merely a design choice made to simplify the specification of angle and distance parameters and the blending process. This requires the reiteration of the optimization process that was already detailed in Section 4.2 to determine a set of optimal angle and distance partitioning parameters. Lastly, any floating-point operations involving trigonometric functions are replaced by integer arithmetic approximations in combination with look-up tables.

Investigations on the optimal blending length performed in Section 5.1.1 already indicated that a sharper blending filter is beneficial in terms of coding gain for pure screen content in class F. This finding is integrated into the GEO design by making the blending filter content-adaptive, e.g. it can be turned on or off depending on the video content by a high-level flag.

The following sections summarize the efforts taken to optimize the individual aspects of the GEO coding tool according to aforementioned criteria and previous findings. The result of these optimizations are contributions to JVET [Ese+19a; Ese+19b] that spawned new core experiments to be conducted [CYX19]. At last, GEO was adopted into the working draft version 8 of VVC as a generalization of TPM. The code basis for these developments was updated to VTM-5.0 which includes new additional coding tools such as Luma mapping and chroma scaling (LMCS) (formerly known as a *reshaper*) as a major novelty, SBT for inter coded blocks, Symmetric motion vector difference (SMVD) and a multitude of improvements and simplifications for existing coding tools. For an overview, the reader shall refer to [BCL18; CYK19].

6.1 Geometric Partitioning Parameters

For the update of the software to VTM-5.0 and the imposed uni-prediction restriction, the number and distribution of geometric partitioning parameters using the angle-distance representation is re-optimized. Based on the findings in Section 4.2.3.2, an angular spacing of $\Delta\varphi = 11.25°$, equal to $N_\varphi = 32$ angles is kept for all block sizes. The quantized angle φ_m is derived accordingly as $\varphi_m = m \cdot \Delta\varphi$ with $m \in \{0...N_\varphi - 1\}$. For the distance ρ, different quantizer settings $N_\rho = \{3,4,5,6,7\}$ are again evaluated according to the JVET CTC. The spacing of the distance $\Delta\rho(\varphi_m, w, h,)$ for a block of size $w \times h$ in luma samples, such that $\rho_n = n \cdot \Delta\rho(\varphi_m, w, h)$ with $n \in \{0...N_\rho - 1\}$ is determined according to:

$$\Delta\rho(\varphi_m, w, h) = \frac{\rho_{\max}(\varphi_m, w, h) - \rho_{\text{th}}(w, h)}{N_\rho} \tag{6.1}$$

$$\rho_{\max}(\varphi_m, w, h) = \cos(\varphi_m)\left[\frac{h}{2\tan(\frac{\pi}{2} - \varphi)} + \frac{w}{2}\right], \quad 0 \leq \varphi_m < \frac{\pi}{2} \tag{6.2}$$

$$\rho_{\text{th}}(w, h) = \rho_{\text{th},0} \cdot 2^{\log_2(\min(w,h)) - 3} \tag{6.3}$$

For the software implementation in integer arithmetic and the specification text, the values of $\Delta\rho(\varphi_m, w, h)$ are not computed at runtime but stored in a look-up table V_{Dist}. The floating-point values as shown in Equation 6.1 are scaled and rounded to integer values:

$$\Delta\rho_{\text{int}}(\varphi_m, w, h) = \left\lfloor \Delta\rho(\varphi_m, w, h) \cdot 2^{B_\text{i}} + 0.5 \right\rfloor \tag{6.4}$$

The value of the scaling factor B_i is matched to the bit-depth required for the per-sample distance equation, as will be discussed in Section 6.6.1 subsequently. In the final proposal for GEO, a bit-depth of $B_i = 7$ is chosen. Note that the size of the look-up table V_{Dist} that

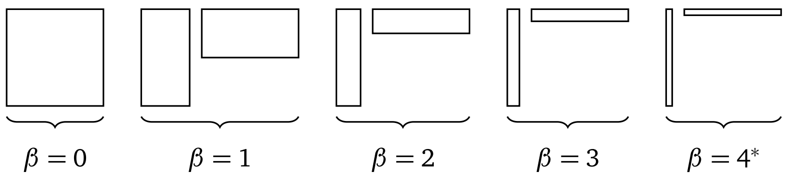


Figure 6.1 Visualization of the block aspect ratios β that are allowed for GEO. *Note that blocks with $\beta = 4$ cannot be reached by the MTT under the JVET CTC for VTM-5.0.

contains the distance spacing dependent on the angle and the block size can be significantly reduced by exploiting symmetries. As shown previously in Figure 4.9, it is sufficient to only specify the angles in the first quadrant, e.g. $0 \leq \varphi_m \leq 90°$ and map the remaining quadrants to the first quadrant accordingly. Furthermore, not all 25 available block sizes from 8×8 to 128×128 need to be specified. It is sufficient to consider only the maximum aspect ratio denoted β of a given block, and scale the distance step size in relation to an 8×8 block:

$$\beta(w,h) = \log_2\left(\frac{\max(w,h)}{\min(w,h)}\right) \tag{6.5}$$

$$\gamma(w,h) = \begin{cases} w/8 & w \geq h \\ h/8 & w < h \end{cases} \tag{6.6}$$

$$\Delta\rho_{\text{int}}(\varphi_m, w, h) = V_{\text{Dist}}(m, \beta) \cdot \gamma \tag{6.7}$$

In total, the look-up table $V_{\text{Dist}}(m,\beta)$ contains 45 entries to fully specify the angle-dependent distance step size $\Delta\rho_{\text{int}}$. The aspect ratios for different values of β are visualized in Figure 6.1.

Equation (6.3) has been updated from Equation (4.21) by also linearly scaling the distance margin $\rho_{\text{th},0}$ relating to an 8×8 block according to the current block size. A larger value of $\rho_{\text{th},0}$ moves the partitioning lines closer to the center of the block. This avoids unequally distributed geometric splits. Base margins of $\rho_{\text{th},0} = 0.0, 0.5, 1.0, 1.5, 2.0, 2.5$ are evaluated according to the JVET CTC, except for classes A1 and A2, for the best combination of N_φ and N_ρ.

The coding results shown in Tables 6.1 and 6.2 lead to the following choice for the geometric partitioning parameters:

- A distance quantization with $N_\rho = 5$ steps is selected for the final codec. The combination of $N_\varphi = 32$ angles and $N_\rho = 5$ distances provides the highest coding gain, measured by a luma BD-rate reduction of −0.24 % with the least number of geometric splits.

- A distance margin of $\rho_{\text{th},0} = 1.5$ is chosen, as this setting provides the highest coding gain for UHD and HD sequences.

In total, this combination of parameters results in $N_{\text{GEO}} = N_\varphi \cdot N_\rho - {}^{N_\varphi}\!/_2 - 4 = 140$ geometric splits per block size $w \times h$, since the diagonal and anti-diagonal splits as well as the horizontal and vertical splits for $\rho = 0$ are excluded to prevent the emulation of these splits. These resulting 140 geometric splits are visualized in Figure 6.2 for a square shaped block, grouped

Class	Luma BD-rate change over VTM-5.0				
	$N_\rho = 3$	$N_\rho = 4$	$N_\rho = 5$	$N_\rho = 6$	$N_\rho = 7$
A1	−0.07 %	−0.07 %	−0.11 %	−0.07 %	−0.10 %
A2	−0.16 %	−0.16 %	−0.15 %	−0.16 %	−0.17 %
B	−0.10 %	−0.10 %	−0.09 %	−0.08 %	−0.11 %
C	−0.46 %	−0.51 %	−0.54 %	−0.54 %	−0.53 %
Overall	**−0.20 %**	**−0.22 %**	**−0.23 %**	**−0.22 %**	**−0.23 %**
D	−0.16 %	−0.15 %	−0.21 %	−0.16 %	−0.17 %
F	−0.14 %	−0.14 %	−0.18 %	−0.18 %	−0.16 %

Table 6.1 Optimization of the distance parameter for the angle-distance representation of GEO.

Class	Luma BD-rate change over VTM-5.0					
	$\rho_{\text{th},0} = 0.0$	$\rho_{\text{th},0} = 0.5$	$\rho_{\text{th},0} = 1.0$	$\rho_{\text{th},0} = 1.5$	$\rho_{\text{th},0} = 2.0$	$\rho_{\text{th},0} = 2.5$
B	−0.09 %	−0.10 %	−0.09 %	−0.11 %	−0.09 %	−0.09 %
C	−0.53 %	−0.54 %	−0.53 %	−0.55 %	−0.54 %	−0.53 %
D	−0.15 %	−0.19 %	−0.15 %	−0.16 %	−0.20 %	−0.14 %
F	−0.16 %	−0.20 %	−0.20 %	−0.17 %	−0.16 %	−0.16 %
Mean	**−0.22 %**	**−0.25 %**	**−0.23 %**	**−0.24 %**	**−0.24 %**	**−0.22 %**

Table 6.2 Optimization of the base margin $\rho_{\text{th},0}$ for the angle distance representation of GEO.

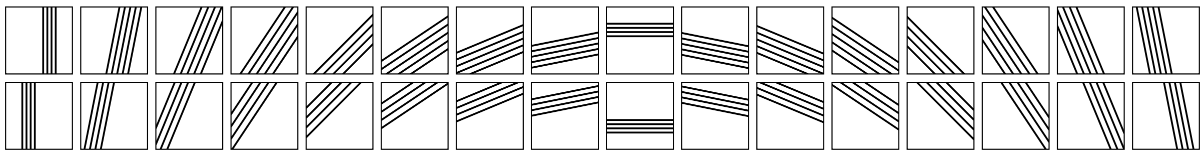


Figure 6.2 Visualization of all 140 geometric splits for VTM-5.0, grouped by identical angles. The splits are shown for a square block size.

by identical angles φ_m, $m \in \{0...31\}$. It can be seen that very small geometric partitions, e.g. segments likely containing only a few samples, are avoided with this scheme.

The code book of allowed partitioning parameters m and n are stored at the encoder and decoder using a look-up table $V_{\text{Param}} = \{\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{A}_k, ..., \boldsymbol{A}_{N_{\text{GEO}}-1}\}$ with $\boldsymbol{A}_k = [m_k \; n_k]$, therefore only the partition index k must be signaled. For simplicity and in accordance with the findings from Section 4.2.2 on entropy coding, a truncated binary binarization and subsequent CABAC bypass-coding of the partitioning index is performed. The truncated binary code $C_{\text{TB},140}(k)$ results in a mean code length of 7.17 bit for the partitioning index k.

If the decoder decodes a partition index k, the look-up table V_{Param} returns the quantized angle and distance indices m and n respectively for the given k, e.g. $V_{\text{Param}} : k \mapsto (m, n)$. According to the quantizer settings for the angle and distance, V_{Param} contains 140 entries. Then, in the next step, the quantized distance step size is reconstructed using the angle index m, the aspect ratio β and the table V_{Dist} as shown in Equations (6.5)-(6.7) above.

The entire signaling of the geometric partitioning mode is performed in the Merge-mode branch. The syntax in accordance with the Versatile Video Coding (Draft 5) specification [BCL18] is shown in Table 6.3. For all inter-predicted coding blocks for which the Merge-mode is signaled and with a width and height larger than 8 luma samples, a syntax element geo_flag is coded. The syntax element geo_flag is CABAC coded using three context models, chosen based on the value of geo_flag in adjacent coding units:

- If geo_flag[xL][yL] or geo_flag[xT][yT] is equal to 1, with xL, yL specifying the coordinates of the left coding unit, adjacent to the current coding unit and xT, yT specifying the coordinates of the above coding, adjacent to the current coding unit, the first context model is used.
- If geo_flag[xL][yL] and geo_flag[xT][yT] is equal to 1, the second context model is used
- Otherwise, the third context model is used.

6.2 Motion Compensation

Despite the reduced coding efficiency, standardization constraints imposed the restriction to uni-prediction for all GEO coded blocks, due to the doubling in theoretically required, worst-case memory bandwidth. Since only uni-prediction is allowed, two predicted samples $\tilde{p}_{\text{L}i,0}$ and $\tilde{p}_{\text{L}j,1}$ are involved in the blending process. The motion compensation process for GEO, detailed in Section 5.1 is implemented for VTM-5.0 as described by Equation (5.5):

merge_data(x0, y0, cbWidth, cbHeight) {	**Descriptor**
if (cbWidth > = 8 && cbHeight >= 8){	
geo_flag[x0][y0]	ae(v)
}	
if (geo_flag[x0][y0]) {	
geo_partition_idx[x0][y0]	ae(v)
geo_merge_idx0[x0][y0]	ae(v)
geo_merge_idx1[x0][y0]	ae(v)
}	
else {	
... *other Merge-modes*	
}	
}	

Table 6.3 Syntax table for the signaling of GEO in the Merge-mode branch, in accordance with the VVC (Draft 5) specification.

$$p_{\mathrm{PB}} = \left\lfloor \frac{w_0 \tilde{p}_{\mathrm{L}i,0} + w_1 \tilde{p}_{\mathrm{L}j,1}}{2^{14+B_\mathrm{f}-B_\mathrm{d}}} + \frac{1}{2} \right\rfloor, \quad i,j \in \{0,1\} \qquad \text{(5.5 revisited)}$$

Prior to the motion compensation step, a per-sample weight $w_{\mathrm{GEO}}(x,y)$ has been derived and assigned to two temporary buffers $w_0(x,y) = w_{\mathrm{GEO}}(x,y)$ and $w_1(x,y) = 8 - w_{\mathrm{GEO}}(x,y)$. Since all required samples and weights are available in arrays of memory, Equation (5.5) can be efficiently implemented using SIMD code. Example code is shown in Listing A.3 for vectors of 4×16 bit length and in Listing A.4 for 8×16 bit vectors in the Appendix .

Similarities between the motion compensation process for GEO and BCW can be noted: As it is the case for GEO, BCW performs a weighted prediction with two weights w_{L0} and w_{L1} in the range from 0 to 8 such that $w_{\mathrm{L0}} + w_{\mathrm{L1}} = 8$. However, for BCW, the weights apply to entire prediction blocks, whereas for GEO they are applied to individual samples.

6.3 Adaptive Blending Filter

The investigations performed in Section 5.1.1 showed that a dependency between the achievable coding gain for a given blending filter length and the video properties exists:

- Coding gain for sub-HD low resolution improves using sharper filters with $d_\mathrm{m} < 5$ luma samples across.
- Coding gain for HD and UHD resolutions improves using more smoothing, e.g. a longer blending filter with $d_\mathrm{m} \geq 5$ across.
- Coding gain for pure screen and computer-generated content improves using a very sharp blending filter.

To balance these different requirements for video resolution and content classes and to keep the specification simple, an adaptive filter is proposed in the final design that has two different settings:

1. A soft masking filter for natural video content is used with a quantized weighting function that approximates a linear transition between the two prediction segments.
2. A hard masking filter for screen and computer-generated content is used with no transition zone between the two segments, e.g. only the weights $w = 0$ or $= 8$ are assigned.

The choice of the filter is signaled at the sequence level in the SPS. Since GEO is a coding tool proposed as an extension of TPM, an adaptive blending filter for the two TPM splits is also proposed.

6.3.1 Blending Filter for GEO

The investigations in Section 5.1.1 showed that a soft masking filter with a transition zone of approximately 3 to 5 luma samples across has good performance for the entire JVET test set containing natural video content. Therefore, a filter $f_\mathrm{q}(\cdot)$ as shown in Figure 6.5 is specified in the final proposal for GEO that quantizes the integer distance d_int – given in sub-pixel accuracy – of each sample to weight values w_GEO in the range from 0 to 8, approximating a linear transition:

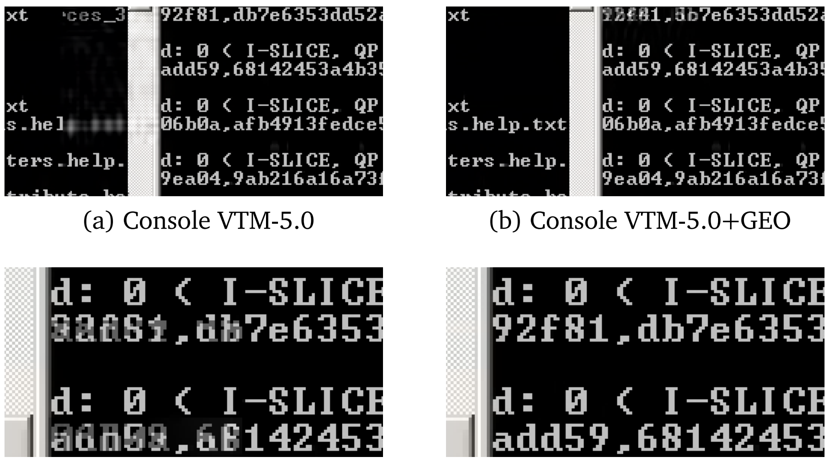
$$w_\mathrm{GEO}(x,y) = f_\mathrm{q}\left(d_\mathrm{int}(x,y)\right) \tag{6.8}$$

The hard masking filter for screen content, also indicated in Figure 6.5, is derived based on the polarity of d_int :

$$w_\mathrm{GEO}(x,y) = \begin{cases} 0 & d_\mathrm{int}(x,y) \leq 0 \\ 8 & d_\mathrm{int}(x,y) > 0 \end{cases} \tag{6.9}$$

Figure 6.4 shows an example of a soft masking mask for natural video content and the corresponding hard masking mask for screen content for a coding block of size 16×32 luma samples. The coding results over VTM-5.0 with hard masking for GEO and soft masking (e.g. unchanged compared to the anchor) for TPM are shown in Table 6.4. For pure screen content, as included in the text with graphics and motion (TGM) sequence class, an overall BD-rate change of −1.95 % over VTM-5.0 can be achieved. Especially for the sequences *Desktop* and *Console* the hard masking is effective, with rate reductions measured at −2.79 % and −2.62 %, respectively. Lower luma BD-rate changes are measured for class F at −0.31 % over VTM-5.0. However, class F also contains the *BasketballDrillText* sequence, which is a natural video sequence with news ticker style graphics at the bottom of the sequence. Accordingly, lower coding efficiency is observed for such a mixed sequence with a luma BD-rate change measured at −0.14 %.

Visual inspection of *Desktop* and *Console* coded with GEO compared to the anchor also revealed that coding artifacts for high QPs are noticeably reduced. Two examples are shown in Figure 6.3. It can be seen that with GEO, phantom prediction errors that occur in the scrolling text disappear. This can be explained by the additional horizontal splits that are available to the coder with GEO. These additional splits are heavily used whenever vertically scrolling graphics or text appear in the video.



```
xt        ces_3   92f81,db7e6353dd52
                  d: 0 < I-SLICE, QP
                  add59,68142453a4b3
xt                d: 0 < I-SLICE, QP
s.hel             06b0a,afb4913fedce
ters.help.        d: 0 < I-SLICE, QP
                  9ea04,9ab216a16a73
```

(a) Console VTM-5.0

```
xt                92f81,db7e6353dd52
                  d: 0 < I-SLICE, QP
                  add59,68142453a4b3
xt                d: 0 < I-SLICE, QP
s.help.txt        06b0a,afb4913fedce
ters.help.        d: 0 < I-SLICE, QP
                  9ea04,9ab216a16a73
```

(b) Console VTM-5.0+GEO

```
d: 0 < I-SLICE
92f81,db7e6353

d: 0 < I-SLICE
add59,68142453
```

(c) Desktop VTM-5.0

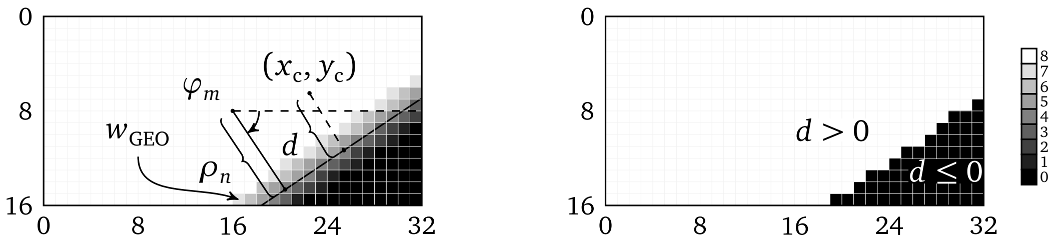
```
d: 0 < I-SLICE
92f81,db7e6353

d: 0 < I-SLICE
add59,68142453
```

(d) Desktop VTM-5.0+GEO

Figure 6.3 Visual examples showing reduced coding artifacts for sequences Console and Desktop at QP37 using GEO with hard masking.

Class	BD-rate change over VTM-5.0				
	Y	U	V	EncT	DecT
F	−0.31 %	−0.36 %	−0.22 %	102 %	101 %
TGM	−1.95 %	−1.61%	−1.59 %	99%	100 %

Table 6.4 Coding results for VTM-5.0+GEO with hard masking.



(a) Soft masking for natural content (b) Hard masking for screen content

Figure 6.4 Examples of soft and hard masking masks for GEO, which are generated depending on the video content type using the adaptive blending filter.

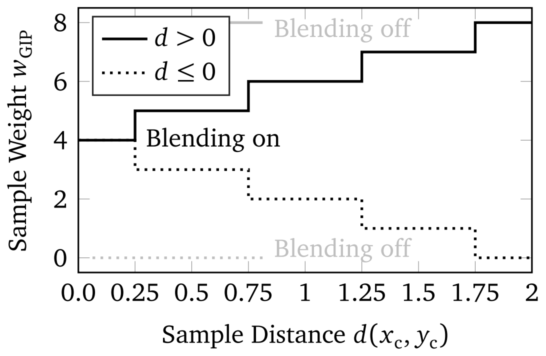


Figure 6.5 Integer mapping function of the sample distance d for derivation of blending weights w_{GEO}. Note that two different filters can be used: a first filter for natural video content and a second filter for screen and computer graphics.

6.3.2 Blending Filter for TPM

Since TPM is regarded as a separate coding tool in this thesis, the adaptive blending filter can also be easily integrated into this coding mode. In TPM, the per-sample weight for each triangle split is derived from the Manhattan distance of each pixel to the diagonal or anti-diagonal partitioning line:

$$d_{\text{TPM},0}(x,y) = \frac{x}{a} - \frac{y}{b} + 4 \qquad \text{(5.19 revisited)}$$

$$d_{\text{TPM},1}(x,y) = \begin{cases} h - 1 - \frac{x}{a} - \frac{y}{b} & w > h \\ w - 1 - \frac{x}{a} - \frac{y}{b} & w \leq h \end{cases} \qquad \text{(5.20 revisited)}$$

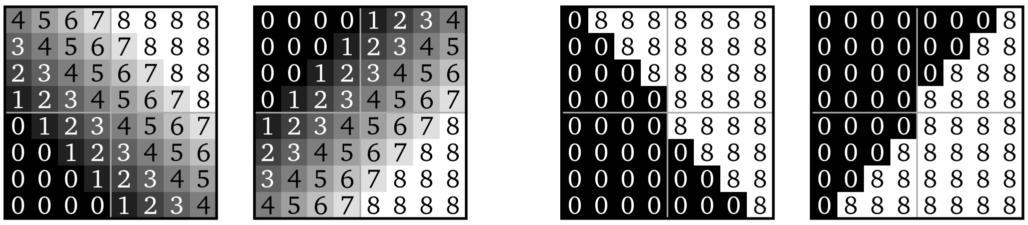
The corresponding sample weight is derived by clipping $d_{\text{TPM,i}}, i \in \{0,1\}$ to the value range from 0 to 8. The hard masking can now be achieved in the following manner, by changing the weight derivation process:

$$d_{\text{TPM,hard},0}(x,y) = \begin{cases} 0 & \frac{x}{a} \leq \frac{y}{b} \text{ and } y < \frac{h}{2} \\ 0 & \frac{x}{y} < \frac{y}{b} \text{ and } y \geq \frac{h}{2} \\ 8 & \text{otherwise} \end{cases} \tag{6.10}$$

$$d_{\text{TPM,hard},1}(x,y) = \begin{cases} 8 & w - 1 - \frac{x}{a} \leq \frac{y}{b} \text{ and } y < \frac{h}{2} \text{ and } w \leq h \\ 8 & w - 1 - \frac{x}{a} < \frac{y}{b} \text{ and } y \geq \frac{h}{2} \text{ and } w \leq h \\ 8 & h - 1 - \frac{x}{a} \leq \frac{y}{b} \text{ and } y < \frac{h}{2} \text{ and } w > h \\ 8 & h - 1 - \frac{x}{a} < \frac{y}{b} \text{ and } y \geq \frac{h}{2} \text{ and } w > h \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

The resulting hard masking filters are shown in Figure 6.6 , next to the soft masking filters. It can be seen that Equations (6.10) and (6.11) generate filters which contain an equal number of samples in each triangle. Furthermore, Equations (6.10) and (6.11) remove the clipping operation.

It is noted that other proposals for hard masking filters have also been made that assign all samples on the diagonal to one specific triangle, resulting in an unequal number of samples



(a) Soft TPM blending for natural content (b) Hard TPM blending for screen content

Figure 6.6 Examples of soft and hard masking masks for TPM. The numbers indicate the per-sample weights $d_{\text{TPM},0}$ for the soft mask and $d_{\text{TPM,hard},0}$ for the hard mask, see Equations (5.19) and (6.10).

Class	BD-rate change over VTM-5.0				
	Y	U	V	EncT	DecT
F	0.34 %	0.76 %	0.05 %	101 %	102 %
TGM	−0.68 %	−0.66 %	−0.54 %	102 %	105 %

Table 6.5 Coding results for VTM-5.0 with hard masking for TPM.

predicted by each triangle [Zhu+19; Che+19a]. The coding results for classes F and TGM using hard masking filters are shown in Table 6.5. It can be seen that the coding gain is significantly lower if only hard masking for TPM is used and no GEO partitioning is available. For the TGM class, a luma BD-rate change of −0.68 % is reported. For class F a luma BD-rate change of 0.34 % can be reported. The loss for class F can be attributed to the *BasketballDrillText* sequence with a 0.71 % luma BD-rate change and the *ArenaOfValor* sequence with 0.24 % luma BD-rate change. These two sequences do not have the special characteristics of pure screen content. For *Desktop* and *Console*, luma BD-rate changes of −1.09 % and −0.93 % can be reported. This result again underlines the benefit of the additional partitioning modes provided by GEO, especially for pure screen content

6.3.3 Combined Results

The two different blending filters for GEO and TPM can also be used in a combined fashion for higher adaptability to the given content. In total, three different configurations are possible:

1. Hard masking with GEO and soft masking with TPM. This is identical to the setup specified in Section 6.3.1.
2. Soft masking with GEO and hard masking with TPM.
3. Hard masking with GEO and hard masking with TPM.

The coding results in terms of luma BD-rate change for these configurations are provided in Table 6.6. If soft masking with GEO and hard masking with TPM is configured, the luma BD-rate for TGM improves from −0.68 % using hard masking with TPM and no geometric partitioning (see Section 6.3.2) to −0.88 %, due to the additional GEO splits. The highest

Class	Luma BD-rate change over VTM-5.0		
	GEO:H, TPM:S	GEO:S, TPM:H	GEO:H, TPM:H
F	−0.31 %	−0.09 %	−0.14 %
TGM	−1.95 %	−0.88 %	−2.15 %

Table 6.6 Coding results for VTM-5.0+GEO for different blending configurations. H: hard masking, S: soft masking.

coding gain for TGM can be achieved if both TPM and GEO are using hard masking – the coding gain then improves to −2.15 %. For the mixed content class F, the best trade-off can be achieved if hard masking is used for GEO and soft masking for TPM. With this setup, the encoder can actually choose which blending fits best for a given block. It is noted that this configuration type would become unavailable if the TPM splits are integrated into GEO.

In summary it can be stated that an adaptive blending filter in combination with GEO significantly improves the coding efficiency for pure screen content. The sharp edges and often geometrical shapes occurring in screen content are a good fit for the GEO coding tool. For high QPs, GEO can also help in removing coding artifacts.

6.4 Motion Vector Prediction, Coding and Storage

The motion vector prediction, coding and storage for GEO are aligned with the methods used by TPM:

- Motion vector prediction is achieved through a merge candidate list of uni-directional motion vectors. The merge candidate list is filled from the five spatial positions A_1, B_1, B_0, A_0 and B_3 (see Figure 4.27). Bi-directional motion vectors are decomposed into the uni-directional components in this scheme. The merge candidate list is filled to a maximum of 5 motion vectors.

- Motion vector coding is performed by signaling two merge indices. A first merge index k_0 in the range from 0...4 is signaled for the first segment using a combination of a context coded bin, indicating $k_0 = 0$ or $k > 0$ and truncated-unary coding of the remainder. Then, a second merge index k_1 in the range from 0...3 is signaled in the same manner. The second merge index has a lower range since it is avoided that both segments are coded with the same vector.

- Motion vector storage is performed by storing the two motion vectors $\boldsymbol{m}_{S,0}$ and $\boldsymbol{m}_{S,1}$ including the reference indices on a 4×4 grid basis, as detailed in Section 5.4. Additionally, a third bi-directional motion vector is constructed from the two uni-directional motion vectors as specified for TPM [BCL18]. This bi-directional motion vector is stored for those 4 × 4 sub-blocks that are located in the transition zone between the two GEO segments. The decision, which motion vector is to be stored for a given 4 × 4

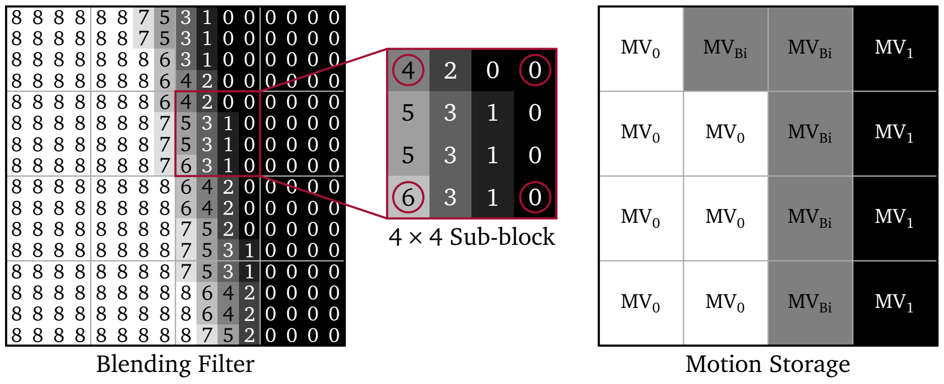


Figure 6.7 Illustration of the GEO motion vector storage derivation using blending weights for $\varphi_m = 11.25°$ and $\rho_n = 1.28$. The sum of the corner weights indicated in red is used to assign the motion information to each of the 4×4 sub-blocks.

sub-block, is made depending on the corresponding filter 4 × 4 sub-block $\boldsymbol{M}_{\mathrm{f},i,j}$:

$$\boldsymbol{M}_{\mathrm{f}} = \begin{bmatrix} \boldsymbol{M}_{\mathrm{f},0,0} & \cdots & \boldsymbol{M}_{\mathrm{f},0,W-1} \\ \vdots & \boldsymbol{M}_{\mathrm{f},i,j} & \vdots \\ \boldsymbol{M}_{\mathrm{f},H-1,0} & \cdots & \boldsymbol{M}_{\mathrm{f},H-1,W-1} \end{bmatrix} \quad 0 \leq i < H,\ 0 \leq j < W \tag{6.12}$$

During the GEO motion information storage process, the sum of four corner weights of each $\boldsymbol{M}_{\mathrm{f},i,j}$ sub-block (relating to the first segment) is calculated. If the sum is larger or equal compared to an upper predetermined threshold γ_{upper} or smaller or equal compared to a lower pre-determined threshold γ_{lower}, the respective uni-directional motion information $\boldsymbol{m}_{\mathrm{S},0}$ or $\boldsymbol{m}_{\mathrm{S},1}$ is stored for the sub-block. Otherwise, a bidirectional motion information $\boldsymbol{m}_{\mathrm{Bi}}$ is assigned, which is a combination of the two uni-directional motion vectors and reference indices. Both thresholds are determined by the size of the coding block and have been designed such that only few 4 × 4 sub-blocks will contain the bi-directional motion vector for small block sizes $w \times h$:

$$\gamma_{\mathrm{lower}} = \frac{4 \cdot 2^{B_{\mathrm{f}}}}{2^{(\log_2 w + \log_2)/2-1}} \tag{6.13}$$

$$\gamma_{\mathrm{upper}} = 4 \cdot 2^{B_{\mathrm{f}}} - \gamma_{\mathrm{lower}} \tag{6.14}$$

Figure 6.7 exemplifies the GEO motion information storage for a given blending matrix and shows the highlighted corner weights used in the process.

6.5 Encoder Complexity

The design of an encoder algorithm that exploits the coding gain potential of a proposed coding tool is a significant task. For the standardization activity conducted by JVET, the encoder algorithm design has to be seen in terms of a *proof of concept*: Unlike a production encoder which may employ many different strategies to select a coding mode for a given block quickly, the HM or VTM reference encoders test all available coding modes for a given block exhaustively, regardless of the coding scenario, e.g. random-access or low-delay applications. Production quality HEVC software encoders such as x265 or Turing often try to predict optimal encoding decisions by means of a look-ahead, 2-pass encoding or texture complexity analysis [Bla+17; Mul19; CBM17]. The complexity of these encoders can be very precisely controlled by the user through selection of encoding presets. E.g. for x265, ten different presets ranging from ultrafast to placebo (full RDO) can be selected with increasing complexity. It is obvious that also special algorithms for GEO, for example relying on object segmentation or edge detection to speed up the selection of the best partitioning, can be developed and potentially greatly reduce the complexity. In the context of AVC, such algorithms have been published in the literature [Wan+13]. For the case of a reference encoder such as HM or VTM however, it is unspoken consensus to avoid heavy pre-analysis and heuristics for mode decisions. Thus, providing an encoding algorithm for a coding mode that operates at low complexity in the HM or VTM framework is proof that at least one encoding scheme exists that can search the available parameter space without excessive requirements. Nevertheless, certain types of speed-ups and heuristics can be frequently found in the VTM reference encoder. A common method is to reuse information on the coding block level from previously tested coding modes, for example:

- For Merge-mode based tools such as TPM, MMVD, CIIP, and Affine Merge, the transform coding stage is skipped if the current best coding mode is already using Skip-mode. Hence, it is assumed that coding a residual will not provide coding gain for the currently tested tool.
- Since a coding block of a given size and position can be reached by different MTT splits, already estimated motion vectors with pixel precision accuracy are reused by AMVP-based coding modes. This effectively prevents the execution of repeated block matching.
- For the BCW tool, not all available weights are exhaustively tested for each inter coding mode. Unequal weights are only tested depending on reference picture POC distances, the QP and the temporal level of the current picture within the hierarchical coding structure. Furthermore, unequal weights are only tested for Affine coding modes, if the current best mode is an Affine mode.

Reliable encoder timing measurements performed separately from the previous experiments show that the encoding runtime with GEO increases to about 125 % across all classes compared to the VTM-3.2 anchor. This is largely attributed to the simple encoding scheme already detailed in Section 5.5 that exhaustively tests all available GEO partitions using a segment-wise distortion computation. Subsequently, candidates are optimized using the SATD metric and fed to the transform coding stage. In order to reduce the encoding complexity, an optimized scheme compared to the scheme detailed in Section 5.5.1 is developed. The following measures are taken to limit the encoding complexity:

- SAD is first computed on a block basis and then estimated for the smaller segment using a hard masking mask. Computing the SAD distortion in a segment-wise manner for every merge candidate and GEO split would require $2 \cdot N_{\text{Merge}} \cdot N_{\text{GEO}} = 1400$ weighted SAD computations. The hard masking eliminates the requirement to multiply each sample difference value with a sample blending weight. The SAD of the larger segment can then be determined from the block-based SAD and the SAD of the smaller segment by simple subtraction, requiring a total of $N_{\text{Merge}} + N_{\text{Merge}} \cdot N_{\text{GEO}} = 705$ non-weighted SAD computations.

- To compensate for the error introduced by not using the sample blending weight, the number of full blending candidates is increased to $N_{\text{Blending}} = 25$. The SATD is computed for these candidates.
- As most of the computation time is spent in the transform coding estimation stage (see Figure 5.13), the number of GEO candidates to be tested is reduced from $N_{\text{Transform}} = 20$ to $N_{\text{Transform}} = 3$.
- Similar as it is performed for TPM, MMVD, CIIP and Affine Merge, the transform coding stage is skipped, and the residual quantized to zero if the current best coding mode is using Skip-mode.

Due to the restriction to uni-prediction only, an additional speed-up can be expected since the encoder also needs to perform fewer motion compensation steps. The overall GEO encoder algorithm for VTM-5.0 is shown in Figure 6.8 and can be described in detail as follows:

1. Perform block-based, uni-directional motion compensation for all available N_{Merge} merge candidates with $N_{\text{Merge}} = 5$. The result is a prediction block $\boldsymbol{P}_l$, $l \in \{0 \dots N_{\text{Merge}} - 1\}$. Compute the block-based SAD $D^{\text{SAD}}_{\text{Block},l}$ for each merge candidate between the prediction $\boldsymbol{P}_l$ and the original picture $\boldsymbol{P}_{\text{Org}}$.
$$D^{\text{SAD}}_{\text{Block},l} = \sum_{x,y} \left| \boldsymbol{P}_l - \boldsymbol{P}_{\text{Org}} \right| \tag{6.15}$$
2. Calculate a segment-based SAD distortion of the luma component for the smaller GEO partition for each of the N_{Merge} motion compensated predictions from Step 1. This gives a total of $N_{\text{Merge}} \cdot N_{\text{GEO}}$ distortion estimates $D^{\text{SAD}}_{\text{Small},l,m}$ with $l \in \{0 \dots N_{\text{Merge}} - 1\}$ and $m \in \{0 \dots N_{\text{GEO}} - 1\}$.
$$D^{\text{SAD}}_{\text{Small},l,m} = \sum_{x,y} \left| \boldsymbol{M}_{\text{b,Small},m} \& (\boldsymbol{P}_l - \boldsymbol{P}_{\text{Org}}) \right| \tag{6.16}$$
Here, $\boldsymbol{M}_{\text{b,Small},m}$ denotes the binary mask of the smaller GEO partition mode m and "&" the binary sample-wise AND operation. Then, calculate the distortion of the larger GEO partition as:
$$D^{\text{SAD}}_{\text{Large},l,m} = D^{\text{SAD}}_{\text{Block},l} - D^{\text{SAD}}_{\text{Small},l,m} \tag{6.17}$$
3. Approximate the RD-cost $J_{l_1,l_2,m}$ with $l_1, l_2 \in l$ and $l_1 \neq l_2$ for each of the $N_{\text{Merge}} \cdot (N_{\text{Merge}} - 1) \cdot N_{\text{GEO}}$ possible combinations of merge candidates and GEO partition modes m using the results from Step 2. Here, R_{l_1} and R_{l_2} are the bits required for signaling the motion information of each segment and R_m are the bits required to signal the GEO partition mode m.
$$\tilde{J}_{l_1,l_2,m} \approx D_{\text{Large},l_1,m} + D_{\text{Small},l_2,m} + \sqrt{\lambda} \cdot (R_{l_1} + R_{l_2} + R_m) \tag{6.18}$$
Then, find the tuples of (l_1, l_2, m) that minimize the RD-cost $\tilde{J}_{l_1,l_2,m}$:
$$\arg \min_{(l_1,l_2,m)} J_{l_1,l_2,m} \tag{6.19}$$

Additionally, exclude all the candidates with $\tilde{J}_{l_1,l_2,m} > D^{\text{SAD}}_{\text{Block},l} + \sqrt{\lambda} \cdot R_l$ with $l_1, l_2, l \in \{0...N_{\text{Merge}} - 1\}$ from further testing. If no candidate remains, abort the entire GEO search.

4. For the $0 \leq k < N_{\text{Blending}} = 25$ candidates from Step 3 with lowest RD-cost, perform the blending process for the luma component and compute the SATD distortion for the resulting block-based predictions. Determine the RD-cost for each candidate.

$$J_k = D^{\text{SATD}}_{l_1,l_2,m} + \sqrt{\lambda} \cdot (R_{l_1} + R_{l_2} + R_m), \quad 0 \leq k < N_{\text{Blending}} \tag{6.20}$$

5. For the $N_{\text{Transform}} = 3$ candidates from Step 4 with lowest RD-cost, perform the blending of all remaining components and apply residual coding and CABAC test-encoding. Compute the actual RD-cost using SSD.

6. Choose the best candidate from Step 5 as the final GEO prediction mode.

The measures taken above result in and overall reduction of encoding time for GEO by 75% compared to the algorithm used in VTM-3.2+GEO, without any loss in coding efficiency. Figure 6.9 shows the relative encoding time for the critical stages (Stage 1: Block-based MC+SAD computation, Stage 2: Segment-based SAD computation, Stage 4: Blending and SATD computation, Stage 5: Transform coding) of the proposed scheme. Compared to the previous algorithm detailed in Section 5.5.1 and the statistics shown in Figure 5.13, the amount of computation time spent in the transform coding stage has been reduced and is now more balanced compared to the time required by the segment-based distortion computation.

Figure 6.10 shows the relative computation time used by GEO in VTM-5.0 compared to other prediction modes of the VVC encoder. It can be seen that the relative encoding time for GEO has been significantly reduced from 33.5% to 5.2%. It is however noted that the encoding complexity of other coding modes has likely also increased from VTM-3.2 to VTM-5.0 due to the adoption of additional tools.

Lastly, it can be summarized that the design of a low-complexity encoder for GEO can be accomplished with the above listed methods. The coding results in Section 6.7 show that encoder runtime increases to 105% for RA and 108% for LB for natural video content. These optimizations have all been performed without any significant loss in coding performance compared to the baseline algorithm detailed in Section 5.5.1. Nevertheless, the author acknowledges that further improvements are possible and have already been demonstrated based on this work. In [Gao+19a] encoder complexity is reported to be 104% for RA, and 105% for LB for the same 140 geometric splits. It is noted that the reduction of 140 geometric splits to 108 or 80, also tested in [Gao+19a], does not result in a measurably lower encoder complexity.

6.6 Decoder Complexity

The main difference in terms of decoding complexity between GEO and TPM is caused by the modified blending filter: While the TPM blending filter weights can be derived in a very simple manner, see Equations (5.19) and (6.11), or stored in memory, the derivation

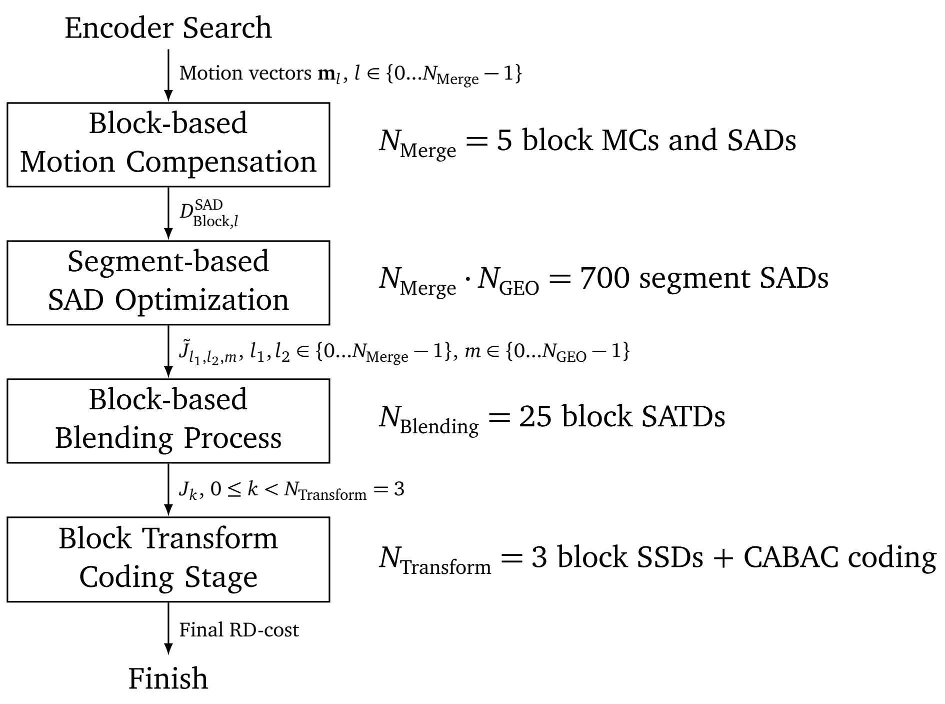

Encoder Search
Motion vectors $\mathbf{m}_l$, $l \in \{0 \ldots N_{\text{Merge}} - 1\}$
Block-based Motion Compensation
$N_{\text{Merge}} = 5$ block MCs and SADs
$D^{\text{SAD}}_{\text{Block},l}$
Segment-based SAD Optimization
$N_{\text{Merge}} \cdot N_{\text{GEO}} = 700$ segment SADs
$\tilde{J}_{l_1,l_2,m}$, $l_1, l_2 \in \{0 \ldots N_{\text{Merge}} - 1\}$, $m \in \{0 \ldots N_{\text{GEO}} - 1\}$
Block-based Blending Process
$N_{\text{Blending}} = 25$ block SATDs
J_k, $0 \leq k < N_{\text{Transform}} = 3$
Block Transform Coding Stage
$N_{\text{Transform}} = 3$ block SSDs + CABAC coding
Final RD-cost
Finish

Figure 6.8 VTM-5.0 mode estimation for GEO using four stages. The optimal GEO mode is found by using a combination of block-based and segment-based distortion computations. Note that the transform coding stage is common to all inter-prediction modes.

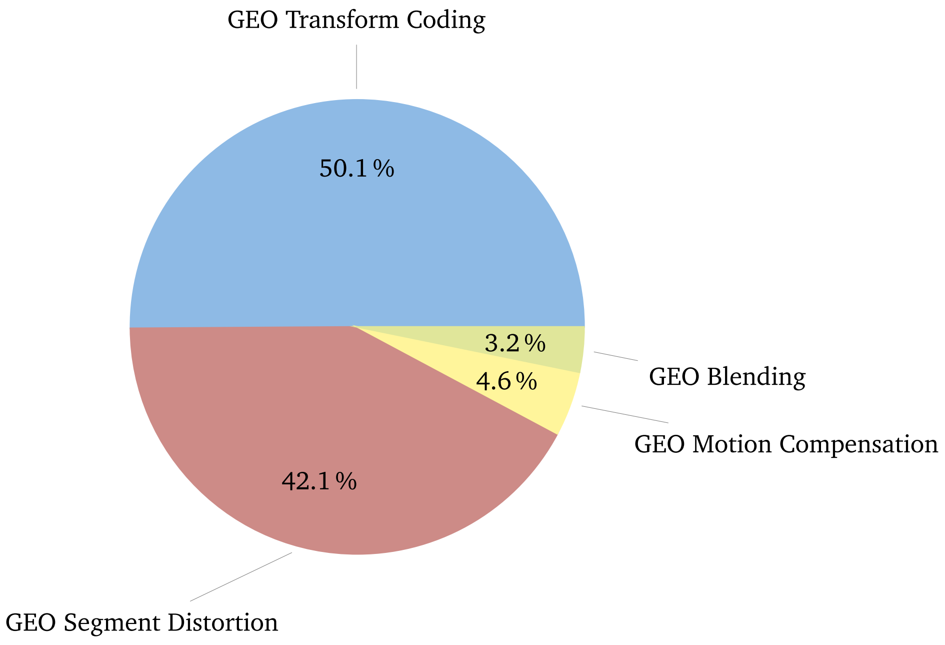


Figure 6.9 Relative encoding runtime distribution for individual stages of the GEO mode estimation. Statistics were gathered by encoding a single CTU at QP32 in a RA configuration for 33 pictures.

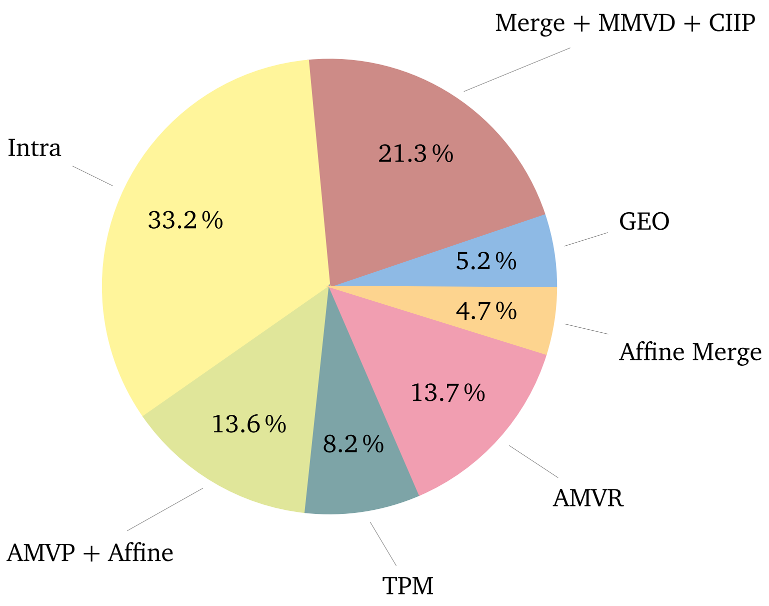


Figure 6.10 Relative encoding runtime distribution for coding modes of the VTM-5.0+GEO encoder. The GEO mode is optimized by performing the search strategy as depicted in Figure 6.8. Statistics were gathered by encoding a single CTU at QP32 in a RA configuration for 33 pictures.

of GEO filter weights involves a distance calculation with trigonometric functions and a mapping operation from a distance value to a filter weight. If this shall be avoided, the GEO filter masks could also be pre-computed and stored in memory. However, the memory requirements for this approach might be prohibitively large for hardware implementation: assuming 4 bit for each weight, utilizing symmetries among the 140 geometric splits and also across different blocks sizes (e.g. the filter masks for an 8×16 and a 16×8 block are coupled by a simple geometric mapping), storing all filter masks in memory would require 1.92 kB. Therefore, it is proposed to utilize the parametric model for the geometric partitioning line as introduced in Section 5.1.1 and generate the weights on the fly while decoding every GEO blocks. For this purpose, the parametric model using polar coordinates is converted to an integer arithmetic based variant using look-up tables. The following sections detail the methodology and approaches taken to ensure low complexity decoding for the GEO coding tool. The success of these methods is demonstrated by the coding results given in Section 6.7, with a decoding time of 100 %-101 % for RA and LB compared to the VTM-5.0 anchor without GEO.

6.6.1 Integer Approximation for Weight Derivation

In standards prior to AVC, certain parts of video codecs were not explicitly specified in integer-based arithmetic, such as the inverse DCT in H.263. Deviations in different floating-point implementations of encoders and decoders may lead to differences when reconstructing a video from a bitstream on different platforms. To solve this conformance issue, all critical decoder parts are specified using integer arithmetic in recent video codecs, such as in AVC, HEVC and VVC. Furthermore, the design and specification of algorithms only using integer additions, subtractions – additions using the respective numbers 2's complement –

multiplications, and bit-shifts simplifies the hardware implementation. The complexity of an algorithm can be assessed by counting the number of integer additions (ADD), bit-shifts (SHIFT) and multiplications (MUL) required per reconstructed sample. For GEO, the critical part performed during decoding is the calculation of the distance of each sample to the partitioning line. Recall that this operation is specified using floating-point operations in Equation (5.9) as follows:

$$d^{\mathrm{AD}}_{\mathrm{float}}(x_{\mathrm{c}}, y_{\mathrm{c}}) = x_{\mathrm{c}} \cos\varphi + y_{\mathrm{c}} \sin\varphi - \rho \tag{5.9 revisited}$$

In this representation, the coordinates $(x_{\mathrm{c}}, y_{\mathrm{c}})$ specify the center of a pixel, relative to the center of the coding block of size $w \times h$, e.g. the top-left coordinate is given by $(-{}^{w}/_{2} + 0.5, -{}^{h}/_{2} + 0.5)$. The same equation can be translated into integer arithmetic as follows:

$$d^{\mathrm{AD}}_{\mathrm{int}}(x, y) = (2x - w + 1) \cdot \cos[m] - (2y - h + 1) \cdot \sin[m] - 2\rho_n \tag{6.21}$$

Equation (6.21) replaces the cos() and sin() functions with cos[ ] and sin[ ] look-up tables, fetching the trigonometric function values by the angular quantization index m. This equation can be further optimized for requiring as few additions, multiplications and shifts as possible:

$$d^{\mathrm{AD}}_{\mathrm{int}}(x, y) = \underbrace{\underbrace{(x << 1)}_{\text{1 SHIFT/sample}} \underbrace{\cdot \cos[m]}_{\text{1 MUL/row}}}_{\text{1 ADD/sample}} \underbrace{- \underbrace{(y << 1)}_{\text{1 SHIFT/row}} \underbrace{\cdot \cos\left[(m + N_{90^\circ})\% N_{\varphi}\right]}_{\text{1 MUL/row}} \underbrace{-(\rho_n << 1)}_{\text{1 SHIFT/block}} + C}_{\text{3 ADD/row}} \tag{6.22}$$

$$C = \underbrace{(h - 1) \cdot \cos\left[(m + N_{90^\circ})\% N_{\varphi}\right] - (w - 1) \cdot \cos[m]}_{\text{2 MUL/block, 4 ADD/block, 1 MOD/block}} \tag{6.23}$$

Here, the sin[ ] look-up table is eliminated using a 90° shifted angular quantization index. For a block of size $w \times h$, Equation (6.22) can be implemented requiring $(1 + {}^{3}/_{w} + {}^{4}/_{wh})$ ADD per sample, $(1 + {}^{1}/_{w} + {}^{1}/_{wh})$ SHIFT per sample and $({}^{2}/_{w} + {}^{2}/_{wh})$ MUL per sample.

The cosine look-up table in the final design contains $N_{\varphi} = 32$ integer values, thereby requiring a 90° shift of $N_{90^\circ} = 8$. Due to the integer arithmetic, Equation (6.22) can be seen as an approximation to the actual distance given by Equation (5.9). The precision of the approximation is determined by the bit-depth B_{i} of the cosine look-up table. The cosine look-up table is generated by using a uniform, mid-riser quantizer $Q(x)$, scaling the look-up values to a dynamic range of $[-2^{B_{\mathrm{i}}-1}, 2^{B_{\mathrm{i}}-1}]$ according to:

$$Q(x) = \min\left(2^{B_{\mathrm{i}}-1}, \max\left(-2^{B_{\mathrm{i}}-1}, \left\lfloor 2^{B_{\mathrm{i}}-1} \Delta \left(\left\lfloor \frac{x}{\Delta} \right\rfloor + \frac{1}{2} \right) + \frac{1}{2} \right\rfloor \right)\right), \quad \Delta = \frac{\max(x) - \min(x)}{2^{B_{\mathrm{i}}}} \tag{6.24}$$

$$\cos[m] = Q\left(\cos\left(m \frac{2\pi}{N_{\varphi}}\right)\right), \quad 0 \le m < N_{\varphi} \tag{6.25}$$

The approximate distance can then be determined by bit-shifting the distance $d^{\mathrm{AD}}_{\mathrm{int}}$ according to:

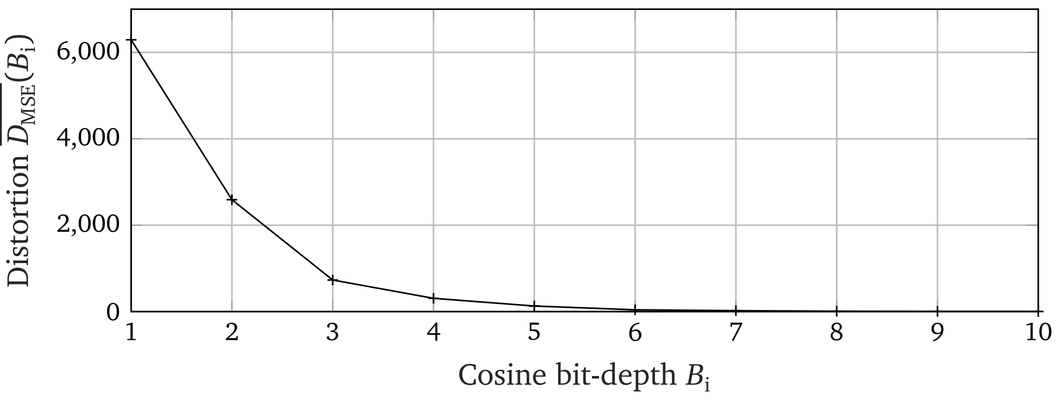


Figure 6.11 Mean distortion of blending filter weights due to quantization of the cosine function with a given bit-depth.

$$d^{\mathrm{AD}}_{\mathrm{float}} \approx d^{\mathrm{AD}}_{\mathrm{int}} + (1 << B_{\mathrm{i}} - 1) >> B_{\mathrm{i}} \tag{6.26}$$

Since the distances $d^{\mathrm{AD}}_{\mathrm{float}}$ or $d^{\mathrm{AD}}_{\mathrm{imt}}$ are subsequently used to derive the sample weights $w_{\mathrm{GEO}}(x,y)$ using the filter quantization function $f_{\mathrm{q}}(\cdot)$ of half-sample accuracy, the required bit-depth B_{i} can be assessed by computing a Mean squared error (MSE) between $\boldsymbol{M}_{\mathrm{f,float}}$ and $\boldsymbol{M}^{B_{\mathrm{i}}}_{\mathrm{f,int}}$ for each geometric split:

$$D_{\mathrm{MSE}}(B_{\mathrm{i}}) = \frac{1}{wh} \sum_{x,y} \left(M_{\mathrm{f,float}}(x,y) - M^{B_{\mathrm{i}}}_{\mathrm{f,int}}(x,y) \right)^2 \tag{6.27}$$

Figure 6.11 shows the mean distortion $\overline{D_{\mathrm{MSE}}}(B_{\mathrm{i}})$, averaged over all geometric splits, for a given cosine bit-depth B_{i}. It can be deduced from the figure that a cosine precision with a bit-depth of $B_{\mathrm{i}} = 7$bit produces a blending filter $\boldsymbol{M}^{7}_{\mathrm{f,int}}$ with nearly identical weights compared to the blending filter $\boldsymbol{M}_{\mathrm{f,float}}$ generated by the floating-point distance equation.

6.6.2 Chroma Weight Derivation

In case the video source format is provided with chroma subsampling, e.g. most commonly as 4:2:0 or 4:2:2, the blending filter for the chroma component has a different size compared to the luma component. In principle, the same process for generating the luma weighting mask can be invoked with different input sizes in terms of width and height for the filter size, e.g. $w_{\mathrm{C}} = {}^{w}\!/_{2}$ and $h_{\mathrm{C}} = {}^{h}\!/_{2}$ for 4:2:0 and $w_{\mathrm{C}} = {}^{w}\!/_{2}$ and $h_{\mathrm{C}} = h$ for 4:2:2 with $w_{\mathrm{C}} \times h_{\mathrm{C}}$ specifying the size of the chroma coding block. In order to simplify the process, the chroma weights can be generated from the luma weights by subsampling:

$$w^{420}_{\mathrm{GEO,Chroma}}(x,y) = w_{\mathrm{GEO}}(2x,2y), \quad 0 \le x < w_{\mathrm{C}}, 0 \le y < h_{\mathrm{C}} \tag{6.28}$$

$$w^{422}_{\mathrm{GEO,Chroma}}(x,y) = w_{\mathrm{GEO}}(2x,y), \quad 0 \le x < w_{\mathrm{C}}, 0 \le y < h \tag{6.29}$$

6.7 Overall Coding Performance

The overall coding results for RA and LDB in Experiment 6.27 over VTM-5.0 are provided in Tables 6.7 and 6.8, respectively. Detailed results are given in Table A.34 and Table A.35. For RA, GEO achieves an overall luma BD-rate change of −0.22 %, measured across the classes A1, A2, B and C according to the JVET CTC. For LDB, higher coding efficiency improvements can be reported at −0.44 %, measured across the classes B, C and E. The coding gain is not evenly distributed among the resolution classes. For RA, largest improvements are observed for class C which contain the *BQMall* and *RaceHorses* sequence. Here, coding efficiency improves to −1.09 % and −0.71 %, respectively. For *BQTerrace*, *SlideEditing* and *SlideShow*, a slight loss in coding efficiency is observed, measured at 0.04 %, 0.10 % and 0.02 % in terms of luma BD-rate change. However, the losses for class F can be completely avoided if an adaptive blending filter is employed for pure screen content as shown in Section 6.3.

The coding results for different sequences largely confirm the previous observations that GEO is a coding tool suitable for video sequences which contain clearly separated, natural moving objects, such as in *BQMall*, *RaceHorses*, *BasketballDrive* or *CatRobot1*. For those sequences and for high QPs, a visual benefit is also assumed since GEO is used predominantly for the coding of object boundaries. Figures 6.12 provides two examples, showing how GEO removes staircase coding artifacts at object boundaries.

Arguably, coding efficiency for GEO can be improved significantly if bi-prediction is utilized at the cost of increased worst-case memory bandwidth. However, analysis of actual memory bandwidth consumption performed in Section 5.1.3 using a realistic cash model showed that only a marginal increase in overall memory bandwidth consumption can be expected for typical GEO mode usages.

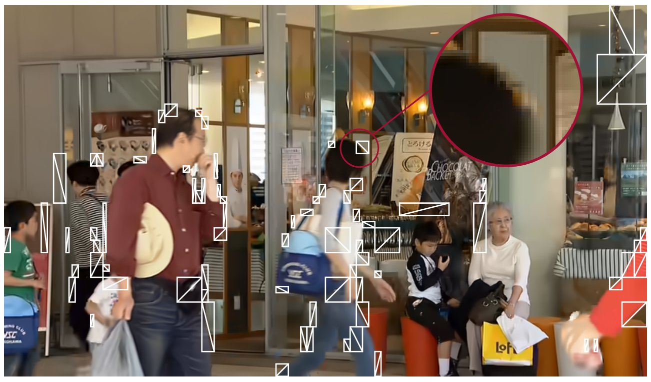
Further enhancements and improvements for GEO are currently conducted in the context of the VVC standardization [Gao+19a; Gao+19b; Gao+19c]. With different numbers and distributions of GEO splits, coding efficiency for GEO is reportedly further increased. For a configuration with 64 geometric splits, using 20 quantized angles and 4, 2 or 1 quantized distances based on the angle, coding efficiency is reported at −0.34 % in terms of luma BD-rate change for RA and −0.67 % for LDB over VTM-6.0. Further improvements made to the encoder lower the encoding time to 102 % and 103 %, respectively.

Class	Over VTM-5.0					
	Y	U	V	EncT	DecT	VMAF
A1	−0.09 %	−0.24 %	−0.05 %	105 %	101 %	0.08 %
A2	−0.14 %	−0.29 %	−0.26 %	106 %	101 %	−0.17 %
B	−0.08 %	−0.06 %	−0.21 %	106 %	100 %	−0.07 %
C	−0.54 %	−1.09 %	−1.13 %	105 %	101 %	−0.60 %
Overall	**−0.22 %**	**−0.42 %**	**−0.43 %**	**106 %**	**101 %**	**−0.20 %**
D	−0.18 %	−0.76 %	−0.35 %	105 %	101 %	−0.12 %
F*	−0.37 %	−0.41 %	−0.32 %	103 %	101 %	−0.43 %
TGM*	−2.15 %	−1.77 %	−1.76 %	99 %	100 %	−2.07 %

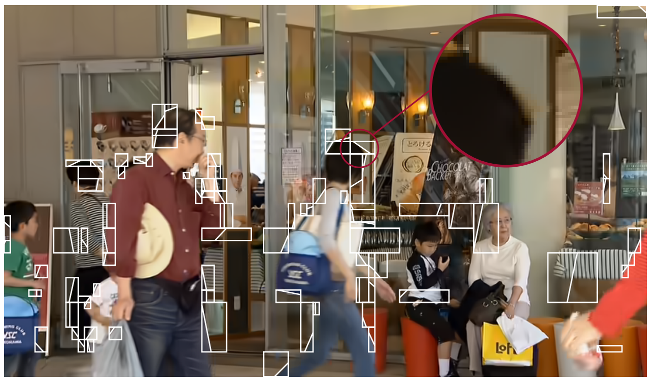
Table 6.7 Coding results for VTM-5.0 + GEO in RA configuration.*Results for screen content sequences with hard masking for GEO and TPM. Detailed coding results are provided in Table A.34.

Class	Over VTM-5.0					
	Y	U	V	EncT	DecT	VMAF
B	−0.22 %	−0.23 %	−0.32 %	108 %	102 %	−0.21 %
C	−0.54 %	−1.12 %	−1.00 %	108 %	101 %	−0.30 %
E	−0.68 %	−0.46 %	−0.10 %	107 %	102 %	−0.84 %
Overall	**−0.44 %**	**−0.59 %**	**−0.49 %**	**108 %**	**102 %**	−0.40 %
D	−0.39 %	−0.84 %	−0.07 %	108 %	100 %	−0.52 %
F*	−0.28 %	−0.42 %	−0.72 %	106 %	101 %	−0.03 %
TGM*	−2.49 %	−2.26 %	−2.25 %	99 %	101 %	−2.37 %

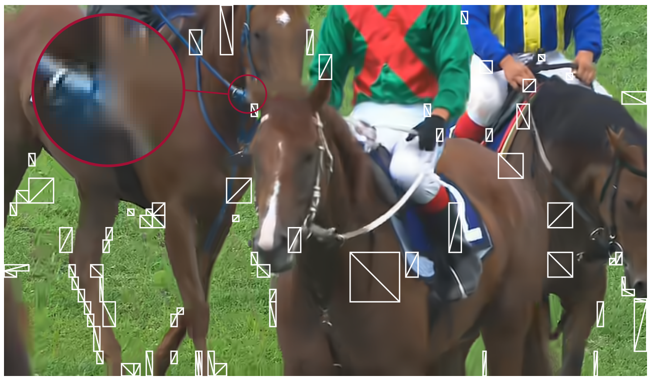
Table 6.8 Coding results for VTM-5.0 + GEO in LDB configuration.*Results for screen content sequences with hard masking for GEO and TPM. Detailed coding results are provided in Table A.35.



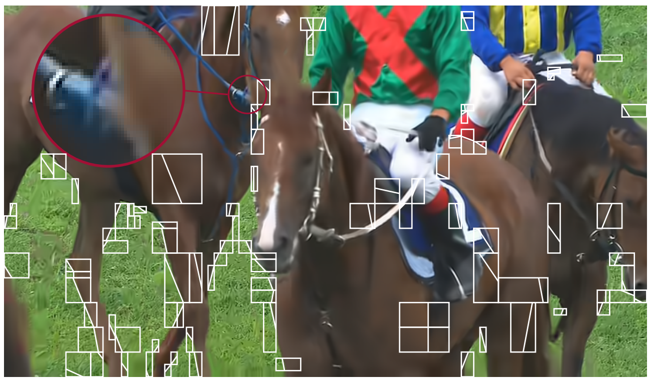
(a) BQMall, Reference



(b) BQMall, GEO



(c) RaceHorses, Reference



(d) RaceHorses, GEO

Figure 6.12 Visual examples showing the geometric partitioning and potential visual impact compared to the VTM-5.0 reference. Examples are coded using RA at QP37.

7 Exploration on Transform Coding for Non-rectangular Partitions

Applying a linear block transform to the prediction error and quantization of the resulting transform coefficients is one essential part of the hybrid video coding scheme. Over 30 years have passed since the development of H.261 and numerous advancements have been made in this area. The core method of transform coding is nevertheless still the application of a two-dimensional, separable DCT-II transform to the prediction error signal, followed by linear quantization of the resulting transform coefficients and subsequent entropy coding. This scheme has been augmented and improved in VVC by additional core transforms, non-separable secondary transforms, and advanced entropy coding methods for transform coefficients and residual samples. Furthermore, the linear quantizer has been replaced by a Trellis-coded quantizer (TCQ) [Sch+19]. In the following, a brief review of past developments in the area of transform coding is given. This review cannot claim to be complete due to the wide scope of the subject but can give the necessary background for the handling of the prediction error resulting from a non-rectangular block partitioning.

It has been known for a long time that the DCT-II is not the optimal decorrelating transform for images, which is the Karhunen-Loève transform (KLT) – the optimal linear transform in terms of coding gain for non-stationary sources. The performance of the DCT-II for AR, specifically AR(1) distributed sources with a high correlation coefficient of $\rho \rightarrow 1$ approaches that of the KLT [CP93]. Furthermore, since the DCT-II does not require additional signaling of side-information compared to the signal-dependent KLT, it is still the RD-optimal transform for naturally occurring prediction error signals that can be modeled by Markov models [DeL+16][Goy01]. This knowledge has been augmented by many empirical and theoretical studies about the properties of prediction error signals and resulting transform coefficients [Mül93; ZWZ10; SEO11; LG00]. Here, the important distinction must be made whether the prediction error for intra-prediction [Wu+07; Zou+13] or inter-prediction [LKK11; CP93; KMO10; Shi92; Wen+00; Wen+02; Zhe+01] is considered, since both have different characteristics. A few key observations and conclusions drawn from these studies can be summarized as follows:

Natural images are often modeled by two-dimensional first order Markov models. The covariance between samples can be written as $r(m,n) = \sigma^2 \rho(m,n)$ (see Chapter 2) with σ^2 being the variance of a natural image block and different assumptions are made on the correlation coefficient function $\rho(m,n)$. Often, separability is assumed, such that $\rho(m,n) = \rho_{\mathrm{h}}^{|m|} \rho_{\mathrm{v}}^{|n|}$ with $\rho_{\mathrm{h}} = \rho_{\mathrm{v}} = 0.95$ for natural images [CP93]. Otherwise, an isotropic, non-separable correlation function is assumed, such that $\rho(m,n) = e^{-\omega_0 \sqrt{m^2+n^2}}$ [Gir87]. It is pointed out that the validity of these models for natural images, assuming a Wide-sense stationary (WSS) Markov process, is already violated by strong edges and directional structures that are known to frequently occur.

The statistical properties of the intra-prediction error signal are different from those of the original image. First, the overall correlation between samples of the intra-prediction error is generally much lower. Secondly, the correlation is in fact often neither separable in horizontal or vertical dimension nor isotropic. Depending on the prediction mode, the correlations in horizontal and vertical direction can be very different from each other. For example, blocks that are coded with a vertical intra-prediction mode show larger correlation coefficients in the vertical direction than their counterparts in horizontal direction. The opposite applies for a horizontal intra-prediction mode. However, also blocks using the same intra-prediction mode can have considerably varying statistical properties over the entire image or video [Zou+13].

In consequence to these findings, adaptations have been suggested to the DCT-II coding of the intra-prediction error. Three different approaches can be mainly distinguished:

The non-isotropic, mode-dependent properties of the intra-prediction residual motivated the use of the KLT and KLT-derived MDDT [Zou+13; Yeo+12] that were proposed for H.264/AVC. In the MDDT scheme, a separable transform is designed based on separable KLTs. A pair of column and row KLTs are derived by using training data for each intra-prediction mode. Therefore, each intra-prediction mode has its corresponding KLT pair. The better coding gain of such a transform is dimmed by the inherent sensitivity towards the properties of the training data and increased hardware implementation cost.

The second approach is the application of a secondary transform after the DCT. Most prominent examples are rotational transforms that are applied to the low-frequency DCT coefficients of a block. Such transforms were suggested for HEVC and ultimately included in the JEM [Che+19b; Che+17]. In JEM, a Mode dependent non-separable secondary transform (MDNSST) is included with a total of 35×3 non-separable transforms, where 35 is the number of transform sets given by the intra-prediction modes and 3 is the number of NSST candidates (transform kernels) for each mode. This scheme required a huge amount of memory for storing the transform kernels and therefore evolved into the Low-frequency non-separable transform (LFNST) concept, now part of VVC. LFNST specifies only 4 transform sets with 2 kernels per set [BCL18; CYK19].

The third approach are additional trigonometric core transforms next to the DCT-II. It was recognized early on that the DST performs better than the DCT under certain conditions. For first-order Markov processes for example, it can be shown that the decorrelation efficiency of the DST is superior to the DCT for sources with a correlation coefficient $\rho < 0.5$ [Cla85]. Han, Saxena & Rose derived the DST-VII to be the actual KLT along the prediction direction for intra modes in H.264/AVC [HSR10]. In HEVC, the DST-VII is the default transform applied to 4×4 intra-predicted transform units [SF13]. To further enhance the coding performance for the JEM, even more core transforms were added next to the known DCT-II/DST-VII scheme: In the Adaptive multi-core transform (AMT) set (or often styled as EMT) they are supplemented by the DCT-V, DCT-VIII and DST-I. Depending on the intra-mode, a different transform set is chosen consisting of two transform candidates. For the horizontal and vertical transform, one of the two transform candidates is selected. For inter-prediction residuals, only one transform set consisting of the DST-VII and DCT-VIII is allowed, next to the default DCT-II applied in both directions [Che+19b; Che+17; Zha+16]. This design was significantly simplified for VVC, resulting in the Multiple transform selection (MTS) scheme: In MTS, next to the DCT-II, the DCT-VIII and the DST-VII can be applied. Four combinations of DCT-VIII and DST-VII in horizontal and vertical direction can be signaled for intra- and inter-predicted blocks. [BCL18; CYK19].

Lastly, the transform-skip mode can also be seen as an additional "core" transform by applying the identity transform before quantization. Transform-skip is beneficial for coding of prediction errors (intra or inter) with few non-zero samples in lossless and near lossless coding scenarios. Such characteristics are often observed for screen content, making transform-skip a coding tool suitable for such content [Nar13] [MX12]. Transform skipping has been a part of HEVC and is also included in the JEM and VVC [Ngu+19].

Many insights have been made regarding the properties of the inter-prediction residual, often termed the motion-compensated frame difference (MCFD). Girod showed that the power density spectrum $S_{ee}(\omega_x, \omega_y)$ of the motion-compensated prediction error signal $e(x, y)$ is dependent on the power density spectrum $S_{ss}(\omega_x, \omega_y)$ of the input video signal s and the power density spectrum $S_{qq}(\omega_x, \omega_y)$ of the quantization error q [Gir93] as follows:

$$S_{ee}(\omega_x, \omega_y) = 2S_{ss}(\omega_x, \omega_y) \cdot (1 - \mathrm{Re}\{\mathscr{F}\{p(\Delta x, \Delta y)\}\}) + S_{qq}(\omega_x, \omega_y) \qquad (7.1)$$

Where $p(\Delta x, \Delta y)$ is the probability density function of the displacement errors Δx and Δy. As the power density spectrum is the Fourier transform of the correlation function, Equation (7.1) also shows that the correlation between the prediction error samples e decreases with better motion compensated prediction. This is for example the case when sub-pixel interpolation filters are introduced. Furthermore, different covariance models have been proposed for the inter-prediction error. Chen and Pang assumed the probability density function $p(\Delta x, \Delta y)$ of the displacement error to be composed of a uniform distribution over an interval $[-a,\ a]$ in both x and y direction and an impulse at the origin to model the probability of achieving perfect motion compensation. Using this model they argued that the optimum KLT of the inter-prediction error is identical to the KLT of the original image and therefore the DCT also remains nearly optimum [CP93] for coding the inter-prediction error. Niehsen and Brünig arrived at the same conclusion with a different covariance model and also taking OBMC into account. They also measured that the block means and standard deviations of the inter-prediction error can vary greatly, preventing the common wide-sense-stationary modeling. Hui and Siu further refined the displacement error modeling by also considering block deformations due to failure of the block-based motion model for moving parts, light variations, inaccuracy of motion compensation, quantization error and noise [HS07]. In this covariance model, the inter-prediction error is assumed to contain directional deformations rather than a uniform distribution.

Early empirical studies have shown that the samples of inter-prediction residual blocks follow a Laplacian distribution [Wen+02]. This was confirmed for the JCT-VC test set sequences used during the standardization of HEVC [Nar13]. In terms of spatial properties within a block of inter-prediction error samples, it was discovered by Zhen and Shishikui that the prediction error tends to be larger at the block boundaries than at the block center [Wen+00; Wen+02; Zhe+01] (see Chapter 5, Section 5.1.1), explained by authors using a statistical motion distribution model theoretically. OBMC has been proven to decrease the prediction error at these block boundaries.

The spatially non-constant variance withing the prediction error blocks also leads to correlation between transform coefficients when the DCT-II is applied [SEO11], which can be exploited for marginally improved coding of these coefficients. A spatially varying distribution of residual energy can also be seen as one motivation for the introduction of the RQT in HEVC. The RQT allows to use large PBs and small TBs, catering to local variations in the residual statistics [Tan+11]. For JEM however, the RQT was dismissed in favor of a unified approach of having identically sized CUs, PUs, and TUs using a more advanced block partitioning scheme (QTBT). For VVC, a SBT mode is reintroduced, allowing one single binary-split for a given coding block.

Naturally, the question arises whether these properties, observed and derived for rectangular coding blocks, also hold true for geometrically partitioned blocks. An analysis of the residual coding mode for TPM and GEO reveals that both of these non-rectangular partitioning schemes more frequently use the Skip-mode – meaning that no residual is coded – compared to other inter-prediction modes. The statistics for the JVET test set are shown in Figure 7.1. Overall, it can be stated that even for a low QP of 22, more than 50 % of all inter-predicted pixels are coded without signaling a prediction error. This significantly increases to more than 80 % for QP37. Differences exists between GEO, TPM, and other inter-prediction modes. For a QP of 22 for example, about 62 % of all GEO coded pixels are using Skip-Mode, compared to 70 % for TPM. For a QP of 37, 89% of all GEO coded pixels are using Skip-mode, compared to 92 % for TPM. Other inter-prediction modes on the other hand signal a residual more frequently. It is to be kept in mind, however, that the combined overall usage in terms of pixels coded using GEO and TPM is mostly below 10 %. It is interesting that for lower QPs, a prediction residual is actually more frequently transmitted for GEO than for TPM, considering the higher mode signaling costs for GEO and the improved motion compensation of GEO over TPM. It is also noted that this measurement may be slightly biased due to different encoder strategies for GEO and TPM concerning the skip-mode selection.

The following sections intend to explore different possibilities of transform coding for both GEO and TPM. Since this is a topic suitable for a thesis of its own, no definite conclusions will be drawn from the results. Modifying a small part of the transform coding scheme may result in a deviation from the well-tuned interaction of DCT-II transform, rate-distortion optimized quantization (RDOQ) of the reference encoder and spatial context-based CABAC-coding of transform coefficients, negating any coding gain that is theoretically expected. Optimizing all of these aspects would be out of the scope of this thesis. Nevertheless, the methods and experiments provided may encourage further investigations into these areas.

In Section 7.1 additional properties for non-rectangular prediction error signals are empirically studied. Section 7.2 explores the possibilities of transforming non-rectangular GEO segments by means of the well-known SADCT. Section 7.3 provides coding results for performing transform-skipping adapted to GEO partitions. In Section 7.4, methods that rely on encoder-side optimization of the regular DCT-II and masking operations at the decoder are presented.

7.1 Properties of Non-rectangular Prediction Residuals

The main difference expected in terms of statistical properties for inter-prediction error blocks is a spatially varying distribution of residual energy. In Section 5.1.1, it was already empirically shown that the prediction error is higher in the transition zone between two segments, where blending is applied. In the following, the prediction error for the two segments composing a block shall be considered.

If the residual error has similar distributions in terms of correlation and energy in both segments S_0 and S_1, a regular 2D block transform such as the DCT-II would suffice for decorrelation. If, however, the two segments differ, the following problems may arise:

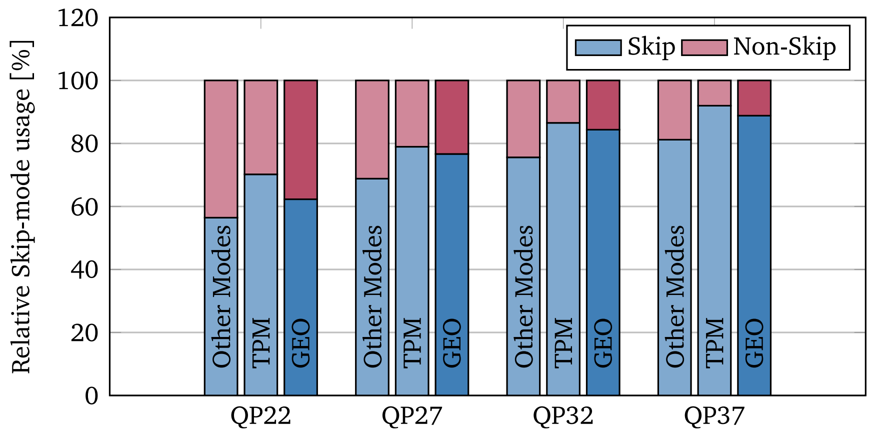


Figure 7.1 Relative usage of the Skip-mode for different inter-prediction coding modes in VTM-3.2. *Skip* statistics also contains blocks with zero-valued luma CBF. The statistics are gathered per class and averaged over the classes.

- If the majority of the residual energy is located in a spatially small area of the residual block, e.g. concentrated in just one segment S_0 or S_1, a block transform based on trigonometric basis functions cannot efficiently represent such data. This is undoubtedly related to a famous result by Heisenberg and Bernstein called the Uncertainty Principle. In short, the Uncertainty Principle states that a function $s(t)$ and its Fourier transform $S(f)$ cannot both be sparse. This also applies to discrete sequences. Let $s(n)$ be a sequence of length N and $S(k)$ be its discrete Fourier transform

$$S(k) = \sum_{n=0}^{N-1} s(n) e^{-\mathrm{j}2\pi k \frac{n}{N}}, \; k = 0, 1, ..., N-1 \tag{7.2}$$

  If N_t and N_f count the number of non-zero entries in $s(n)$ and $S(k)$, respectively, then the Uncertainty relations

$$N_\mathrm{t} \cdot N_\mathrm{f} \geq N \tag{7.3}$$

  and

$$N_\mathrm{t} + N_\mathrm{f} \geq 2\sqrt{N} \tag{7.4}$$

  must be obeyed [WC14]. This can be extended to 2D signals accordingly.

- Likewise, segments of prediction residuals with low inter-sample correlation may not be suitable for DCT-II based transform coding. This could become a relevant issue for low QPs and specific video content.

The distribution of residual energy $\sigma^2_{\mathrm{S}.k}$, $k \in \{0, 1\}$ among the two GEO segments S_0 and S_1 can be analyzed using the JVET test set. Measuring the residual energy at the encoder-side for every tested GEO blocks could increase the overall variance of the distribution and therefore reduce the ability to detect any statistical significance. Since the encoder is performing a nearly full search of all possible combinations of GEO splits and motion vectors per split, many of these combinations will not be a *good* prediction in terms of their RD-cost and dismissed by the encoder. Only GEO blocks with competitive RD-cost are therefore considered relevant. However, also analyzing the single, best GEO block for every position and block size in terms of RD-cost before transform coding is applied, could distort the distribution. Therefore, it is proposed to measure the properties of actually coded GEO blocks that were chosen to be locally RD-optimal. This can of course be seen as a biased measurement, since the properties of these selected blocks are apparently suitable for transform coding, if a transform is chosen at all.

For coded GEO blocks, the variance of a segment S_k defined by its binary mask $\boldsymbol{M}_{\mathrm{b},k}$ and the prediction block $\boldsymbol{P}$ shall be measured as

$$\sigma^2_{\mathrm{S},k} = \frac{1}{N_{\mathrm{S},k}} \sum_{x,y} \boldsymbol{M}_{\mathrm{b},k} \circ \left(\boldsymbol{P} - \mu_{\mathrm{S},k}\right)^2 \tag{7.5}$$

$$N_{\mathrm{S},k} = \sum_{x,y} \boldsymbol{M}_{\mathrm{b},k} \tag{7.6}$$

$$\mu_{\mathrm{S},k} = \frac{1}{N_{\mathrm{S},k}} \sum_{x,y} \boldsymbol{M}_{\mathrm{b},k} \circ \boldsymbol{P} \tag{7.7}$$

with $k \in \{0,1\}$. To assess whether the two segments contain different amount of power, the empirical joint discrete probability mass function $p(\sigma_{\mathrm{S},0}, \sigma_{\mathrm{S},1})$ in terms of a normalized histogram is measured for all coded GEO blocks for the first 65 pictures of all video sequences in the JVET test set. The distribution is measured by counting the relative occurrences of $(\sigma_{\mathrm{S},0}, \sigma_{\mathrm{S},1})$ pairs in equally spaced bins with a width of $\Delta\sigma = 1$ in the half-open interval $[0, 100)$ and normalizing each bin count by the total occurrence. Therefore, the last bins in each direction contain the occurrences for $[100, \infty)$. The distributions per QP are shown in Figure 7.2. An elliptical probability mass function along the main diagonal would indicate that the two segments always contain equal energy. The farther the probability mass function is centered along the diagonal, the higher the overall prediction error variance.

In general, it can be observed from Figure 7.2 that for a low QP of 22, most of the prediction error is – as to be expected – quantitatively small and centered around $\sigma_{\mathrm{S},0} = \sigma_{\mathrm{S},1} \approx 10$[1]. The distribution also shows a stronger linear dependence between $\sigma_{\mathrm{S},0}$ and $\sigma_{\mathrm{S},1}$, indicating in fact that GEO coded blocks often have similar residual energy in both segments. This behavior becomes qualitatively less pronounced for higher QPs. Here, the distributions along the diagonal generally become more smeared and shifted to the bottom right, indicating an overall higher prediction error energy in both segments. It is also noteworthy that with increasing QPs, the distributions masses are slightly more concentrated above the diagonal. This indicates that the segment S_0 tends to have lower residual energy compared to segment S_1. At this point the reader shall be reminded that by convention, segment S_0 always relates to the segment containing the top-left sample. Therefore, except for a few corner cases, most GEO segments S_0 are located closer to the top and left block boundary than the corresponding S_1 segments. This also explains why the residual energy is lower in the S_0 segments for higher QPs, since the correlation between motion vectors typically decreases when moving away from the top-left block boundaries.

For each GEO block, the ratio between $\sigma^2_{\mathrm{S},0}$ and $\sigma^2_{\mathrm{S},1}$ of each coded block can also be measured and analyzed in a histogram. For simplicity, the ratio is computed as $\beta_{\mathrm{r}} = {}^{\max(\sigma^2_{\mathrm{S},0}, \sigma^2_{\mathrm{S},1})}/{}_{\min(\sigma^2_{\mathrm{S},0}, \sigma^2_{\mathrm{S},1})}$, such that $\beta_{\mathrm{r}} \geq 1$ for all blocks. The histogram bins for this measurement are using a spacing of $\Delta\beta_{\mathrm{r}} = 1$ in the interval from $[1, 9)$. Therefore, the last

[1]All sequences were scaled to 10 bit.

bin contains all ratios in the interval $[10, \infty)$. The measurement results are shown in Figure 7.3. For QP22, the ratio between the variances of the two segments is in the ranges of 1-2 in almost 60% of the cases, 2-3 in 30% of the cases and larger than 3 in the remaining interval. This distribution becomes flatter with increasing QP, showing that segments with different residual variances are selected by the encoder more frequently.

Lastly, the per-sample inter-prediction error variances $\sigma^2(x, y)$ for a specific non-rectangular partitioning mode can be analyzed. The two TPM splits are particularly suited for this task since they occur more frequently compared to a single GEO split mode. Figure 7.4 shows the per-sample luma variance $\sigma^2(x, y)$ for the two TPM splits for 16×16 blocks. The variance has been computed for each sample across all TPM residuals for a given TPM split mode and block size.

First of all, the measurements confirm the observations from Zheng and Shishikui that the residual energy is not uniformly distributed across the whole block. The variance is visibly lowest at the center of each triangular segment and larger at the partitioning line and the block edges. This extends the observations made in Section 5.1.1 that the prediction error is highest at the partitioning boundary. It can also be seen that differences exists between the two triangular split modes. In particular, the triangular split mode from the bottom-left to the top-right, shown on the right of each sub-figure, has a different residual characteristics in each triangle. The top-left triangle for QPs 27 and 32 has a visibly lower variance compared to the bottom-right triangle. Again, this can be explained by the higher correlation in terms of motion vectors that is expected for the top-left triangle, leading to a better motion compensation. For the triangular split from the top-left to the bottom-right, shown on the left of each sub-figure, this difference is much less pronounced.

Both of these results suggest that, despite the biased measurement due to analyzing the RD-optimal, coded non-rectangular partitioned blocks, the two segments may have different residual characteristics that are currently not considered during transform coding. Therefore, in the following, three approaches are presented for the residual coding of GEO and TPM blocks, specifically for the coding of single residual segments.

7.2 Shape-adaptive DCT Coding for GEO

The first approach presented for coding of GEO residuals utilizes the well-known Shape-adaptive DCT (SADCT). The SADCT has originally been developed by Sikora, Makai, Kauff and Schüür [SM95; KS98] for the coding of residuals with arbitrary shapes in M4V. [04]. M4V defines the concept of video objects (VO) which are described by texture, shape and motion (see Section (3.1)). Two types of VOs are distinguished in M4V: For opaque objects, a binary shape information is transmitted using bitmap-based, contour-based, or implicit shape coding techniques. Transparent objects are described by grayscale alpha-maps with 8 bit/sample, defining the shape as well as the transparency of an object. The shape information in M4V is coded on a macroblock basis. The SADCT uses the binary shape information in order to apply transform coding only to the residual samples that are contained in the VO. If the VO contains N active samples, the SADCT will generate N transform coefficients.

In the following, a short review of the forward SADCT algorithm is given. For more information, the reader is referred to [SM95; KS98]. The basic concept of the SADCT is the application of DCT basis functions of varying length l to the residual samples. Compared

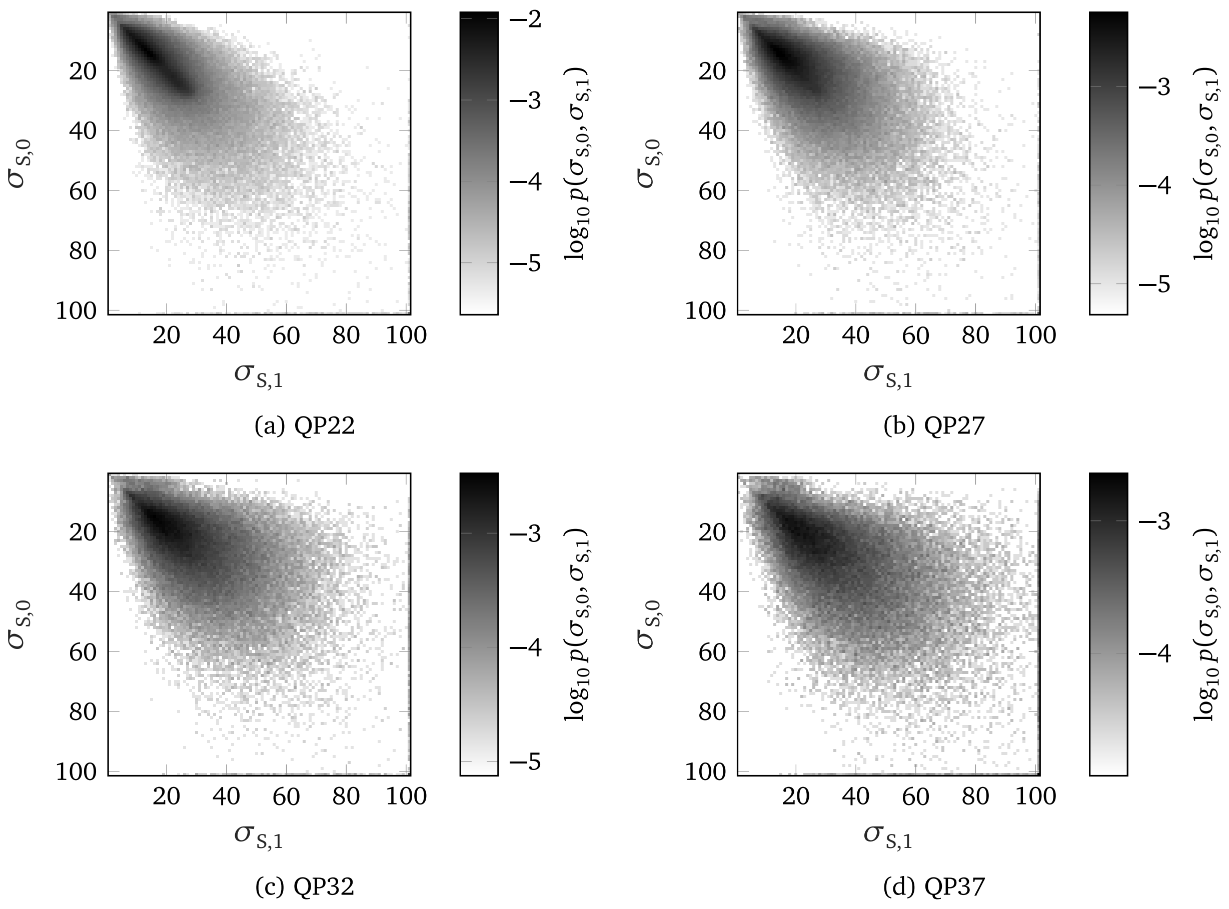


Figure 7.2 Normalized 2D histogram estimating the empirical joint probability density $p(\sigma_{S,0}, \sigma_{S,1})$ of the luma residual energy for GEO blocks. Statistics were gathered from all VTM-3.2 coded GEO blocks for the first 65 pictures of all sequences from the JVET test set.

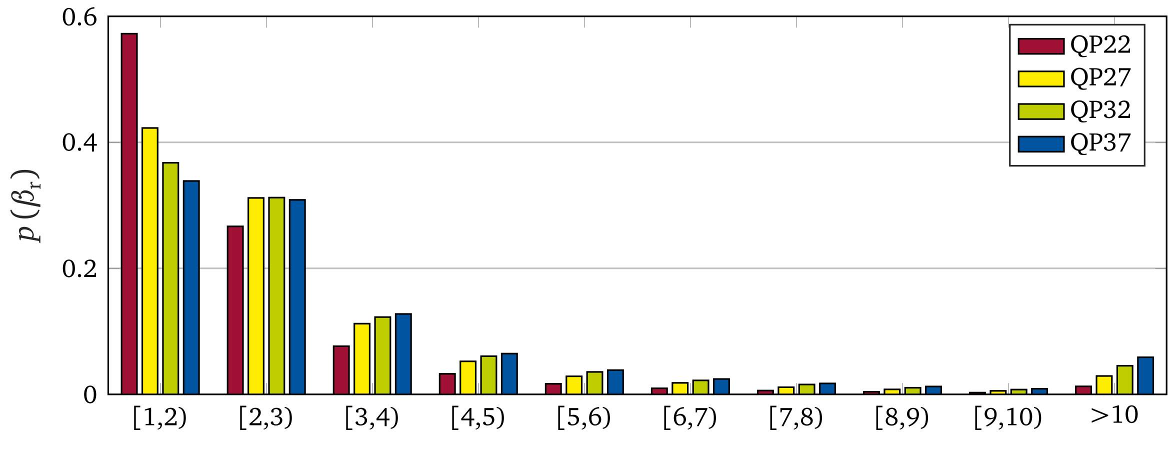
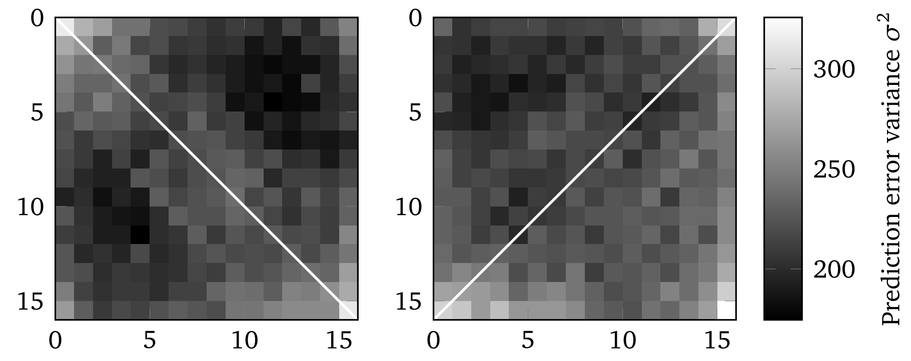
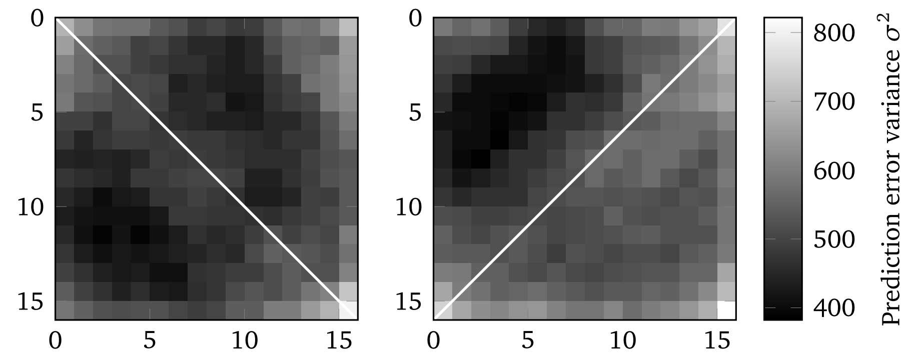


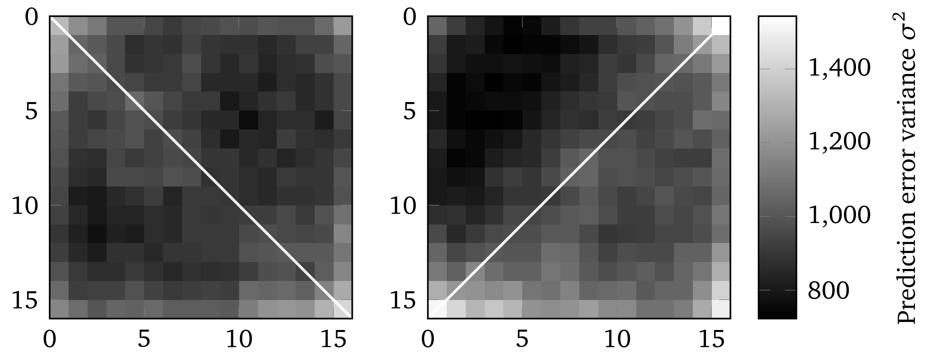
Figure 7.3 Histogram of the variance ratio $\beta_r = \frac{\max(\sigma^2_{S,0}, \sigma^2_{S,1})}{\min(\sigma^2_{S,0}, \sigma^2_{S,1})}$ for the luma residual of all GEO coded blocks. Statistics were gathered from all VTM-3.2 coded GEO blocks for the first 65 pictures of all sequences from the JVET test set.



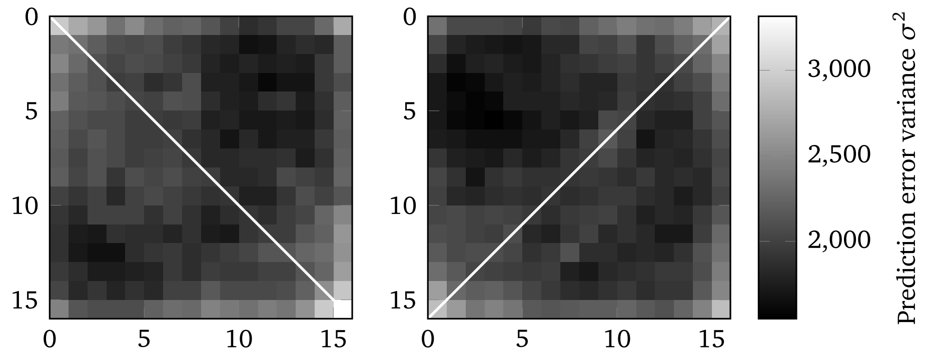
(a) QP22



(b) QP27



(c) QP32



(d) QP37

Figure 7.4 Distribution of the per-sample luma residual error energy $\sigma^2(x, y)$ for TPM blocks of size 16×16. Statistics were gathered from all VTM-3.2 coded TPM blocks for the first 65 pictures of all sequences from the JVET test set.

to the regular, block-based DCT, additional sample-shifting processes are required for two purposes:

- to close any holes in the video object,
- to align the resulting SADCT coefficients at the top-left corner of the block such that regular entropy coding as used for DCT coefficients can be applied.

The SADCT basis functions of length L are defined by $t_L(i,j)$ and grouped in the corresponding transform matrices $\boldsymbol{T}_L$:

$$t_L(i,j) = \sqrt{\frac{2}{L}} \cdot c_0 \cdot \cos\left(i\left(j+\frac{1}{2}\right)\frac{\pi}{L}\right) \tag{7.8}$$

$$c_0 = \begin{cases} \sqrt{\frac{1}{2}}, & \text{if } i = 0 \\ 1, & \text{otherwise} \end{cases} \tag{7.9}$$

$$\boldsymbol{T}_L = \begin{bmatrix} t_L(0,0) & \cdots & t_L(0,L-1) \\ \vdots & \ddots & \vdots \\ t_L(L-1,0) & \cdots & t_L(L-1,L-1) \end{bmatrix} \tag{7.10}$$

For each row i in the binary mask $\boldsymbol{M}_\text{b}$, the value L_i shall denote the number of active samples defined in that row and for each column j, the value L_j denotes the number of samples active in that column.

$$L_i = \sum_x M_\text{b}(x,i) \tag{7.11}$$

$$L_j = \sum_y M_\text{b}(j,y) \tag{7.12}$$

The input samples shall be given by $s(x,y)$ with $\boldsymbol{s}_j$ defining the column vector of samples in column j. The forward SADCT operates as follows:

1. For each column vector $\boldsymbol{s}_j$, the L_j samples of the given partition are shifted to the topmost position and grouped to the column vector $\boldsymbol{s}_j^*$.
2. For each shifted column vector $\boldsymbol{s}_j^*$, the vertical 1D-DCT of length L_j is computed. The output DCT coefficients are denoted as $\boldsymbol{a}_j$:

$$\boldsymbol{a}_j = \boldsymbol{T}_{L_j} \cdot \boldsymbol{s}_j^* \tag{7.13}$$

3. The same procedure is applied in horizontal direction. First, the L_i elements of column vectors $\boldsymbol{a}_j$ that belong to the same row i are shifted to the leftmost position and grouped to row vectors $\boldsymbol{b}_i$.
4. For each shifted row vector, the horizontal 1D-DCT of length L_i is computed.

$$\boldsymbol{c}_i = \boldsymbol{T}_{L_i} \cdot \boldsymbol{b}_i \tag{7.14}$$

5. The output DCT coefficients are denoted as $\mathbf{c}_i$, containing all DCT coefficients.

These steps are visualized in Figure 7.5. In order to compensate for the *mean weighting defect*, the SADCT variant with DC coefficient separation and ΔDC correction is used for the remainder of the thesis. In this scheme, the DC coefficient, e.g. $c(0,0)$ from the definitions above, is calculated separately from the samples using the mean $\mu_{\mathrm{S},k}$ of the given segment S_k. Then, the mean is subtracted from the input samples to the SADCT forward transform process, e.g. $\widetilde{s}(x,y) = s(x,y) - \mu_{\mathrm{S},k}$. For transmission, $c(0,0)$ is replaced with $\mu_{\mathrm{S},k}$, e.g. $c^*(0,0) = \mu_{\mathrm{S},k}$. At the decoder side, $c^*(0,0)$ is set to zero and the SADCT backward transform is performed. This step requires ΔDC correction, since the original $c(0,0)$ can have a non-zero value depending on the shape parameters L_i and L_j, even if the input samples $\widetilde{s}(x,y)$ are zero-mean.

Other deficiencies of the SADCT are well-known and have been pointed out in the literature [BOA96]:

- The shifting operation that aligns the columns with the top block boundary can decrease the correlation between neighboring samples.
- The grouping of AC coefficients into rows after the vertical DCT has been computed does not reflect the correlation expected between DCT coefficients resulting from different transform lengths. It would be beneficial to group together coefficients of different length but similar frequency.

Nevertheless, the SADCT is a simple and straightforward transform that also fulfills the important property that transform coefficients are located at the top-left position just like regular DCT coefficients. This allows the same entropy coding process based on 4×4 coefficient sub-blocks (CSBs) and pre-defined scan-orders of VVC / VTM-3.2 to be reused for coding of SADCT coefficients. Furthermore, the transform basis functions $t_L(i,j)$ of different length L are defined in such a way that the overall transform is considered *pseudo-orthonormal*. Therefore, the same quantizer as for regular transform coefficients can also be reused and no block-size dependent scaling or quantization is required.

In order to adapt the SADCT for GEO, an additional SADCT transform coding mode is introduced which is evaluated by the encoder next to the default DCT-II and Skip-mode coding. This allows the usage of DCT-II or Skip-mode as a fallback option. The signaling of the SADCT mode is performed for transform units and is shown in syntax Table 7.1. Three additional syntax elements are introduced:

1. tu_sadct_flag[x0][y0], a single-context CABAC-coded flag specifying if the SADCT is enabled for the current luma residual of the transform block.
2. sps_sadct_code_partitions_flag, an SPS flag specifying if the SADCT codes a single partition or the area around the partitioning line
3. tu_sadct_partition_flag[x0][y0], a bypass-coded flag coded specifying if the residual of the first segment S_0 or the second segment S_1 is coded. If tu_sadct_partition_flag[x0][y0]=0 is coded, all residual samples belonging to segment S_1 are assumed to be zero. If tu_sadct_partition_flag[x0][y0]=1 is coded, all residual samples belonging to segment S_0 are assumed to be zero. If

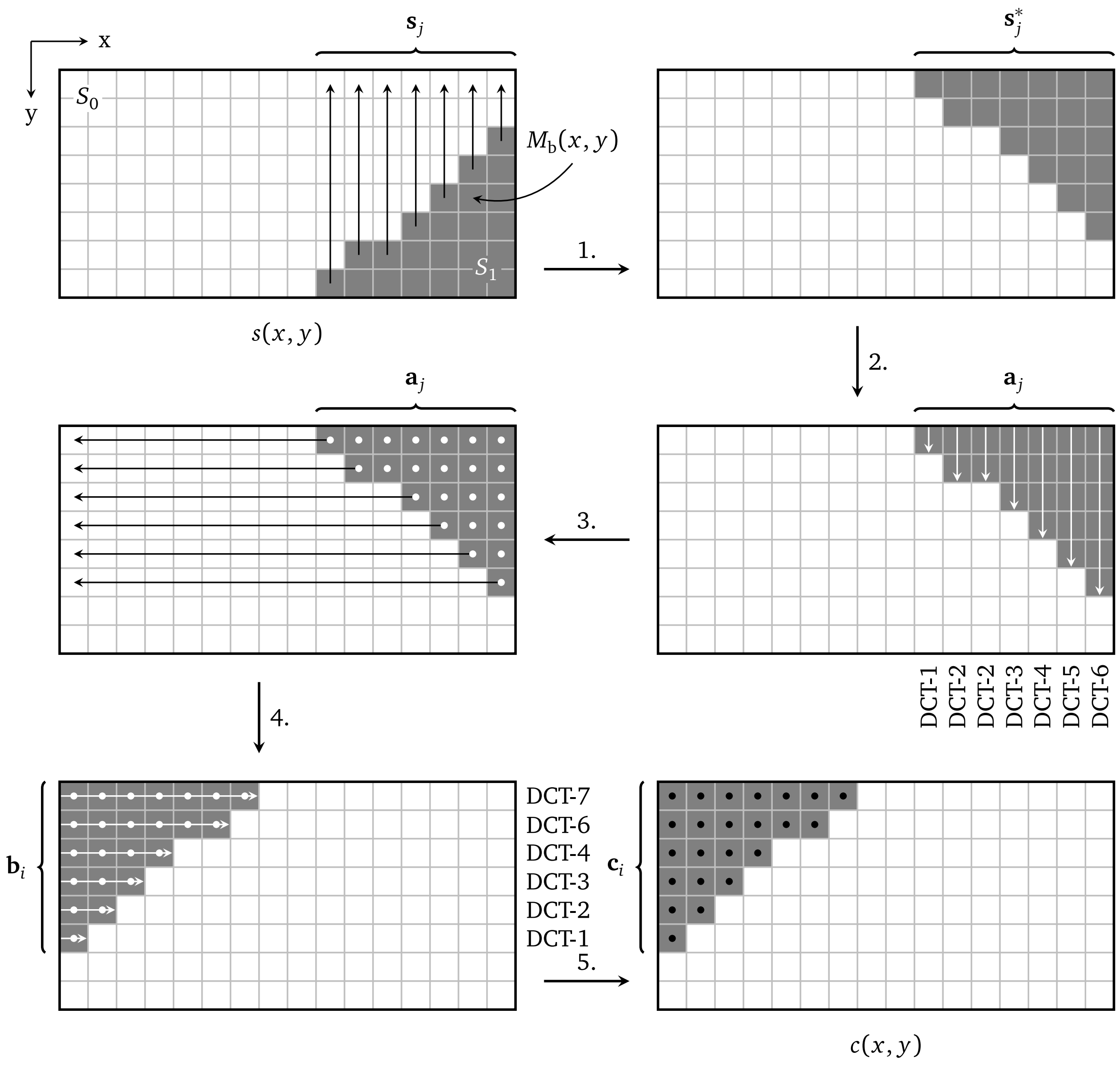


Figure 7.5 Visualization of the SADCT algorithm for an exemplified GEO block.

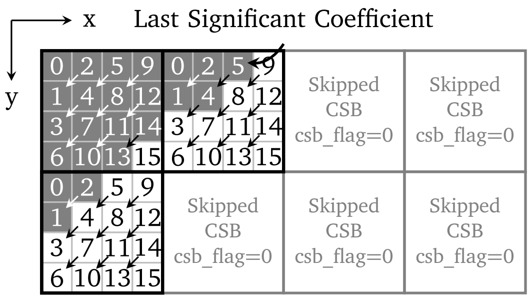


Figure 7.6 Visualization of the SADCT coefficient coding, based on 4×4 CSGs in VTM-3.2. Note that the CSG flags for skipped CSGs are not signaled due to the last significant coefficient position coding beforehand.

sps_sadct_code_partitions_flag is false, then tu_sadct_partition_flag is omitted, and a stripe of residual samples is always transform coded.

The scanning process for CSBs and SADCT coefficients is visualized in Figure 7.6. The last significant coefficient position is signaled first. Derived from the last significant coefficient position, coefficient sub-group flags csg_flag[xS][yS] are either coded or skipped. For coefficient sub-groups that contain at least one non-zero coefficient, csg_flag[xS][yS]=1 is coded. Then, transform coefficient levels of a sub-block are coded in five passes over the scan positions.

As mentioned above, two types of SADCT coding are considered in the next experiments:

1. Experiment 7.28: Coding of the residual samples for segment S_0 or S_1, thereby assuming that the residual in the other segment S_1 or S_0 is equal to zero. This type of coding is performed when sps_sadct_code_partitions_flag is true. The underlying assumption of this coding mode is that motion compensated prediction performs well in one segment but fails or is sub-optimal in the other segment. This coding mode is visualized in Figure 7.7a. The detailed coding results are shown in Table A.36. For this experiment, a marginal improvement of −0.04 % in terms of luma BD-rate change over VTM-3.2 + GEO is reported. The largest relative improvements are measured for *BasketballDrive* at −0.09 % and for the two screen content sequences *SlideEditing* and *SlideShow,* measured at −0.08 % and −0.10 %, respectively.

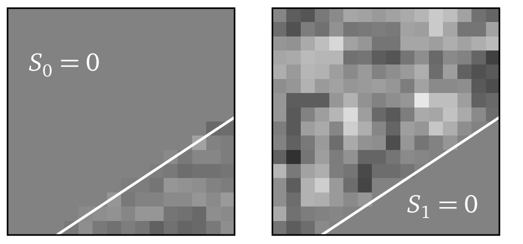
2. Experiment 7.29: Coding of residual samples that are contained in a rectangular stripe of samples around the partitioning line. This type of coding is performed when sps_sadct_code_partitions_flag is false. Here, the assumption is that the highest prediction error is occurring in the blending region, as it was analyzed in Section 5.1.1. This type of coding is visualized in Figure 7.7b. The detailed coding results are shown in Table A.37. A lower improvement in coding efficiency compared to Experiment 7.28 can be reported with a luma BD-rate change of −0.02 % over VTM-3.2 + GEO. Here, the largest improvements are measured for *BQMall* at −0.07 %, *BasketballDrive* and *RitualDance,* both at −0.06 %.

The small impact on coding efficiency can be explained by the low mode utilization, shown in Figure 7.8. Overall, only about 3 % of all luma samples coded with GEO are using the SADCT and the majority of blocks is still coded using the DCT-II, in cases where a residual is signaled. Multiple reasons can be assumed for this behavior, such as higher coefficient coding costs for SADCT coefficients compared to DCT coefficients and slightly worse decorrelation properties.

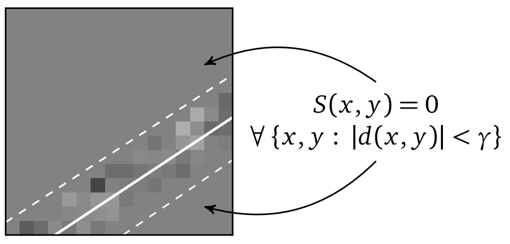
Examples of 32 × 32 luma transform blocks that are using the SADCT are given in Figure 7.9. In each sub-figure, the original, uncompressed block is shown on the left. Next to the original, the prediction block using GEO and resulting uncoded residual is shown. For visual reference, the binary partitioning mask is also shown. On the right side, the quantized SADCT coefficients and the corresponding coded residual in the spatial domain is depicted. Lastly, the reconstructed block (before loop-filtering is applied) is shown. The examples were selected from all SADCT coded blocks conditioned on having a qualitatively high residual energy and are randomly picked. Figures 7.9a and 7.9b both show examples, where motion compensation results in a low prediction error in one segment and a high prediction error

transform_unit(x0, y0, tbWidth, tbHeight, treeType) {	**Descriptor**
if(treeType = = SINGLE_TREE | | treeType = = DUAL_TREE_LUMA)	
tu_cbf_luma[x0][y0]	ae(v)
if(treeType = = SINGLE_TREE | | treeType = = DUAL_TREE_CHROMA) {	
tu_cbf_cb[x0][y0]	ae(v)
tu_cbf_cr[x0][y0]	ae(v)
}	
if((tu_cbf_luma[x0][y0] | | tu_cbf_cb[x0][y0] | | tu_cbf_cr[x0][y0]) && treeType != DUAL_TREE_CHROMA) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	ae(v)
}	
}	
if((((CuPredMode[x0][y0] = = MODE_INTRA) && sps_mts_intra_enabled_flag) | | ((CuPredMode[x0][y0] = = MODE_INTER) && sps_mts_inter_enabled_flag)) && tu_cbf_luma[x0][y0] && treeType != DUAL_TREE_CHROMA && (tbWidth <= 32) && (tbHeight <= 32))	
tu_mts_flag[x0][y0]	ae(v)
if(tu_cbf_luma[x0][y0])	
residual_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
if(tu_cbf_cb[x0][y0])	
residual_coding(x0, y0, Log2(tbWidth / 2), Log2(tbHeight / 2), 1)	
if(tu_cbf_cr[x0][y0])	
residual_coding(x0, y0, Log2(tbWidth / 2), Log2(tbHeight / 2), 2)	
}	
if(CuPredMode[x0][y0] = = MODE_INTER && tu_cbf_luma[x0][y0] && tbWidth <= 64 && tbHeight <= 64 && !transform_skip_flag[x0][y0][0] && !tu_mts_flag[x0][y0]) {	
tu_sadct_flag[x0][y0]	ae(v)
if(tu_sadct_flag[x0][y0] && sps_sadct_code_partitions_flag**)**	
tu_sadct_partition_flag[x0][y0]	ae(v)
}	

Table 7.1 Syntax table for the signaling of SADCT for a transform unit in accordance with the VVC (Draft 3) specification.



(a) SADCT transform coding of residuals S_0 or S_1



(b) SADCT transform coding of transition zone

Figure 7.7 Visualization of the two SADCT coding modes, depending on the value of the SPS syntax element sps_sadct_code_partitions_flag. Coding mode (a) is tested in Experiment 7.28, coding mode (b) is tested in Experiment 7.29.

in the other segment, due to a highly textured content, likely belonging to the background. For these two cases, the reconstruction quality is improved from about 26 dB for the prediction block to over 30 dB by coding a residual using the SADCT. Figures 7.9c and 7.9d show two examples of disocclusions, where new content is being uncovered. For Figure 7.9c, the reconstruction quality improves only slightly from 29.23 dB to 30.01 dB but considerably more for the larger disocclusion depicted in Figure 7.9d. For this selected example, the coding of an SADCT-based residual improves the prediction from 22.56 dB to 31.96 dB for the reconstruction.

Although these are only a few selected examples, they nevertheless show promising results that the SADCT performs as designed for cases where a significant residual is present in only one GEO segment. However, the SADCT seems not to be competitive for the vast majority of GEO coded blocks and the resulting prediction errors. Further research and specific adaptations regarding quantization and entropy coding of SADCT coefficients may boost the performance. It is also noted that the shifting operation in the SADCT algorithm is actually not required, since the residual samples are always arranged in a triangular shape or a shape that can be decomposed into one or two rectangles and a triangle. Rotating and flipping the residual can therefore provide a similar concentration of samples at the top-left corner and would not impact the correlation of samples.

7.3 Transform-skipping for GEO

Another option for residual coding of non-rectangular partitions with distinct distributions is transform-skipping. As introduced in Chapter 2, transform-skipping omits the 2D blocks transform and encodes the quantized residual directly. This can also be seen as the application of an identity transform, e.g. $\boldsymbol{T} = \boldsymbol{I}_n$ of size n in horizontal or vertical direction. Transform-skipping is one essential coding tool for screen content, due to the very weakly correlated prediction error observed for such content. For GEO, transform-skipping might be favorable for very small segments with a high prediction error. Since transform-skipping is already part of VVC and implemented in VTM-3.2[2], it is sufficient to explicitly allow the us-

[2]It is noted that the transform skip coding has changed for VVC / VTM versions > 3.2. For example, the last significant coefficient coding is omitted in the most recent versions.

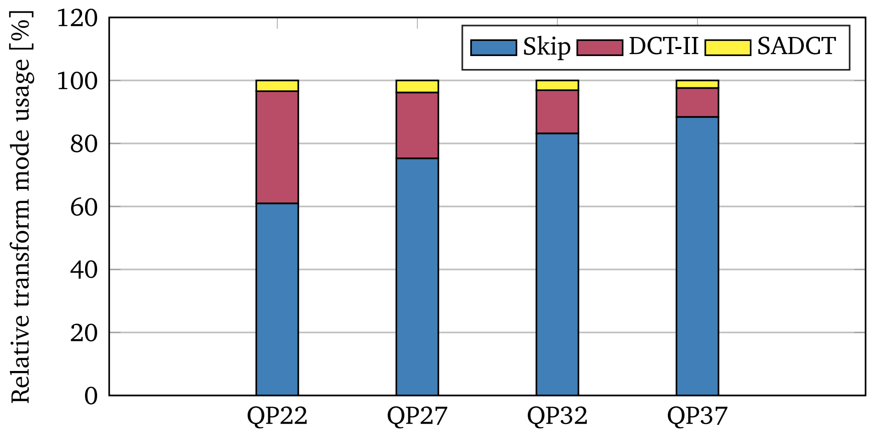


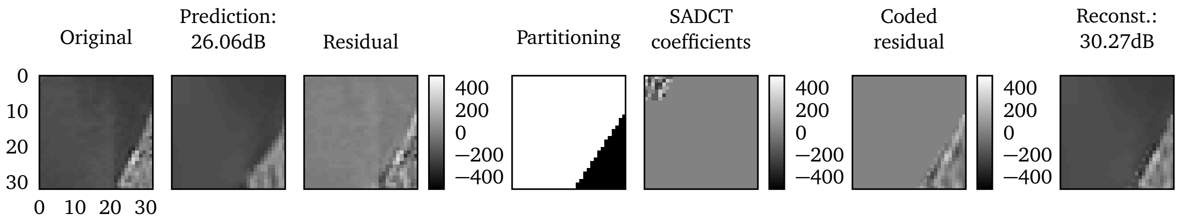
Figure 7.8 Relative transform coding mode usages for GEO blocks across all classes, measured for Experiment 7.28.

age of transform skip with GEO for any given block size. Three different coding experiments are conducted to assess the impact of transform-skipping:

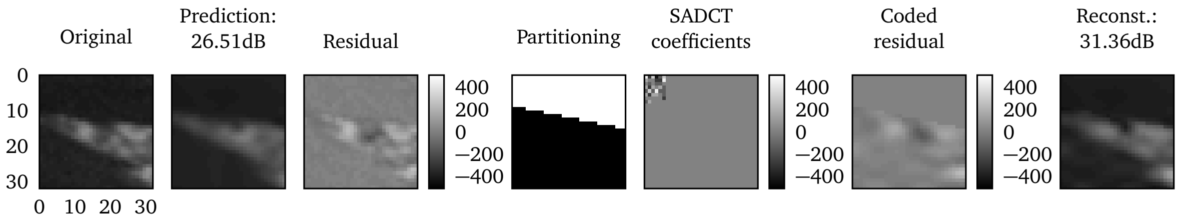
1. Experiment 7.30: Coding of the residual using transform-skip for the entire $w \times h$ sized transform block or regular DCT-based coding. The detailed coding results are presented in Table A.38. Overall, this improves the coding efficiency slightly compared to VTM-3.2 + GEO. The luma BD-rate changes by −0.05 % according to the JVET CTC. For classes A1 and A2, no improvement in coding efficiency can be reported, likely due to a higher percentage of large block sizes utilized for UHD content. The largest improvements were measured for class D and F sequences with −0.14 % and −0.10 %,respectively. For individual sequences, the coding performance was improved specifically for *BQTerrace* (−0.35 %), *BQSquare* (−0.31 %) and SlideEditing (−0.22 %). However, *BQTerrace* and *BQSquare* are both considered to be quite noisy sequences, which might explain the larger coding gain for these sequences. The higher coding gain measured for screen content sequences is to be expected on the other hand.

2. Experiment 7.31: Coding of one residual segment, signaled by an additional flag and quantization of the other segment residual to zero. The detailed coding results are presented in Table A.39. In this setup, the encoder determines which segment shall be coded with transform-skip and which shall be quantized to zero based on RD-optimization. As a fallback option, DCT coding of the entire residual block is also possible. Overall, the impact of coding efficiency is lower, measured at −0.02 % according to the JVET CTC. Even smaller coding efficiency reductions are measured for the sequence classes compared to Experiment 7.30.

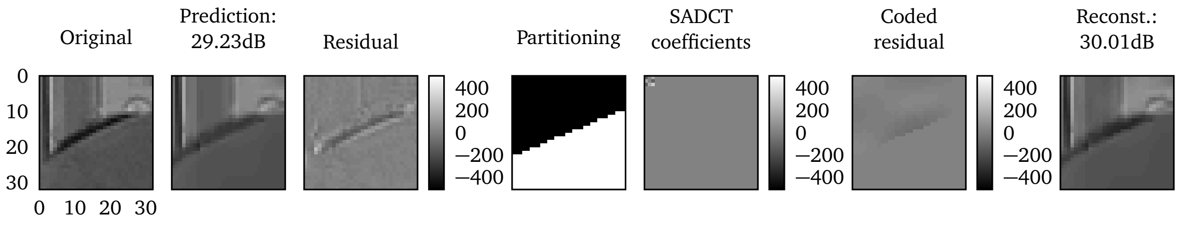
3. Experiment 7.32: Coding of one residual segment, signaled by an additional flag and quantization of the other segment residual to zero as in Experiment 7.31. In addition, the residual is shifted to the top-left corner, similarly as it is performed for SADCT coding. The reasoning behind this approach is visualized in Figure 7.10: In order to improve the coding of residuals that are not entirely located at the top-left corner of the transform block, the residual samples are shifted to the top block-boundary



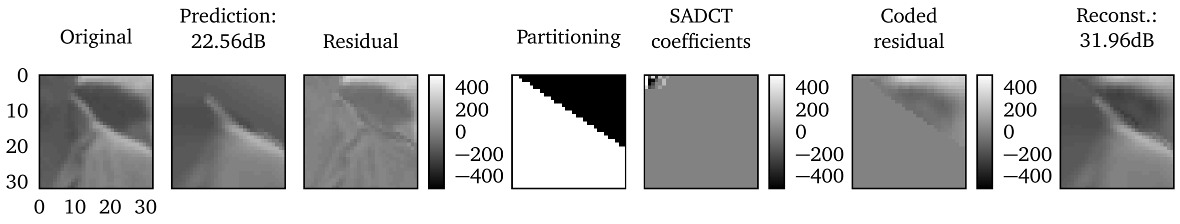
(a) RaceHorsesC, POC90, [448 224]



(b) MarketPlace, POC14, [1024 320]



(c) BQMall, POC2, [96 192]



(d) RitualDance, POC60, [896 608]

Figure 7.9 Selected examples for 32 × 32 GEO predicted and SADCT coded blocks for JVET test sequences, coded at QP32.

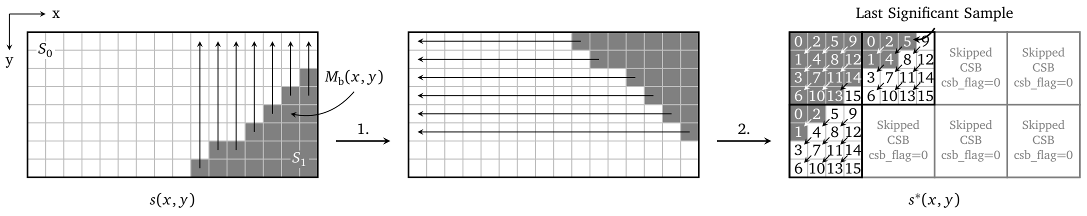


Figure 7.10 Visualization of residual sample shifting applied before transform-skip coding used in Experiment 7.31.

first, followed by a shift to the left. Therefore, the last significant coefficient coding can be efficiently applied and Coefficient sub-group (CSG) flag bits for irrelevant CSG are saved. Compared to Experiment 7.31 without this modification, coding efficiency again increases to −0.05 % over the JVET test set. Furthermore, the coding gain for class D improves to −0.20 %.

Overall, it can be stated that transform-skipping in combination with GEO only impacts sequences of small resolutions (class D) with measurable significance. Transform-skipping applied to larger GEO predicted blocks does not seem to be competitive in the rate-distortion sense. However, allowing transform-skipping for larger, regular partitioned blocks might also result in similar results and should be investigated in the future.

7.4 Encoder-side Transform Optimizations with Decoder-side Masking

Alternative transform coding approaches for non-rectangular shapes that still rely on the regular DCT-II are also possible. The general idea behind such approaches involves an encoder side optimization in the spatial or transform domain such that the block transform is only applied to the relevant segment. At the decoder side, a masking operation is required for reconstruction, in order to set the reconstructed sample outside of the relevant segment to zero. Here, different methods can be distinguished:

1. Padding or extension algorithms that insert samples into the region outside of the current segment, e.g. for the case of GEO, if segment S_0 is to be coded, the samples of segment S_1 are being padded or extended and vice versa[3]. For certain shapes, such as triangular partitions, symmetric extensions can be easily determined that correspondingly exploit the symmetry properties of the DCT. After the padding has been generated, the regular DCT-II can be applied. At the decoder side, only the relevant reconstructed samples are masked out.

2. Iterative encoder-side optimization that chooses the best DCT bases considering only the approximation error in the relevant segment. Since the DCT bases are overcomplete for a subspace smaller than the given block, this essentially becomes a sparse

[3]Zero-padding is considered the most trivial approach and therefore not further investigated. It is obvious that a zero-padded residual will increase the number of high-frequency coefficients. If those are quantized to zero, a serious degradation in compression performance is to be expected.

approximation problem. A well-known method for such an optimization is Orthogonal matching pursuit (OMP). This optimization can be formalized as,

$$\min_{c} ||\boldsymbol{s} - \boldsymbol{T} \cdot \boldsymbol{c}||_2^2 \text{ subject to } ||\boldsymbol{c}||_0 < L \tag{7.15}$$

where $\boldsymbol{s}$ in this case is the column-wise scanned 2D signal, to be represented by the column-wise scanned DCT transform basis images in $\boldsymbol{T}$ and the coefficients $\boldsymbol{c}$. To get a sparse representation, the L0 norm is chosen as the imposed optimization criterion. Only L non-zero coefficients are retained. Equation 7.15 can be easily extended to arbitrary shapes using a *masked* version of OMP using a binary mask $\boldsymbol{M}$,also scanned column-wise into the vector $\boldsymbol{m}$:

$$\min_{c} ||\boldsymbol{m} \circ (\boldsymbol{s} - \boldsymbol{T} \cdot \boldsymbol{c})||_2^2 \text{ subject to } ||\boldsymbol{c}||_0 < L \tag{7.16}$$

The downside of this approach is given by the fact that the masked DCT transform basis images $\boldsymbol{m} \circ \boldsymbol{T}$ are an overcomplete set for the masked residual.

Lastly, transforms with arbitrarily shaped DCT basis images can also be easily found by projecting each 2D block DCT basis image $\boldsymbol{T}_{j_1,j_2}^{M,N}$ of size $M \times N$ with $0 \leq j_1 < M$ and $0 \leq j_2 < N$ onto the subspace defined by a segment S_k, $k \in \{0,1\}$ using the binary mask $\boldsymbol{M}_{\mathrm{b},k}$ as follows:

$$t_M(i,j) = \sqrt{\frac{2}{M}} c_0 \cos\left(j\left(i+\frac{1}{2}\right)\frac{\pi}{M}\right),\ c_0 = \begin{cases}\sqrt{\frac{1}{2}} & \text{if } j=0\\ 1 & \text{otherwise}\end{cases} \tag{7.17}$$

$$\boldsymbol{t}_j^M = \begin{bmatrix} t_M(0,j) & t_M(1,j) & \ldots & t_M(M-1,j) \end{bmatrix}^{\mathrm{T}} \tag{7.18}$$

$$\boldsymbol{T}_{j_1,j_2}^{M,N} = \boldsymbol{t}_{j_1}^M \cdot \left[\boldsymbol{t}_{j_2}^N\right]^{\mathrm{T}} \tag{7.19}$$

$$\hat{\boldsymbol{T}}_{j_1,j_2}^{M,N} = \boldsymbol{T}_{j_1,j_2}^{M,N} \circ \boldsymbol{M}_{\mathrm{b},k} \tag{7.20}$$

The full set of new basis images $\hat{\boldsymbol{T}}_{j_1,j_2}^{M,N}$, which are zero outside of the given segment S_k, is not optimal since it represents an overcomplete set of basis images in general. Due to the lower rank of $\hat{\boldsymbol{T}}_{j_1,j_2}^{M,N}$, some basis images are actually redundant and mutually dependent of each other. An optimal set of basis images would correspond to the full rank of the subspace. A well-known method to reduce the number of $M \cdot N$ basis images and find a smaller, orthogonal subset from a larger set of (non-orthogonal) functions is the Gram-Schmidt orthogonalization. This concept has been discussed by Gilge and extended to arbitrary bases, such as polynomials, splines and sinusoidal functions [Gil90]. The drawback of such an approach is the loss of separability and different sets of transforms existing for each block size and shape combination. Considering that 15 unique block sizes are available for VVC and about 140 GEO partitions per block size, this would require the precomputation of 4200 transform sets.

In the following, details are presented regarding the first two practical approaches, using a symmetric extension of the residual and the consideration of OMP for coefficient selection.

7.4.1 Symmetric Extension for TPM

For residual blocks that are partitioned by TPM or specific GEO modes passing through the center of the given block, a symmetric extension can be easily derived as follows: A prediction error block of size $w \times h$ shall be considered, partitioned into two triangular shaped segments S_0 and S_1. If the segment S_0, containing the residual samples $s_0(x,y)$, shall be transform coded using the DCT-II and the segment S_1 is considered to contain no residual, e.g. $s_1(x,y) = 0,\ \forall\ x,y \in S_1$, a point-symmetric extension of S_0 is given by $\tilde{S_0}$ with samples $\tilde{s}_0(x,y)$ by:

$$\tilde{s}_0(x,y) = s_0(w-1-x, h-1-y) \tag{7.21}$$

Then, a symmetric extension $S_{0,\mathrm{E}}$ is generated by $S_{0,\mathrm{E}} = S_0 + \tilde{S_0}$. It can now be shown in accordance with [Din+13] that the DCT-II transform bases $\sqrt{2}\boldsymbol{T}^{M,N}_{j_1,j_2}$ with $0 \le j_1 < M$ and $0 \le j_2 < N$ and all $j_1 + j_2$ even, $M = h$ and $N = w$, form an orthogonal set for the block $S_{0,\mathrm{E}}$. For completeness, the derivation of this assertion is given below. The two-dimensional DCT-II basis images $\boldsymbol{T}^{M,N}_{j_1,j_2}$ can also be written as:

$$t^{M,N}_{j_1,j_2}(m,n) = c_{j,1} c_{j,2} \cos\left(j_1\left(m+\frac{1}{2}\right)\frac{\pi}{M}\right)\cos\left(j_2\left(n+\frac{1}{2}\right)\frac{\pi}{N}\right) \tag{7.22}$$

$$c_{j,1} = \begin{cases} \sqrt{\frac{1}{M}} & \text{if } j_1 = 0 \\ \sqrt{\frac{2}{M}} & \text{otherwise} \end{cases} \tag{7.23}$$

$$c_{j,2} = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j_2 = 0 \\ \sqrt{\frac{2}{N}} & \text{otherwise} \end{cases} \tag{7.24}$$

Two transform bases $t^{M,N}_{j_1,j_2}(m,n)$ and $t^{M,N}_{j_3,j_4}(m,n)$ are orthogonal if the condition

$$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} t^{M,N}_{j_1,j_2}(m,n)\, t^{M,N}_{j_3,j_4}(m,n) = \delta(j_3 - j_1)\delta(j_4 - j_2) \tag{7.25}$$

is met. Using Equation 7.21 and considering that $S_{0,\mathrm{E}}$ is the union of S_0 and $\tilde{S_0}$, Equation 7.25 can also be written as:

$$\sum_{m,n\in S_0} t^{M,N}_{j_1,j_2}(m,n)\, t^{M,N}_{j_3,j_4}(m,n) + \sum_{m,n\in S_0} t^{M,N}_{j_1,j_2}(M-1-m, N-1-n)\, t^{M,N}_{j_3,j_4}(M-1-m, N-1-n) = \delta(j_3 - j_1)\delta(j_4 - j_2) \tag{7.26}$$

The mirrored transform bases $t^{M,N}_{j_1,j_2}(M-1-m, N-1-n)$ can then be rewritten as

$$t^{M,N}_{j_1,j_2}(M-1-m, N-1-n) = (-1)^{j_1+j_2}\, t^{M,N}_{j_1,j_2}(m,n) \tag{7.27}$$

using the symmetry properties of the cosine function. With the condition that $j_1 + j_2$ and $j_3 + j_4$ are even and using Equation 7.27, Equation 7.26 can be rewritten as

$$2\sum_{m,n\in S_0} t^{M,N}_{j_1,j_2}(m,n)\, t^{M,N}_{j_3,j_4}(m,n) = \delta(j_3 - j_1)\delta(j_4 - j_2), \tag{7.28}$$

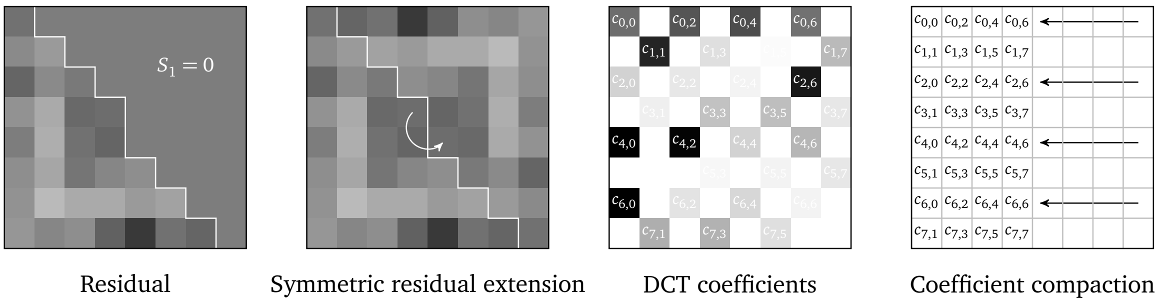

$S_1 = 0$
Residual
Symmetric residual extension
DCT coefficients
Coefficient compaction
$c_{0,0}$ $c_{0,2}$ $c_{0,4}$ $c_{0,6}$
$c_{1,1}$ $c_{1,3}$ $c_{1,5}$ $c_{1,7}$
$c_{2,0}$ $c_{2,2}$ $c_{2,4}$ $c_{2,6}$
$c_{3,1}$ $c_{3,3}$ $c_{3,5}$ $c_{3,7}$
$c_{4,0}$ $c_{4,2}$ $c_{4,4}$ $c_{4,6}$
$c_{5,1}$ $c_{5,3}$ $c_{5,5}$ $c_{5,7}$
$c_{6,0}$ $c_{6,2}$ $c_{6,4}$ $c_{6,6}$
$c_{7,1}$ $c_{7,3}$ $c_{7,5}$ $c_{7,7}$

Figure 7.11 Visualization of symmetric residual extension for a TPM block and resulting DCT-II transform coefficients.

proving that $\sqrt{2}\boldsymbol{T}^{M,N}_{j_1,j_2}$ for all $j_1 + j_2$ even, form an orthogonal set over the symmetrically extended block $S_{0,\mathrm{E}}$. Intuitively, the result is quite obvious, since a point-symmetric signal can be perfectly represented by all basis images with even symmetry. Figure 7.11 visualizes the method for a TPM block of sizes 8 × 8. Note how the transform coefficients $c_{m,n}$ are arranged in a checkerboard pattern, fulfilling the above condition that no transform base with $m + n$ odd is required.

Since coding of these coefficients using the default context modeling would not be a sensible approach, a final coefficient compaction process is applied, by shifting all coefficients to the left:

$$\hat{c}(m,n) = \begin{cases} c\left(m, \frac{n}{2}\right) & \text{if } m \text{ even} \\ c\left(m, \frac{n-1}{2}\right) & \text{otherwise} \end{cases} \tag{7.29}$$

The compacted coefficients $\hat{c}(m,n)$ are then coded using the default entropy coding approaches of VVC. The symmetric residual extension of a luma segment is signaled in the same manner as the usage of the SADCT in Section 7.2. A first flag, coded per TU, indicates the usage of symmetric extension for the current block. If the value of the flag is one, a second flag indicates which triangle shall be symmetrically extended. Two coding setups are now experimentally tested, enabling the symmetric extension mode depending on the block size. In both coding experiments, only TPM with the optional symmetric residual extension is enabled and the GEO coding tool is turned off.

1. In Experiment 7.33, symmetric residual extension is enabled for all TPM block sizes. The detailed coding results are shown in Table A.41. The coding results indicate that this approach is not competitive. Overall, a slight loss in coding efficiency is reported with an almost negligible luma BD-rate change of +0.01 % overall.

2. In Experiment 7.34, symmetric residual extension is enabled only for small TPM blocks with $w \leq 16$ and $h \leq 16$, for which the signaling of a residual becomes more likely. The detailed coding results are shown in Table A.42. The coding efficiency is again only marginally effected with a luma BD-rate change of +0.01 %. Although the losses for UHD sequences are now closer to zero, this experiments validates that this is not a competitive way of coding triangular shaped residuals, even for smaller resolutions.

The common reason behind the almost non-existent impact on coding efficiency is again the very low usage of this coding mode, shown in Figure 7.12 for Experiment 7.34. For a QP

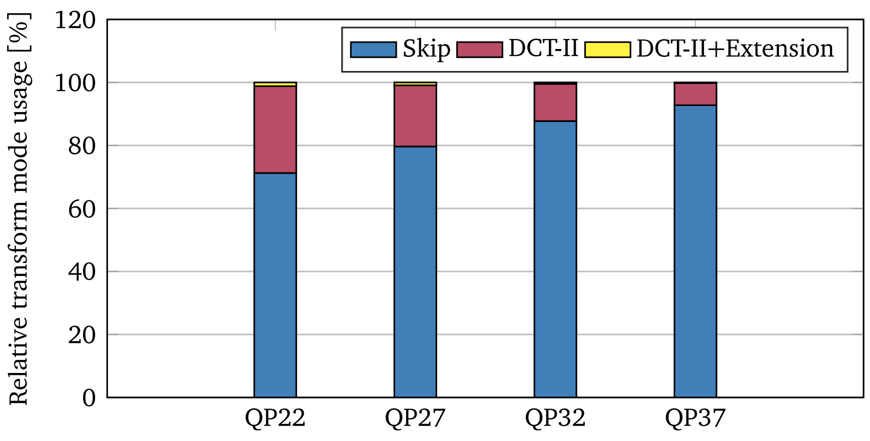


Figure 7.12 Relative transform coding mode usages for TPM blocks across all classes, measured for Experiment 7.34.

of 22, only 1.17 % of all TPM samples are coded using the symmetric extension. This low usage declines even further to 0.18 % for QP37.

In summary, it must be stated that a symmetric extension of the residual according to the TPM split is not a competitive method. Although improvements to the coefficient entropy coding can probably be investigated, there seems to be no evidence that this will increase the coding efficiency of TPM substantially.

7.4.2 Orthogonal Matching Pursuit (OMP)

The possibility of using OMP is only discussed theoretically in the context of this thesis. As it was explained above, the goal of the iterative OMP process is to select sparse coefficients $\boldsymbol{c}$ from an overcomplete transform $\boldsymbol{T}$ to approximate a signal $\boldsymbol{s}$. Considering that $s(x,y)$ is the discrete two-dimensional function to be approximated and $t_\lambda(x,y)$ (or given as column vectors $\boldsymbol{t}_\lambda$) are the basis functions of $\boldsymbol{T}$ with $0 \leq \lambda < N$, e.g. $N = w \cdot h$ for a given block. The goal of OMP is to minimize the error $e(s,g)$ in a least-squares sense given by:

$$g(x,y) = c_0 t_0 + c_1 t_1 + \ldots + c_{N-1} t_{N-1} \tag{7.30}$$

$$e(s,g) = \sum_x \sum_y (g(x,y) - s(x,y))^2 \tag{7.31}$$

The coefficients $\boldsymbol{c}$ that best approximate the signal $s(x,y)$ are found by setting the partial derivatives of $e(s,g)$ to zero, leading to the following set of normal equations [Gil90]:

$$\sum_{n=0}^{N-1} c_n \sum_x \sum_y t_n(x,y) t_m(x,y) \overset{!}{=} \sum_x \sum_y s(x,y) t_m(x,y), \text{ for } m = 0 \ldots N-1 \tag{7.32}$$

It can be easily seen using Equation 7.25 that this set of coupled equations simplifies for orthogonal bases to

$$c_m = \sum_x \sum_y s(x,y) t_m(x,y) \tag{7.33}$$

which is nothing but the well-known multiplication of the signal by the transform matrix, e.g. $\boldsymbol{c} = \boldsymbol{T}\boldsymbol{s}$. For non-orthogonal bases however, the problem given by Equation 7.32 remains ill posed. Matching pursuit (MP) [MZ93] tries to solve this problem iteratively by subtracting one weighted basis $c_{\lambda_k} \cdot \boldsymbol{t}_{\lambda_k}$ from $\boldsymbol{s}$ in each step k, where c_{λ_k} is determined successively from the transform bases that have the highest inner product, e.g. correlation, with the signal $\boldsymbol{s}$ in the first step and the residual $\boldsymbol{r}$ in subsequent steps, updating a prediction $\hat{\boldsymbol{s}}_k$ according to:

$$\boldsymbol{r}_k = \boldsymbol{r}_{k-1} - c_{\lambda_k} \boldsymbol{t}_{\lambda_k}, \text{ with } \boldsymbol{r}_0 = \boldsymbol{s} \tag{7.34}$$

$$\hat{\boldsymbol{s}}_k = \hat{\boldsymbol{s}}_{k-1} + c_{\lambda_k} \boldsymbol{t}_{\lambda_k}, \text{ with } \hat{\boldsymbol{s}_0} = 0 \tag{7.35}$$

In each step, MP chooses a different transform base given by its index λ_k. The residual $\boldsymbol{r}$ quickly converges to zero as stated in [Tro04].

OMP [Tro04; TG07] initializes the approximation in the same way but adds a least squares minimization to each step of MP to obtain the best approximation over transform bases that have already been chosen. In each step, the best transform index λ_k is determined from the transform set with highest inner product between the residual of the previous step and the transform bases:

$$\lambda_k = \underset{j \in \{0,\ldots,N-1\}}{\arg\max} \left| \boldsymbol{r}_{k-1} \cdot \boldsymbol{t}_j \right| \tag{7.36}$$

Then, the transform matrix of selected basis function $\boldsymbol{T}_k$ is updated by adding the transform base $\boldsymbol{t}_{\lambda_k}$ to it, e.g. $\boldsymbol{T}_k = [\boldsymbol{T}_{k-1}\ \boldsymbol{t}_{\lambda_k}]$ with $\boldsymbol{T}_0$ defined as an empty matrix. The approximation $\hat{\boldsymbol{s}}_k$ and residual $\boldsymbol{r}_k$ are then derived by:

$$\boldsymbol{c}_k = \arg\min_{\boldsymbol{c}} \|\boldsymbol{s} - \boldsymbol{T}_k \boldsymbol{c}\|_2^2 \tag{7.37}$$

$$\hat{\boldsymbol{s}}_k = \boldsymbol{T}_k \boldsymbol{c}_k \tag{7.38}$$

$$\boldsymbol{r}_k = \boldsymbol{s} - \hat{\boldsymbol{s}}_k \tag{7.39}$$

Therefore, the residual $\boldsymbol{r}_k$ is always orthogonal to the columns of $\boldsymbol{T}_k$, hence the name of the algorithm.

The stopping criterion for OMP can be defined in multiple ways, e.g. based on a maximum number K of steps, e.g. coefficients, or a threshold for the MSE of $\boldsymbol{r}_k$.

In the following, a comparison is made between DCT, SADCT and OMP-based DCT coding of residual segments. For a fair comparison, the reconstruction quality of all three approaches is measured by the MSE between the original segment $s(x,y)$ and the reconstruction estimate $\hat{s}_k(x,y)$, where k denotes the number of high energy coefficients $\boldsymbol{c}$ of the full transform that are kept. All other coefficients are quantized to zero:

$$\hat{\boldsymbol{s}}_{k,\{\text{DCT, SADCT, OMP}\}} = \boldsymbol{T}_{\{\text{DCT, SADCT}\}} \boldsymbol{c}_{k,\{\text{DCT, SADCT, OMP}\}} \tag{7.40}$$

$$\boldsymbol{c}_{k+1} := [\max(\boldsymbol{c} - \boldsymbol{c}_k)] + \boldsymbol{c}_k \tag{7.41}$$

$$\boldsymbol{c}_0 := [\max(\boldsymbol{c})] \tag{7.42}$$

In this notation, $[\max(\boldsymbol{a})]$ returns a vector of same size as $\boldsymbol{a}$ and containing the single highest value of $\boldsymbol{a}$ at the corresponding position. Figure 7.13 shows the mean squared error of the reconstruction for a given number of coefficients of an AR(1) distributed residual segment with $\rho = 0.6$, partitioned by a geometric split. It can be seen from the simulation that an OMP-based DCT coefficient selection outperforms even the SADCT, meaning that qualitatively fewer transform coefficients are required for (nearly) perfect reconstruction. Unsurprisingly, the block-based DCT is not competitive to the SADCT and OMP-based DCT, due to the effect of zero-padding. Figure 7.14 shows the corresponding transform coefficients for DCT, SADCT and OMP-based DCT required for perfect reconstruction of the masked residual. Here, however, a severe drawback of OMP compared to the SADCT becomes evident: Although the coefficients are very sparse, they are also scattered across the transform block. The entropy coding scheme however expects transform coefficients to be compact and located in the top-left corner where most of the transform domain signal energy is concentrated in low frequency base functions. Therefore, coding experiments using OMP performed during the work on this thesis have been unsuccessful in providing any significant coding gain. A potential solution for this problem could be to constrain the OMP algorithm further and add a rate regularization term to the minimization problem that reflects the required property:

$$\boldsymbol{c}_k = \arg\min_{\boldsymbol{c}} \|\boldsymbol{s} - \boldsymbol{T}_k\boldsymbol{c}\|_2^2 + \lambda R(\boldsymbol{c}_{\mathrm{q}}) \tag{7.43}$$

Where $R(\cdot)$ is an estimate of the bits required for coding the quantized coefficients $\boldsymbol{c}_{\mathrm{q}}$. However, this introduces multiple problems to finding the solution for Equation 7.43, since the coefficient coding requires quantization to be applied first. This is a non-differentiable operation and therefore difficult to model as well as the dependent entropy coding process. Similar problems have been recognized in [Jia+16] in the context of using OMP for still image coding. It is stated by the authors that the distribution of coded transform indices λ for OMP-determined coefficients – although using transforms or *dictionaries* other than the DCT for their specific problem – is rather uniform. Therefore, a fixed-length coding of non-zero transform indices is proposed instead of a scan-based, bit-plane coding as used in HEVC and VVC. The effect of such an adaptation remains to be investigated.

Lastly it is to be summarized that OMP-based coefficients selection for an overcomplete, masked DCT base could be an interesting aspect to study further in the future. If a solution to the problem of efficiently coding sparse and scattered coefficients can be found or if the sparse optimization can be regularized accordingly, this approach remains attractive since only an additional masking is required for reconstruction at the decoder. Unlike the SADCT case, no new transform bases need to be stored or computed at run time. An iterative algorithm such as OMP however, could be an additional complexity burden for an encoder.

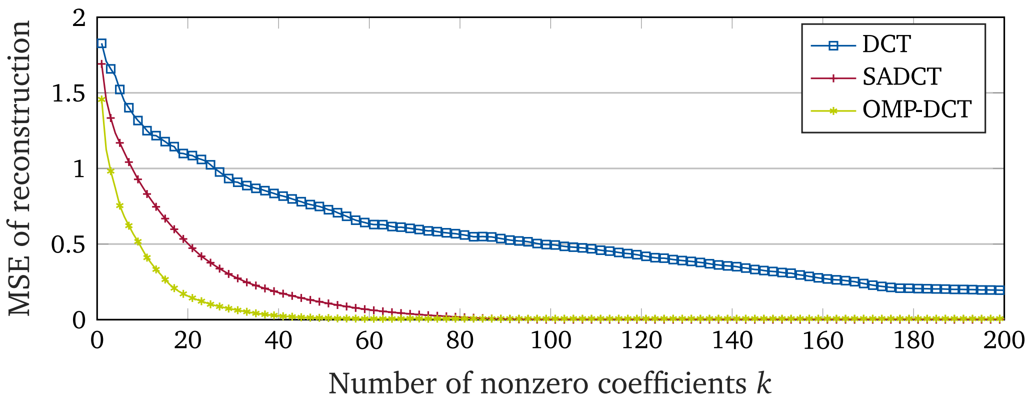


Figure 7.13 Comparison of reconstruction quality for different transform schemes in terms of MSE for a geometrically partitioned, AR(1) distributed residual, $\rho = 0.6$, depending on the number of retained transform coefficients.

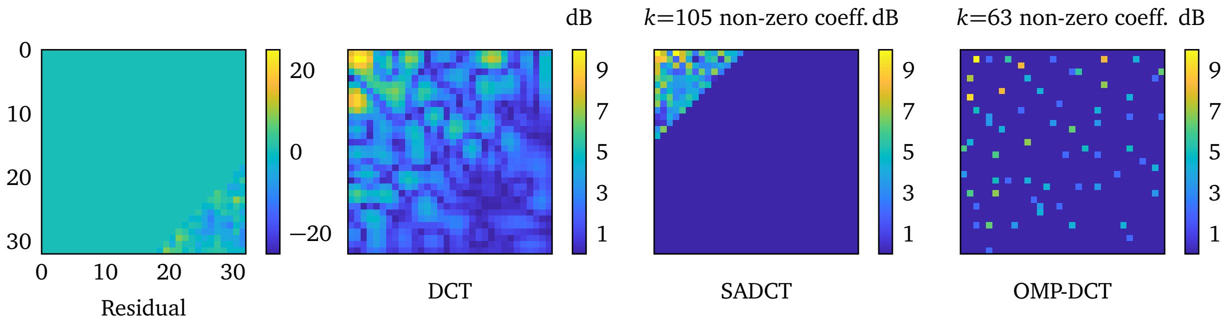


Figure 7.14 Visualization of resulting transform coefficients for the residual segment on the left for different transform schemes using DCT, SADCT and OMP-DCT.

7.5 Conclusions

The coding of the prediction error for geometrically partitioned segments remains an interesting topic that should be further investigated in the future. Although the analysis of the spatial distribution of the prediction error suggests that local, segment-dependent differences exist, specialized transform coding approaches that target these cases may not result in any significant coding gain over the plain DCT-II based block transforms. SADCT and transform-skipping both result in marginal overall improvements of about $-0.05\,\%$ in terms of luma BD-rate change. Multiple reasons could explain this behavior, such as the non-optimized entropy coding of transform coefficients for cases different from the separable DCT-II and MTS scheme. Since GEO already requires more side-information bits to signal the partitioning, these bits are “missing” for coding the residual. If a strong residual exists in a GEO or TPM segment, it is likely that the encoder simply chooses to segment the local area into smaller blocks using the MTT. Nevertheless, future research may result in competitive and practical solutions to these problems.

8 Summary and Outlook

Continued improvements in motion compensated prediction have been one of the determining factors for better coding efficiency in hybrid video coding schemes. In particular, the ability of combining highly flexible block partitioning structures with non-rectangular block partitioning as realized by geometric block partitioning (GEO) has been demonstrated in this thesis to improve the coding efficiency of state-of-the-art hybrid video coding. Overall, the coding efficiency in a random-access configuration can be improved, measured by an average luma BD-rate change of about −0.5 % for the JVET CTC test set over VVC, the most recent video coding standard in development. Higher gains are observed for sequences containing rigid objects in motion, such as people, cars, or animals. For these type of sequences, geometric block partitioning can better approximate object boundaries and model the signal characteristics more precisely. This ability is particularly useful at low bitrates, where coding artifacts impair the perceived visual quality. Although the relative coding efficiency improvements provided by GEO are lower compared to previous approaches reviewed in Chapter 3, the additional gains require negligible additional decoding and manageable additional encoding complexity for the reference encoder. The main contribution of this thesis is a low-complexity geometric partitioning coding tool suitable for practical video coding applications. Consequently, the technology presented in this thesis was accepted to be included into VVC. Since the implementation of geometric partitioning in the VVC Test Model is closely aligned with the discussed concepts, this thesis may serve as an in-depth guideline beyond the usual algorithm or encoder description provided.

Important general design aspects regarding the representation, quantization, entropy coding, and prediction of geometric partitioning parameters were analyzed and discussed in Chapter 4. First, two different parametrizations using block boundary intercept points and angle-distance pairs were being compared, with the conclusion that angle-distance pairs offer an overall easier controllable quantization and higher coding efficiency. An angle-dependent quantization resulting in about 140 geometric splits per block size was determined to be optimal, although it is recognized that other optimization minima could be found. The observed, non-equiprobable distribution of quantized angle and distance values can be exploited by common entropy coding methods, such as Huffman coding in combination with binary arithmetic coding. Simpler coding techniques, such as truncated-binary or fixed-length coding, provide slightly lower coding efficiency but require less decoding logic. It was also shown that the geometric splits determined for the coding of moving objects across multiple pictures display a correlation that can be exploited by temporal prediction. This further improves the coding efficiency by about −0.1 % on top of the previous results for specific sequences. Although smooth and slowly changing object boundaries would suggest that also spatial correlation exists across neighboring GEO blocks, no additional coding gain could be realized with simple methods of prediction and coding, as considered in this thesis.

Chapter 5 detailed inter-prediction modifications required by GEO and also investigated methods of motion vector prediction and coding that are offered by VVC and can be utilized by GEO. Blending – the per-sample weighting of pixels according to their distance to the partitioning line – is the essential process that distinguishes GEO from other inter-prediction coding tools. The amount of blending that is applied to smoothly combine two inter-predicted segments has an impact on coding efficiency and is recognized to be resolution and content dependent. It was shown in this thesis that a linear averaging with a transition zone of approximately 5 samples centered at the partitioning boundary provides a good compromise for UHD as well as HD and sub-HD sequences.

By adapting to the content characteristics and disabling the blending, GEO was also developed into a coding tool suitable for the compression of screen content. Significant coding efficiency improvements of −2.15 % in terms of luma BD-rate change can be reported for sequences containing text with graphics and motion.

Chapter 6 detailed simplifications and algorithmic improvements applied to the GEO coding tool that were developed in the context of the JVET standardization activity. These developments enabled the inclusion of GEO into VVC. Although the increased memory bandwidth requirement for GEO when performing bi-prediction led to a restriction of the motion compensation process to uni-prediction and an overall drop in coding efficiency improvements, coding gain can still be retained for sequences showing the characteristics targeted by GEO. While previous attempts of including extensive geometric partitioning into video coding standards were not successful due to high complexity, an improved encoder search strategy to determine the rate-distortion optimal partitioning was developed that causes an increase of the encoding run time to only 106 % and 108 % for random access and low delay coding scenarios.

In Chapter 7, methods of transform coding for the inter-prediction error signal resulting from geometric partitioning were briefly explored. Shape-adaptive DCT coding, transform-skipping, and encoder-side optimizations were evaluated, but provided little additional coding efficiency improvement. Future investigations into these topics may build upon these findings.

Most experimental results in this thesis were obtained by conducting encoding and decoding simulations according to the CTC by JVET. These specify 26 video sequences of different resolution, frame rate and content-type for the evaluation of proposed coding methods and also standardized methods of computing the relative coding efficiency against a reference. All methods were implemented in the VVC Test Model reference software, versions 3.2 and 5.0. Furthermore, perceived subjective quality was measured using VMAF. These measurements and visual inspection of selected, coded still pictures indicate that geometric block partitioning could also be visually beneficial, leading to fewer blocking artifacts and sharper object boundaries at low data rates.

The development of GEO as a coding tool for VVC is certainly not finished. Further optimizations regarding the quantization, prediction and entropy coding are to be expected in the future. The combination of GEO with other coding tools, such as affine motion models or intra-prediction, offers additional research opportunities. At last, the extension of GEO to a parametric block partitioning approach with even higher flexibility, e.g. using polygons, Bézier curves, or B-splines, could build a bridge between coding efficiency-oriented block tools and object-based coding.

A Appendix

List of experiments

- Experiment 4.1: Basic GEO performance with fixed quantization, Table A.1
- Experiment 4.2: Angle-dependent distance quantization, Table A.3
- Experiment 4.3: Intercept representation (80 partitions), Table A.5
- Experiment 4.4: Intercept representation (266-352 partitions), Table A.6
- Experiment 4.5: Intercept representation (266-1472 partitions), Table A.7
- Experiment 4.6: Huffman coding of GEO partition, Table A.8
- Experiment 4.7: Truncated-binary coding of GEO partition, Table A.9
- Experiment 4.8: Separate coding of angle-distance parameters, Table A.10
- Experiment 4.9: AD, spatial prediction, full refinement coding, Table A.11
- Experiment 4.10: AD, spatial prediction, limited refinement coding, Table A.13
- Experiment 4.11: AD, spatial prediction, no refinement coding, Table A.14
- Experiment 4.12: BI, spatial prediction, full refinement coding, Table A.15
- Experiment 4.13: BI, spatial prediction, limited refinement coding, Table A.17
- Experiment 4.14: BI, spatial prediction, no refinement coding, Table A.18
- Experiment 4.15: AD, temporal prediction, full refinement coding, Table A.19
- Experiment 4.16: AD, temporal prediction, limited refinement coding, Table A.21
- Experiment 4.17: AD, temporal prediction, no refinement coding, Table A.22
- Experiment 4.18: BI, temporal prediction, full refinement coding, Table A.23
- Experiment 4.19: BI, temporal prediction, limited refinement coding, Table A.25
- Experiment 4.20: BI, temporal prediction, no refinement coding, Table A.26
- Experiment 5.21: Blending filter length $d_{\mathrm{m}} = 1$, Table A.27
- Experiment 5.22: Blending filter length $d_{\mathrm{m}} = 3$, Table A.28
- Experiment 5.23: Blending filter length $d_{\mathrm{m}} = 5$, Table A.29

- Experiment 5.24: Blending filter length $d_\mathrm{m} = 7$, Table A.30
- Experiment 5.25: Restriction of GEO to uni-directional prediction, Table A.33
- Experiment 5.26: Combination of GEO with MMVD, Table A.32
- Experiment 6.27: VTM-5.0 simplifications, RA: Table A.34, LDB: Table A.35
- Experiment 7.28: SADCT coding of residual segments, Table A.36
- Experiment 7.28: SADCT coding of transition zone residual, Table A.37
- Experiment 7.30: Transform skip for GEO TB, Table A.38
- Experiment 7.31: Transform skip for GEO residual segments, Table A.39
- Experiment 7.32: Transform skip for GEO residual segments + shifting, Table A.40
- Experiment 7.33: Symmetric extension for TPM residual segments, all blocks, Table A.41
- Experiment 7.34: Symmetric extension for TPM residual segments, blocks $\{w, h\} \leq 16$, Table A.42

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.36 %	−1.04 %	−0.76 %	109.50 %	124.76 %	−0.22 %
FoodMarket4	−0.32 %	−0.42 %	−0.28 %	107.27 %	150.25 %	−0.24 %
Campfire	−0.08 %	−0.11 %	−0.35 %	113.84 %	151.52 %	−0.20 %
CatRobot1	−0.32 %	−0.68 %	−0.84 %	106.52 %	159.42 %	−0.25 %
DaylightRoad2	−0.11 %	−0.57 %	−0.30 %	106.11 %	158.22 %	−0.06 %
ParkRunning3	−0.27 %	−0.28 %	−0.24 %	110.87 %	146.68 %	−0.21 %
MarketPlace	−0.23 %	−0.05 %	−0.62 %	117.48 %	146.41 %	−0.02 %
RitualDance	−0.15 %	−0.42 %	−0.75 %	123.15 %	135.71 %	−0.17 %
Cactus	−0.39 %	−0.64 %	−0.53 %	125.32 %	164.82 %	−0.23 %
BasketballDrive	−0.22 %	−0.67 %	−0.53 %	118.05 %	152.29 %	−0.25 %
BQTerrace	−0.41 %	−0.64 %	−0.28 %	111.11 %	164.69 %	−0.20 %
BasketballDrill	−0.52 %	−0.97 %	−0.82 %	116.64 %	159.71 %	−0.36 %
BQMall	−0.92 %	−1.08 %	−1.17 %	121.70 %	165.21 %	−0.50 %
PartyScene	−0.29 %	−0.48 %	−0.57 %	126.30 %	127.99 %	−0.11 %
RaceHorsesL	−0.76 %	−1.41 %	−1.05 %	118.45 %	114.04 %	−1.07 %
BasketballPass	−0.35 %	−1.19 %	−0.78 %	118.78 %	193.70 %	0.11 %
BQSquare	−0.08 %	0.16 %	0.32 %	126.88 %	203.06 %	−0.26 %
BlowingBubbles	−0.23 %	−0.27 %	−0.27 %	128.55 %	181.58 %	0.04 %
RaceHorsesM	−0.62 %	−1.23 %	−1.36 %	113.68 %	162.64 %	−0.42 %
BasketballDrillText	−0.53 %	−0.68 %	−0.76 %	115.61 %	160.35 %	−0.15 %
ArenaOfValor	−0.52 %	−0.85 %	−0.48 %	119.65 %	143.10 %	−0.56 %
SlideEditing	0.00 %	0.00 %	−0.02 %	110.74 %	78.59 %	−0.05 %
SlideShow	−0.25 %	−0.44 %	−0.26 %	117.34 %	94.01 %	−0.45 %
Mean	**−0.34** %	**−0.61** %	**−0.55** %	**116.68** %	**149.51** %	**−0.25** %
Class A1	−0.25 %	−0.52 %	−0.46 %	109.55 %	141.11 %	−0.22 %
Class A2	−0.23 %	−0.51 %	−0.46 %	107.25 %	154.05 %	−0.18 %
Class B	−0.28 %	−0.48 %	−0.54 %	118.45 %	151.92 %	−0.17 %
Class C	−0.62 %	−0.98 %	−0.90 %	120.45 %	139.52 %	−0.51 %
Class D	−0.32 %	−0.63 %	−0.53 %	120.58 %	184.07 %	−0.13 %
Class F	−0.32 %	−0.49 %	−0.38 %	115.58 %	113.77 %	−0.30 %
JVET Overall	**−0.36** %	**−0.63** %	**−0.61** %	**114.83** %	**146.74** %	**−0.27** %

Table A.1 Coding results for Experiment 4.1 with angle-distance representation and no prediction used. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $\Delta\rho = \max(w, h)/8$. Coding of GEO partitions using $C_{\mathrm{FL},\{7,8\}}$ code.

		GEO usage for QP				
Class	Sequence	QP22	QP27	QP32	QP37	Mean
A1	Tango2	6.37 %	4.57 %	4.38 %	4.22 %	4.88 %
	FoodMarket4	4.93 %	3.49 %	3.07 %	2.73 %	3.56 %
	Campfire	4.72 %	2.33 %	1.84 %	1.59 %	2.62 %
A2	CatRobot1	5.25 %	2.79 %	2.37 %	2.11 %	3.13 %
	DaylightRoad2	3.58 %	2.13 %	1.84 %	1.88 %	2.36 %
	ParkRunning3	5.51 %	4.30 %	3.95 %	4.14 %	4.48 %
B	MarketPlace	5.67 %	3.40 %	2.96 %	2.80 %	3.71 %
	RitualDance	3.74 %	3.14 %	3.03 %	3.11 %	3.25 %
	Cactus	9.60 %	4.64 %	3.75 %	3.41 %	5.35 %
	BasketballDrive	5.31 %	3.86 %	3.51 %	3.17 %	3.96 %
	BQTerrace	15.97 %	9.58 %	5.47 %	3.07 %	8.52 %
C	BasketballDrill	9.25 %	8.93 %	8.43 %	8.30 %	8.73 %
	BQMall	5.21 %	5.12 %	5.64 %	6.46 %	5.61 %
	PartyScene	3.75 %	3.51 %	3.32 %	3.43 %	3.51 %
	RaceHorsesL	7.36 %	7.79 %	7.80 %	7.60 %	7.64 %
D	BasketballPass	4.88 %	5.58 %	5.85 %	6.25 %	5.64 %
	BQSquare	2.94 %	1.66 %	0.84 %	0.49 %	1.48 %
	BlowingBubbles	4.00 %	3.83 %	3.81 %	3.50 %	3.79 %
	RaceHorsesM	7.24 %	7.61 %	7.97 %	7.16 %	7.50 %
F	BasketballDrillText	9.25 %	8.93 %	8.43 %	8.30 %	8.73 %
	ArenaOfValor	6.88 %	6.27 %	5.73 %	5.19 %	6.02 %
	SlideEditing	1.05 %	0.99 %	0.90 %	1.00 %	0.99 %
	SlideShow	1.16 %	1.22 %	1.29 %	1.62 %	1.32 %
	Overall	**5.81** %	**4.59** %	**4.18** %	**3.98** %	**4.64** %

Table A.2 Usage of GEO in percentage of area coded for Experiment 4.1

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.46 %	−0.98 %	−0.82 %	141.79 %	134.74 %	−0.33 %
FoodMarket4	−0.32 %	−0.35 %	−0.44 %	126.67 %	131.71 %	−0.17 %
Campfire	−0.13 %	−0.14 %	−0.45 %	125.25 %	132.16 %	−0.23 %
CatRobot1	−0.47 %	−0.89 %	−0.95 %	126.09 %	138.79 %	−0.31 %
DaylightRoad2	−0.30 %	−0.59 %	−0.44 %	115.04 %	139.14 %	−0.29 %
ParkRunning3	−0.30 %	−0.28 %	−0.25 %	139.98 %	131.90 %	−0.22 %
MarketPlace	−0.23 %	−0.15 %	−0.54 %	116.27 %	125.08 %	−0.13 %
RitualDance	−0.26 %	−0.39 %	−0.68 %	121.12 %	127.21 %	−0.25 %
Cactus	−0.46 %	−0.65 %	−0.65 %	131.94 %	145.49 %	−0.43 %
BasketballDrive	−0.27 %	−0.64 %	−0.67 %	119.32 %	135.99 %	−0.36 %
BQTerrace	−0.58 %	−0.70 %	−0.36 %	112.17 %	143.31 %	−0.46 %
BasketballDrill	−0.60 %	−0.97 %	−0.89 %	118.74 %	135.99 %	−0.39 %
BQMall	−1.13 %	−1.45 %	−1.28 %	113.92 %	142.15 %	−0.70 %
PartyScene	−0.30 %	−0.50 %	−0.62 %	133.12 %	97.88 %	−0.26 %
RaceHorsesL	−0.89 %	−1.48 %	−1.04 %	119.54 %	95.21 %	−1.02 %
BasketballPass	−0.36 %	−1.07 %	−0.72 %	118.19 %	154.48 %	0.10 %
BQSquare	0.01 %	−0.12 %	0.21 %	127.33 %	160.88 %	0.27 %
BlowingBubbles	−0.38 %	−0.36 %	−0.34 %	138.99 %	154.26 %	−0.02 %
RaceHorsesM	−0.73 %	−1.54 %	−1.33 %	124.28 %	129.56 %	−0.39 %
BasketballDrillText	−0.71 %	−0.87 %	−0.87 %	116.03 %	137.47 %	−0.38 %
ArenaOfValor	−0.68 %	−1.04 %	−0.61 %	121.64 %	138.66 %	−0.53 %
SlideEditing	−0.01 %	−0.03 %	−0.01 %	115.15 %	84.07 %	−0.07 %
SlideShow	−0.25 %	−0.61 %	−0.41 %	119.17 %	101.80 %	−0.32 %
Mean	**−0.43** %	**−0.69** %	**−0.62** %	**123.55** %	**131.21** %	**−0.30** %
Class A1	−0.30 %	−0.49 %	−0.57 %	130.40 %	132.83 %	−0.24 %
Class A2	−0.36 %	−0.59 %	−0.55 %	126.10 %	136.53 %	−0.28 %
Class B	−0.36 %	−0.51 %	−0.58 %	119.67 %	135.05 %	−0.33 %
Class C	−0.73 %	−1.10 %	−0.96 %	120.82 %	115.65 %	−0.59 %
Class D	−0.36 %	−0.77 %	−0.54 %	126.17 %	148.89 %	−0.01 %
Class F	−0.41 %	−0.64 %	−0.48 %	117.81 %	112.89 %	−0.33 %
JVET Overall	**−0.45** %	**−0.68** %	**−0.67** %	**123.34** %	**129.43** %	**−0.37** %

Table A.3 Coding results for Experiment 4.2 with angle-distance representation and no prediction used. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$. Coding of GEO partitions using $C_{\mathrm{FL},\{7,8\}}$ code.

		GEO usage for QP				
Class	Sequence	QP22	QP27	QP32	QP37	Mean
A1	Tango2	7.52 %	5.05 %	4.78 %	4.71 %	5.51 %
	FoodMarket4	5.55 %	3.74 %	3.36 %	2.99 %	3.91 %
	Campfire	5.36 %	2.71 %	2.14 %	1.86 %	3.02 %
A2	CatRobot1	6.14 %	3.29 %	2.74 %	2.55 %	3.68 %
	DaylightRoad2	4.40 %	2.55 %	2.26 %	2.27 %	2.87 %
	ParkRunning3	5.88 %	4.68 %	4.30 %	4.61 %	4.87 %
B	MarketPlace	6.31 %	3.68 %	3.26 %	3.08 %	4.08 %
	RitualDance	4.14 %	3.50 %	3.35 %	3.48 %	3.62 %
	Cactus	10.88 %	5.12 %	3.98 %	3.68 %	5.91 %
	BasketballDrive	6.27 %	4.40 %	4.00 %	3.60 %	4.57 %
	BQTerrace	18.04 %	10.61 %	6.20 %	3.56 %	9.60 %
C	BasketballDrill	9.86 %	9.36 %	9.08 %	8.83 %	9.28 %
	BQMall	6.04 %	5.67 %	5.91 %	6.66 %	6.07 %
	PartyScene	4.10 %	3.71 %	3.54 %	3.62 %	3.74 %
	RaceHorsesL	8.09 %	8.62 %	8.50 %	8.62 %	8.46 %
D	BasketballPass	5.32 %	5.65 %	6.16 %	6.68 %	5.95 %
	BQSquare	3.40 %	1.87 %	0.98 %	0.64 %	1.72 %
	BlowingBubbles	4.32 %	4.17 %	4.16 %	3.63 %	4.07 %
	RaceHorsesM	7.77 %	8.26 %	8.28 %	7.81 %	8.03 %
F	BasketballDrillText	9.86 %	9.36 %	9.08 %	8.83 %	9.28 %
	ArenaOfValor	7.74 %	7.05 %	6.49 %	5.87 %	6.79 %
	SlideEditing	1.30 %	1.02 %	0.97 %	1.04 %	1.08 %
	SlideShow	1.24 %	1.36 %	1.51 %	1.74 %	1.46 %
	Overall	**6.50** %	**5.02** %	**4.57** %	**4.36** %	**5.11** %

Table A.4 Usage of GEO in percentage of area coded for Experiment 4.2

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.37 %	−0.69 %	−0.98 %	246.90 %	167.88 %	−0.28 %
FoodMarket4	−0.31 %	−0.48 %	−0.49 %	212.85 %	223.00 %	−0.21 %
Campfire	−0.09 %	−0.14 %	−0.33 %	210.40 %	333.86 %	−0.18 %
CatRobot1	−0.49 %	−0.96 %	−0.96 %	234.36 %	281.27 %	−0.28 %
DaylightRoad2	−0.23 %	−0.69 %	−0.55 %	217.91 %	249.84 %	−0.14 %
ParkRunning3	−0.25 %	−0.30 %	−0.27 %	217.74 %	161.09 %	−0.18 %
MarketPlace	−0.17 %	−0.09 %	−0.50 %	193.18 %	200.25 %	−0.02 %
RitualDance	−0.21 %	−0.37 %	−0.69 %	212.09 %	283.69 %	−0.14 %
Cactus	−0.38 %	−0.40 %	−0.63 %	195.15 %	385.31 %	−0.29 %
BasketballDrive	−0.23 %	−0.59 %	−0.61 %	171.79 %	252.52 %	−0.33 %
BQTerrace	−0.45 %	−0.69 %	−0.41 %	180.23 %	296.94 %	−0.17 %
BasketballDrill	−0.48 %	−1.06 %	−0.99 %	186.21 %	431.90 %	−0.20 %
BQMall	−1.00 %	−1.17 %	−1.22 %	160.97 %	257.71 %	−0.66 %
PartyScene	−0.25 %	−0.55 %	−0.41 %	154.78 %	169.89 %	−0.14 %
RaceHorsesL	−0.78 %	−1.38 %	−1.36 %	158.38 %	168.36 %	−0.85 %
BasketballPass	−0.34 %	−0.83 %	−0.86 %	162.22 %	316.50 %	−0.03 %
BQSquare	0.12 %	−0.03 %	0.33 %	169.00 %	274.20 %	−0.23 %
BlowingBubbles	−0.27 %	−0.34 %	−0.13 %	157.43 %	315.13 %	0.01 %
RaceHorsesM	−0.67 %	−1.20 %	−1.23 %	138.50 %	197.89 %	−0.38 %
BasketballDrillText	−0.52 %	−0.72 %	−0.75 %	154.66 %	437.97 %	−0.14 %
ArenaOfValor	−0.53 %	−0.87 %	−0.69 %	209.66 %	469.27 %	−0.44 %
SlideEditing	−0.06 %	−0.03 %	−0.01 %	358.21 %	535.40 %	−0.05 %
SlideShow	−0.22 %	−0.39 %	−0.21 %	236.25 %	564.41 %	−0.25 %
Mean	**−0.36** %	**−0.61** %	**−0.61** %	**197.34** %	**303.23** %	**−0.24** %
Class A1	−0.26 %	−0.44 %	−0.60 %	219.47 %	228.20 %	−0.22 %
Class A2	−0.33 %	−0.65 %	−0.60 %	220.70 %	212.95 %	−0.20 %
Class B	−0.29 %	−0.43 %	−0.57 %	187.96 %	274.22 %	−0.19 %
Class C	−0.63 %	−1.04 %	−1.00 %	162.98 %	233.01 %	−0.46 %
Class D	−0.29 %	−0.60 %	−0.47 %	154.01 %	267.51 %	−0.16 %
Class F	−0.33 %	−0.50 %	−0.41 %	225.90 %	487.34 %	−0.22 %
JVET Overall	**−0.38** %	**−0.64** %	**−0.70** %	**192.73** %	**240.61** %	**−0.27** %

Table A.5 Coding results for Experiment 4.3 with intercept representation and no prediction used. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/4$, $\Delta h = h/4$, $C_{\mathrm{FL},7}$ code (80 partitions).

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.46 %	−0.78 %	−0.87 %	313.17 %	196.64 %	−0.41 %
FoodMarket4	−0.35 %	−0.49 %	−0.62 %	253.11 %	253.05 %	−0.27 %
Campfire	−0.10 %	−0.11 %	−0.36 %	260.51 %	288.00 %	−0.19 %
CatRobot1	−0.56 %	−1.16 %	−1.07 %	227.03 %	276.68 %	−0.39 %
DaylightRoad2	−0.28 %	−0.51 %	−0.37 %	253.86 %	234.99 %	−0.10 %
ParkRunning3	−0.25 %	−0.25 %	−0.20 %	249.34 %	223.02 %	−0.21 %
MarketPlace	−0.14 %	−0.23 %	−0.71 %	212.21 %	207.43 %	0.05 %
RitualDance	−0.20 %	−0.31 %	−0.64 %	262.81 %	255.21 %	−0.08 %
Cactus	−0.38 %	−0.34 %	−0.49 %	217.98 %	509.53 %	−0.18 %
BasketballDrive	−0.24 %	−0.59 %	−0.55 %	204.79 %	249.42 %	−0.38 %
BQTerrace	−0.43 %	−0.74 %	−0.14 %	215.32 %	304.41 %	−0.42 %
BasketballDrill	−0.52 %	−0.91 %	−0.84 %	225.82 %	499.48 %	−0.30 %
BQMall	−1.16 %	−1.56 %	−1.12 %	209.65 %	339.84 %	−0.78 %
PartyScene	−0.25 %	−0.57 %	−0.35 %	210.34 %	176.55 %	−0.19 %
RaceHorsesL	−0.92 %	−1.35 %	−1.17 %	214.66 %	195.95 %	−1.04 %
BasketballPass	−0.36 %	−1.17 %	−0.59 %	198.06 %	382.48 %	−0.11 %
BQSquare	−0.05 %	−0.24 %	0.14 %	205.86 %	270.38 %	−0.05 %
BlowingBubbles	−0.25 %	−0.48 %	−0.57 %	199.88 %	315.43 %	−0.04 %
RaceHorsesM	−0.65 %	−1.38 %	−1.34 %	192.71 %	240.15 %	−0.20 %
BasketballDrillText	−0.55 %	−0.74 %	−0.90 %	229.77 %	415.41 %	−0.33 %
ArenaOfValor	−0.50 %	−0.75 %	−0.42 %	246.13 %	470.20 %	−0.42 %
SlideEditing	−0.03 %	−0.07 %	−0.05 %	386.90 %	466.47 %	0.03 %
SlideShow	−0.29 %	−0.48 %	−0.46 %	238.61 %	451.92 %	−0.51 %
Mean	**−0.39** %	**−0.66** %	**−0.60** %	**236.02** %	**314.03** %	**−0.28** %
Class A1	−0.31 %	−0.46 %	−0.62 %	271.33 %	235.80 %	−0.29 %
Class A2	−0.36 %	−0.64 %	−0.54 %	241.63 %	237.28 %	−0.24 %
Class B	−0.28 %	−0.44 %	−0.51 %	220.43 %	286.69 %	−0.20 %
Class C	−0.71 %	−1.10 %	−0.87 %	212.56 %	275.03 %	−0.57 %
Class D	−0.32 %	−0.82 %	−0.59 %	196.02 %	292.53 %	−0.10 %
Class F	−0.34 %	−0.51 %	−0.46 %	267.63 %	443.76 %	−0.31 %
JVET Overall	**−0.42** %	**−0.66** %	**−0.63** %	**231.78** %	**262.54** %	**−0.33** %

Table A.6 Coding results for Experiment 4.4 with intercept representation and no prediction used. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/8$, $\Delta h = h/8$, $C_{\mathrm{FL},9}$ code (266-352 partitions).

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.36 %	−0.65 %	−0.62 %	555.82 %	215.76 %	−0.20 %
FoodMarket4	−0.32 %	−0.36 %	−0.67 %	474.04 %	207.17 %	−0.29 %
Campfire	−0.06 %	−0.12 %	−0.35 %	431.78 %	307.26 %	−0.19 %
CatRobot1	−0.52 %	−1.19 %	−1.01 %	446.09 %	263.10 %	−0.31 %
DaylightRoad2	−0.21 %	−0.53 %	−0.27 %	423.82 %	236.78 %	−0.09 %
ParkRunning3	−0.21 %	−0.19 %	−0.18 %	449.39 %	228.17 %	−0.18 %
MarketPlace	−0.14 %	0.03 %	−0.30 %	400.72 %	212.15 %	0.01 %
RitualDance	−0.21 %	−0.50 %	−0.63 %	489.61 %	284.86 %	−0.13 %
Cactus	−0.33 %	−0.47 %	−0.57 %	414.39 %	517.57 %	−0.24 %
BasketballDrive	−0.18 %	−0.66 %	−0.45 %	326.69 %	221.66 %	−0.13 %
BQTerrace	−0.43 %	−0.58 %	−0.40 %	404.76 %	294.79 %	−0.13 %
BasketballDrill	−0.38 %	−0.65 %	−0.81 %	362.94 %	497.92 %	−0.12 %
BQMall	−1.06 %	−1.44 %	−1.15 %	359.27 %	274.06 %	−0.71 %
PartyScene	−0.24 %	−0.41 %	−0.39 %	340.14 %	116.19 %	−0.09 %
RaceHorsesL	−0.82 %	−1.41 %	−1.17 %	328.30 %	125.79 %	−0.88 %
BasketballPass	−0.26 %	−1.09 %	−0.66 %	301.03 %	440.55 %	−0.00 %
BQSquare	0.11 %	0.16 %	0.44 %	396.06 %	476.80 %	−0.29 %
BlowingBubbles	−0.23 %	−0.32 %	−0.14 %	354.82 %	465.66 %	0.17 %
RaceHorsesM	−0.56 %	−1.05 %	−1.11 %	295.16 %	284.45 %	0.10 %
BasketballDrillText	−0.46 %	−0.51 %	−0.64 %	332.28 %	399.24 %	−0.24 %
ArenaOfValor	−0.40 %	−0.68 %	−0.27 %	392.04 %	433.87 %	−0.42 %
SlideEditing	−0.03 %	−0.06 %	−0.03 %	548.47 %	562.29 %	−0.07 %
SlideShow	−0.28 %	−0.49 %	−0.27 %	390.92 %	463.71 %	−0.31 %
Mean	**−0.33** %	**−0.57** %	**−0.51** %	**400.81** %	**327.38** %	**−0.21** %
Class A1	−0.25 %	−0.37 %	−0.55 %	472.07 %	222.43 %	−0.23 %
Class A2	−0.31 %	−0.64 %	−0.49 %	432.66 %	226.93 %	−0.20 %
Class B	−0.26 %	−0.44 %	−0.47 %	398.03 %	286.34 %	−0.12 %
Class C	−0.63 %	−0.98 %	−0.88 %	339.43 %	205.71 %	−0.45 %
Class D	−0.24 %	−0.57 %	−0.37 %	324.69 %	401.49 %	−0.01 %
Class F	−0.29 %	−0.44 %	−0.30 %	405.42 %	451.97 %	−0.26 %
JVET Overall	**−0.37** %	**−0.61** %	**−0.60** %	**401.36** %	**237.93** %	**−0.25** %

Table A.7 Coding results for Experiment 4.5 with intercept representation and no prediction used. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/16$, $\Delta h = h/16$, $C_{\mathrm{FL},\{9\dots11\}}$ code (266-1472 partitions).

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.48 %	−1.01 %	−0.81 %	121.00 %	108.77 %	−0.42 %
FoodMarket4	−0.35 %	−0.47 %	−0.51 %	119.60 %	109.48 %	−0.15 %
Campfire	−0.15 %	−0.15 %	−0.42 %	128.28 %	97.82 %	−0.24 %
CatRobot1	−0.55 %	−0.79 %	−0.91 %	111.85 %	89.09 %	−0.41 %
DaylightRoad2	−0.32 %	−0.78 %	−0.40 %	113.24 %	107.36 %	−0.18 %
ParkRunning3	−0.34 %	−0.30 %	−0.32 %	123.78 %	86.11 %	−0.27 %
MarketPlace	−0.24 %	0.04 %	−0.65 %	127.09 %	108.50 %	−0.07 %
RitualDance	−0.27 %	−0.54 %	−0.70 %	138.44 %	107.60 %	−0.26 %
Cactus	−0.51 %	−0.52 %	−0.59 %	132.68 %	87.97 %	−0.37 %
BasketballDrive	−0.31 %	−0.61 %	−0.72 %	123.35 %	120.49 %	−0.41 %
BQTerrace	−0.63 %	−0.67 %	−0.47 %	137.38 %	110.18 %	0.11 %
BasketballDrill	−0.71 %	−1.05 %	−0.94 %	153.45 %	96.75 %	−0.48 %
BQMall	−1.27 %	−1.50 %	−1.33 %	134.17 %	111.95 %	−0.84 %
PartyScene	−0.36 %	−0.56 %	−0.53 %	143.39 %	132.82 %	−0.26 %
RaceHorsesL	−0.96 %	−1.42 %	−1.31 %	131.21 %	110.50 %	−1.27 %
BasketballPass	−0.36 %	−1.23 %	−1.01 %	127.34 %	148.02 %	−0.04 %
BQSquare	−0.02 %	−0.07 %	0.26 %	147.86 %	158.88 %	0.10 %
BlowingBubbles	−0.41 %	−0.20 %	−0.16 %	135.25 %	146.03 %	−0.43 %
RaceHorsesM	−0.77 %	−1.43 %	−1.27 %	123.50 %	135.03 %	−0.07 %
BasketballDrillText	−0.64 %	−0.85 %	−1.01 %	136.97 %	111.22 %	−0.36 %
ArenaOfValor	−0.73 %	−1.06 %	−0.64 %	121.57 %	106.72 %	−0.61 %
SlideEditing	−0.06 %	−0.06 %	−0.04 %	115.52 %	113.71 %	−0.15 %
SlideShow	−0.35 %	−0.75 %	−0.55 %	115.32 %	119.80 %	−0.35 %
Mean	**−0.47** %	**−0.69** %	**−0.65** %	**128.79** %	**114.12** %	**−0.32** %
Class A1	−0.33 %	−0.54 %	−0.58 %	121.93 %	102.02 %	−0.27 %
Class A2	−0.40 %	−0.62 %	−0.54 %	115.78 %	91.26 %	−0.29 %
Class B	−0.39 %	−0.46 %	−0.63 %	131.49 %	104.77 %	−0.20 %
Class C	−0.83 %	−1.13 %	−1.03 %	139.93 %	110.35 %	−0.71 %
Class D	−0.39 %	−0.74 %	−0.55 %	132.07 %	143.41 %	−0.11 %
Class F	−0.44 %	−0.68 %	−0.56 %	121.80 %	111.40 %	−0.37 %
JVET Overall	**−0.50** %	**−0.69** %	**−0.71** %	**128.38** %	**102.79** %	**−0.37** %

Table A.8 Coding results for Experiment 4.6 with Huffman coding of the partitioning information. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, partitioning coded with C_H code (Huffman trees).

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.47 %	−1.19 %	−1.11 %	117.48 %	100.18 %	−0.30 %
FoodMarket4	−0.36 %	−0.35 %	−0.43 %	124.10 %	118.22 %	−0.26 %
Campfire	−0.11 %	−0.14 %	−0.44 %	116.53 %	93.34 %	−0.21 %
CatRobot1	−0.56 %	−0.91 %	−0.99 %	123.60 %	93.49 %	−0.45 %
DaylightRoad2	−0.33 %	−0.83 %	−0.42 %	120.52 %	107.83 %	−0.15 %
ParkRunning3	−0.34 %	−0.31 %	−0.28 %	109.81 %	91.10 %	−0.35 %
MarketPlace	−0.26 %	−0.25 %	−0.27 %	131.88 %	102.90 %	−0.24 %
RitualDance	−0.24 %	−0.44 %	−0.71 %	129.13 %	95.95 %	−0.22 %
Cactus	−0.49 %	−0.45 %	−0.81 %	130.43 %	102.06 %	−0.28 %
BasketballDrive	−0.28 %	−0.62 %	−0.65 %	122.86 %	101.14 %	−0.45 %
BQTerrace	−0.57 %	−0.52 %	−0.41 %	131.57 %	101.97 %	−0.29 %
BasketballDrill	−0.65 %	−1.07 %	−1.02 %	123.31 %	102.74 %	−0.48 %
BQMall	−1.22 %	−1.65 %	−1.35 %	125.06 %	108.97 %	−0.89 %
PartyScene	−0.35 %	−0.68 %	−0.57 %	133.96 %	89.96 %	−0.32 %
RaceHorsesL	−0.90 %	−1.64 %	−1.43 %	135.45 %	100.89 %	−0.82 %
BasketballPass	−0.43 %	−0.94 %	−0.97 %	125.67 %	113.15 %	0.10 %
BQSquare	−0.03 %	−0.08 %	0.17 %	128.58 %	115.35 %	−0.22 %
BlowingBubbles	−0.36 %	−0.40 %	−0.26 %	131.91 %	104.83 %	0.01 %
RaceHorsesM	−0.84 %	−1.41 %	−1.41 %	128.07 %	109.53 %	−0.70 %
BasketballDrillText	−0.69 %	−0.63 %	−0.78 %	122.09 %	105.49 %	−0.41 %
ArenaOfValor	−0.67 %	−0.93 %	−0.69 %	115.74 %	108.51 %	−0.66 %
SlideEditing	−0.03 %	−0.03 %	0.00 %	121.70 %	107.18 %	−0.10 %
SlideShow	−0.33 %	−0.65 %	−0.45 %	125.42 %	103.02 %	−0.55 %
Mean	**−0.46** %	**−0.70** %	**−0.66** %	**124.99** %	**103.38** %	**−0.36** %
Class A1	−0.31 %	−0.56 %	−0.66 %	118.84 %	101.93 %	−0.26 %
Class A2	−0.41 %	−0.68 %	−0.56 %	117.73 %	96.62 %	−0.32 %
Class B	−0.37 %	−0.46 %	−0.57 %	128.90 %	100.10 %	−0.30 %
Class C	−0.78 %	−1.26 %	−1.09 %	129.24 %	99.82 %	−0.63 %
Class D	−0.41 %	−0.71 %	−0.62 %	128.43 %	109.95 %	−0.20 %
Class F	−0.43 %	−0.56 %	−0.48 %	121.09 %	105.74 %	−0.43 %
JVET Overall	**−0.47** %	**−0.74** %	**−0.73** %	**124.63** %	**99.68** %	**−0.38** %

Table A.9 Coding results for Experiment 4.7 with truncated binary coding of the partitioning information. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, partitioning coded with C_{TB} code.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.46 %	−0.93 %	−0.95 %	124.62 %	96.95 %	−0.29 %
FoodMarket4	−0.33 %	−0.43 %	−0.44 %	124.88 %	115.91 %	−0.23 %
Campfire	−0.11 %	−0.17 %	−0.41 %	123.51 %	103.15 %	−0.18 %
CatRobot1	−0.54 %	−0.97 %	−1.03 %	129.30 %	99.47 %	−0.42 %
DaylightRoad2	−0.32 %	−0.69 %	−0.37 %	119.46 %	107.50 %	−0.13 %
ParkRunning3	−0.33 %	−0.30 %	−0.28 %	116.36 %	97.13 %	−0.27 %
MarketPlace	−0.25 %	−0.17 %	−0.72 %	133.59 %	101.58 %	−0.02 %
RitualDance	−0.25 %	−0.48 %	−0.77 %	128.17 %	96.75 %	−0.23 %
Cactus	−0.49 %	−0.65 %	−0.74 %	134.51 %	100.86 %	−0.53 %
BasketballDrive	−0.28 %	−0.69 %	−0.66 %	130.78 %	89.06 %	−0.31 %
BQTerrace	−0.58 %	−0.94 %	−0.56 %	130.20 %	101.99 %	−0.51 %
BasketballDrill	−0.62 %	−1.05 %	−1.13 %	125.38 %	101.78 %	−0.41 %
BQMall	−1.21 %	−1.41 %	−1.13 %	122.58 %	95.19 %	−0.84 %
PartyScene	−0.33 %	−0.58 %	−0.52 %	134.83 %	115.84 %	−0.09 %
RaceHorsesL	−0.84 %	−1.53 %	−1.27 %	133.69 %	105.24 %	−1.01 %
BasketballPass	−0.37 %	−1.14 %	−0.83 %	125.93 %	112.65 %	−0.15 %
BQSquare	−0.01 %	−0.15 %	0.41 %	128.01 %	111.34 %	0.00 %
BlowingBubbles	−0.39 %	−0.29 %	−0.31 %	128.02 %	117.64 %	−0.11 %
RaceHorsesM	−0.82 %	−1.31 %	−1.33 %	124.01 %	104.71 %	−0.32 %
BasketballDrillText	−0.66 %	−0.68 %	−0.80 %	126.61 %	110.21 %	−0.40 %
ArenaOfValor	−0.69 %	−1.09 %	−0.67 %	124.98 %	109.48 %	−0.59 %
SlideEditing	−0.08 %	−0.10 %	−0.04 %	123.08 %	91.82 %	0.01 %
SlideShow	−0.30 %	−0.53 %	−0.40 %	121.59 %	102.91 %	−0.48 %
Mean	**−0.45** %	**−0.71** %	**−0.65** %	**126.70** %	**103.88** %	**−0.33** %
Class A1	−0.30 %	−0.51 %	−0.60 %	123.96 %	104.08 %	−0.23 %
Class A2	−0.39 %	−0.65 %	−0.56 %	121.37 %	99.78 %	−0.27 %
Class B	−0.37 %	−0.59 %	−0.69 %	130.96 %	97.50 %	−0.32 %
Class C	−0.75 %	−1.14 %	−1.01 %	128.80 %	103.03 %	−0.59 %
Class D	−0.40 %	−0.72 %	−0.52 %	126.19 %	110.89 %	−0.14 %
Class F	−0.43 %	−0.60 %	−0.48 %	123.99 %	102.82 %	−0.37 %
JVET Overall	**−0.46** %	**−0.73** %	**−0.73** %	**127.01** %	**100.71** %	**−0.36** %

Table A.10 Coding results for Experiment 4.8 with separate parameter coding of the partitioning information. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, partitioning coded with $C_{\mathrm{FL},5}$ (angle), $C_{\mathrm{FL},1} + C_{\mathrm{TU},\mathrm{N}_\rho-1}$ (distance).

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.49 %	−0.89 %	−0.99 %	329.81 %	173.56 %	−0.36 %
FoodMarket4	−0.36 %	−0.45 %	−0.53 %	265.52 %	172.45 %	−0.27 %
Campfire	−0.19 %	−0.20 %	−0.52 %	251.76 %	173.52 %	−0.29 %
CatRobot1	−0.60 %	−1.17 %	−1.15 %	260.00 %	164.93 %	−0.42 %
DaylightRoad2	−0.34 %	−0.79 %	−0.45 %	284.22 %	152.13 %	−0.10 %
ParkRunning3	−0.37 %	−0.41 %	−0.37 %	233.46 %	129.09 %	−0.29 %
MarketPlace	−0.27 %	−0.09 %	−0.56 %	263.60 %	142.25 %	−0.16 %
RitualDance	−0.27 %	−0.56 %	−0.78 %	284.88 %	159.33 %	−0.26 %
Cactus	−0.49 %	−0.57 %	−0.71 %	298.68 %	257.19 %	−0.45 %
BasketballDrive	−0.28 %	−0.77 %	−0.75 %	261.81 %	123.67 %	−0.34 %
BQTerrace	−0.68 %	−0.83 %	−0.52 %	261.07 %	134.93 %	−0.31 %
BasketballDrill	−0.61 %	−1.27 %	−1.27 %	307.23 %	211.46 %	−0.40 %
BQMall	−1.12 %	−1.22 %	−1.50 %	271.15 %	161.69 %	−0.79 %
PartyScene	−0.34 %	−0.58 %	−0.60 %	259.31 %	124.43 %	−0.10 %
RaceHorsesL	−0.89 %	−1.64 %	−1.62 %	284.28 %	141.46 %	−1.18 %
BasketballPass	−0.34 %	−1.26 %	−0.83 %	253.16 %	182.73 %	−0.09 %
BQSquare	0.01 %	−0.11 %	0.08 %	278.10 %	125.29 %	0.02 %
BlowingBubbles	−0.34 %	−0.36 %	−0.29 %	266.75 %	177.18 %	−0.35 %
RaceHorsesM	−0.77 %	−1.20 %	−1.81 %	248.96 %	147.31 %	−0.43 %
BasketballDrillText	−0.65 %	−0.78 %	−1.09 %	309.06 %	256.09 %	−0.38 %
ArenaOfValor	−0.70 %	−1.01 %	−0.80 %	284.50 %	313.12 %	−0.58 %
SlideEditing	−0.09 %	−0.08 %	−0.05 %	504.13 %	597.46 %	−0.06 %
SlideShow	−0.28 %	−0.52 %	−0.45 %	346.49 %	475.59 %	−0.46 %
Mean	**−0.45** %	**−0.73** %	**−0.76** %	**287.30** %	**204.21** %	**−0.35** %
Class A1	−0.35 %	−0.51 %	−0.68 %	275.45 %	170.15 %	−0.31 %
Class A2	−0.44 %	−0.79 %	−0.66 %	256.22 %	145.19 %	−0.27 %
Class B	−0.40 %	−0.57 %	−0.67 %	270.94 %	153.99 %	−0.30 %
Class C	−0.74 %	−1.18 %	−1.25 %	277.69 %	153.39 %	−0.62 %
Class D	−0.36 %	−0.73 %	−0.71 %	259.96 %	154.56 %	−0.21 %
Class F	−0.43 %	−0.60 %	−0.60 %	350.64 %	379.97 %	−0.37 %
JVET Overall	**−0.49** %	**−0.76** %	**−0.82** %	**270.59** %	**155.10** %	**−0.38** %

Table A.11 Coding results for Experiment 4.9 with spatial prediction and full refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, $C_{\mathrm{FL},\{7,8\}}$ explicit partition coding. Spatial prediction, predictor: $C_{\mathrm{TU},2}$, refinement: $C_{\mathrm{EG},1}$.

		GEO spatial prediction mode usage for QP				
Class	Sequence	QP22	QP27	QP32	QP37	Mean
A1	Tango2	2.23 %	6.67 %	7.75 %	7.82 %	6.12 %
	FoodMarket4	3.14 %	6.42 %	7.89 %	8.89 %	6.59 %
	Campfire	1.50 %	1.79 %	2.05 %	2.59 %	1.98 %
A2	CatRobot1	3.32 %	7.40 %	8.59 %	8.55 %	6.96 %
	DaylightRoad2	1.85 %	5.06 %	7.46 %	10.74 %	6.28 %
	ParkRunning3	2.24 %	3.90 %	5.45 %	6.39 %	4.50 %
B	MarketPlace	2.44 %	5.28 %	6.63 %	7.23 %	5.39 %
	RitualDance	4.18 %	5.72 %	6.35 %	6.99 %	5.81 %
	Cactus	2.74 %	5.54 %	7.55 %	7.19 %	5.75 %
	BasketballDrive	3.04 %	4.06 %	5.08 %	5.69 %	4.47 %
	BQTerrace	1.89 %	4.13 %	5.81 %	6.01 %	4.46 %
C	BasketballDrill	4.45 %	4.97 %	5.05 %	5.41 %	4.97 %
	BQMall	5.05 %	8.27 %	10.69 %	11.99 %	9.00 %
	PartyScene	2.43 %	4.09 %	5.06 %	6.28 %	4.47 %
	RaceHorsesL	4.62 %	9.90 %	11.31 %	11.02 %	9.21 %
D	BasketballPass	4.74 %	6.04 %	5.69 %	5.86 %	5.58 %
	BQSquare	2.65 %	2.61 %	2.96 %	4.37 %	3.15 %
	BlowingBubbles	3.99 %	5.11 %	5.35 %	6.42 %	5.22 %
	RaceHorsesM	6.74 %	9.09 %	8.03 %	6.94 %	7.70 %
F	BasketballDrillText	4.67 %	4.78 %	5.13 %	5.49 %	5.02 %
	ArenaOfValor	5.26 %	5.87 %	6.84 %	7.61 %	6.39 %
	SlideEditing	20.06 %	21.84 %	29.50 %	27.09 %	24.62 %
	SlideShow	10.46 %	10.43 %	10.90 %	9.42 %	10.30 %
	Overall	**4.51** %	**6.48** %	**7.70** %	**8.09** %	**6.69** %

Table A.12 Relative usage of spatial prediction in percentage of area coded with GEO for Experiment 4.9.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.47 %	−0.71 %	−1.06 %	314.67 %	193.60 %	−0.37 %
FoodMarket4	−0.38 %	−0.49 %	−0.55 %	272.98 %	162.54 %	−0.23 %
Campfire	−0.19 %	−0.21 %	−0.57 %	267.42 %	202.78 %	−0.22 %
CatRobot1	−0.53 %	−1.10 %	−1.16 %	276.96 %	168.86 %	−0.42 %
DaylightRoad2	−0.29 %	−0.78 %	−0.53 %	281.34 %	155.60 %	−0.15 %
ParkRunning3	−0.37 %	−0.42 %	−0.35 %	225.01 %	126.49 %	−0.32 %
MarketPlace	−0.21 %	−0.15 %	−0.71 %	267.16 %	129.91 %	−0.12 %
RitualDance	−0.29 %	−0.38 %	−0.81 %	272.22 %	134.79 %	−0.17 %
Cactus	−0.52 %	−0.66 %	−0.83 %	294.40 %	251.86 %	−0.27 %
BasketballDrive	−0.29 %	−0.82 %	−0.84 %	253.89 %	115.96 %	−0.50 %
BQTerrace	−0.65 %	−0.98 %	−0.62 %	268.79 %	143.88 %	−0.37 %
BasketballDrill	−0.59 %	−1.16 %	−1.31 %	290.58 %	208.82 %	−0.41 %
BQMall	−1.17 %	−1.23 %	−1.63 %	246.70 %	162.41 %	−0.80 %
PartyScene	−0.31 %	−0.62 %	−0.55 %	264.76 %	106.98 %	−0.15 %
RaceHorsesL	−0.91 %	−1.67 %	−1.70 %	249.31 %	147.59 %	−1.16 %
BasketballPass	−0.36 %	−1.31 %	−0.97 %	259.68 %	169.82 %	−0.05 %
BQSquare	0.01 %	−0.11 %	0.19 %	275.63 %	114.26 %	0.16 %
BlowingBubbles	−0.33 %	−0.41 %	−0.40 %	267.50 %	173.81 %	−0.24 %
RaceHorsesM	−0.78 %	−1.44 %	−1.49 %	239.14 %	144.20 %	−0.52 %
BasketballDrillText	−0.63 %	−0.85 %	−0.91 %	276.25 %	234.42 %	−0.42 %
ArenaOfValor	−0.70 %	−1.17 %	−0.83 %	301.71 %	248.38 %	−0.64 %
SlideEditing	−0.09 %	−0.08 %	−0.04 %	511.65 %	611.88 %	−0.06 %
SlideShow	−0.24 %	−0.53 %	−0.41 %	338.94 %	460.64 %	−0.55 %
Mean	**−0.45** %	**−0.75** %	**−0.79** %	**283.33** %	**198.67** %	**−0.35** %
Class A1	−0.35 %	−0.47 %	−0.73 %	278.62 %	181.40 %	−0.27 %
Class A2	−0.40 %	−0.76 %	−0.68 %	257.33 %	145.49 %	−0.30 %
Class B	−0.39 %	−0.60 %	−0.76 %	267.67 %	146.50 %	−0.29 %
Class C	−0.75 %	−1.17 %	−1.30 %	260.92 %	150.76 %	−0.63 %
Class D	−0.36 %	−0.82 %	−0.67 %	258.66 %	145.16 %	−0.16 %
Class F	−0.42 %	−0.66 %	−0.55 %	345.48 %	350.10 %	−0.42 %
JVET Overall	**−0.48** %	**−0.76** %	**−0.88** %	**265.89** %	**153.86** %	**−0.38** %

Table A.13 Coding results for Experiment 4.10 with spatial prediction and limited refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, $C_{\mathrm{FL},\{7,8\}}$ explicit partition coding. Spatial prediction, predictor: $C_{\mathrm{TU},2}$, refinement: $C_{\mathrm{FL},1}$.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.51 %	−1.12 %	−1.13 %	167.74 %	92.84 %	−0.38 %
FoodMarket4	−0.40 %	−0.43 %	−0.61 %	152.39 %	108.36 %	−0.20 %
Campfire	−0.21 %	−0.22 %	−0.59 %	158.21 %	94.45 %	−0.36 %
CatRobot1	−0.54 %	−1.20 %	−1.24 %	158.15 %	89.90 %	−0.36 %
DaylightRoad2	−0.33 %	−0.68 %	−0.57 %	161.62 %	94.85 %	−0.19 %
ParkRunning3	−0.40 %	−0.42 %	−0.37 %	144.42 %	85.95 %	−0.34 %
MarketPlace	−0.25 %	−0.19 %	−0.80 %	173.23 %	101.19 %	−0.18 %
RitualDance	−0.25 %	−0.41 %	−0.71 %	156.94 %	96.55 %	−0.09 %
Cactus	−0.55 %	−0.79 %	−0.83 %	162.83 %	103.12 %	−0.33 %
BasketballDrive	−0.29 %	−0.89 %	−0.87 %	157.53 %	81.77 %	−0.30 %
BQTerrace	−0.64 %	−0.58 %	−0.51 %	173.57 %	109.65 %	−0.17 %
BasketballDrill	−0.63 %	−1.29 %	−1.17 %	166.93 %	100.90 %	−0.40 %
BQMall	−1.20 %	−1.24 %	−1.53 %	161.23 %	101.75 %	−0.86 %
PartyScene	−0.33 %	−0.67 %	−0.57 %	174.07 %	100.92 %	−0.16 %
RaceHorsesL	−0.89 %	−1.79 %	−1.66 %	169.65 %	107.07 %	−0.95 %
BasketballPass	−0.37 %	−1.15 %	−0.77 %	163.37 %	105.04 %	−0.10 %
BQSquare	0.01 %	−0.20 %	−0.15 %	186.36 %	100.53 %	0.15 %
BlowingBubbles	−0.38 %	−0.49 %	−0.47 %	166.02 %	108.56 %	−0.01 %
RaceHorsesM	−0.75 %	−1.30 %	−1.65 %	162.75 %	108.06 %	−0.47 %
BasketballDrillText	−0.65 %	−0.83 %	−1.04 %	168.70 %	110.25 %	−0.41 %
ArenaOfValor	−0.70 %	−1.11 %	−0.82 %	149.04 %	100.74 %	−0.77 %
SlideEditing	−0.08 %	−0.08 %	−0.05 %	144.69 %	107.95 %	−0.05 %
SlideShow	−0.32 %	−0.52 %	−0.49 %	158.24 %	97.04 %	−0.60 %
Mean	**−0.46** %	**−0.77** %	**−0.81** %	**162.51** %	**100.32** %	**−0.33** %
Class A1	−0.37 %	−0.59 %	−0.78 %	158.81 %	97.15 %	−0.31 %
Class A2	−0.42 %	−0.77 %	−0.73 %	154.35 %	89.71 %	−0.30 %
Class B	−0.40 %	−0.57 %	−0.74 %	164.32 %	97.10 %	−0.22 %
Class C	−0.77 %	−1.25 %	−1.23 %	167.68 %	102.15 %	−0.59 %
Class D	−0.37 %	−0.78 %	−0.76 %	169.15 %	104.95 %	−0.11 %
Class F	−0.44 %	−0.63 %	−0.60 %	154.75 %	103.35 %	−0.45 %
JVET Overall	**−0.49** %	**−0.80** %	**−0.88** %	**162.04** %	**96.88** %	**−0.35** %

Table A.14 Coding results for Experiment 4.11 with spatial prediction and no refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, $C_{\mathrm{FL},\{7,8\}}$ explicit partition coding. Spatial prediction, predictor: $C_{\mathrm{TU},2}$.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.45 %	−1.10 %	−1.05 %	277.03 %	185.26 %	−0.38 %
FoodMarket4	−0.40 %	−0.47 %	−0.58 %	173.76 %	145.37 %	−0.34 %
Campfire	−0.11 %	−0.18 %	−0.45 %	191.45 %	184.24 %	−0.22 %
CatRobot1	−0.61 %	−1.17 %	−1.24 %	166.25 %	166.01 %	−0.54 %
DaylightRoad2	−0.29 %	−0.62 %	−0.47 %	175.94 %	138.28 %	−0.33 %
ParkRunning3	−0.30 %	−0.30 %	−0.25 %	196.06 %	145.55 %	−0.23 %
MarketPlace	−0.20 %	−0.21 %	−0.67 %	156.56 %	116.12 %	−0.06 %
RitualDance	−0.23 %	−0.37 %	−0.77 %	189.05 %	147.73 %	−0.20 %
Cactus	−0.41 %	−0.58 %	−0.64 %	167.96 %	259.04 %	−0.25 %
BasketballDrive	−0.24 %	−0.68 %	−0.58 %	151.04 %	150.95 %	−0.32 %
BQTerrace	−0.50 %	−0.60 %	−0.37 %	155.10 %	154.00 %	−0.00 %
BasketballDrill	−0.51 %	−1.18 %	−1.03 %	168.26 %	267.21 %	−0.20 %
BQMall	−1.12 %	−1.42 %	−1.56 %	168.97 %	179.37 %	−0.60 %
PartyScene	−0.24 %	−0.51 %	−0.48 %	160.19 %	91.01 %	0.02 %
RaceHorsesL	−0.92 %	−1.67 %	−1.73 %	154.88 %	109.80 %	−1.13 %
BasketballPass	−0.23 %	−1.31 %	−0.66 %	138.46 %	196.04 %	−0.11 %
BQSquare	0.03 %	−0.03 %	0.33 %	158.06 %	161.52 %	−0.54 %
BlowingBubbles	−0.16 %	−0.22 %	−0.11 %	164.68 %	183.58 %	0.39 %
RaceHorsesM	−0.65 %	−1.70 %	−1.46 %	143.11 %	151.85 %	0.09 %
BasketballDrillText	−0.55 %	−0.73 %	−1.04 %	169.16 %	272.99 %	−0.13 %
ArenaOfValor	−0.52 %	−0.99 %	−0.64 %	194.80 %	280.29 %	−0.42 %
SlideEditing	−0.06 %	−0.07 %	−0.05 %	314.55 %	328.07 %	−0.12 %
SlideShow	−0.34 %	−0.61 %	−0.52 %	193.83 %	276.20 %	−0.43 %
Mean	**−0.39** %	**−0.73** %	**−0.70** %	**179.53** %	**186.54** %	**−0.26** %

Table A.15 Coding results for Experiment 4.12 with spatial prediction and full refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/8$, $\Delta h = h/8$, $C_{\mathrm{FL},9}$ explicit partition coding. Spatial prediction, predictor: $C_{\mathrm{TU},2}$, refinement: $C_{\mathrm{EG},1}$.

		GEO spatial prediction mode usage for QP				
Class	Sequence	QP22	QP27	QP32	QP37	Mean
A1	Tango2	3.68 %	10.54 %	12.53 %	12.56 %	9.83 %
	FoodMarket4	4.61 %	8.95 %	11.82 %	13.74 %	9.78 %
	Campfire	1.79 %	2.63 %	3.08 %	3.13 %	2.66 %
A2	CatRobot1	5.31 %	11.21 %	12.89 %	12.97 %	10.59 %
	DaylightRoad2	2.87 %	8.10 %	12.04 %	15.27 %	9.57 %
	ParkRunning3	3.02 %	6.27 %	8.65 %	11.16 %	7.27 %
B	MarketPlace	3.42 %	8.09 %	10.96 %	10.88 %	8.34 %
	RitualDance	5.99 %	8.74 %	10.53 %	10.17 %	8.86 %
	Cactus	3.65 %	8.29 %	10.37 %	10.46 %	8.19 %
	BasketballDrive	4.54 %	6.26 %	6.88 %	7.50 %	6.29 %
	BQTerrace	2.34 %	5.43 %	7.75 %	7.41 %	5.73 %
C	BasketballDrill	5.68 %	6.78 %	7.51 %	7.69 %	6.91 %
	BQMall	6.36 %	11.44 %	14.78 %	15.98 %	12.14 %
	PartyScene	3.14 %	5.85 %	7.09 %	8.56 %	6.16 %
	RaceHorsesL	7.44 %	13.43 %	16.87 %	15.08 %	13.20 %
D	BasketballPass	6.41 %	8.08 %	7.54 %	8.57 %	7.65 %
	BQSquare	3.06 %	2.00 %	1.51 %	2.69 %	2.31 %
	BlowingBubbles	5.51 %	6.55 %	7.35 %	7.47 %	6.72 %
	RaceHorsesM	9.76 %	12.28 %	11.61 %	10.64 %	11.07 %
F	BasketballDrillText	6.48 %	6.65 %	7.33 %	6.94 %	6.85 %
	ArenaOfValor	6.36 %	8.29 %	9.40 %	10.18 %	8.56 %
	SlideEditing	19.79 %	22.34 %	24.38 %	29.54 %	24.01 %
	SlideShow	14.48 %	13.25 %	14.05 %	10.29 %	13.02 %
	Overall	**5.90** %	**8.76** %	**10.30** %	**10.82** %	**8.95** %

Table A.16 Relative usage of spatial prediction in percentage of area coded with GEO for Experiment 4.12.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.44 %	−0.98 %	−1.02 %	263.28 %	195.46 %	−0.33 %
FoodMarket4	−0.37 %	−0.31 %	−0.54 %	180.86 %	158.20 %	−0.26 %
Campfire	−0.14 %	−0.17 %	−0.47 %	192.40 %	184.96 %	−0.24 %
CatRobot1	−0.62 %	−1.30 %	−1.32 %	178.29 %	174.77 %	−0.43 %
DaylightRoad2	−0.30 %	−0.82 %	−0.51 %	188.57 %	159.23 %	−0.18 %
ParkRunning3	−0.28 %	−0.32 %	−0.27 %	199.88 %	128.35 %	−0.18 %
MarketPlace	−0.19 %	−0.28 %	−0.52 %	163.73 %	132.43 %	−0.17 %
RitualDance	−0.24 %	−0.37 %	−0.78 %	195.99 %	161.68 %	−0.19 %
Cactus	−0.42 %	−0.74 %	−0.69 %	174.92 %	281.39 %	−0.35 %
BasketballDrive	−0.27 %	−0.65 %	−0.73 %	145.28 %	143.17 %	−0.28 %
BQTerrace	−0.48 %	−0.78 %	−0.37 %	168.55 %	172.93 %	−0.37 %
BasketballDrill	−0.49 %	−1.08 %	−1.05 %	180.18 %	264.69 %	−0.23 %
BQMall	−1.16 %	−1.28 %	−1.33 %	165.75 %	190.30 %	−0.87 %
PartyScene	−0.27 %	−0.49 %	−0.50 %	160.77 %	102.31 %	−0.12 %
RaceHorsesL	−0.91 %	−1.67 %	−1.62 %	164.81 %	123.79 %	−0.93 %
BasketballPass	−0.22 %	−1.38 %	−0.87 %	147.74 %	210.45 %	−0.03 %
BQSquare	0.04 %	0.00 %	0.38 %	158.07 %	173.32 %	−0.46 %
BlowingBubbles	−0.22 %	−0.23 %	−0.37 %	165.89 %	208.25 %	0.18 %
RaceHorsesM	−0.68 %	−1.78 %	−1.55 %	148.26 %	151.71 %	−0.03 %
BasketballDrillText	−0.52 %	−0.84 %	−1.04 %	174.83 %	287.50 %	−0.08 %
ArenaOfValor	−0.53 %	−0.97 %	−0.68 %	179.14 %	284.67 %	−0.41 %
SlideEditing	−0.04 %	−0.06 %	−0.02 %	305.80 %	317.90 %	−0.09 %
SlideShow	−0.30 %	−0.60 %	−0.49 %	198.94 %	372.26 %	−0.43 %
Mean	**−0.39** %	**−0.74** %	**−0.71** %	**182.69** %	**199.12** %	**−0.28** %
Class A1	−0.32 %	−0.49 %	−0.67 %	207.74 %	176.63 %	−0.27 %
Class A2	−0.40 %	−0.81 %	−0.70 %	187.88 %	149.56 %	−0.26 %
Class B	−0.32 %	−0.56 %	−0.62 %	167.52 %	169.09 %	−0.27 %
Class C	−0.71 %	−1.13 %	−1.13 %	166.73 %	157.06 %	−0.54 %
Class D	−0.27 %	−0.85 %	−0.60 %	153.26 %	182.91 %	−0.08 %
Class F	−0.35 %	−0.62 %	−0.56 %	207.39 %	307.81 %	−0.25 %
JVET Overall	**−0.44** %	**−0.75** %	**−0.78** %	**178.72** %	**163.19** %	**−0.34** %

Table A.17 Coding results for Experiment 4.13 with spatial prediction and limited refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/8, \Delta h = h/8$, $C_{\mathrm{FL},9}$ explicit partition coding. Spatial prediction, predictor: $C_{\mathrm{TU},2}$, refinement: $C_{\mathrm{FL},1}$.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.47 %	−1.09 %	−0.96 %	142.22 %	96.54 %	−0.34 %
FoodMarket4	−0.39 %	−0.47 %	−0.69 %	108.90 %	98.43 %	−0.42 %
Campfire	−0.13 %	−0.16 %	−0.48 %	116.59 %	101.42 %	−0.26 %
CatRobot1	−0.58 %	−1.10 %	−1.17 %	107.37 %	93.62 %	−0.37 %
DaylightRoad2	−0.28 %	−0.68 %	−0.47 %	111.87 %	102.06 %	−0.32 %
ParkRunning3	−0.27 %	−0.31 %	−0.30 %	122.38 %	104.88 %	−0.21 %
MarketPlace	−0.18 %	−0.09 %	−0.53 %	108.54 %	86.65 %	−0.09 %
RitualDance	−0.25 %	−0.52 %	−0.87 %	125.21 %	95.64 %	−0.24 %
Cactus	−0.44 %	−0.48 %	−0.66 %	99.66 %	115.61 %	−0.45 %
BasketballDrive	−0.25 %	−0.75 %	−0.78 %	97.09 %	108.61 %	−0.23 %
BQTerrace	−0.53 %	−0.54 %	−0.38 %	112.50 %	108.93 %	−0.32 %
BasketballDrill	−0.52 %	−1.05 %	−1.10 %	106.94 %	126.44 %	−0.22 %
BQMall	−1.15 %	−1.45 %	−1.65 %	107.47 %	122.24 %	−0.83 %
PartyScene	−0.28 %	−0.49 %	−0.50 %	112.34 %	91.54 %	−0.05 %
RaceHorsesL	−0.91 %	−1.68 %	−1.63 %	107.99 %	92.22 %	−1.03 %
BasketballPass	−0.23 %	−1.27 %	−0.70 %	93.78 %	148.23 %	−0.02 %
BQSquare	0.04 %	0.07 %	0.19 %	115.37 %	129.96 %	−0.63 %
BlowingBubbles	−0.25 %	−0.23 %	−0.34 %	113.39 %	135.57 %	0.05 %
RaceHorsesM	−0.66 %	−1.71 %	−1.41 %	97.89 %	111.54 %	−0.28 %
BasketballDrillText	−0.54 %	−0.73 %	−0.86 %	100.84 %	118.56 %	−0.21 %
ArenaOfValor	−0.55 %	−0.93 %	−0.71 %	103.49 %	107.93 %	−0.41 %
SlideEditing	−0.07 %	−0.08 %	−0.06 %	101.47 %	59.03 %	−0.17 %
SlideShow	−0.31 %	−0.60 %	−0.45 %	101.30 %	74.92 %	−0.47 %
Mean	**−0.40** %	**−0.71** %	**−0.72** %	**109.33** %	**105.68** %	**−0.33** %
Class A1	−0.33 %	−0.57 %	−0.71 %	121.12 %	98.39 %	−0.34 %
Class A2	−0.38 %	−0.70 %	−0.65 %	113.53 %	99.77 %	−0.30 %
Class B	−0.33 %	−0.48 %	−0.65 %	107.86 %	102.35 %	−0.27 %
Class C	−0.71 %	−1.17 %	−1.22 %	108.46 %	106.20 %	−0.53 %
Class D	−0.28 %	−0.79 %	−0.56 %	104.15 %	130.40 %	−0.22 %
Class F	−0.36 %	−0.58 %	−0.52 %	101.57 %	86.62 %	−0.31 %
JVET Overall	**−0.44** %	**−0.72** %	**−0.81** %	**111.69** %	**102.03** %	**−0.36** %

Table A.18 Coding results for Experiment 4.14 with spatial prediction and no refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/8, \Delta h = h/8$, $C_{\mathrm{FL},9}$ explicit partition coding. Spatial prediction, predictor: $C_{\mathrm{TU},2}$.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.51 %	−0.73 %	−1.03 %	303.99 %	182.20 %	−0.42 %
FoodMarket4	−0.39 %	−0.52 %	−0.59 %	279.66 %	172.85 %	−0.21 %
Campfire	−0.17 %	−0.24 %	−0.59 %	275.12 %	174.89 %	−0.27 %
CatRobot1	−0.57 %	−1.11 %	−1.15 %	295.84 %	172.90 %	−0.36 %
DaylightRoad2	−0.31 %	−0.90 %	−0.45 %	264.20 %	150.99 %	−0.20 %
ParkRunning3	−0.40 %	−0.44 %	−0.39 %	233.53 %	133.69 %	−0.41 %
MarketPlace	−0.26 %	−0.05 %	−0.66 %	277.13 %	144.34 %	−0.18 %
RitualDance	−0.29 %	−0.39 %	−0.76 %	284.20 %	160.62 %	−0.25 %
Cactus	−0.56 %	−0.59 %	−0.78 %	318.97 %	266.51 %	−0.44 %
BasketballDrive	−0.29 %	−0.76 %	−0.71 %	265.18 %	128.45 %	−0.36 %
BQTerrace	−0.71 %	−0.87 %	−0.57 %	287.77 %	137.07 %	−0.56 %
BasketballDrill	−0.64 %	−1.29 %	−1.10 %	303.47 %	233.26 %	−0.51 %
BQMall	−1.22 %	−1.44 %	−1.46 %	259.32 %	165.78 %	−0.94 %
PartyScene	−0.31 %	−0.53 %	−0.53 %	258.32 %	123.58 %	−0.05 %
RaceHorsesL	−0.93 %	−1.69 %	−1.54 %	271.20 %	146.41 %	−0.95 %
BasketballPass	−0.32 %	−1.30 %	−0.94 %	264.24 %	176.14 %	0.05 %
BQSquare	−0.03 %	−0.12 %	0.38 %	276.33 %	125.30 %	−0.17 %
BlowingBubbles	−0.35 %	−0.20 %	−0.35 %	265.61 %	158.03 %	0.02 %
RaceHorsesM	−0.77 %	−1.61 %	−1.88 %	252.28 %	161.21 %	−0.58 %
BasketballDrillText	−0.63 %	−0.91 %	−0.92 %	313.89 %	232.44 %	−0.24 %
ArenaOfValor	−0.79 %	−1.21 %	−0.88 %	327.09 %	305.20 %	−0.92 %
SlideEditing	−0.09 %	−0.07 %	−0.03 %	543.14 %	568.74 %	−0.13 %
SlideShow	−0.29 %	−0.36 %	−0.19 %	370.75 %	481.45 %	−0.61 %
Mean	**−0.47** %	**−0.75** %	**−0.74** %	**295.27** %	**204.44** %	**−0.38** %
Class A1	−0.36 %	−0.49 %	−0.74 %	281.42 %	173.14 %	−0.30 %
Class A2	−0.43 %	−0.82 %	−0.67 %	260.91 %	148.23 %	−0.32 %
Class B	−0.42 %	−0.53 %	−0.70 %	283.62 %	158.40 %	−0.36 %
Class C	−0.78 %	−1.24 %	−1.16 %	271.04 %	161.54 %	−0.61 %
Class D	−0.37 %	−0.81 %	−0.70 %	262.78 %	151.27 %	−0.17 %
Class F	−0.45 %	−0.64 %	−0.50 %	378.05 %	367.03 %	−0.48 %
JVET Overall	**−0.51** %	**−0.77** %	**−0.82** %	**275.14** %	**159.95** %	**−0.41** %

Table A.19 Coding results for Experiment 4.15 with temporal prediction and full refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, $C_{\mathrm{FL},\{7,8\}}$ explicit partition coding. Temporal prediction, predictor $C_{\mathrm{TU},2}$, refinement $C_{\mathrm{EG},1}$.

		GEO temporal prediction mode usage for QP				
Class	Sequence	QP22	QP27	QP32	QP37	Mean
A1	Tango2	13.28 %	29.72 %	33.32 %	32.72 %	27.26 %
	FoodMarket4	16.64 %	20.72 %	23.45 %	25.40 %	21.55 %
	Campfire	12.88 %	11.19 %	13.41 %	12.78 %	12.57 %
A2	CatRobot1	24.70 %	37.48 %	38.61 %	37.80 %	34.65 %
	DaylightRoad2	14.03 %	24.74 %	28.60 %	31.70 %	24.77 %
	ParkRunning3	16.63 %	22.40 %	25.93 %	28.58 %	23.39 %
B	MarketPlace	18.66 %	26.60 %	28.96 %	28.45 %	25.66 %
	RitualDance	17.20 %	21.13 %	22.06 %	22.20 %	20.64 %
	Cactus	18.73 %	28.15 %	33.11 %	33.64 %	28.41 %
	BasketballDrive	17.00 %	22.99 %	25.46 %	26.64 %	23.02 %
	BQTerrace	20.06 %	27.51 %	35.09 %	37.40 %	30.01 %
C	BasketballDrill	24.76 %	28.59 %	31.70 %	33.07 %	29.53 %
	BQMall	25.96 %	34.56 %	42.55 %	45.21 %	37.07 %
	PartyScene	19.35 %	26.00 %	32.26 %	33.66 %	27.82 %
	RaceHorsesL	18.45 %	28.29 %	32.87 %	33.01 %	28.16 %
D	BasketballPass	24.54 %	30.70 %	33.50 %	32.16 %	30.23 %
	BQSquare	25.05 %	24.74 %	28.00 %	29.38 %	26.80 %
	BlowingBubbles	28.83 %	33.26 %	39.28 %	36.82 %	34.55 %
	RaceHorsesM	24.92 %	28.43 %	31.40 %	30.52 %	28.82 %
F	BasketballDrillText	24.72 %	27.47 %	29.27 %	32.46 %	28.48 %
	ArenaOfValor	28.76 %	35.33 %	41.59 %	43.88 %	37.39 %
	SlideEditing	27.52 %	38.26 %	40.21 %	42.22 %	37.05 %
	SlideShow	18.10 %	19.76 %	21.00 %	24.81 %	20.92 %
	Overall	**20.90** %	**27.31** %	**30.94** %	**31.94** %	**27.77** %

Table A.20 Relative usage of temporal prediction in percentage of area coded with GEO for Experiment 4.15.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.54 %	−0.73 %	−1.15 %	304.90 %	184.86 %	−0.41 %
FoodMarket4	−0.45 %	−0.61 %	−0.63 %	292.12 %	176.37 %	−0.42 %
Campfire	−0.15 %	−0.24 %	−0.57 %	269.74 %	189.37 %	−0.24 %
CatRobot1	−0.62 %	−1.22 %	−1.21 %	267.08 %	160.12 %	−0.43 %
DaylightRoad2	−0.38 %	−0.92 %	−0.51 %	263.42 %	157.52 %	−0.29 %
ParkRunning3	−0.38 %	−0.42 %	−0.37 %	238.84 %	133.80 %	−0.31 %
MarketPlace	−0.30 %	−0.15 %	−0.67 %	288.17 %	151.16 %	−0.20 %
RitualDance	−0.28 %	−0.49 %	−0.84 %	278.37 %	160.52 %	−0.22 %
Cactus	−0.54 %	−0.74 %	−0.74 %	321.56 %	264.60 %	−0.32 %
BasketballDrive	−0.30 %	−0.74 %	−0.75 %	252.48 %	139.86 %	−0.28 %
BQTerrace	−0.69 %	−0.83 %	−0.64 %	294.31 %	148.11 %	−0.27 %
BasketballDrill	−0.65 %	−1.19 %	−1.11 %	309.21 %	235.09 %	−0.47 %
BQMall	−1.30 %	−1.34 %	−1.44 %	276.41 %	176.29 %	−1.11 %
PartyScene	−0.32 %	−0.58 %	−0.43 %	253.98 %	118.14 %	−0.09 %
RaceHorsesL	−0.88 %	−1.73 %	−1.66 %	279.95 %	160.92 %	−1.18 %
BasketballPass	−0.34 %	−1.17 %	−0.93 %	264.28 %	186.41 %	0.14 %
BQSquare	−0.01 %	−0.24 %	0.26 %	277.11 %	140.30 %	0.13 %
BlowingBubbles	−0.41 %	−0.43 %	−0.36 %	270.76 %	180.81 %	−0.14 %
RaceHorsesM	−0.82 %	−1.75 %	−1.93 %	256.35 %	154.37 %	−0.40 %
BasketballDrillText	−0.73 %	−0.88 %	−1.09 %	308.34 %	275.06 %	−0.43 %
ArenaOfValor	−0.82 %	−1.19 %	−0.83 %	295.44 %	312.08 %	−0.77 %
SlideEditing	−0.09 %	−0.07 %	−0.04 %	526.15 %	645.64 %	−0.14 %
SlideShow	−0.31 %	−0.43 %	−0.25 %	362.87 %	483.72 %	−0.66 %
Mean	**−0.49** %	**−0.79** %	**−0.78** %	**293.56** %	**214.57** %	**−0.37** %
Class A1	−0.38 %	−0.53 %	−0.78 %	284.71 %	178.94 %	−0.36 %
Class A2	−0.46 %	−0.85 %	−0.70 %	254.59 %	147.61 %	−0.34 %
Class B	−0.42 %	−0.59 %	−0.73 %	283.86 %	164.98 %	−0.26 %
Class C	−0.79 %	−1.21 %	−1.16 %	277.76 %	165.87 %	−0.71 %
Class D	−0.40 %	−0.90 %	−0.74 %	265.71 %	162.44 %	−0.07 %
Class F	−0.49 %	−0.64 %	−0.55 %	362.33 %	400.17 %	−0.50 %
JVET Overall	**−0.52** %	**−0.79** %	**−0.85** %	**276.31** %	**164.23** %	**−0.42** %

Table A.21 Coding results for Experiment 4.16 with temporal prediction and limited refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, $C_{\text{FL},\{7,8\}}$ explicit partition coding. Temporal prediction, predictor: $C_{\text{TU},2}$, refinement: $C_{\text{FL},1}$.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.55 %	−0.86 %	−1.10 %	166.58 %	95.89 %	−0.42 %
FoodMarket4	−0.43 %	−0.51 %	−0.60 %	157.11 %	108.65 %	−0.39 %
Campfire	−0.20 %	−0.23 %	−0.60 %	159.57 %	98.22 %	−0.30 %
CatRobot1	−0.59 %	−1.12 %	−1.19 %	161.92 %	92.86 %	−0.40 %
DaylightRoad2	−0.36 %	−0.86 %	−0.64 %	169.76 %	97.91 %	−0.29 %
ParkRunning3	−0.39 %	−0.43 %	−0.39 %	141.83 %	92.80 %	−0.33 %
MarketPlace	−0.25 %	−0.22 %	−1.06 %	179.16 %	105.28 %	−0.25 %
RitualDance	−0.33 %	−0.61 %	−0.77 %	187.79 %	98.32 %	−0.25 %
Cactus	−0.58 %	−0.78 %	−0.86 %	181.23 %	116.98 %	−0.35 %
BasketballDrive	−0.34 %	−0.64 %	−0.83 %	178.69 %	97.94 %	−0.34 %
BQTerrace	−0.70 %	−0.69 %	−0.47 %	182.81 %	103.73 %	−0.25 %
BasketballDrill	−0.69 %	−1.33 %	−1.26 %	182.86 %	109.01 %	−0.46 %
BQMall	−1.26 %	−1.41 %	−1.57 %	177.87 %	126.97 %	−0.87 %
PartyScene	−0.38 %	−0.63 %	−0.65 %	188.10 %	102.37 %	−0.26 %
RaceHorsesL	−0.97 %	−1.75 %	−1.58 %	186.78 %	112.40 %	−1.32 %
BasketballPass	−0.43 %	−1.27 %	−0.85 %	169.14 %	118.94 %	−0.12 %
BQSquare	−0.04 %	−0.32 %	0.06 %	190.60 %	111.58 %	−0.33 %
BlowingBubbles	−0.42 %	−0.26 %	−0.30 %	188.69 %	125.84 %	−0.20 %
RaceHorsesM	−0.77 %	−1.60 %	−1.59 %	171.15 %	120.56 %	−0.52 %
BasketballDrillText	−0.74 %	−0.99 %	−1.06 %	176.85 %	119.91 %	−0.45 %
ArenaOfValor	−0.79 %	−1.18 %	−0.90 %	168.06 %	109.88 %	−0.83 %
SlideEditing	−0.09 %	−0.09 %	−0.06 %	145.64 %	108.90 %	−0.10 %
SlideShow	−0.32 %	−0.62 %	−0.53 %	174.04 %	112.94 %	−0.58 %
Mean	**−0.51** %	**−0.80** %	**−0.82** %	**173.31** %	**108.17** %	**−0.42** %
Class A1	−0.39 %	−0.53 %	−0.76 %	160.60 %	99.91 %	−0.37 %
Class A2	−0.45 %	−0.80 %	−0.74 %	157.23 %	94.34 %	−0.34 %
Class B	−0.44 %	−0.59 %	−0.80 %	181.57 %	103.28 %	−0.29 %
Class C	−0.82 %	−1.28 %	−1.27 %	183.66 %	111.91 %	−0.73 %
Class D	−0.42 %	−0.86 %	−0.67 %	179.14 %	118.70 %	−0.29 %
Class F	−0.49 %	−0.72 %	−0.64 %	165.55 %	112.22 %	−0.49 %
JVET Overall	**−0.53** %	**−0.81** %	**−0.90** %	**172.67** %	**102.94** %	**−0.43** %

Table A.22 Coding results for Experiment 4.17 with temporal prediction and no refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, $C_{\mathrm{FL},\{7,8\}}$ explicit partition coding. Temporal prediction, predictor: $C_{\mathrm{TU},2}$.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.56 %	−0.80 %	−1.07 %	261.83 %	197.84 %	−0.44 %
FoodMarket4	−0.41 %	−0.42 %	−0.53 %	191.36 %	158.33 %	−0.36 %
Campfire	−0.12 %	−0.17 %	−0.46 %	193.84 %	182.90 %	−0.27 %
CatRobot1	−0.69 %	−1.39 %	−1.28 %	176.45 %	161.46 %	−0.41 %
DaylightRoad2	−0.34 %	−0.94 %	−0.45 %	179.38 %	146.93 %	−0.28 %
ParkRunning3	−0.31 %	−0.34 %	−0.32 %	200.19 %	144.05 %	−0.27 %
MarketPlace	−0.18 %	−0.09 %	−0.86 %	173.24 %	128.40 %	−0.04 %
RitualDance	−0.23 %	−0.54 %	−0.83 %	205.56 %	145.54 %	−0.16 %
Cactus	−0.50 %	−0.57 %	−0.78 %	184.97 %	268.90 %	−0.28 %
BasketballDrive	−0.24 %	−0.69 %	−0.64 %	153.68 %	150.04 %	−0.42 %
BQTerrace	−0.57 %	−0.63 %	−0.42 %	179.34 %	157.50 %	−0.35 %
BasketballDrill	−0.61 %	−1.21 %	−1.28 %	188.21 %	267.95 %	−0.43 %
BQMall	−1.31 %	−1.45 %	−1.86 %	174.71 %	180.39 %	−0.81 %
PartyScene	−0.30 %	−0.58 %	−0.47 %	163.77 %	109.72 %	−0.20 %
RaceHorsesL	−1.02 %	−1.77 %	−1.74 %	170.20 %	110.47 %	−1.22 %
BasketballPass	−0.31 %	−1.15 %	−0.61 %	148.74 %	215.18 %	−0.11 %
BQSquare	−0.10 %	0.05 %	0.24 %	169.45 %	183.51 %	−0.49 %
BlowingBubbles	−0.33 %	−0.41 %	−0.42 %	161.55 %	219.79 %	0.21 %
RaceHorsesM	−0.67 %	−1.59 %	−1.48 %	159.14 %	164.03 %	−0.02 %
BasketballDrillText	−0.59 %	−0.93 %	−1.05 %	177.85 %	289.93 %	−0.10 %
ArenaOfValor	−0.67 %	−1.11 %	−0.70 %	185.35 %	275.59 %	−0.69 %
SlideEditing	−0.08 %	−0.09 %	−0.04 %	319.83 %	350.76 %	−0.15 %
SlideShow	−0.31 %	−0.66 %	−0.45 %	201.92 %	354.79 %	−0.52 %
Mean	**−0.45** %	**−0.76** %	**−0.76** %	**187.85** %	**198.43** %	**−0.34** %
Class A1	−0.37 %	−0.46 %	−0.69 %	211.96 %	176.41 %	−0.35 %
Class A2	−0.45 %	−0.89 %	−0.68 %	184.37 %	148.04 %	−0.32 %
Class B	−0.34 %	−0.50 %	−0.71 %	177.27 %	162.39 %	−0.25 %
Class C	−0.81 %	−1.25 %	−1.34 %	173.04 %	153.72 %	−0.67 %
Class D	−0.35 %	−0.78 %	−0.57 %	157.96 %	192.46 %	−0.10 %
Class F	−0.41 %	−0.70 %	−0.56 %	213.49 %	314.66 %	−0.37 %
JVET Overall	**−0.49** %	**−0.77** %	**−0.87** %	**183.98** %	**159.72** %	**−0.40** %

Table A.23 Coding results for Experiment 4.18 with temporal prediction and full refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/8$, $\Delta h = h/8$, $C_{\mathrm{FL},9}$ explicit partition coding. Temporal prediction, predictor: $C_{\mathrm{TU},2}$ refinement: $C_{\mathrm{EG},1}$.

		GEO temporal prediction mode usage for QP				
Class	Sequence	QP22	QP27	QP32	QP37	Mean
A1	Tango2	27.73 %	45.61 %	49.95 %	49.91 %	43.30 %
	FoodMarket4	29.20 %	33.16 %	37.17 %	38.67 %	34.55 %
	Campfire	32.33 %	24.05 %	24.70 %	25.32 %	26.60 %
A2	CatRobot1	45.01 %	56.03 %	59.46 %	58.70 %	54.80 %
	DaylightRoad2	31.42 %	41.03 %	47.16 %	50.56 %	42.54 %
	ParkRunning3	28.57 %	38.08 %	43.29 %	48.03 %	39.49 %
B	MarketPlace	35.77 %	41.57 %	44.88 %	46.94 %	42.29 %
	RitualDance	28.77 %	31.89 %	35.29 %	34.71 %	32.67 %
	Cactus	37.23 %	45.52 %	50.04 %	49.46 %	45.56 %
	BasketballDrive	31.73 %	36.13 %	37.55 %	39.15 %	36.14 %
	BQTerrace	41.93 %	49.08 %	55.12 %	56.73 %	50.72 %
C	BasketballDrill	36.72 %	39.74 %	41.68 %	44.95 %	40.77 %
	BQMall	39.39 %	48.35 %	54.74 %	60.59 %	50.77 %
	PartyScene	32.88 %	38.46 %	44.79 %	48.05 %	41.04 %
	RaceHorsesL	30.64 %	37.34 %	41.77 %	44.24 %	38.50 %
D	BasketballPass	32.82 %	38.99 %	43.53 %	47.06 %	40.60 %
	BQSquare	40.93 %	41.61 %	46.46 %	52.51 %	45.38 %
	BlowingBubbles	41.84 %	49.39 %	53.86 %	55.47 %	50.14 %
	RaceHorsesM	35.69 %	39.87 %	44.25 %	46.63 %	41.61 %
F	BasketballDrillText	39.10 %	40.83 %	40.57 %	43.98 %	41.12 %
	ArenaOfValor	43.46 %	51.98 %	57.50 %	60.33 %	53.32 %
	SlideEditing	42.42 %	39.62 %	45.08 %	52.05 %	44.79 %
	SlideShow	29.52 %	29.34 %	34.13 %	40.66 %	33.41 %
	Overall	**35.44** %	**40.77** %	**44.91** %	**47.60** %	**42.18** %

Table A.24 Relative usage of temporal prediction in percentage of area coded with GEO for Experiment 4.18.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.57 %	−0.80 %	−1.09 %	260.46 %	199.57 %	−0.45 %
FoodMarket4	−0.46 %	−0.42 %	−0.59 %	178.83 %	171.77 %	−0.47 %
Campfire	−0.14 %	−0.16 %	−0.51 %	189.03 %	163.58 %	−0.31 %
CatRobot1	−0.69 %	−1.28 %	−1.30 %	181.94 %	158.64 %	−0.42 %
DaylightRoad2	−0.37 %	−0.96 %	−0.51 %	178.54 %	147.26 %	−0.31 %
ParkRunning3	−0.31 %	−0.33 %	−0.28 %	188.48 %	139.63 %	−0.23 %
MarketPlace	−0.24 %	−0.16 %	−0.83 %	163.25 %	131.46 %	−0.10 %
RitualDance	−0.25 %	−0.48 %	−0.91 %	198.66 %	151.35 %	−0.20 %
Cactus	−0.50 %	−0.59 %	−0.72 %	171.10 %	294.27 %	−0.37 %
BasketballDrive	−0.25 %	−0.80 %	−0.69 %	147.89 %	172.92 %	−0.34 %
BQTerrace	−0.56 %	−0.80 %	−0.46 %	162.26 %	152.41 %	−0.21 %
BasketballDrill	−0.57 %	−1.28 %	−1.04 %	177.36 %	282.84 %	−0.41 %
BQMall	−1.39 %	−1.56 %	−2.04 %	171.52 %	201.46 %	−0.99 %
PartyScene	−0.31 %	−0.56 %	−0.58 %	159.14 %	109.15 %	−0.02 %
RaceHorsesL	−0.99 %	−1.77 %	−1.73 %	164.84 %	134.46 %	−1.35 %
BasketballPass	−0.34 %	−1.32 %	−0.61 %	148.87 %	212.70 %	−0.02 %
BQSquare	−0.03 %	0.14 %	0.27 %	168.70 %	175.07 %	0.03 %
BlowingBubbles	−0.34 %	−0.41 %	−0.42 %	168.07 %	191.18 %	−0.05 %
RaceHorsesM	−0.74 %	−1.70 %	−1.57 %	151.34 %	165.82 %	−0.31 %
BasketballDrillText	−0.65 %	−0.86 %	−1.08 %	177.11 %	297.37 %	−0.19 %
ArenaOfValor	−0.69 %	−1.17 %	−0.67 %	208.52 %	306.25 %	−0.57 %
SlideEditing	−0.07 %	−0.08 %	−0.04 %	310.18 %	344.35 %	−0.16 %
SlideShow	−0.32 %	−0.70 %	−0.55 %	202.69 %	347.18 %	−0.50 %
Mean	**−0.47** %	**−0.78** %	**−0.78** %	**183.86** %	**202.20** %	**−0.35** %
Class A1	−0.39 %	−0.46 %	−0.73 %	204.97 %	175.09 %	−0.41 %
Class A2	−0.46 %	−0.86 %	−0.70 %	182.21 %	146.88 %	−0.32 %
Class B	−0.36 %	−0.56 %	−0.72 %	166.63 %	169.96 %	−0.24 %
Class C	−0.82 %	−1.29 %	−1.35 %	166.29 %	168.82 %	−0.69 %
Class D	−0.36 %	−0.82 %	−0.58 %	157.67 %	183.73 %	−0.09 %
Class F	−0.43 %	−0.70 %	−0.59 %	218.51 %	319.12 %	−0.35 %
JVET Overall	**−0.51** %	**−0.80** %	**−0.89** %	**176.71** %	**165.76** %	**−0.41** %

Table A.25 Coding results for Experiment 4.19 with temporal prediction and limited refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/8$, $\Delta h = h/8$, $C_{\mathrm{FL},9}$ explicit partition coding. Temporal prediction, predictor: $C_{\mathrm{TU},2}$, refinement: $C_{\mathrm{FL},1}$.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.56 %	−0.87 %	−0.91 %	141.24 %	101.52 %	−0.38 %
FoodMarket4	−0.40 %	−0.47 %	−0.50 %	128.06 %	110.56 %	−0.41 %
Campfire	−0.15 %	−0.17 %	−0.53 %	114.99 %	106.34 %	−0.28 %
CatRobot1	−0.71 %	−1.30 %	−1.38 %	110.08 %	96.90 %	−0.54 %
DaylightRoad2	−0.39 %	−0.92 %	−0.53 %	114.48 %	104.27 %	−0.37 %
ParkRunning3	−0.31 %	−0.35 %	−0.32 %	138.31 %	104.60 %	−0.28 %
MarketPlace	−0.20 %	−0.18 %	−0.87 %	109.08 %	97.77 %	−0.17 %
RitualDance	−0.26 %	−0.42 %	−0.74 %	136.89 %	93.98 %	−0.19 %
Cactus	−0.53 %	−0.56 %	−0.75 %	111.94 %	105.66 %	−0.36 %
BasketballDrive	−0.27 %	−0.60 %	−0.70 %	104.95 %	112.17 %	−0.22 %
BQTerrace	−0.58 %	−0.94 %	−0.55 %	106.27 %	110.31 %	0.05 %
BasketballDrill	−0.64 %	−1.13 %	−1.03 %	115.40 %	152.29 %	−0.49 %
BQMall	−1.32 %	−1.51 %	−1.85 %	115.08 %	147.44 %	−1.11 %
PartyScene	−0.31 %	−0.60 %	−0.60 %	121.19 %	90.53 %	−0.23 %
RaceHorsesL	−1.02 %	−1.84 %	−1.60 %	113.60 %	90.54 %	−1.27 %
BasketballPass	−0.33 %	−1.33 %	−0.95 %	100.77 %	156.63 %	−0.15 %
BQSquare	0.04 %	0.00 %	0.39 %	125.27 %	162.88 %	−0.59 %
BlowingBubbles	−0.35 %	−0.39 %	−0.24 %	116.34 %	145.39 %	0.10 %
RaceHorsesM	−0.76 %	−1.77 %	−1.43 %	109.48 %	123.89 %	−0.42 %
BasketballDrillText	−0.68 %	−0.83 %	−1.06 %	108.57 %	122.76 %	−0.30 %
ArenaOfValor	−0.71 %	−1.15 %	−0.71 %	105.97 %	108.08 %	−0.67 %
SlideEditing	−0.08 %	−0.09 %	−0.05 %	99.92 %	65.08 %	−0.15 %
SlideShow	−0.31 %	−0.60 %	−0.44 %	106.24 %	77.40 %	−0.50 %
Mean	**−0.47** %	**−0.78** %	**−0.75** %	**115.40** %	**112.48** %	**−0.39** %
Class A1	−0.37 %	−0.50 %	−0.65 %	127.05 %	105.31 %	−0.36 %
Class A2	−0.47 %	−0.86 %	−0.74 %	120.01 %	101.64 %	−0.40 %
Class B	−0.37 %	−0.54 %	−0.72 %	112.94 %	103.55 %	−0.18 %
Class C	−0.82 %	−1.27 %	−1.27 %	115.96 %	115.78 %	−0.77 %
Class D	−0.35 %	−0.87 %	−0.56 %	111.65 %	145.97 %	−0.26 %
Class F	−0.44 %	−0.67 %	−0.57 %	105.02 %	90.16 %	−0.41 %
JVET Overall	**−0.51** %	**−0.79** %	**−0.86** %	**117.87** %	**106.64** %	**−0.42** %

Table A.26 Coding results for Experiment 4.20 with temporal prediction and no refinement coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. BI: $\Delta w = w/8$, $\Delta h = h/8$, $C_{\mathrm{FL},9}$ explicit partition coding. Temporal prediction, predictor: $C_{\mathrm{TU},2}$.

	BD-rate change			Relative complexity	
Sequence / Class	Y	U	V	EncT	DecT
Tango2	−0.05 %	−1.04 %	−0.07 %	106.46 %	96.28 %
FoodMarket4	−0.02 %	0.36 %	−0.03 %	104.43 %	77.46 %
Campfire	−0.11 %	−0.09 %	−0.26 %	103.51 %	79.05 %
CatRobot1	−0.27 %	−0.47 %	−0.36 %	104.50 %	75.41 %
DaylightRoad2	−0.19 %	−0.61 %	−0.52 %	104.36 %	74.36 %
ParkRunning3	−0.20 %	−0.20 %	−0.18 %	102.92 %	86.46 %
MarketPlace	−0.12 %	0.53 %	−0.93 %	114.21 %	103.68 %
RitualDance	0.03 %	−0.25 %	−0.46 %	100.25 %	78.52 %
Cactus	−0.23 %	−0.48 %	−0.49 %	107.43 %	99.97 %
BasketballDrive	−0.03 %	−0.17 %	−0.09 %	111.10 %	101.94 %
BQTerrace	−0.38 %	−0.52 %	0.04 %	105.45 %	98.29 %
BasketballDrill	−0.56 %	−0.80 %	−0.31 %	103.57 %	137.92 %
BQMall	−1.20 %	−1.76 %	−1.21 %	102.72 %	122.61 %
PartyScene	−0.31 %	−0.58 %	−0.67 %	102.32 %	112.20 %
RaceHorsesL	−1.20 %	−1.62 %	−1.81 %	102.21 %	122.99 %
BasketballPass	−0.32 %	−0.07 %	−1.79 %	95.63 %	211.81 %
BQSquare	−0.07 %	0.42 %	0.83 %	99.92 %	238.79 %
BlowingBubbles	−0.32 %	−0.95 %	−1.40 %	110.79 %	232.89 %
RaceHorsesM	−1.21 %	−1.82 %	−1.66 %	111.99 %	199.43 %
BasketballDrillText	−0.82 %	−0.71 %	−0.33 %	105.26 %	130.48 %
ArenaOfValor	−0.69 %	−0.83 %	−0.65 %	110.16 %	103.94 %
SlideEditing	−0.29 %	−0.21 %	−0.22 %	90.45 %	82.94 %
SlideShow	−0.17 %	0.66 %	1.31 %	95.25 %	82.68 %
Mean	**−0.38** %	**−0.49** %	**−0.49** %	**104.13** %	**119.57** %
Class A1	−0.06 %	−0.26 %	−0.12 %	104.76 %	83.79 %
Class A2	−0.22 %	−0.42 %	−0.36 %	103.91 %	78.18 %
Class B	−0.14 %	−0.18 %	−0.39 %	105.64 %	95.89 %
Class C	−0.82 %	−1.19 %	−1.00 %	100.21 %	123.25 %
Class D	−0.48 %	−0.61 %	−1.00 %	102.76 %	218.70 %
Class F	−0.50 %	−0.27 %	0.03 %	98.75 %	98.01 %
JVET Overall	**−0.32** %	**−0.51** %	**−0.49** %	**103.65** %	**95.81** %

Table A.27 Coding results for Experiment 5.21 with filter length $d_\mathrm{m} = 1$. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC with only the first 65 pictures coded. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$.

	BD-rate change			Relative complexity	
Sequence / Class	Y	U	V	EncT	DecT
Tango2	−0.28 %	−1.22 %	−0.35 %	105.92 %	97.19 %
FoodMarket4	−0.12 %	−0.14 %	−0.18 %	105.22 %	75.35 %
Campfire	−0.09 %	−0.07 %	−0.22 %	103.09 %	76.27 %
CatRobot1	−0.42 %	−0.42 %	−0.56 %	105.10 %	73.44 %
DaylightRoad2	−0.33 %	−0.88 %	−0.58 %	104.48 %	72.86 %
ParkRunning3	−0.26 %	−0.28 %	−0.29 %	105.12 %	85.93 %
MarketPlace	−0.34 %	0.76 %	−0.60 %	115.79 %	98.25 %
RitualDance	−0.16 %	−0.19 %	−0.55 %	95.08 %	80.72 %
Cactus	−0.42 %	−0.34 %	−0.55 %	106.91 %	101.46 %
BasketballDrive	−0.21 %	−1.15 %	−0.30 %	102.92 %	102.67 %
BQTerrace	−0.27 %	−0.40 %	−0.16 %	93.41 %	100.87 %
BasketballDrill	−0.74 %	−0.98 %	−0.90 %	102.45 %	135.24 %
BQMall	−1.83 %	−3.13 %	−2.69 %	104.06 %	125.41 %
PartyScene	−0.40 %	−1.04 %	−0.45 %	96.34 %	119.53 %
RaceHorsesL	−1.57 %	−2.00 %	−2.13 %	97.57 %	119.58 %
BasketballPass	−0.32 %	0.56 %	−2.12 %	94.60 %	223.38 %
BQSquare	−0.08 %	0.21 %	0.72 %	95.02 %	241.93 %
BlowingBubbles	−0.41 %	−0.54 %	−1.08 %	91.95 %	238.25 %
RaceHorsesM	−1.40 %	−2.33 %	−1.38 %	121.95 %	208.25 %
BasketballDrillText	−0.83 %	−0.90 %	−0.43 %	111.89 %	130.87 %
ArenaOfValor	−0.79 %	−1.20 %	−0.79 %	109.24 %	106.82 %
SlideEditing	−0.15 %	−0.25 %	−0.19 %	81.71 %	82.46 %
SlideShow	0.26 %	0.65 %	0.51 %	101.37 %	81.42 %
Mean	**−0.49** %	**−0.66** %	**−0.66** %	**102.23** %	**120.79** %
Class A1	−0.17 %	−0.48 %	−0.25 %	104.72 %	82.31 %
Class A2	−0.34 %	−0.53 %	−0.47 %	104.89 %	76.87 %
Class B	−0.28 %	−0.26 %	−0.43 %	98.59 %	96.36 %
Class C	−1.13 %	−1.79 %	−1.54 %	96.72 %	124.23 %
Class D	−0.56 %	−0.52 %	−0.97 %	95.09 %	225.82 %
Class F	−0.38 %	−0.42 %	−0.23 %	98.67 %	98.17 %
JVET Overall	**−0.50** %	**−0.76** %	**−0.70** %	**100.52** %	**95.50** %

Table A.28 Coding results for Experiment 5.22 with filter length $d_\mathrm{m} = 3$. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC with only the first 65 pictures coded. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$.

	BD-rate change			Relative complexity	
Sequence / Class	Y	U	V	EncT	DecT
Tango2	−0.39 %	−1.03 %	−0.40 %	68.04 %	94.89 %
FoodMarket4	−0.18 %	−0.10 %	−0.35 %	64.41 %	76.59 %
Campfire	−0.03 %	−0.05 %	−0.13 %	102.43 %	79.42 %
CatRobot1	−0.41 %	−0.86 %	−0.72 %	104.43 %	77.57 %
DaylightRoad2	−0.37 %	−0.89 %	−0.39 %	67.10 %	74.78 %
ParkRunning3	−0.29 %	−0.28 %	−0.21 %	70.71 %	87.61 %
MarketPlace	−0.38 %	0.32 %	−0.85 %	113.56 %	100.71 %
RitualDance	−0.18 %	−0.50 %	−0.92 %	106.11 %	79.76 %
Cactus	−0.40 %	−0.85 %	−0.93 %	108.59 %	100.93 %
BasketballDrive	−0.20 %	−0.41 %	−0.16 %	126.96 %	105.48 %
BQTerrace	−0.29 %	−0.98 %	0.10 %	112.85 %	95.07 %
BasketballDrill	−0.83 %	−0.74 %	−0.42 %	86.15 %	140.10 %
BQMall	−1.64 %	−2.76 %	−2.26 %	90.51 %	121.90 %
PartyScene	−0.34 %	−0.73 %	−0.78 %	103.02 %	115.09 %
RaceHorsesL	−1.23 %	−1.76 %	−1.39 %	102.27 %	114.80 %
BasketballPass	−0.27 %	0.20 %	−1.96 %	91.30 %	220.67 %
BQSquare	−0.07 %	−0.07 %	−0.02 %	99.62 %	250.54 %
BlowingBubbles	−0.20 %	−0.49 %	−0.59 %	118.75 %	235.09 %
RaceHorsesM	−1.00 %	−1.52 %	−1.72 %	114.02 %	205.27 %
BasketballDrillText	−0.80 %	−0.94 %	−0.08 %	113.89 %	131.20 %
ArenaOfValor	−0.53 %	−0.75 %	−0.58 %	90.61 %	105.78 %
SlideEditing	−0.11 %	−0.10 %	−0.11 %	86.07 %	82.69 %
SlideShow	0.25 %	0.62 %	0.83 %	92.58 %	82.50 %
Mean	**−0.43** %	**−0.64** %	**−0.61** %	**97.13** %	**120.80** %
Class A1	−0.20 %	−0.39 %	−0.30 %	75.66 %	83.20 %
Class A2	−0.36 %	−0.67 %	−0.44 %	79.02 %	79.39 %
Class B	−0.29 %	−0.48 %	−0.55 %	108.46 %	95.86 %
Class C	−1.01 %	−1.50 %	−1.21 %	90.96 %	122.34 %
Class D	−0.38 %	−0.47 %	−1.07 %	99.22 %	225.17 %
Class F	−0.30 %	−0.29 %	0.02 %	91.74 %	98.40 %
JVET Overall	**−0.48** %	**−0.77** %	**−0.65** %	**90.38** %	**95.77** %

Table A.29 Coding results for Experiment 5.23 with filter length $d_\mathrm{m} = 5$. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC with only the first 65 pictures coded. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$.

	BD-rate change			Relative complexity	
Sequence / Class	Y	U	V	EncT	DecT
Tango2	−0.48 %	−1.35 %	−0.89 %	69.56 %	157.41 %
FoodMarket4	−0.14 %	−0.25 %	−0.29 %	65.26 %	115.98 %
Campfire	0.00 %	−0.06 %	−0.18 %	75.07 %	110.84 %
CatRobot1	−0.36 %	−0.43 %	−0.58 %	64.48 %	101.50 %
DaylightRoad2	−0.30 %	−0.79 %	−0.25 %	65.71 %	116.13 %
ParkRunning3	−0.24 %	−0.22 %	−0.20 %	72.07 %	140.11 %
MarketPlace	−0.34 %	0.30 %	−0.13 %	119.47 %	101.80 %
RitualDance	−0.21 %	−0.64 %	−0.91 %	97.16 %	81.99 %
Cactus	−0.42 %	−0.47 %	−0.72 %	108.38 %	102.89 %
BasketballDrive	−0.09 %	−0.51 %	−0.22 %	109.73 %	107.61 %
BQTerrace	−0.34 %	−0.75 %	−0.42 %	103.42 %	102.25 %
BasketballDrill	−0.67 %	−0.52 %	−0.38 %	101.74 %	141.12 %
BQMall	−1.03 %	−1.54 %	−1.66 %	105.94 %	128.27 %
PartyScene	−0.22 %	−0.29 %	−0.52 %	98.90 %	115.56 %
RaceHorsesL	−0.79 %	−0.99 %	−1.03 %	98.31 %	126.59 %
BasketballPass	−0.22 %	1.07 %	−1.76 %	100.97 %	224.24 %
BQSquare	0.10 %	1.05 %	1.54 %	105.66 %	243.62 %
BlowingBubbles	−0.05 %	−0.95 %	−1.09 %	113.61 %	238.66 %
RaceHorsesM	−0.46 %	−1.32 %	−0.95 %	108.58 %	203.89 %
BasketballDrillText	−0.69 %	−0.37 %	−0.11 %	101.31 %	130.65 %
ArenaOfValor	−0.35 %	−0.50 %	−0.41 %	115.65 %	107.96 %
SlideEditing	−0.05 %	−0.05 %	0.02 %	86.19 %	82.55 %
SlideShow	0.25 %	0.56 %	0.68 %	90.54 %	84.84 %
Mean	**−0.31** %	**−0.39** %	**−0.45** %	**94.68** %	**133.32** %
Class A1	−0.20 %	−0.55 %	−0.45 %	68.67 %	125.86 %
Class A2	−0.30 %	−0.48 %	−0.34 %	66.26 %	117.07 %
Class B	−0.28 %	−0.41 %	−0.48 %	104.00 %	98.79 %
Class C	−0.68 %	−0.83 %	−0.90 %	97.23 %	127.19 %
Class D	−0.16 %	−0.04 %	−0.56 %	103.41 %	224.83 %
Class F	−0.21 %	−0.09 %	0.04 %	94.13 %	99.53 %
JVET Overall	**−0.37** %	**−0.57** %	**−0.56** %	**85.91** %	**114.75** %

Table A.30 Coding results for Experiment 5.24 with filter length $d_\mathrm{m} = 7$. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC with only the first 65 pictures coded. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$.

Table A.31 Memory bandwidth measurements for VTM-3.2 (Anchor) and VTM-3.2+GEO, showing the average and worst-case bandwidth requirements in MB and the percental change.

		Anchor Bw. [MB]		GEO Bw. [MB]		Change [%]	
Sequence	QP	Mean	Max	Mean	Max	Mean	Max
Tango2	22	101.85	131.00	107.58	139.70	5.62	6.64
	27	92.48	116.50	95.28	120.30	3.03	3.26
	32	84.98	108.10	87.75	111.30	3.25	2.96
	37	78.31	101.20	80.44	102.40	2.73	1.19
FoodMarket4	22	85.60	157.00	89.65	164.20	4.74	4.59
	27	74.80	109.30	76.68	116.30	2.51	6.40
	32	69.42	104.10	71.30	104.50	2.72	0.38
	37	65.86	97.50	67.21	105.00	2.04	7.69
Campfire	22	91.37	138.40	100.72	156.00	10.23	12.72
	27	57.77	90.40	60.38	95.80	4.51	5.97
	32	55.38	72.40	57.28	75.30	3.42	4.01
	37	52.64	69.30	55.00	74.80	4.50	7.94
CatRobot1	22	128.72	178.90	134.09	190.70	4.17	6.60
	27	111.78	131.70	114.65	134.00	2.57	1.75
	32	101.17	119.10	102.91	124.70	1.73	4.70
	37	92.89	111.00	93.96	115.40	1.16	3.96
DaylightRoad2	22	96.20	142.60	99.43	145.80	3.35	2.24
	27	82.07	99.80	83.59	99.60	1.85	-0.20
	32	72.93	90.90	74.06	94.70	1.55	4.18
	37	67.15	88.30	68.54	91.00	2.07	3.06
ParkRunning3	22	130.60	155.00	138.41	165.30	5.98	6.65
	27	109.48	128.10	113.53	136.00	3.70	6.17
	32	95.52	114.10	98.55	120.50	3.17	5.61
	37	82.83	101.10	85.40	102.30	3.10	1.19
MarketPlace	22	26.44	46.40	27.44	54.90	3.75	**18.32**
	27	21.78	43.10	22.27	49.30	2.29	14.39
	32	19.26	37.50	19.60	40.00	1.75	6.67
	37	17.24	30.70	17.55	34.90	1.80	13.68
RitualDance	22	22.17	31.30	22.87	34.20	3.18	9.27
	27	19.52	30.80	20.06	30.20	2.81	-1.95
	32	17.82	27.00	18.29	28.20	2.65	4.44
	37	16.32	23.80	16.77	26.10	2.73	9.66
Cactus	22	29.54	42.60	31.41	47.70	6.33	11.97
	27	23.65	29.10	24.49	29.80	3.56	2.41
	32	21.53	27.30	22.20	27.30	3.10	0.00

	37	19.57	26.30	20.20	26.70	3.21	1.52
BasketballDrive	22	28.34	46.30	29.39	50.60	3.71	9.29
	27	23.33	38.60	23.95	39.10	2.67	1.30
	32	20.85	31.20	21.35	30.70	2.37	-1.60
	37	19.18	24.90	19.64	25.30	2.41	1.61
BQTerrace	22	44.05	85.60	49.40	95.20	**12.14**	11.21
	27	24.46	48.30	25.75	51.30	5.28	6.21
	32	18.94	30.40	19.50	33.50	2.97	10.20
	37	16.57	22.40	16.89	23.80	1.96	6.25
BasketballDrill	22	5.34	6.70	5.65	7.50	5.80	11.94
	27	4.69	5.90	4.97	6.30	5.89	6.78
	32	4.29	5.50	4.55	6.00	6.03	9.09
	37	4.00	5.50	4.22	6.10	5.43	10.91
BQMall	22	6.56	10.10	6.72	10.20	2.35	0.99
	27	5.63	7.90	5.80	7.90	2.97	0.00
	32	5.17	7.40	5.34	8.00	3.31	8.11
	37	4.77	7.40	4.98	7.30	4.38	-1.35
PartyScene	22	8.12	12.30	8.31	12.40	2.29	0.81
	27	6.50	10.20	6.65	9.90	2.31	-2.94
	32	5.55	7.90	5.65	8.00	1.73	1.27
	37	4.95	7.70	5.05	7.70	2.04	0.00
RaceHorsesL	22	6.84	11.50	7.17	12.80	4.89	11.30
	27	5.76	8.90	6.03	10.00	4.71	12.36
	32	4.94	6.70	5.21	7.20	5.31	7.46
	37	4.43	6.40	4.63	6.40	4.33	0.00
BasketballPass	22	1.67	2.60	1.71	2.70	2.13	3.85
	27	1.50	2.20	1.56	2.30	4.32	4.55
	32	1.39	2.20	1.42	2.30	2.77	4.55
	37	1.27	2.20	1.31	2.10	2.89	-4.55
BQSquare	22	2.29	3.40	2.33	3.50	1.64	2.94
	27	1.65	2.80	1.66	2.90	0.70	3.57
	32	1.28	2.40	1.28	2.40	0.27	0.00
	37	1.17	2.50	1.15	2.60	**-1.91**	4.00
BlowingBubbles	22	2.14	3.50	2.18	3.70	1.91	5.71
	27	1.76	3.00	1.79	3.10	1.55	3.33
	32	1.51	2.50	1.55	2.90	3.00	16.00
	37	1.31	2.60	1.35	2.80	2.79	7.69
RaceHorsesM	22	2.07	3.00	2.13	3.00	3.15	0.00
	27	1.74	2.30	1.79	2.50	2.98	8.70
	32	1.53	2.30	1.58	2.30	3.22	0.00

	37	1.34	2.00	1.38	2.10	2.97	5.00
BasketballDrillText	22	5.28	6.60	5.58	7.50	5.68	13.64
	27	4.59	5.70	4.85	6.40	5.59	12.28
	32	4.15	5.60	4.42	5.60	6.46	0.00
	37	3.89	5.40	4.13	6.00	6.35	11.11
ArenaOfValor	22	24.24	32.10	25.12	33.20	3.59	3.43
	27	20.48	27.60	21.19	28.60	3.47	3.62
	32	18.37	25.00	19.03	27.90	3.55	11.60
	37	16.52	23.60	17.15	25.90	3.82	9.75
SlideEditing	22	4.82	7.90	4.81	8.80	-0.15	11.39
	27	5.01	10.60	5.05	8.40	0.83	**-20.75**
	32	5.14	9.50	5.18	8.60	0.71	-9.47
	37	5.38	10.60	5.40	9.70	0.51	-8.49
SlideShow	22	6.23	11.20	6.30	11.80	1.11	5.36
	27	6.13	10.60	6.18	10.80	0.78	1.89
	32	5.90	10.30	5.96	10.70	0.90	3.88
	37	5.76	9.50	5.78	10.90	0.41	14.74
Overall	22					**4.24**	**18.32**
	27					**3.08**	**14.39**
	32					**2.87**	**16.00**
	37					**2.68**	**14.74**

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.44 %	−0.98 %	−1.13 %	375.87 %	171.48 %	−0.28 %
FoodMarket4	−0.36 %	−0.49 %	−0.46 %	319.67 %	149.84 %	−0.22 %
Campfire	−0.12 %	−0.11 %	−0.68 %	304.41 %	169.49 %	−0.24 %
CatRobot1	−0.62 %	−1.04 %	−1.13 %	288.42 %	128.04 %	−0.46 %
DaylightRoad2	−0.47 %	−1.26 %	−0.69 %	277.10 %	144.46 %	−0.14 %
ParkRunning3	−0.56 %	−0.52 %	−0.51 %	320.72 %	156.27 %	−0.45 %
MarketPlace	−0.36 %	−0.15 %	−0.77 %	249.15 %	135.63 %	−0.02 %
RitualDance	−0.25 %	−0.54 %	−0.83 %	318.56 %	140.02 %	−0.11 %
Cactus	−0.51 %	−0.72 %	−0.74 %	263.55 %	117.49 %	−0.38 %
BasketballDrive	−0.34 %	−0.95 %	−0.92 %	243.46 %	135.24 %	−0.17 %
BQTerrace	−0.76 %	−1.14 %	−0.44 %	275.65 %	138.56 %	−0.12 %
BasketballDrill	−0.66 %	−1.19 %	−1.29 %	249.04 %	83.49 %	−0.20 %
BQMall	−1.25 %	−1.61 %	−1.87 %	256.19 %	112.42 %	−0.60 %
PartyScene	−0.46 %	−0.60 %	−0.73 %	247.40 %	122.38 %	−0.12 %
RaceHorsesL	−1.17 %	−2.05 %	−1.91 %	242.33 %	93.03 %	−1.21 %
BasketballPass	−0.47 %	−1.71 %	−1.53 %	231.22 %	174.10 %	−0.12 %
BQSquare	−0.16 %	−0.02 %	0.14 %	295.80 %	158.72 %	0.05 %
BlowingBubbles	−0.55 %	−0.24 %	−0.58 %	287.84 %	133.49 %	−0.10 %
RaceHorsesM	−1.13 %	−2.04 %	−1.95 %	232.09 %	131.80 %	−0.47 %
BasketballDrillText	−0.65 %	−0.92 %	−1.06 %	244.96 %	123.04 %	−0.15 %
ArenaOfValor	−0.72 %	−1.11 %	−0.76 %	256.90 %	104.21 %	−0.71 %
SlideEditing	−0.06 %	−0.08 %	−0.05 %	304.85 %	116.83 %	−0.14 %
SlideShow	−0.36 %	−0.66 %	−0.54 %	259.06 %	133.72 %	−0.66 %
Mean	**−0.54** %	**−0.88** %	**−0.89** %	**275.84** %	**133.64** %	**−0.31** %
Class A1	−0.31 %	−0.52 %	−0.76 %	326.56 %	158.92 %	−0.25 %
Class A2	−0.55 %	−0.94 %	−0.78 %	292.61 %	136.42 %	−0.35 %
Class B	−0.45 %	−0.70 %	−0.74 %	267.53 %	129.83 %	−0.16 %
Class C	−0.89 %	−1.36 %	−1.45 %	247.17 %	99.96 %	−0.53 %
Class D	−0.58 %	−1.00 %	−0.98 %	258.33 %	144.21 %	−0.16 %
Class F	−0.45 %	−0.69 %	−0.60 %	264.54 %	116.88 %	−0.41 %
JVET Overall	**−0.56** %	**−0.89** %	**−0.94** %	**277.53** %	**127.34** %	**−0.31** %

Table A.32 Coding results for Experiment 5.26 using Merge-mode and MMVD motion vector coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, $C_{\mathrm{FL},\{7,8\}}$ explicit partition coding. Full encoder search of all possible MMVD+GEO combinations.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.22 %	0.04 %	−0.51 %	162.55 %	161.25 %	0.02 %
FoodMarket4	−0.15 %	−0.09 %	−0.21 %	134.98 %	155.34 %	−0.05 %
Campfire	−0.11 %	−0.06 %	−0.28 %	129.31 %	139.18 %	−0.32 %
CatRobot1	−0.29 %	−0.78 %	−0.67 %	116.92 %	116.57 %	−0.28 %
DaylightRoad2	−0.17 %	−0.51 %	−0.14 %	120.77 %	135.32 %	−0.23 %
ParkRunning3	−0.17 %	−0.11 %	−0.10 %	135.79 %	129.06 %	−0.24 %
MarketPlace	−0.09 %	0.25 %	−0.43 %	115.65 %	128.12 %	−0.08 %
RitualDance	−0.19 %	−0.30 %	−0.53 %	132.19 %	138.62 %	−0.24 %
Cactus	−0.26 %	−0.11 %	−0.32 %	107.86 %	116.62 %	−0.53 %
BasketballDrive	−0.15 %	−0.55 %	−0.39 %	114.81 %	130.27 %	−0.30 %
BQTerrace	−0.00 %	−0.30 %	−0.02 %	118.46 %	136.32 %	−0.06 %
BasketballDrill	−0.33 %	−0.88 %	−0.66 %	120.07 %	97.10 %	−0.63 %
BQMall	−0.96 %	−0.94 %	−0.85 %	106.59 %	96.56 %	−1.06 %
PartyScene	−0.24 %	−0.34 %	−0.27 %	114.04 %	127.59 %	−0.30 %
RaceHorsesL	−0.77 %	−1.08 %	−1.31 %	106.54 %	97.88 %	−1.11 %
BasketballPass	−0.26 %	−0.94 %	−0.74 %	100.33 %	153.36 %	−0.76 %
BQSquare	−0.02 %	−0.02 %	0.24 %	109.26 %	144.84 %	−0.20 %
BlowingBubbles	−0.12 %	−0.04 %	−0.10 %	116.50 %	156.33 %	−0.03 %
RaceHorsesM	−0.78 %	−1.49 %	−1.47 %	98.68 %	127.81 %	−0.70 %
BasketballDrillText	−0.40 %	−0.32 %	−0.50 %	103.47 %	129.15 %	−0.48 %
ArenaOfValor	−0.50 %	−0.50 %	−0.20 %	109.58 %	111.67 %	−0.81 %
SlideEditing	−0.03 %	−0.04 %	−0.04 %	99.69 %	121.09 %	0.02 %
SlideShow	0.03 %	−0.32 %	−0.14 %	104.10 %	124.96 %	−0.24 %
Mean	**−0.27** %	**−0.41** %	**−0.42** %	**116.44** %	**129.35** %	**−0.37** %
Class A1	−0.16 %	−0.04 %	−0.33 %	140.10 %	144.29 %	−0.12 %
Class A2	−0.21 %	−0.46 %	−0.30 %	123.83 %	122.49 %	−0.25 %
Class B	−0.14 %	−0.20 %	−0.34 %	117.07 %	125.67 %	−0.24 %
Class C	−0.58 %	−0.81 %	−0.77 %	111.22 %	102.35 %	−0.78 %
Class D	−0.29 %	−0.62 %	−0.52 %	105.19 %	138.57 %	−0.42 %
Class F	−0.23 %	−0.29 %	−0.22 %	104.01 %	119.90 %	−0.38 %
JVET Overall	**−0.27** %	**−0.38** %	**−0.45** %	**121.05** %	**121.68** %	**−0.36** %

Table A.33 Coding results for Experiment 5.25 using uni-prediction restriction and TPM motion vector coding. Test: VTM-3.2 + GEO, Reference: VTM-3.2, RA according to JVET CTC. AD: $\Delta\varphi = 11.25°$, $N_\rho = 4$, $C_{\text{FL},\{7,8\}}$ explicit partition coding.

```
Distortion getMaskedSAD8_SSE(short* pOrg, short* pPred, short* pMask, int
    width, int height, int strideOrg, int stridePred, int strideMask, int
    log2MaxWeight)
{
  __m128i vZero = _mm_setzero_si128();
  __m128i vSum  = vzero;
  for(int y = 0; y < height; y++)
  {
    for(int x = 0; x < width; x+=8)
    {
      __m128i vOrg  = _mm_loadu_si128((const __m128i*)&pOrg[x]);
      __m128i vPred = _mm_lddqu_si128((const __m128i*)&pPred[x]);
      __m128i vMask = _mm_lddqu_si128((const __m128i*)&pMask[x]);

      vSum = _mm_add_epi32(vSum, _mm_madd_epi16(vMask, _mm_abs_epi16(
          _mm_sub_epi16(vOrg, vPred))));
    }
    pOrg  += strideOrg;
    pPred += stridePred;
    pMask += strideMask;
  }
  vSum = _mm_hadd_epi32(vSum, vZero);
  vSum = _mm_hadd_epi32(vSum, vZero);
  unsigned uiSum = _mm_cvtsi128_si32(vSum);
  return uiSum >>= log2MaxWeight;
}
```

Listing A.1 Code example for SAD-based distortion metric using SIMD instructions for vectors of 8×16 bit samples.

```
Distortion getMaskedSAD16_SSE(short* pOrg, short* pPred, short* pMask,
    int width, int height, int strideOrg, int stridePred, int strideMask,
    int log2MaxWeight)
{
  __m256i vZero = _mm256_setzero_si256();
  __m256i vSum  = vZero;
  for(int y = 0; y < height; y++)
  {
    for(int x = 0; x < width; x+=16)
    {
      __m256i vsrc1 = _mm256_lddqu_si256((const __m256i*)&pOrg[x]));
      __m256i vsrc2 = _mm256_lddqu_si256((const __m256i*)&pPred[x]));
      __m256i vmask = _mm256_lddqu_si256((const __m256i*)&pMask[x]));

      vSum = _mm256_add_epi32(vSum, _mm256_madd_epi16(vMask,
          _mm256_abs_epi16(_mm256_sub_epi16(vOrg, vPred))));
    }
    pOrg   += strideOrg;
    pPred  += stridePred;
    pMask  += strideMask;
  }
  vSum = _mm256_hadd_epi32(vSum, vZero);
  vSum = _mm256_hadd_epi32( vsum32, vzero );
  uiSum = _mm_cvtsi128_si32(_mm256_castsi256_si128(vSum))+
      _mm_cvtsi128_si32(_mm256_castsi256_si128(_mm256_permute2x128_si256(
      vSum, vSum, 0x11)));
  return uiSum >>= log2MaxWeight;
}
```
Listing A.2 Code example for SAD-based distortion metric using SIMD instructions for vectors of 16×16 bit samples.

```
void blendingFilter4_SSE(const int16_t* pSrc0, int src0Stride, const
    int16_t* pSrc1, int src1Stride, const int16_t* pMask0, int mask0Stride
    , const int16_t* pMask1, int mask1Stride, int16_t *pDst, int dstStride
    , int width, int height, int shift, int offset)
{
  __m128i vZero   = _mm_setzero_si128();
  __m128i vOffset = _mm_set1_epi32(offset);
  for( int row = 0; row < height; row++ )
  {
    for( int col = 0; col < width; col += 4 )
    {
      __m128i vSum = _mm_loadl_epi64((const __m128i*)&pSrc0[col]);
      __m128i vDst = _mm_loadl_epi64((const __m128i*)&pSrc1[col]);
      __m128i vMask0 = _mm_loadl_epi64((const __m128i*)&pMask0[col]);
      __m128i vMask1 = _mm_loadl_epi64((const __m128i*)&pMask1[col]);
      vSum   = _mm_cvtepi16_epi32(vSum);
      vDst   = _mm_cvtepi16_epi32(vDst);
      vMask0 = _mm_cvtepi16_epi32(vMask0);
      vMask1 = _mm_cvtepi16_epi32(vMask1);
      vSum   = _mm_mullo_epi32(vSum, vMask0);
      vDst   = _mm_mullo_epi32(vDst, vMask1);
      vSum   = _mm_add_epi32(vSum, vDst);
      vSum   = _mm_add_epi32(vSum, vOffset);
      vSum   = _mm_srai_epi32(vSum, shift);
      vSum   = _mm_packs_epi32(vSum, vZero);
      _mm_storel_epi64((__m128i*)&pDst[col], vSum);
    }
    pSrc0  += src0Stride;
    pSrc1  += src1Stride;
    pMask0 += mask0Stride;
    pMask1 += mask1Stride;
    pDst   += dstStride;
  }
}
```

Listing A.3 Code example for SIMD-based blending filtering for vectors of 4×16 bit samples.

```
void blendingFilter8_SSE(const int16_t* pSrc0, int src0Stride, const
    int16_t* pSrc1, int src1Stride, const int16_t* pMask0, int mask0Stride
    , const int16_t* pMask1, int mask1Stride, int16_t *pDst, int dstStride
    , int width, int height, int shift, int offset)
{
  __m128i vZero   = _mm_setzero_si128();
  __m128i vOffset = _mm_set1_epi32(offset);
  for( int row = 0; row < height; row++ )
  {
    for( int col = 0; col < width; col += 8 )
    {
      __m128i vSrc0 = _mm_loadu_si128((const __m128i*)&pSrc0[col]);
      __m128i vSrc1 = _mm_loadu_si128((const __m128i*)&pSrc1[col]);
      __m128i vMask0 = _mm_loadu_si128((const __m128i*)&pMask0[col]);
      __m128i vMask1 = _mm_loadu_si128((const __m128i*)&pMask1[col]);
      __m128i vTmp, vSum, vDst, vM0, vM1;
      // first 64 bytes
      vSum = _mm_cvtepi16_epi32(vSrc0);
      vDst = _mm_cvtepi16_epi32(vsrc1);
      vM0  = _mm_cvtepi16_epi32(vMask0);
      vM1  = _mm_cvtepi16_epi32(vMask1);
      vSum = _mm_mullo_epi32(vSum, vM0);
      vDst = _mm_mullo_epi32(vDst, vM1);
      vSum = _mm_add_epi32(vSum, vDst);
      vSum = _mm_add_epi32(vSum, vOffset);
      vTmp = _mm_srai_epi32(vSum, shift);
      // next 64 bytes
      vSrc0  = _mm_unpackhi_epi64(vSrc0, vzero );
      vSrc1  = _mm_unpackhi_epi64(vSrc1, vzero );
      vMask0 = _mm_unpackhi_epi64(vMask0,vzero );
      vMask1 = _mm_unpackhi_epi64(vMask1,vzero );
      vSum   = _mm_cvtepi16_epi32(vSrc0);
      vDst   = _mm_cvtepi16_epi32(vSrc1);
      vM0    = _mm_cvtepi16_epi32(vMask0);
      vM1    = _mm_cvtepi16_epi32(vMask1);
      vSum   = _mm_mullo_epi32(vSum, vM0);
      vDst   = _mm_mullo_epi32(vDst, vM1);
      vSum   = _mm_add_epi32(vSum, vDst);
      vSum   = _mm_add_epi32(vSum, vOffset);
      vSum   = _mm_srai_epi32(vSum, shift);
      vSum   = _mm_packs_epi32(vTmp, vSum );
      _mm_storeu_si128((__m128i*)&pDst[col], vSum);
    }
    pSrc0  += src0Stride;
    pSrc1  += src1Stride;
    pMask0 += mask0Stride;
    pMask1 += mask1Stride;
    pDst   += dstStride;
  }
}
```

Listing A.4 Code example for SIMD-based blending filtering for vectors of 8×16 bit samples.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.13 %	−0.74 %	−0.13 %	105.22 %	100.26 %	0.10 %
FoodMarket4	−0.11 %	0.04 %	0.13 %	105.25 %	100.39 %	0.17 %
Campfire	−0.02 %	−0.02 %	−0.16 %	104.77 %	102.27 %	−0.03 %
CatRobot1	−0.20 %	−0.49 %	−0.58 %	105.84 %	100.17 %	−0.21 %
DaylightRoad2	−0.08 %	−0.22 %	−0.12 %	105.81 %	101.98 %	−0.13 %
ParkRunning3	−0.14 %	−0.15 %	−0.08 %	106.42 %	101.72 %	−0.16 %
MarketPlace	−0.03 %	0.27 %	−0.15 %	106.12 %	101.36 %	−0.02 %
RitualDance	−0.14 %	−0.32 %	−0.56 %	105.01 %	100.66 %	−0.06 %
Cactus	−0.20 %	−0.30 %	−0.35 %	106.07 %	102.91 %	−0.45 %
BasketballDrive	−0.04 %	−0.29 %	−0.03 %	105.10 %	100.69 %	−0.13 %
BQTerrace	0.02 %	0.31 %	0.06 %	106.65 %	101.12 %	0.29 %
BasketballDrill	−0.30 %	−0.76 %	−0.82 %	104.90 %	99.95 %	−0.60 %
BQMall	−1.05 %	−1.92 %	−1.96 %	106.01 %	101.40 %	−0.81 %
PartyScene	−0.17 %	−0.28 %	−0.31 %	105.61 %	101.03 %	−0.07 %
RaceHorses	−0.64 %	−1.41 %	−1.42 %	105.24 %	101.04 %	−0.91 %
BasketballPass	−0.17 %	−1.60 %	−0.98 %	104.98 %	101.36 %	−0.51 %
BQSquare	0.15 %	0.08 %	0.41 %	105.65 %	101.25 %	1.11 %
BlowingBubbles	−0.07 %	−0.37 %	−0.31 %	105.93 %	101.59 %	−0.25 %
RaceHorses	−0.61 %	−1.14 %	−0.50 %	105.33 %	100.17 %	−0.83 %
BasketballDrillText	−0.36 %	−0.59 %	−0.53 %	106.05 %	103.86 %	−0.50 %
ArenaOfValor	−0.39 %	−0.54 %	−0.32 %	102.76 %	100.31 %	−0.19 %
SlideEditing*	−0.07 %	−0.09 %	−0.05 %	101.42 %	100.90 %	−0.12 %
SlideShow*	−0.65 %	−0.40 %	−0.36 %	101.65 %	100.57 %	−0.89 %
FlyingGraphic*	−2.15 %	−1.60 %	−1.56 %	99.38 %	99.23 %	−1.94 %
Desktop*	−3.06 %	−2.70 %	−2.70 %	97.39 %	98.46 %	−2.98 %
Console*	−2.89 %	−2.38 %	−2.43 %	99.45 %	99.44 %	−2.90 %
ChineseEditing*	−0.51 %	−0.40 %	−0.33 %	100.59 %	101.61 %	−0.45 %
Mean	**−0.52** %	**−0.67** %	**−0.60** %	**104.24** %	**100.95** %	**−0.50** %
Class A1	−0.09 %	−0.24 %	−0.05 %	105.07 %	100.96 %	0.08 %
Class A2	−0.14 %	−0.29 %	−0.26 %	106.02 %	101.28 %	−0.17 %
Class B	−0.08 %	−0.06 %	−0.21 %	105.79 %	101.34 %	−0.07 %
Class C	−0.54 %	−1.09 %	−1.13 %	105.44 %	100.84 %	−0.60 %
Class D	−0.18 %	−0.76 %	−0.35 %	105.47 %	101.08 %	−0.12 %
Class F	−0.37 %	−0.41 %	−0.32 %	102.95 %	101.39 %	−0.43 %
Class TGM	−2.15 %	−1.77 %	−1.76 %	99.19 %	99.65 %	−2.07 %
JVET Overall	**−0.22** %	**−0.42** %	**−0.43** %	**105.60** %	**101.12** %	**−0.20** %

Table A.34 Coding results for Experiment 6.27. Test: VTM-5.0 + GEO, Reference: VTM-5.0, RA according to JVET CTC. Simplified GEO with 140 partitions per block, integer arithmetic and uni-prediction restriction. *Screen content sequences coded with hard masking.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
MarketPlace	−0.10 %	−0.21 %	0.10 %	107.75 %	103.12 %	−0.25 %
RitualDance	−0.31 %	−0.10 %	−0.52 %	107.32 %	102.84 %	−0.11 %
Cactus	−0.44 %	−0.33 %	−0.56 %	107.82 %	100.34 %	−0.35 %
BasketballDrive	−0.14 %	−0.84 %	−0.17 %	107.77 %	102.01 %	0.03 %
BQTerrace	−0.08 %	0.31 %	−0.43 %	107.88 %	102.23 %	−0.37 %
BasketballDrill	−0.61 %	−1.15 %	−0.63 %	108.57 %	101.72 %	−0.09 %
BQMall	−0.64 %	−1.54 %	−1.57 %	107.76 %	99.82 %	−0.64 %
PartyScene	−0.25 %	−0.30 %	−0.29 %	108.31 %	100.86 %	0.12 %
RaceHorses	−0.64 %	−1.48 %	−1.51 %	107.93 %	102.38 %	−0.57 %
BasketballPass	−0.35 %	−0.91 %	−0.81 %	107.69 %	102.14 %	−0.19 %
BQSquare	−0.08 %	−0.87 %	2.55 %	107.85 %	101.37 %	−0.55 %
BlowingBubbles	−0.45 %	−0.24 %	−0.65 %	108.09 %	98.16 %	−1.04 %
RaceHorses	−0.69 %	−1.34 %	−1.37 %	107.64 %	100.34 %	−0.31 %
FourPeople	−0.38 %	0.13 %	−0.06 %	109.02 %	104.59 %	−0.35 %
Johnny	−1.04 %	−1.33 %	−0.22 %	106.87 %	102.47 %	−1.59 %
KristenAndSara	−0.62 %	−0.18 %	−0.02 %	106.51 %	99.54 %	−0.58 %
BasketballDrillText	−0.70 %	−1.28 %	−0.47 %	106.44 %	98.67 %	−0.27 %
ArenaOfValor	−0.52 %	−1.11 %	−1.02 %	105.39 %	100.45 %	−0.45 %
SlideEditing*	0.05 %	0.08 %	0.12 %	105.93 %	104.03 %	−0.39 %
SlideShow*	0.05 %	0.63 %	−1.52 %	105.25 %	101.01 %	0.99 %
FlyingGraphic*	−2.34 %	−1.47 %	−1.75 %	99.45 %	100.19 %	−2.44 %
Desktop*	−4.23 %	−4.25 %	−4.21 %	99.02 %	103.67 %	−3.77 %
Console*	−3.06 %	−2.99 %	−2.82 %	99.06 %	100.08 %	−3.27 %
ChineseEditing*	−0.33 %	−0.34 %	−0.21 %	99.49 %	99.28 %	0.01 %
Mean	**−0.75** %	**−0.88** %	**−0.75** %	**106.03** %	**101.30** %	**−0.68** %
Class B	−0.22 %	−0.23 %	−0.32 %	107.70 %	102.03 %	−0.21 %
Class C	−0.54 %	−1.12 %	−1.00 %	108.13 %	101.14 %	−0.30 %
Class E	−0.68 %	−0.46 %	−0.10 %	107.43 %	102.05 %	−0.84 %
Class D	−0.39 %	−0.84 %	−0.07 %	107.81 %	100.46 %	−0.52 %
Class F	−0.28 %	−0.42 %	−0.72 %	105.75 %	101.00 %	−0.03 %
Class TGM	−2.49 %	−2.26 %	−2.25 %	99.23 %	100.73 %	−2.37 %
JVET Overall	**−0.44** %	**−0.59** %	**−0.49** %	**107.78** %	**101.74** %	**−0.40** %

Table A.35 Coding results for Experiment 6.27. Test: VTM-5.0 + GEO, Reference: VTM-5.0, LDB according to JVET CTC. Simplified GEO with 140 partitions per block, integer arithmetic and uni-prediction restriction. *Screen content sequences coded with hard masking.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.06 %	−0.12 %	0.17 %	187.64 %	108.96 %	−0.11 %
FoodMarket4	−0.02 %	−0.06 %	0.00 %	182.25 %	109.59 %	−0.08 %
Campfire	−0.03 %	0.00 %	0.03 %	165.87 %	135.03 %	0.06 %
CatRobot1	−0.02 %	0.03 %	0.00 %	170.18 %	122.43 %	−0.11 %
DaylightRoad2	−0.04 %	−0.13 %	−0.14 %	205.31 %	116.38 %	−0.20 %
ParkRunning3	0.00 %	0.00 %	0.00 %	177.54 %	125.73 %	−0.04 %
MarketPlace	−0.04 %	0.01 %	0.13 %	179.44 %	146.68 %	0.10 %
RitualDance	−0.06 %	0.04 %	−0.05 %	190.85 %	142.64 %	0.01 %
Cactus	−0.06 %	0.10 %	0.11 %	150.07 %	145.63 %	−0.01 %
BasketballDrive	−0.09 %	0.13 %	−0.05 %	151.16 %	168.37 %	−0.07 %
BQTerrace	−0.06 %	0.13 %	−0.09 %	188.39 %	140.48 %	−0.18 %
BasketballDrill	−0.05 %	0.13 %	0.09 %	165.36 %	144.68 %	0.14 %
BQMall	−0.07 %	0.12 %	−0.08 %	200.24 %	181.09 %	0.12 %
PartyScene	−0.01 %	0.04 %	−0.02 %	155.51 %	151.70 %	−0.12 %
RaceHorsesL	−0.04 %	0.02 %	−0.12 %	152.14 %	131.94 %	−0.09 %
BasketballPass	−0.06 %	−0.03 %	0.17 %	151.66 %	182.65 %	0.00 %
BQSquare	0.03 %	0.28 %	0.23 %	171.20 %	183.64 %	0.57 %
BlowingBubbles	−0.06 %	0.03 %	−0.03 %	154.53 %	196.73 %	0.31 %
RaceHorsesM	−0.02 %	0.06 %	0.04 %	147.20 %	148.98 %	−0.04 %
BasketballDrillText	−0.01 %	0.09 %	−0.09 %	157.21 %	162.65 %	−0.05 %
ArenaOfValor	−0.02 %	0.06 %	0.07 %	166.99 %	150.46 %	−0.08 %
SlideEditing	−0.08 %	−0.07 %	−0.07 %	115.95 %	101.66 %	0.04 %
SlideShow	−0.10 %	−0.01 %	−0.04 %	150.83 %	121.90 %	−0.06 %
Mean	**−0.04** %	**0.04** %	**0.01** %	**166.85** %	**144.35** %	**0.00** %
Class A1	−0.04 %	−0.06 %	0.07 %	176.99 %	115.26 %	−0.04 %
Class A2	−0.02 %	−0.03 %	−0.04 %	183.18 %	117.46 %	−0.12 %
Class B	−0.06 %	0.08 %	0.01 %	170.33 %	147.19 %	−0.03 %
Class C	−0.04 %	0.08 %	−0.03 %	166.81 %	149.99 %	0.01 %
Class D	−0.03 %	0.08 %	0.10 %	155.16 %	176.10 %	0.21 %
Class F	−0.05 %	0.02 %	−0.03 %	146.10 %	130.80 %	−0.04 %
JVET Overall	**−0.04** %	**0.03** %	**−0.00** %	**173.19** %	**134.66** %	**−0.04** %

Table A.36 Coding results for Experiment 7.28. Test: VTM-3.2 + GEO + SADCT, Reference: VTM-3.2 + GEO, RA according to JVET CTC. Optional SADCT coding of residual segment.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	−0.01 %	−0.05 %	0.03 %	152.28 %	105.12 %	−0.00 %
FoodMarket4	0.02 %	−0.01 %	0.12 %	148.31 %	121.31 %	−0.06 %
Campfire	−0.04 %	0.00 %	−0.04 %	147.04 %	131.49 %	−0.07 %
CatRobot1	0.02 %	−0.03 %	−0.04 %	137.07 %	130.43 %	0.03 %
DaylightRoad2	−0.04 %	−0.11 %	−0.08 %	162.18 %	113.55 %	−0.08 %
ParkRunning3	−0.00 %	−0.01 %	0.01 %	146.23 %	124.28 %	−0.07 %
MarketPlace	−0.00 %	−0.16 %	−0.08 %	146.27 %	122.30 %	0.14 %
RitualDance	−0.06 %	0.15 %	−0.04 %	167.38 %	151.32 %	−0.02 %
Cactus	−0.03 %	0.03 %	0.04 %	115.21 %	148.64 %	−0.04 %
BasketballDrive	−0.06 %	−0.04 %	−0.07 %	128.20 %	159.49 %	−0.18 %
BQTerrace	0.01 %	−0.07 %	−0.26 %	143.14 %	172.06 %	0.27 %
BasketballDrill	−0.00 %	0.03 %	0.08 %	134.60 %	167.91 %	−0.03 %
BQMall	−0.07 %	0.12 %	−0.32 %	146.22 %	165.78 %	−0.08 %
PartyScene	−0.03 %	0.02 %	−0.06 %	136.44 %	169.15 %	0.03 %
RaceHorsesL	0.01 %	0.01 %	−0.45 %	144.77 %	182.86 %	0.15 %
BasketballPass	−0.03 %	0.29 %	−0.01 %	130.31 %	192.51 %	0.21 %
BQSquare	−0.02 %	0.10 %	0.06 %	127.64 %	179.21 %	0.65 %
BlowingBubbles	−0.03 %	0.29 %	−0.02 %	121.01 %	163.63 %	0.14 %
RaceHorsesM	0.03 %	0.22 %	−0.28 %	138.47 %	155.36 %	0.06 %
BasketballDrillText	−0.04 %	0.10 %	0.06 %	136.72 %	146.40 %	−0.15 %
ArenaOfValor	−0.01 %	0.02 %	0.05 %	130.99 %	129.15 %	−0.08 %
SlideEditing	−0.04 %	−0.03 %	−0.04 %	90.55 %	97.21 %	0.02 %
SlideShow	−0.03 %	0.07 %	0.04 %	116.67 %	103.98 %	0.02 %
Mean	**−0.02** %	**0.04** %	**−0.06** %	**136.86** %	**144.92** %	**0.04** %
Class A1	−0.01 %	−0.02 %	0.04 %	147.12 %	117.32 %	−0.04 %
Class A2	−0.01 %	−0.05 %	−0.03 %	147.43 %	119.58 %	−0.04 %
Class B	−0.03 %	−0.02 %	−0.09 %	137.79 %	148.88 %	0.04 %
Class C	−0.02 %	0.04 %	−0.19 %	140.00 %	169.53 %	0.02 %
Class D	−0.01 %	0.23 %	−0.06 %	128.47 %	168.53 %	0.27 %
Class F	−0.03 %	0.04 %	0.03 %	116.95 %	117.05 %	−0.05 %
JVET Overall	**−0.02** %	**−0.01** %	**−0.08** %	**142.11** %	**140.66** %	**0.00** %

Table A.37 Coding results for Experiment 7.29. Test: VTM-3.2 + GEO + SADCT, Reference: VTM-3.2 + GEO, RA according to JVET CTC. Optional SADCT coding of transition zone.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	0.05 %	0.02 %	–0.06 %	115.04 %	108.08 %	0.05 %
FoodMarket4	0.00 %	0.01 %	–0.05 %	139.85 %	106.01 %	0.06 %
Campfire	–0.10 %	0.06 %	0.02 %	147.45 %	107.73 %	0.09 %
CatRobot1	0.03 %	–0.02 %	–0.02 %	148.82 %	99.85 %	0.06 %
DaylightRoad2	–0.04 %	–0.20 %	–0.12 %	159.11 %	99.27 %	–0.14 %
ParkRunning3	0.00 %	–0.01 %	0.00 %	140.97 %	93.97 %	–0.09 %
MarketPlace	0.02 %	–0.15 %	–0.26 %	151.07 %	122.89 %	0.17 %
RitualDance	–0.03 %	0.05 %	0.02 %	173.20 %	106.07 %	0.02 %
Cactus	–0.00 %	–0.04 %	0.06 %	127.16 %	135.21 %	0.02 %
BasketballDrive	–0.05 %	0.06 %	0.07 %	148.47 %	157.71 %	–0.21 %
BQTerrace	–0.35 %	0.21 %	0.08 %	159.52 %	145.64 %	–0.01 %
BasketballDrill	–0.11 %	0.11 %	0.07 %	144.63 %	183.64 %	0.07 %
BQMall	–0.05 %	0.12 %	0.04 %	165.52 %	160.19 %	0.27 %
PartyScene	–0.17 %	0.42 %	0.34 %	138.88 %	144.73 %	0.06 %
RaceHorsesL	0.00 %	–0.07 %	–0.06 %	159.26 %	170.08 %	0.14 %
BasketballPass	–0.03 %	0.13 %	0.02 %	129.89 %	159.66 %	0.22 %
BQSquare	–0.31 %	0.57 %	0.80 %	145.64 %	146.59 %	0.47 %
BlowingBubbles	–0.19 %	0.36 %	–0.05 %	133.62 %	182.36 %	0.01 %
RaceHorsesM	–0.02 %	0.06 %	0.19 %	146.33 %	155.75 %	0.09 %
BasketballDrillText	–0.08 %	0.27 %	0.12 %	151.46 %	169.09 %	–0.26 %
ArenaOfValor	–0.07 %	0.25 %	0.26 %	140.01 %	122.19 %	0.09 %
SlideEditing	–0.22 %	–0.10 %	–0.01 %	103.76 %	115.69 %	–0.02 %
SlideShow	–0.01 %	–0.04 %	–0.08 %	133.84 %	113.74 %	0.02 %
Mean	**–0.08** %	**0.09** %	**0.06** %	**143.63** %	**135.05** %	**0.05** %
Class A1	–0.01 %	0.03 %	–0.03 %	132.84 %	102.87 %	0.07 %
Class A2	–0.00 %	–0.07 %	–0.05 %	148.97 %	95.43 %	–0.06 %
Class B	–0.09 %	0.03 %	–0.00 %	149.90 %	131.26 %	–0.00 %
Class C	–0.08 %	0.15 %	0.10 %	151.27 %	162.37 %	0.14 %
Class D	–0.14 %	0.28 %	0.24 %	137.82 %	158.54 %	0.20 %
Class F	–0.10 %	0.10 %	0.07 %	130.79 %	127.60 %	–0.04 %
JVET Overall	**–0.05** %	**0.04** %	**0.01** %	**146.50** %	**124.14** %	**0.04** %

Table A.38 Coding results for Experiment 7.30. Test: VTM-3.2 + GEO + Transform Skip, Reference: VTM-3.2 + GEO, RA according to JVET CTC. Optional transform skip applied to entire GEO TB.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	0.05 %	0.07 %	−0.07 %	118.73 %	80.05 %	0.10 %
FoodMarket4	0.00 %	−0.00 %	−0.03 %	129.27 %	99.16 %	0.06 %
Campfire	−0.03 %	0.03 %	0.02 %	151.04 %	91.56 %	0.05 %
CatRobot1	0.02 %	0.01 %	−0.03 %	140.32 %	88.90 %	0.05 %
DaylightRoad2	−0.05 %	−0.32 %	−0.11 %	148.44 %	92.41 %	−0.01 %
ParkRunning3	−0.00 %	0.01 %	−0.00 %	133.39 %	82.76 %	−0.12 %
MarketPlace	0.04 %	−0.10 %	−0.19 %	166.97 %	96.82 %	0.12 %
RitualDance	0.01 %	−0.04 %	−0.03 %	160.74 %	91.74 %	0.04 %
Cactus	0.00 %	0.00 %	0.13 %	142.42 %	144.36 %	0.02 %
BasketballDrive	−0.03 %	−0.06 %	0.06 %	147.58 %	169.69 %	−0.14 %
BQTerrace	−0.24 %	0.06 %	−0.04 %	185.45 %	120.39 %	−0.01 %
BasketballDrill	−0.04 %	0.10 %	0.13 %	151.50 %	170.29 %	−0.08 %
BQMall	−0.02 %	0.34 %	0.16 %	177.30 %	149.99 %	0.05 %
PartyScene	−0.12 %	0.31 %	0.05 %	155.00 %	162.05 %	−0.06 %
RaceHorsesL	0.06 %	−0.19 %	−0.32 %	163.61 %	139.79 %	0.19 %
BasketballPass	−0.08 %	−0.01 %	−0.17 %	146.62 %	155.40 %	0.19 %
BQSquare	−0.16 %	0.25 %	0.81 %	185.84 %	191.53 %	0.45 %
BlowingBubbles	−0.15 %	0.24 %	0.36 %	145.10 %	187.47 %	−0.01 %
RaceHorsesM	−0.00 %	0.05 %	−0.14 %	149.38 %	187.89 %	−0.25 %
BasketballDrillText	−0.05 %	0.14 %	−0.00 %	143.34 %	172.20 %	−0.10 %
ArenaOfValor	−0.03 %	0.22 %	0.18 %	157.52 %	155.54 %	0.17 %
SlideEditing	−0.12 %	−0.05 %	−0.02 %	109.97 %	128.61 %	−0.07 %
SlideShow	0.05 %	0.04 %	0.03 %	148.04 %	103.62 %	−0.07 %
Mean	**−0.04** %	**0.05** %	**0.03** %	**150.33** %	**133.14** %	**0.02** %
Class A1	0.01 %	0.03 %	−0.03 %	131.82 %	88.79 %	0.07 %
Class A2	−0.01 %	−0.10 %	−0.05 %	140.11 %	86.95 %	−0.02 %
Class B	−0.05 %	−0.02 %	−0.01 %	158.15 %	119.93 %	0.01 %
Class C	−0.03 %	0.14 %	0.01 %	161.28 %	152.51 %	0.03 %
Class D	−0.10 %	0.13 %	0.22 %	155.12 %	178.64 %	0.10 %
Class F	−0.04 %	0.09 %	0.05 %	138.13 %	136.27 %	−0.02 %
JVET Overall	**−0.02** %	**0.01** %	**−0.02** %	**149.62** %	**112.91** %	**0.02** %

Table A.39 Coding results for Experiment 7.31. Test: VTM-3.2 + GEO + Transform Skip, Reference: VTM-3.2 + GEO, RA according to JVET CTC. Optional transform skip applied to signaled GEO residual segment.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	0.05 %	0.12 %	0.08 %	169.09 %	126.03 %	0.05 %
FoodMarket4	−0.02 %	−0.01 %	−0.02 %	144.25 %	103.07 %	−0.00 %
Campfire	−0.07 %	0.05 %	0.07 %	162.72 %	131.81 %	0.00 %
CatRobot1	0.03 %	−0.05 %	−0.03 %	152.40 %	113.41 %	0.04 %
DaylightRoad2	0.01 %	−0.06 %	−0.11 %	180.46 %	123.22 %	0.06 %
ParkRunning3	0.01 %	−0.01 %	0.01 %	160.49 %	129.56 %	−0.01 %
MarketPlace	0.00 %	−0.08 %	−0.13 %	162.49 %	140.66 %	0.15 %
RitualDance	−0.01 %	0.17 %	0.03 %	168.73 %	148.32 %	−0.05 %
Cactus	−0.02 %	−0.06 %	0.04 %	127.56 %	142.97 %	0.07 %
BasketballDrive	−0.03 %	−0.03 %	0.04 %	144.84 %	160.03 %	−0.08 %
BQTerrace	−0.38 %	0.32 %	0.17 %	170.19 %	148.41 %	−0.17 %
BasketballDrill	−0.05 %	0.20 %	0.09 %	150.53 %	167.20 %	−0.02 %
BQMall	−0.09 %	0.27 %	0.22 %	156.25 %	166.83 %	0.08 %
PartyScene	−0.16 %	0.14 %	0.17 %	144.47 %	166.75 %	0.01 %
RaceHorsesL	−0.06 %	−0.05 %	−0.16 %	152.70 %	147.38 %	0.01 %
BasketballPass	−0.12 %	0.17 %	0.06 %	127.91 %	161.78 %	0.28 %
BQSquare	−0.38 %	0.73 %	0.67 %	167.79 %	165.57 %	0.59 %
BlowingBubbles	−0.23 %	0.41 %	0.23 %	138.00 %	191.32 %	0.02 %
RaceHorsesM	−0.05 %	0.00 %	0.45 %	136.45 %	173.20 %	0.14 %
BasketballDrillText	−0.16 %	0.09 %	0.14 %	151.57 %	177.89 %	−0.18 %
ArenaOfValor	−0.08 %	0.25 %	0.24 %	142.59 %	123.58 %	0.00 %
SlideEditing	−0.14 %	−0.09 %	−0.07 %	107.99 %	101.86 %	−0.04 %
SlideShow	−0.05 %	0.04 %	0.06 %	132.99 %	98.43 %	−0.11 %
Mean	**−0.09** %	**0.11** %	**0.10** %	**150.11** %	**143.88** %	**0.04** %
Class A1	−0.01 %	0.05 %	0.04 %	157.31 %	115.99 %	0.02 %
Class A2	0.02 %	−0.04 %	−0.05 %	163.17 %	118.07 %	0.03 %
Class B	−0.09 %	0.06 %	0.03 %	152.58 %	147.43 %	−0.02 %
Class C	−0.09 %	0.14 %	0.08 %	150.33 %	160.36 %	0.02 %
Class D	−0.20 %	0.33 %	0.35 %	141.36 %	170.77 %	0.26 %
Class F	−0.11 %	0.07 %	0.09 %	132.48 %	121.34 %	−0.08 %
JVET Overall	**−0.05** %	**0.06** %	**0.03** %	**154.97** %	**137.47** %	**0.01** %

Table A.40 Coding results for Experiment 7.32. Test: VTM-3.2 + GEO + Transform Skip, Reference: VTM-3.2 + GEO, RA according to JVET CTC. Optional transform skip applied to signaled GEO residual segment and shifting to top-left corner of TB.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	0.07 %	0.00 %	−0.18 %	98.82 %	119.26 %	0.07 %
FoodMarket4	0.04 %	−0.02 %	0.04 %	88.72 %	125.07 %	0.09 %
Campfire	−0.01 %	−0.02 %	0.06 %	92.44 %	128.29 %	−0.08 %
CatRobot1	0.04 %	0.05 %	0.21 %	88.19 %	134.54 %	−0.04 %
DaylightRoad2	0.03 %	−0.06 %	−0.04 %	83.03 %	130.35 %	0.10 %
ParkRunning3	−0.00 %	−0.02 %	−0.01 %	95.21 %	115.10 %	−0.02 %
MarketPlace	−0.02 %	0.06 %	−0.06 %	91.30 %	149.46 %	−0.11 %
RitualDance	0.02 %	0.01 %	−0.18 %	98.96 %	143.60 %	−0.06 %
Cactus	0.04 %	−0.08 %	0.03 %	85.06 %	181.86 %	−0.10 %
BasketballDrive	−0.00 %	−0.05 %	0.00 %	85.97 %	195.68 %	−0.20 %
BQTerrace	0.02 %	−0.36 %	0.11 %	89.49 %	197.99 %	0.06 %
BasketballDrill	0.04 %	−0.08 %	−0.18 %	94.54 %	237.92 %	0.07 %
BQMall	−0.00 %	0.09 %	0.13 %	88.57 %	172.75 %	0.08 %
PartyScene	−0.02 %	−0.17 %	−0.08 %	92.13 %	128.84 %	0.01 %
RaceHorsesL	−0.03 %	−0.02 %	0.09 %	90.72 %	146.94 %	−0.19 %
BasketballPass	0.03 %	0.12 %	−0.17 %	82.83 %	205.13 %	0.07 %
BQSquare	−0.03 %	−0.14 %	0.05 %	82.47 %	211.54 %	0.10 %
BlowingBubbles	0.00 %	−0.09 %	−0.03 %	89.06 %	180.57 %	0.05 %
RaceHorsesM	0.02 %	−0.28 %	−0.38 %	80.87 %	159.00 %	0.35 %
BasketballDrillText	−0.05 %	−0.03 %	−0.15 %	84.98 %	204.26 %	0.10 %
ArenaOfValor	−0.02 %	−0.08 %	−0.04 %	85.60 %	172.58 %	−0.22 %
SlideEditing	−0.01 %	−0.04 %	−0.06 %	84.18 %	77.76 %	−0.13 %
SlideShow	0.04 %	−0.12 %	−0.00 %	80.97 %	99.15 %	0.02 %
Mean	**0.01** %	**−0.06** %	**−0.04** %	**88.44** %	**157.29** %	**0.00** %
Class A1	0.03 %	−0.01 %	−0.03 %	92.69 %	122.45 %	0.03 %
Class A2	0.02 %	−0.01 %	0.05 %	87.99 %	125.08 %	0.01 %
Class B	0.01 %	−0.08 %	−0.02 %	89.83 %	170.60 %	−0.08 %
Class C	−0.00 %	−0.05 %	−0.01 %	91.32 %	165.89 %	−0.01 %
Class D	0.01 %	−0.09 %	−0.13 %	83.39 %	186.83 %	0.14 %
Class F	−0.01 %	−0.07 %	−0.06 %	83.31 %	126.88 %	−0.06 %
JVET Overall	**0.01** %	**−0.04** %	**−0.01** %	**90.42** %	**148.93** %	**−0.02** %

Table A.41 Coding results for Experiment 7.33. Test: VTM-3.2 + TPM symmetric residual extension for all block sizes, Reference: VTM-3.2, RA according to JVET CTC.

	BD-rate change			Relative complexity		Visual
Sequence / Class	Y	U	V	EncT	DecT	VMAF
Tango2	0.01 %	−0.02 %	−0.07 %	108.89 %	123.46 %	0.05 %
FoodMarket4	−0.01 %	−0.09 %	0.03 %	95.43 %	145.94 %	0.05 %
Campfire	−0.00 %	0.02 %	0.02 %	101.08 %	145.86 %	−0.12 %
CatRobot1	−0.02 %	0.16 %	0.04 %	97.04 %	157.32 %	0.03 %
DaylightRoad2	0.04 %	−0.17 %	0.12 %	96.08 %	132.34 %	0.14 %
ParkRunning3	0.00 %	−0.01 %	−0.03 %	106.62 %	133.82 %	0.06 %
MarketPlace	−0.00 %	0.19 %	0.01 %	94.41 %	164.15 %	−0.02 %
RitualDance	0.03 %	0.08 %	−0.24 %	108.99 %	147.59 %	−0.04 %
Cactus	0.02 %	−0.04 %	0.12 %	93.32 %	238.55 %	−0.10 %
BasketballDrive	0.01 %	−0.04 %	0.03 %	96.85 %	194.13 %	−0.04 %
BQTerrace	0.05 %	−0.24 %	0.00 %	83.60 %	184.85 %	0.06 %
BasketballDrill	0.01 %	−0.11 %	0.00 %	94.48 %	188.73 %	0.12 %
BQMall	−0.03 %	0.22 %	0.22 %	78.03 %	178.02 %	0.14 %
PartyScene	−0.02 %	−0.25 %	−0.05 %	89.61 %	133.22 %	−0.05 %
RaceHorsesL	−0.00 %	−0.08 %	−0.25 %	87.58 %	110.31 %	−0.01 %
BasketballPass	0.02 %	0.05 %	−0.13 %	87.57 %	206.01 %	−0.06 %
BQSquare	−0.04 %	−0.17 %	0.09 %	74.89 %	217.69 %	0.41 %
BlowingBubbles	0.05 %	0.07 %	0.01 %	82.26 %	190.50 %	0.38 %
RaceHorsesM	0.01 %	−0.14 %	−0.28 %	77.67 %	152.03 %	0.06 %
BasketballDrillText	−0.03 %	0.21 %	0.06 %	101.95 %	225.82 %	0.02 %
ArenaOfValor	−0.03 %	−0.06 %	0.08 %	91.06 %	168.61 %	−0.21 %
SlideEditing	0.01 %	0.01 %	−0.01 %	76.92 %	73.53 %	−0.11 %
SlideShow	−0.03 %	−0.00 %	0.03 %	86.48 %	119.66 %	0.09 %
Mean	**0.00** %	**−0.02** %	**−0.01** %	**91.77** %	**162.27** %	**0.04** %
Class A1	−0.00 %	−0.03 %	−0.00 %	100.87 %	135.20 %	−0.01 %
Class A2	0.01 %	−0.00 %	0.04 %	99.11 %	137.81 %	0.07 %
Class B	0.02 %	−0.01 %	−0.01 %	94.73 %	181.83 %	−0.03 %
Class C	−0.01 %	−0.05 %	−0.02 %	86.75 %	148.23 %	0.05 %
Class D	0.01 %	−0.04 %	−0.08 %	80.02 %	186.49 %	0.20 %
Class F	−0.02 %	0.04 %	0.04 %	88.39 %	133.51 %	−0.05 %
JVET Overall	**0.01** %	**−0.02** %	**−0.00** %	**94.55** %	**153.53** %	**0.02** %

Table A.42 Coding results for Experiment 7.34. Test: VTM-3.2 + TPM symmetric residual extension for block sizes with $w \leq 16$ and $h \leq 16$, Reference: VTM-3.2, RA according to JVET CTC.

Bibliography

[04] *Information technology - Coding of audio-visual objects - Part 2: Visual*. Standard. Geneva, CH: International Organization for Standardization, June 2004 (cited on page 155).

[93] *Information technology - Coded representation of picture and audio information - Progressive bi-level image compression*. Standard. Geneva, CH: International Organization for Standardization, June 1993 (cited on page 34).

[Ahm+13a] A. Ahmmed, M. J. Alam, M. Pickering, R. Xu, A. T. Naman, and D. Taubman. "Motion hints based inter-frame prediction for hybrid video coding." In: *2013 Picture Coding Symposium (PCS)*. Dec. 2013, pages 177–180. DOI: 10.1109/PCS.2013.6737712 (cited on page 39).

[Ahm+13b] A. Ahmmed, R. Xu, A. T. Naman, M. J. Alam, M. Pickering, and D. Taubman. "Motion segmentation initialization strategies for bi-directional inter-frame prediction." In: *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*. Sept. 2013, pages 058–063. DOI: 10.1109/MMSP.2013.6659264 (cited on page 39).

[And+16] K. Andersson, P. Wennersten, J. Samuelsson, J. Ström, P. Hermansson, and M. Pettersson. *AHG 3: Recommended settings for HM*. Technical report JCTVC-X0038. Geneva, Switzerland, 24th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, May 2016 (cited on page 121).

[AS18a] Yongjo Ahn and Donggyu Sim. *CE10: Diagonal motion partitions with uni-prediction constraint (Test 10.3.3)*. Technical report JVET-L0125. Macau SR, China, 12th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Oct. 2018 (cited on page 116).

[AS18b] Yongjo Ahn and Donggyu Sim. *CE10-related: Diagonal motion partitions on top of MTT block structure*. Technical report JVET-K0270. Ljubljana, Slovenia, 11th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, July 2018 (cited on page 116).

[BCL18] Benjamin Bross, Jianle Chen, and Shan Liu. *Versatile Video Coding (Draft 5)*. Technical report JVET-N1001. Geneva, Switzerland, 14th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Mar. 2018 (cited on pages 17, 18, 28, 128, 131, 137, 150).

[BFT11] P. Bordes, E. Francois, and D. Thoreau. "Fast encoding algorithms for geometry-adaptive block partitioning." In: *2011 18th IEEE International Conference on Image Processing*. Sept. 2011, pages 1205–1208. DOI: 10.1109/ICIP.2011.6115647 (cited on page 37).

[BHW16] Max Bläser, Cordula Heithausen, and Mathias Wien. “Segmentation-based Partitioning for Motion Compensated Prediction in Video Coding.” In: *2016 Picture Coding Symposium*. Nuremberg, Germany, Dec. 2016 (cited on page 39).

[BHW17] Max Bläser, Cordula Heithausen, and Mathias Wien. “Geometry-Adaptive Motion Partitioning Using Improved Temporal Prediction.” In: *2017 Visual Communications And Image Processing*. St Petersburg, USA, Dec. 2017 (cited on page 84).

[Bjø01] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. Technical report VCEG-M33. Austin, USA: ITU-T SG16/Q6 VCEG, 2001 (cited on page 27).

[BK97] D. B. Bradshaw and N. G. Kingsbury. “Combined affine and translational motion compensation scheme using triangular tessellations.” In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Volume 4. Apr. 1997, 2645–2648 vol.4. DOI: 10.1109/ICASSP.1997.595332 (cited on page 38).

[Bla+17] S. Blasi, M. Naccari, R. Weerakkody, J. Funnell, and M. Mrak. “The Open-Source Turing Codec: Toward Fast, Flexible, and Parallel HEVC Encoding.” In: *SMPTE Motion Imaging Journal* 126.9 (Nov. 2017), pages 1–8. ISSN: 2160-2492. DOI: 10.5594/JMI.2017.2744578 (cited on page 139).

[BOA96] M. Bi, S. H. Ong, and Y. H. Ang. “Comment on "Shape-adaptive DCT for generic coding of video".” In: *IEEE Transactions on Circuits and Systems for Video Technology* 6.6 (Dec. 1996), pages 686–688. ISSN: 1558-2205. DOI: 10.1109/76.544740 (cited on page 159).

[Bor+11] P. Bordes, P. Chen, I.-K. Kim, L. Guo, H. Yu, and X. Zheng. *CE2: Unified solution of flexible motion partitioning*. Technical report JCTVC-E374. Geneva, Switzerland, 5th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Mar. 2011 (cited on page 37).

[Bos+18a] Frank Bossen, Jill Boyce, Karsten Suehring, Xiang Li, and Vadim Seregin. *JVET common test conditions and software reference configurations for SDR video*. Technical report JVET-L1010. Macao, SAR China, 12th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Oct. 2018 (cited on page 25).

[Bos+18b] Frank Bossen, Jill Boyce, Karsten Suehring, Xiang Li, and Vadim Seregin. *JVET common test conditions and software reference configurations for SDR video*. Technical report JVET-N1010. Geneva, Switzerland, 14th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Mar. 2018 (cited on page 25).

[BS18a] Max Bläser and Johannes Sauer. *CE10: Results on Geometric block partitioning (Test 3.3)*. Technical report JVET-K0146. Ljubljana, Slovenia, 11th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, July 2018 (cited on page 116).

[BS18b] Max Bläser and Johannes Sauer. *CE10: Results on Geometric Partitioning (Experiments 3.2.a - 3.2.c)*. Technical report JVET-L0417. Macau SR, China, 12th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Oct. 2018 (cited on page 116).

[BSW18] Max Bläser, Johannes Sauer, and Mathias Wien. *Description of SDR and 360° video coding technology proposal by RWTH Aachen University*. Technical report JVET-J0023. San Diego, USA, 10th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Apr. 2018 (cited on page 84).

[BYR07] Vladimir Britanak, Patrick C. Yip, and K.R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. 1st edition. Academic Press, 2007. ISBN: 9780123736246 (cited on page 10).

[CBM17] J. Carter, S. Blasi, and M. Mrak. "Complexity-Driven Rate-Control for Parallel HEVC Coding." In: *2017 IEEE Visual Communications and Image Processing (VCIP)*. Dec. 2017, pages 1–4. DOI: `10.1109/VCIP.2017.8305101` (cited on page 139).

[Che+07] Jianle Chen, SangRae Lee, Kyo-Hyuk Lee, and Woo-Jin Han. "Object Boundary Based Motion Partition For Video Coding." In: *Picture Coding Symposium (PCS)*. Nov. 2007 (cited on page 39).

[Che+10] P. Chen, W.-J. Chien, R. Panchal, and M. Karczewicz. *Geometry Motion Partition*. Technical report JCTVC-B049. Geneva, Switzerland, 2nd meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, July 2010 (cited on page 36).

[Che+14] Y. Chen, D. Mukherjee, J. Han, and K. Rose. "Joint inter-intra prediction based on mode-variant and edge-directed weighting approaches in video coding." In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2014, pages 7372–7376. DOI: `10.1109/ICASSP.2014.6855032` (cited on page 37).

[Che+17] J. Chen, E. Alshina, G. Sullivan, J.-R. Ohm, and J. Boyce. *Algorithm Description of Joint Exploration Test Model 7 (JEM 7)*. Technical report JVET-G1001. Torino, Italy, 7th meeting: Joint Video Exploration Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, July 2017 (cited on page 150).

[Che+18] Y. Chen et al. "An Overview of Core Coding Tools in the AV1 Video Codec." In: *2018 Picture Coding Symposium (PCS)*. June 2018, pages 41–45. DOI: `10.1109/PCS.2018.8456249` (cited on pages 17, 18, 38).

[Che+19a] C.-C. Chen, Y. Zhang, K. Reuze, Y.-J. Chang, W.-J. Chien, and M. Karczewicz. *Non-CE4/8: Blending-off Switch for TPM Mode*. Doc. JVET-O0645. Gothenburg, Sweden, 15th meeting: Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG, July 2019 (cited on page 136).

[Che+19b] J. Chen, M. Karczewicz, Y. Huang, K. Choi, J. Ohm, and G. J. Sullivan. "The Joint Exploration Model (JEM) for Video Compression with Capability beyond HEVC." In: *IEEE Transactions on Circuits and Systems for Video Technology* (2019), pages 1–1. ISSN: 1558-2205. DOI: `10.1109/TCSVT.2019.2945830` (cited on pages 28, 150).

[Chi+19] Wei-Jung Chien, Jill Boyce, Roman Chernyak, Kiho Choi Francois, Ryoji Hashimoto, Yu-Wen Huang, Shan Liu, and Daniel Luo. *JVET AHG report: Tool reporting procedure (AHG13)*. Technical report JVET-M0013. Marrakech, Morocco, 13th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Jan. 2019 (cited on page 116).

[Cla85] R. J. Clarke. *Transform Coding of Images*. Microelectronics and Signal Processing. Academic Press Limited, 1985. ISBN: 0121757307 (cited on pages 8, 150).

[CLC06] Tung-Chien Chen, Chung-Jr Lian, and Liang-Gee Chen. "Hardware Architecture Design of an H.264/AVC Video Codec." In: *Asia and South Pacific Conference on Design Automation*. Jan. 2006, 8 pp.-. DOI: 10.1109/ASPDAC.2006.1594776 (cited on page 69).

[CP93] Chi-Fa Chen and Khee K. Pang. "The optimal transform of motion-compensated frame difference images in a hybrid coder." In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 40.6 (June 1993), pages 393–397. ISSN: 1558-125X. DOI: 10.1109/82.277884 (cited on pages 149, 151).

[CYK19] Jianle Chen, Yan Ye, and Seung Hwan Kim. *Algorithm description for Versatile Video Coding and Test Model 3 (VTM 3)*. Technical report JVET-N1002. Geneva, Switzerland, 14th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Mar. 2019 (cited on pages 17, 18, 24, 28, 128, 150).

[CYX19] C.-C. Chen, H. Yang, and X. Xiu. *Description of Core Experiment 4 (CE4): Inter prediction*. Doc. JVET-O2024. Gothenburg, Sweden, 15th meeting: Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG, July 2019 (cited on page 128).

[Dai+07] C. Dai, O. D. Escoda, P. Yin, X. Li, and C. Gomila. "Geometry-Adaptive Block Partitioning for Intra Prediction in Image Video Coding." In: *2007 IEEE International Conference on Image Processing*. Volume 6. Sept. 2007, pages VI - 85-VI - 88. DOI: 10.1109/ICIP.2007.4379527 (cited on page 35).

[DeL+16] S. De-Luxán-Hernández, D. Marpe, H. Schwarz, K. Muller, M. Wien, J. Ohm, and T. Wiegand. "Block adaptive selection of multiple core transforms for video coding." In: *2016 Picture Coding Symposium (PCS)*. Dec. 2016, pages 1–5. DOI: 10.1109/PCS.2016.7906349 (cited on page 149).

[Din+13] J. Ding, Y. Huang, P. Lin, S. Pei, H. Chen, and Y. Wang. "Two-Dimensional Orthogonal DCT Expansion in Trapezoid and Triangular Blocks and Modified JPEG Image Compression." In: *IEEE Transactions on Image Processing* 22.9 (Sept. 2013), pages 3664–3675. ISSN: 1941-0042. DOI: 10.1109/TIP.2013.2268971 (cited on page 168).

[Doa+17] Nghia Doan, Tae Kim, Chae Eun Rhee, and Hyuk-Jae Lee. "A hardware-oriented concurrent TZ search algorithm for High-Efficiency Video Coding." In: *EURASIP Journal on Advances in Signal Processing* 78 (Nov. 2017). DOI: 10.1186/s13634-017-0513-9 (cited on page 113).

[DYG07] Oscar Divorra, Peng Yin, and Cristina Gomila. *Geometry-adaptive Block Partioning*. Technical report VCEG-AF10. San Jose, USA, 32nd meeting: ITU-T SG16/Q6 VCEG, Apr. 2007 (cited on page 35).

[Esc+07] O. Divorra Escoda, P. Yin, C. Dai, and X. Li. "Geometry-Adaptive Block Partitioning for Video Coding." In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Volume 1. Apr. 2007, pages I-657-I-660. DOI: 10.1109/ICASSP.2007.365993 (cited on pages 35, 48, 49).

[Ese+19a] Semih Esenlik, Han Gao, Alexey Filippov, Vasily Rufitskiy, Anand Meher Kotra, Biao Wang, Elena Alshina, Max Bläser, and Johannes Sauer. *Non-CE4: Geometrical partitioning for inter blocks*. Doc. JVET-O0489. Gothenburg, Sweden, 15th meeting: Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG, July 2019 (cited on page 128).

[Ese+19b] Semih Esenlik, Han Gao, Biao Wang, Anand Meher Kotra, Zhijie Zhao, Elena Alshina, Max Bläser, and Johannes Sauer. *Non-CE4: Adaptive blending filtering for TPM*. Doc. JVET-O0513. Gothenburg, Sweden, 15th meeting: Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG, July 2019 (cited on page 128).

[Fer+09] R. U. Ferreira, E. M. Hung, R. L. de Queiroz, and D. Mukherjee. "Efficiency improvements for a geometric-partition-based video coder." In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. Nov. 2009, pages 1009–1012. DOI: 10.1109/ICIP.2009.5413818 (cited on page 36).

[Fra+11] E. Francois, P. Bordes, L. Guo, and M. Karczewicz. *CE2: Simplified Geometry Block Partitioning*. Technical report JCTVC-D230. Daegu, Korea, 4th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Jan. 2011 (cited on page 36).

[Fu+12] C. Fu, E. Alshina, A. Alshin, Y. Huang, C. Chen, C. Tsai, C. Hsu, S. Lei, J. Park, and W. Han. "Sample Adaptive Offset in the HEVC Standard." In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pages 1755–1764. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2012.2221529 (cited on page 24).

[FWG98] M. Flierl, T. Wiegand, and B. Girod. "A locally optimal design algorithm for block-based multi-hypothesis motion-compensated prediction." In: *Proceedings DCC '98 Data Compression Conference (Cat. No.98TB100225)*. Mar. 1998, pages 239–248. DOI: 10.1109/DCC.1998.672152 (cited on page 107).

[FZC11] E. Francois, X. Zheng, and P. Chen. *CE2: Summary of Core Experiment 2 on Flexible Motion Partitioning*. Technical report JCTVC-D229. Daegu, Korea, 4th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Jan. 2011 (cited on page 36).

[Gao+19a] Han Gao, Semih Esenlik, Elena Alshina, Anand Meher Kotra, Biao Wang, Max Bläser, and Johannes Sauer. *CE4: CE4-1.1, CE4-1.2 and CE4-1.14: Geometric Merge Mode (GEO)*. Doc. JVET-P0068. Geneva, Switzerland, 16th meeting: Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG, Oct. 2019 (cited on pages 141, 146).

[Gao+19b] Han Gao, Semih Esenlik, Elena Alshina, Anand Meher Kotra, Biao Wang, Max Bläser, and Johannes Sauer. *CE4: CE4-1.7, CE4-1.8: GEO and TPM Blending Off for SCC*. Doc. JVET-P0069. Geneva, Switzerland, 16th meeting: Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG, Oct. 2019 (cited on page 146).

[Gao+19c] Han Gao, Semih Esenlik, Elena Alshina, Anand Meher Kotra, Biao Wang, Max Bläser, and Johannes Sauer. *CE4-Related: Geometric Merge Mode (GEO) Simplifications*. Doc. JVET-P0107. Geneva, Switzerland, 16th meeting: Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG, Oct. 2019 (cited on page 146).

[Gil90] M. Gilge. "Region-oriented transform coding (ROTC) of images." In: *International Conference on Acoustics, Speech, and Signal Processing*. Apr. 1990, 2245–2248 vol.4. DOI: 10.1109/ICASSP.1990.116016 (cited on pages 167, 170).

[Gir87] B. Girod. "The Efficiency of Motion-Compensating Prediction for Hybrid Coding of Video Sequences." In: *IEEE Journal on Selected Areas in Communications* 5.7 (Aug. 1987), pages 1140–1154. ISSN: 1558-0008. DOI: 10.1109/JSAC.1987.1146632 (cited on page 149).

[Gir93] B. Girod. "Motion-compensating prediction with fractional-pel accuracy." In: *IEEE Transactions on Communications* 41.4 (Apr. 1993), pages 604–612. ISSN: 1558-0857. DOI: 10.1109/26.223785 (cited on page 151).

[Gir98] B. Girod. "Why B-pictures work: a theory of multi-hypothesis motion-compensated prediction." In: *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*. Volume 2. Oct. 1998, 213–217 vol.2. DOI: 10.1109/ICIP.1998.723350 (cited on pages 107, 112).

[Goy01] V. K. Goyal. "Theoretical foundations of transform coding." In: *IEEE Signal Processing Magazine* 18.5 (Sept. 2001), pages 9–21. ISSN: 1558-0792. DOI: 10.1109/79.952802 (cited on page 149).

[Guo+10] L. Guo, P. Yin, Y. Zheng, X. Lu, Q. Xu, and J. Sole. "Simplified geometry-adaptive block partitioning for video coding." In: *2010 IEEE International Conference on Image Processing*. Sept. 2010, pages 965–968. DOI: 10.1109/ICIP.2010.5649913 (cited on page 36).

[GW08] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Upper Saddle River, N.J.: Prentice Hall, 2008. ISBN: 9780131687288 013168728X 9780135052679 013505267X (cited on page 5).

[GYF10] L. Guo, P. Yin, and E. Francois. *TE3: Simplified geometry block partitioning*. Technical report JCTVC-B085. Geneva, Switzerland, 2nd meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, July 2010 (cited on page 36).

[HDM06] E. M. Hung, R. L. De Queiroz, and D. Mukherjee. "On Macroblock Partition for Motion Compensation." In: *2006 International Conference on Image Processing*. Oct. 2006, pages 1697–1700. DOI: 10.1109/ICIP.2006.312686 (cited on page 35).

[HLC97] P. Hsu, K. J. R. Liu, and T. Chen. “2-D mesh motion compensation with adaptive interpolation.” In: *Proceedings of First Signal Processing Society Workshop on Multimedia Signal Processing*. June 1997, pages 213–218. DOI: 10.1109/MMSP.1997.602638 (cited on page 38).

[HM18a] Ryoji Hashimoto and Seiji Mochizuki. *AHG5: How to use the software to evaluate memory bandwidth*. Technical report JVET-K0451. Ljubljana, Slovenia, 11th meeting: Joint Video Experts Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, July 2018 (cited on page 110).

[HM18b] Ryoji Hashimoto and Seiji Mochizuki. *AHG5: Measurement result of memory bandwidth with anchor streams*. Technical report JVET-I0033. Gwangju, Korea, 9th meeting: Joint Video Exploration Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Jan. 2018 (cited on page 110).

[HMI17] Ryoji Hashimoto, Seiji Mochizuki, and Tomohiro Ikai. *AHG5: How to measure memory bandwidth*. Technical report JVET-H0043. Macao, China, 8th meeting: Joint Video Exploration Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, Oct. 2017 (cited on page 109).

[HS07] K. Hui and W. Siu. “Extended Analysis of Motion-Compensated Frame Difference for Block-Based Motion Prediction Error.” In: *IEEE Transactions on Image Processing* 16.5 (May 2007), pages 1232–1245. ISSN: 1941-0042. DOI: 10.1109/TIP.2007.894263 (cited on page 151).

[HSR10] J. Han, A. Saxena, and K. Rose. “Towards jointly optimal spatial prediction and adaptive transform in video/image coding.” In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. Mar. 2010, pages 726–729. DOI: 10.1109/ICASSP.2010.5495043 (cited on page 150).

[HWZ11] H. Huang, J. W. Woods, and Y. Zhao. “Motion compensated prediction using partial mesh generation.” In: *2011 18th IEEE International Conference on Image Processing*. Sept. 2011, pages 1677–1680. DOI: 10.1109/ICIP.2011.6115778 (cited on page 38).

[Int19] Intel. *Intel Intrinsics Guide*. 2019. URL: https://software.intel.com/sites/landingpage/IntrinsicsGuide/ (visited on 10/30/2019) (cited on page 114).

[Jäg13] F. Jäger. “Depth-based block partitioning for 3D video coding.” In: *2013 Picture Coding Symposium (PCS)*. Dec. 2013, pages 410–413. DOI: 10.1109/PCS.2013.6737770 (cited on page 39).

[Jia+16] M. Jiang, M. Kalluri, N. Ling, J. Zheng, and P. Zhang. “An approach to image compression using R-D optimal OMP selection.” In: *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. May 2016, pages 2230–2233. DOI: 10.1109/ISCAS.2016.7539026 (cited on page 172).

[Joi20] Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG. *VVC Test Model (VTM)*. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM. 2018–2020 (cited on page 25).

[Jos+19] Urvang Joshi, Debargha Mukherjee, Debargha Chen, Sarah Parker, and Adrian Grange. “In-loop Frame Super-resolution in AV1.” In: *2019 Picture Coding Symposium (PCS)*. Nov. 2019 (cited on page 18).

[Kan+10] M. Kang, C. Lee, J. Y. Lee, and Y. Ho. “Adaptive geometry-based intra prediction for depth video coding.” In: *2010 IEEE International Conference on Multimedia and Expo*. July 2010, pages 1230–1235. DOI: 10.1109/ICME.2010.5583876 (cited on page 37).

[Kar+10a] M. Karczewicz, P. Chen, R. L. Joshi, X. Wang, W. J. Chien, R. Panchal, Y. Reznik, M. Coban, and I. S. Chong. “A Hybrid Video Coder Based on Extended Macroblock Sizes, Improved Interpolation, and Flexible Motion Representation.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 20.12 (Dec. 2010), pages 1698–1708. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2010.2092614 (cited on page 36).

[Kar+10b] M. Karczewicz, P. Chen, R. Joshi, K. Wang, and W.-J. Chien. *Video coding technology proposal by Qualcomm*. Technical report JCTVC-A121. Dresden, Germany, 1st meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Apr. 2010 (cited on page 36).

[Kat+98] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann, and G. M. Schuster. “MPEG-4 and rate-distortion-based shape-coding techniques.” In: *Proceedings of the IEEE* 86.6 (June 1998), pages 1126–1154. ISSN: 1558-2256. DOI: 10.1109/5.687833 (cited on page 34).

[KH12] M. Kang and Y. Ho. “Depth Video Coding Using Adaptive Geometry Based Intra Prediction for 3-D Video Systems.” In: *IEEE Transactions on Multimedia* 14.1 (Feb. 2012), pages 121–128. ISSN: 1520-9210. DOI: 10.1109/TMM.2011.2169238 (cited on page 37).

[Kim+08] J. H. Kim, A. Ortega, P. Yin, P. Pandit, and C. Gomila. “Motion compensation based on implicit block segmentation.” In: *2008 15th IEEE International Conference on Image Processing*. Oct. 2008, pages 2452–2455. DOI: 10.1109/ICIP.2008.4712289 (cited on page 39).

[KMO10] S. Klomp, M. Munderloh, and J. Ostermann. “Block size dependent error model for motion compensation.” In: *2010 IEEE International Conference on Image Processing*. Sept. 2010, pages 969–972. DOI: 10.1109/ICIP.2010.5649414 (cited on page 149).

[Kru+10] A. Krutz et al. *Tool Experiment 3: Inter Prediction in HEVC*. Technical report JCTVC-A303. Dresden, Germany, 1st meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Apr. 2010 (cited on page 36).

[KS05] S. Kondo and H. Sasai. “A motion compensation technique using sliced blocks in hybrid video coding.” In: *IEEE International Conference on Image Processing 2005*. Volume 2. Sept. 2005, pages II-305-8. DOI: 10.1109/ICIP.2005.1530052 (cited on page 34).

[KS10a] A. Krutz and T. Sikora. *Summary report for TE3 on inter prediction in HEVC*. Technical report JCTVC-B053. Geneva, Switzerland, 2nd meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, July 2010 (cited on page 36).

[KS10b] A. Krutz and T. Sikora. *Summary report for TE3 on inter prediction in HEVC*. Technical report JCTVC-C220. Guangzhou, China, 3rd meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Oct. 2010 (cited on page 36).

[KS98] Peter Kauff and Klaas Schüür. “Shape-adaptive DCT with block-based DC separation and Delta DC correction.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 8.3 (June 1998), pages 237–242. ISSN: 1051-8215. DOI: 10.1109/76.678616 (cited on page 155).

[LG00] E. Y. Lam and J. W. Goodman. “A mathematical analysis of the DCT coefficient distributions for images.” In: *IEEE Transactions on Image Processing* 9.10 (Oct. 2000), pages 1661–1666. ISSN: 1941-0042. DOI: 10.1109/83.869177 (cited on page 149).

[Li+11] S. Li, F. Zhang, L. Ma, and K. N. Ngan. “Image Quality Assessment by Separately Evaluating Detail Losses and Additive Impairments.” In: *IEEE Transactions on Multimedia* 13.5 (Oct. 2011), pages 935–949. ISSN: 1941-0077. DOI: 10.1109/TMM.2011.2152382 (cited on page 27).

[Li+17] Xiang Li, Ted Hsieh, Jianle Chen, and Marta Karczewicz. *AHG5 External Memory Access Evaluation with the Consideration of Cache*. Technical report JVET-G0061. Torino, Italy, 7th meeting: Joint Video Exploration Team (JVET) of ITU-T VCEG and ISO/IEC MPEG, July 2017 (cited on page 109).

[LKK11] H. J. Leu, S. Kim, and W. Kim. “Statistical Modeling of Inter-Frame Prediction Error and Its Adaptive Transform.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 21.4 (Apr. 2011), pages 519–523. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2011.2125470 (cited on page 149).

[MC11] S. Milani and G. Calvagno. “Segmentation-based motion compensation for enhanced video coding.” In: *2011 18th IEEE International Conference on Image Processing*. Sept. 2011, pages 1649–1652. DOI: 10.1109/ICIP.2011.6115769 (cited on page 39).

[Mer+16] P. Merkle, K. Müller, D. Marpe, and T. Wiegand. “Depth Intra Coding for 3D Video Based on Geometric Primitives.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.3 (Mar. 2016), pages 570–582. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2015.2407791 (cited on page 37).

[MNW98] Alistair Moffat, Radford M. Neal, and Ian H. Witten. “Arithmetic Coding Revisited.” In: *ACM Trans. Inf. Syst.* 16.3 (July 1998), pages 256–294. ISSN: 1046-8188. DOI: 10.1145/290159.290162. URL: http://doi.acm.org/10.1145/290159.290162 (cited on page 23).

[MPF09a] A. A. Muhit, M. R. Pickering, and M. R. Frater. “A fast approach for geometry-adaptive block partitioning.” In: *2009 Picture Coding Symposium*. May 2009, pages 1–4. DOI: 10.1109/PCS.2009.5167365 (cited on page 37).

[MPF09b] A. A. Muhit, M. R. Pickering, and M. R. Frater. “Motion compensation using geometry and an elastic motion model.” In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. Nov. 2009, pages 621–624. DOI: `10.1109/ICIP.2009.5413849` (cited on page 37).

[MSW03] D. Marpe, H. Schwarz, and T. Wiegand. “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7 (July 2003), pages 620–636. ISSN: 1558-2205. DOI: `10.1109/TCSVT.2003.815173` (cited on page 23).

[Mul19] MulticoreWare Inc. *x265 Manual*. English. Version 8.2. MulticoreWare Inc. Sept. 25, 2019. URL: `http://x265.org/` (cited on page 139).

[Mül93] F. Müller. “Distribution shape of two-dimensional DCT coefficients of natural images.” In: *Electronics Letters* 29.22 (Oct. 1993), pages 1935–1936. ISSN: 0013-5194. DOI: `10.1049/el:19931288` (cited on page 149).

[MX12] M. Mrak and J. Xu. “Improving screen content coding in HEVC by transform skipping.” In: *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. Aug. 2012, pages 1209–1213 (cited on page 151).

[MZ93] S. G. Mallat and Zhifeng Zhang. “Matching pursuits with time-frequency dictionaries.” In: *IEEE Transactions on Signal Processing* 41.12 (Dec. 1993), pages 3397–3415. ISSN: 1941-0476. DOI: `10.1109/78.258082` (cited on page 171).

[Nar13] M. Narroschke. “Coding Efficiency of the DCT and DST in Hybrid Video Coding.” In: *IEEE Journal of Selected Topics in Signal Processing* 7.6 (Dec. 2013), pages 1062–1071. ISSN: 1941-0484. DOI: `10.1109/JSTSP.2013.2272192` (cited on page 151).

[Net20] Netflix, Inc. *VMAF - Video Multi-Method Assessment Fusion*. `https://github.com/Netflix/vmaf`. 2018–2020 (cited on page 25).

[Ngu+19] Tung Nguyen, Benjamin Bross, Paul Keydel, Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. “Extended Transform Skip Mode and Fast Multiple Transform Set Selection in VVC.” In: *2019 Picture Coding Symposium (PCS)*. Nov. 2019 (cited on page 151).

[Nor+12] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. Van der Auwera. “HEVC Deblocking Filter.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pages 1746–1754. ISSN: 1051-8215. DOI: `10.1109/TCSVT.2012.2223053` (cited on page 107).

[Ohm15] Jens-Rainer Ohm. *Multimedia Signal Coding and Transmission*. Springer, Heidelberg/Berlin, 2015. ISBN: 3-540-01249-4. DOI: `10.1007/978-3-662-46691-9` (cited on page 5).

[OS09] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN: 0131988425, 9780131988422 (cited on page 5).

[Ost97] J. Ostermann. “Feedback loop for coder control in a block-based hybrid coder with mesh-based motion compensation.” In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Volume 4. Apr. 1997, 2673–2676 vol.4. DOI: `10.1109/ICASSP.1997.595339` (cited on page 38).

[Pen+16] W. Peng, F. G. Walls, R. A. Cohen, J. Xu, J. Ostermann, A. MacInnis, and T. Lin. “Overview of Screen Content Video Coding: Technologies, Standards, and Beyond.” In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 6.4 (Dec. 2016), pages 393–408. ISSN: 2156-3365. DOI: `10.1109/JETCAS.2016.2608971` (cited on page 18).

[RH19] Oeter de Rivaz and Jack Haughton. *AV1 Bitstream & Decoding Process Specification*. Technical report. The Alliance for Open Media, 2019. URL: `https://aomediacodec.github.io/av1-spec/av1-spec.pdf` (cited on pages 18, 38).

[Ric03] Iain E. G. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. New York, NY, USA: John Wiley & Sons, Inc., 2003 (cited on page 33).

[San] Sandvine, Inc. *The Global Internet Phenomena Report 2019*. `https://www.sandvine.com/press-releases/sandvine-releases-2019-global-internet-phenomena-report`. Accessed: 2019-12-30 (cited on page 1).

[SB06] H. R. Sheikh and A. C. Bovik. “Image Information and Visual Quality.” In: *Trans. Img. Proc.* 15.2 (Feb. 2006), pages 430–444. ISSN: 1057-7149. DOI: `10.1109/TIP.2005.859378`. URL: `http://dx.doi.org/10.1109/TIP.2005.859378` (cited on page 27).

[SB12] V. Sze and M. Budagavi. “High Throughput CABAC Entropy Coding in HEVC.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pages 1778–1791. ISSN: 1558-2205. DOI: `10.1109/TCSVT.2012.2221526` (cited on page 23).

[Sch+19] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand. “Hybrid Video Coding with Trellis-Coded Quantization.” In: *2019 Data Compression Conference (DCC)*. Mar. 2019, pages 182–191. DOI: `10.1109/DCC.2019.00026` (cited on pages 19, 149).

[Seg+17] Andrew Segall, Vittorio Baroncini, Jill Boyce, Jianle Chen, and Teruhiko Suzuki. *Joint Call for Proposals on Video Compression with Capability beyond HEVC*. JVET-H1002. Macao, CN, 8th meeting: Oct. 2017 (cited on page 28).

[SEO11] J. Schmidt, B. Edler, and J. Ostermann. “Prediction of DCT coefficients considering motion compensation error distributions.” In: *2011 Visual Communications and Image Processing (VCIP)*. Nov. 2011, pages 1–4. DOI: `10.1109/VCIP.2011.6115991` (cited on pages 149, 151).

[SF13] A. Saxena and F. C. Fernandes. “DCT/DST-Based Transform Coding for Intra Prediction in Image/Video Coding.” In: *IEEE Transactions on Image Processing* 22.10 (Oct. 2013), pages 3974–3981. ISSN: 1941-0042. DOI: `10.1109/TIP.2013.2265882` (cited on page 150).

[Shi92] Yoshiaki Shishikui. "A Study on Modeling of the Motion Compensation Prediction Error Signal." In: *IEICE TRANSACTIONS on Communications* Vol.E75-B.5 (Apr. 1992), pages 368–376. ISSN: 0916-8516 (cited on pages 105, 149).

[SM00] J. K. Su and R. M. Mersereau. "Motion Estimation Methods for Overlapped Block Motion Compensation." In: *IEEE Transactions on Image Processing* 9.9 (Sept. 2000), pages 1509–1521. DOI: 10.1109/83.862628 (cited on page 114).

[SM95] T. Sikora and B. Makai. "Shape-adaptive DCT for generic coding of video." In: *IEEE Transactions on Circuits and Systems for Video Technology* 5.1 (Feb. 1995), pages 59–62. ISSN: 1051-8215. DOI: 10.1109/76.350781 (cited on page 155).

[Sul+12] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand. "Overview of the High Efficiency Video Coding (HEVC) Standard." In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pages 1649–1668 (cited on page 17).

[Tan+11] Y. H. Tan, C. Yeo, H. L. Tan, and Z. Li. "On residual quad-tree coding in HEVC." In: *2011 IEEE 13th International Workshop on Multimedia Signal Processing*. Oct. 2011, pages 1–4. DOI: 10.1109/MMSP.2011.6093805 (cited on page 151).

[Tec+16] G. Tech, Y. Chen, K. Müller, J. Ohm, A. Vetro, and Y. Wang. "Overview of the Multiview and 3D Extensions of High Efficiency Video Coding." In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (Jan. 2016), pages 35–49. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2015.2477935 (cited on page 37).

[TG07] J. A. Tropp and A. C. Gilbert. "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit." In: *IEEE Transactions on Information Theory* 53.12 (Dec. 2007), pages 4655–4666. ISSN: 1557-9654. DOI: 10.1109/TIT.2007.909108 (cited on page 171).

[Tro04] J. A. Tropp. "Greed is good: algorithmic results for sparse approximation." In: *IEEE Transactions on Information Theory* 50.10 (Oct. 2004), pages 2231–2242. ISSN: 1557-9654. DOI: 10.1109/TIT.2004.834793 (cited on page 171).

[VTP97] P. J. L. Van Beek, A. M. Tekalp, and A. Puri. "2-D mesh geometry and motion compression for efficient object-based video representation." In: *Proceedings of International Conference on Image Processing*. Volume 3. Oct. 1997, 440–443 vol.3. DOI: 10.1109/ICIP.1997.632151 (cited on page 38).

[Wan+12] Q. Wang, M. T. Sun, G. J. Sullivan, and J. Li. "Complexity-reduced geometry partition search and high efficiency prediction for video coding." In: *2012 IEEE International Symposium on Circuits and Systems*. May 2012, pages 133–136. DOI: 10.1109/ISCAS.2012.6271486 (cited on page 37).

[Wan+13] Q. Wang, X. Ji, M. T. Sun, G. J. Sullivan, J. Li, and Q. Dai. "Complexity Reduction and Performance Improvement for Geometry Partitioning in Video Coding." In: *IEEE Transactions on Circuits and Systems for Video Technology* 23.2 (Feb. 2013), pages 338–352. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2203743 (cited on pages 37, 139).

[WC14] Z. Wu and C. W. Chen. "Signal Reconstruction from Partial Frequency Coefficients for Image/Video Frame Upsampling." In: *IEEE Transactions on Broadcasting* 60.3 (Sept. 2014), pages 575–581. ISSN: 1557-9611. DOI: 10.1109/TBC.2014.2347871 (cited on page 153).

[Wen+00] Wentao Zheng, Y. Kanatsugu, S. Itoh, and Y. Tanaka. "Analysis of space-dependent characteristics of motion-compensated frame differences." In: *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)*. Volume 3. Sept. 2000, 158–161 vol.3. DOI: 10.1109/ICIP.2000.899319 (cited on pages 105, 149, 151).

[Wen+02] Wentao Zheng, Y. Shishikui, M. Naemura, Y. Kanatsugu, and S. Itoh. "Analysis of space-dependent characteristics of motion-compensated frame differences based on a statistical motion distribution model." In: *IEEE Transactions on Image Processing* 11.4 (Apr. 2002), pages 377–386. ISSN: 1057-7149. DOI: 10.1109/TIP.2002.999672 (cited on pages 105, 149, 151).

[Wie14] Mathias Wien. *High Efficiency Video Coding – Coding Tools and Specification*. Berlin, Heidelberg: Springer, Sept. 2014. ISBN: 978-3-662-44275-3 (cited on pages 10, 19, 30, 119).

[WNC87] Ian H. Witten, Radford M. Neal, and John G. Cleary. "Arithmetic Coding for Data Compression." In: *Commun. ACM* 30.6 (June 1987), pages 520–540. ISSN: 0001-0782. DOI: 10.1145/214762.214771. URL: http://doi.acm.org/10.1145/214762.214771 (cited on page 23).

[Wu+07] X. Wu, Q. Sun, K. Zhang, and Lu Yu. "Modeling Natural Image for Estimating DCT Coefficient Properties of Intra Prediction." In: *2007 IEEE International Conference on Multimedia and Expo*. July 2007, pages 476–479. DOI: 10.1109/ICME.2007.4284690 (cited on page 149).

[Yan+10] H. Yang et al. *Description of video coding technology proposal by Huawei Technologies & Hisilicon Technologies*. Technical report JCTVC-A111. Dresden, Germany, 1st meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Apr. 2010 (cited on page 36).

[Yeo+12] C. Yeo, Y. H. Tan, Z. Li, and S. Rahardja. "Mode-Dependent Transforms for Coding Directional Intra Prediction Residuals." In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.4 (Apr. 2012), pages 545–554. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2011.2168291 (cited on page 150).

[Yu+15] Haoping Yu, Robert Cohen, Krishna Rapaka, and Jizheng Xu. *Common test conditions for screen content coding*. Technical report JCTVC-U1015. Warsaw, Poland, 21st meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, June 2015 (cited on page 26).

[ZCK11] X. Zheng, I. S. Chong, and I.-K. Kim. *CE2: Summary report of core experiment on Motion Partitioning and OBMC*. Technical report JCTVC-G032. Geneva, Switzerland, 7th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Nov. 2011 (cited on page 36).

[Zha+09] Y. Zhang, W. Wang, L. Zheng, and M. Wu. "Motion Compensation Using Polyline Based Block Partition." In: *2009 2nd International Congress on Image and Signal Processing*. Oct. 2009, pages 1–5. DOI: 10.1109/CISP.2009.5304312 (cited on page 38).

[Zha+16] X. Zhao, J. Chen, M. Karczewicz, L. Zhang, X. Li, and W. Chien. "Enhanced Multiple Transform for Video Coding." In: *2016 Data Compression Conference (DCC)*. Mar. 2016, pages 73–82. DOI: 10.1109/DCC.2016.9 (cited on pages 10, 150).

[Zhe+01] W. Zheng, Y. Shishikui, M. Naemura, Y. Kanatsugu, and S. Itoh. "Analysis of overlapped block motion compensation based on a statistical motion distribution model." In: *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*. Volume 3. Oct. 2001, 522–525 vol.3. DOI: 10.1109/ICIP.2001.958166 (cited on pages 105, 149, 151).

[Zhe+11a] X. Zheng, P. Bordes, P. Chen, and I.-K. Kim. *CE2: Summary of Core Experiment 2 on Flexible Motion Partitioning*. Technical report JCTVC-E022. Geneva, Switzerland, 5th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Mar. 2011 (cited on page 36).

[Zhe+11b] X. Zheng, P. Bordes, P. Chen, I.-K. Kim, W.-H. Peng, and L. Guo. *CE2: Summary report of Core Experiment on motion partitioning and OBMC*. Technical report JCTVC-F022. Torino, Italy, 6th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, July 2011 (cited on page 36).

[Zhe+11c] X. Zheng, H. Yu, S. Li, Y. He, and P. Bordes. *CE2: Non-rectangular motion partitioning*. Technical report JCTVC-F415. Torino, Italy, 6th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, July 2011 (cited on page 37).

[Zhu+19] W. Zhu, L. Zhang, J. Xu, and K. Zhang. *Non-CE8: disabling TPM blending*. Doc. JVET-O0563. Gothenburg, Sweden, 15th meeting: Joint Video Experts Team of ITU-T VCEG and ISO/IEC MPEG, July 2019 (cited on page 136).

[Zou+13] F. Zou, O. C. Au, C. Pang, J. Dai, X. Zhang, and L. Fang. "Rate-Distortion Optimized Transforms Based on the Lloyd-Type Algorithm for Intra Block Coding." In: *IEEE Journal of Selected Topics in Signal Processing* 7.6 (Dec. 2013), pages 1072–1083. ISSN: 1941-0484. DOI: 10.1109/JSTSP.2013.2274173 (cited on pages 149, 150).

[ZWZ10] T. Zhu, J. Wang, and X. Zhang. "Research on statistical distributions of transform coefficients for H.264/SVC." In: *2010 3rd International Congress on Image and Signal Processing*. Volume 1. Oct. 2010, pages 198–201. DOI: 10.1109/CISP.2010.5648005 (cited on page 149).

[ZY10] X. Zheng and H. Yu. *TE3: Huawei & Hisilicon report on flexible motion partitioning coding*. Technical report JCTVC-B041. Geneva, Switzerland, 2nd meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, July 2010 (cited on page 36).