University of California
Santa Barbara

Towards optimal prediction and transforms for video compression

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Bharath Vishwanath

Committee in charge:

Professor Kenneth Rose, Chair
Professor B.S. Manjunath
Professor Shiv Chandrasekaran
Professor Nikhil Jayant

June 2021

The Dissertation of Bharath Vishwanath is approved.

Professor B.S. Manjunath

Professor Shiv Chandrasekaran

Professor Nikhil Jayant

Professor Kenneth Rose, Committee Chair

June 2021

Towards optimal prediction and transforms for video compression

by

Bharath Vishwanath

To my beloved family

Acknowledgements

I profusely thank Prof. Rose for his guidance during my PhD. Discussions with him have strengthened my fundamentals and made me look at research problems from a different perspective. During many hiccups in research, he has been greatly supportive and encouraging. I'm deeply indebted for all his support and guidance.

My sincere thanks to Professor Manjunath, Professor Shiv, and Professor Jayant, for serving on my doctoral committee. I would like to thank all professors of the ECE department for their excellent courses that helped me gain strong fundamentals. Thanks to Val for her consistent support.

This research has been partly funded by Interdigital and Google. Thanks to our sponsors. I would like to thank Yan Ye for her efforts in supporting our research.

I thank all my lab-mates for all the discussions and making my experience at SCL a memorable one. I would like to thank Tejaswi for his support and guidance during the initial phase of PhD.

I would like to thank all my teachers, lecturers and professors back in India who played a major role in moulding me. Thanks to all my friends here in US and back in India for their constant support.

I'm deeply thankful to my family for sailing through all the hardships and constantly supporting me. Thanks for the unconditional love and affection.

Curriculum Vitæ

Bharath Vishwanath

Education

2021	Ph.D. in Electrical and Computer Engineerng, University of California, Santa Barbara.
2016	M.S. in Electrical and Computer Engineering, University of California, Santa Barbara.
2014	B.Tech in Electronics and Communication, National Institute of Technology Karnataka, India

Experience

2016-2021	Graduate Research Assistant, Signal Compression Lab, UCSB
2019	Video Processing Research Intern, Dolby Laboratories, Sunnyvale
2017	Video Coding Intern, Interdigital Communications, San Diego
2016	Video Coding Intern, Interdigital Communications, San Diego
2013	Summer Intern, University of Southern California, Los Angeles

Publications

- B. Vishwanath, and K. Rose, "Transform Design for Temporal Prediction Residual in Video Coding ," to be submitted to IEEE Transactions on Image Processing

- B. Vishwanath, T. Nanjundaswamy, and K. Rose, "A Geodesic Translation Model for Motion Compensation in Spherical Video Coding," submitted to IEEE Transactions on Image Processing

- B. Vishwanath, T. Nanjundaswamy, and K. Rose, "Deterministic Annealing Based Switched Predictor Design for Adaptive Compression Systems with Applications in Video Coding," submitted to IEEE Transactions on Image Processing

- K. Sivakumar, B. Vishwanath and K. Rose, "Geodesic Disparity Compensation for Inter-View Prediction in VR180," in VCIP 2020

- B. Vishwanath, S. Li and K. Rose, "Asymptotic Closed-Loop Design of Transform Modes for the Inter-Prediction Residual in Video Coding," in ICIP 2020

- B. Vishwanath and K. Rose, "Spherical Video Coding with Geometry and Region Adaptive Transform Domain Temporal Prediction," in ICASSP 2020

- B. Vishwanath and K. Rose, "Spherical Video Coding with Motion Vector Modulation to Account for Camera Motion," in VCIP 2019

- B. Vishwanath, T. Nanjundaswamy, and K. Rose, "Deterministic Annealing Based Transform Domain Temporal Predictor Design for Adaptive Video Coding," in DCC 2019

- B. Vishwanath, T. Nanjundaswamy, and K. Rose, "A Deterministic Annealing Approach to Switched Predictor Design for Adaptive Compression Systems," in IEEE ICASSP 2019

- B. Vishwanath, T. Nanjundaswamy, and K. Rose, "Motion Compensated Prediction for Translational Camera Motion in Spherical Video Coding," IEEE MMSP 2018

- B. Vishwanath, K. Rose, Y. He, and Y. Ye, "Rotational Motion Compensated Prediction in HEVC Based Omnidirectional Video Coding," Picture Coding Symposium (PCS), June 2018

- B. Vishwanath, T. Nanjundaswamy, and K. Rose, "Rotational Motion Model for Temporal Prediction in 360 Video Coding," IEEE MMSP 2017

- X. Xiu, Y. He, Y. Ye, B. Vishwanath, "An Evaluation Framework for 360 Video Compression," in VCIP 2017

- B. Vishwanath, T. Nanjundaswamy, S. Zamani, and K. Rose, "Deterministic Annealing Based Design of Error Resilient Predictive Compression Systems," ICASSP 2017

- B. Vishwanath and C. S.Seelamantula,"Cell tracking using Particle Filters and Level Sets," IEEE TENCON 2013

Awards

2020	Dissertation Fellowship, Department of Electrical and Computer Engineering, University of California, Santa Barbara
2017	Top 10% paper award at IEEE MMMSP
2013	Viterbi-India Fellowship

Abstract

Towards optimal prediction and transforms for video compression

by

Bharath Vishwanath

The focus of the dissertation is on optimal prediction paradigms and the complementary design of transforms for video compression. One main line of research is on motion compensated prediction for spherical videos. Standard approaches project spherical videos onto planes for processing with traditional 2D video coding standards. Such approaches are significantly sub-optimal as standard video coders only allow for block translations in the critical tool of motion compensated prediction, which is incompatible with the expected motion in projected spherical video. Specifically, the effective sampling density varies over the sphere and the resulting locally varying warping yields complex non-linear motion in the projected domain. Moreover, motion vector in the projected domain does not have a useful physical interpretation. To address these shortcomings, the thesis presents a rotational motion model that performs motion compensation in terms of rotations along geodesics on the sphere. Rotation preserves object shape and size on the sphere. A motion vector in this model implicitly specifies an axis of rotation and the degree of rotation about that axis, to convey the actual motion of objects on the sphere. Complementary to the novel motion model, an effective motion search technique that is tailored to the sphere's geometry is presented for improved motion estimation. The thesis then considers an important class of spherical videos whose dynamics involve camera motion. The thesis presents a new geodesic translation motion model that captures the motion field on the sphere, and capitalizes on insights into the perceived motion on the sphere due to camera translation. Specifically, surrounding static points are perceived as

moving along their respective geodesics, which all intersect at the poles corresponding to the camera velocity axis. The method further exploits insights into the displacement rate of static points, which depends on object depth and degree of elevation on the sphere (with respect to the camera velocity axis). Complementary to the new motion model, a search grid tailored to capture expected geodesic motion on the sphere for effective motion estimation is presented.

Another focus related to predictor optimization is on design of prediction filters for adaptive compression of non-stationary signals with applications to video coding. The design poses several challenges including: i) catastrophic instability due to statistical mismatch driven by error propagation through the prediction loop, and ii) severe non-convexity of the cost surface that is often riddled with poor local minima. Motivated by these challenges, the thesis presents a near-optimal method for designing prediction modes for adaptive compression. The design builds on a stable, open-loop platform, but with a subterfuge that ensures that it is asymptotically optimized for closed-loop operation. The non-convexity is handled by deterministic annealing, a powerful optimization tool to avoid poor local minima. The impact of the design paradigm on practical applications is demonstrated by designing temporal prediction filters in video coding.

The second line of research focuses on offline design of transforms for inter-prediction residuals, a complementary step to the effective prediction paradigms. Existing codecs for regular 2D videos use standard trigonometric transforms. These transforms are only optimal under certain assumptions that are highly questionable for inter-prediction residue. For projected spherical videos, derivation of transforms even under certain assumptions is a highly challenging task. Thus, there is a strong motivation for a data-driven approach to learn these transforms. The joint design of multiple transform modes is highly challenging due to critical stability problems inherent to feedback through the codec's prediction loop, wherein training updates inadvertently impact the signal statistics the

transform ultimately operates on, and are often counter-productive. It is the premise of this work that a truly effective switched transform design procedure must account for and circumvent this shortcoming. We introduce a data-driven approach to design optimal transform modes for adaptive switching by the encoder. Most importantly, to overcome the critical stability issues, the approach is derived within an asymptotic closed loop (ACL) design framework. The design yields transforms that outperform the transforms obtained by standard design procedures.

Contents

Curriculum Vitae **vi**

Abstract **ix**

List of Figures **xiv**

List of Tables **xvi**

1 Introduction **1**

2 Rotational Motion Model for Spherical Video Coding **6**
2.1 Introduction 6
2.2 Background 8
2.3 Prediction Framework with a Rotational Motion Model 12
2.4 Experimental Results 19
2.5 Conclusions 21

3 Geodesic Translation Motion Model for Spherical Video Coding **25**
3.1 Introduction 25
3.2 Geodesic Motion Compensated Prediction 27
3.3 On efficient implementation of the geodesic model 38
3.4 Experimental Results 41
3.5 Conclusions 44

4 Efficient Predictor Mode Design for Adaptive Prediction **49**
4.1 Introduction 49
4.2 Background 53
4.3 Deterministic Annealing Based Predictor Design 58
4.4 Predictor Design in Video coding 63
4.5 Experimental Results 73
4.6 Conclusions 77

5 Transform Design for the Inter-Prediction Residual in Video Coding 79
5.1 Introduction 79
5.2 Background 81
5.3 Relevant Work 82
5.4 Proposed Method 85
5.5 Experimental Results 89
5.6 Conclusions 91

Bibliography 95

List of Figures

2.1 Sampling pattern induced on sphere due to equirectangular projection (top) and corresponding 2-D projection (bottom) 9
2.2 Equatorial Cylindrical Cube-map: Mapping of equatorial sphere region onto cylinder (left) and mapping of polar discs to square faces (right) . . 10
2.3 Standard spherical video coding pipeline 11
2.4 Comparison of motion search grids: Search pattern defined in projected domain in HEVC mapped to sphere (top) and proposed location invariant search pattern (bottom) 14
2.5 Steps in motion compensation with rotational motion model 16
2.6 Comparison of motion models: Model effect on block shape as seen on sphere 18
2.7 RD curves for (a) *glacier* and (b) *chair* sequences with ERP as the projection format 22
2.8 RD curves for (a) *bicyclist* and (b) *glacier* sequences with ECP as the projection format 23
2.9 Subjective comparison for chairlift sequence encoded at constant bit-rate with ERP as projection format: HEVC (top) and proposed method (bottom) 24

3.1 A static point's perceived trajectory on the sphere due to camera motion 29
3.2 The geodesic translation motion model 30
3.3 Relation between geodesic displacement, elevation, depth and camera translation 33
3.4 Comparison of the proposed EAP based distribution of motion vectors along the azimuth and uniform distribution of motion vectors 39
3.5 Comparison of motion search patterns: (a) HEVC search pattern defined in projection format, (b) Search pattern aligned with the expected geodesic motion of the center of the block and (c) Geodesic aligned search pattern optimized to account for object motion 40
3.6 RD curves for (a) *bicyclist* and (b) *balboa* sequences with ERP as the projection format 46

3.7 RD curves for (a) *bicyclist* and (b) *balboa* sequences with ECP as the projection format 47
3.8 Subjective comparison for balboa sequence encoded at constant bit-rate with ERP as projection format. anchor (top) and proposed method (bottom) 48

4.1 A simple first order predictive compression system 55
4.2 Standard closed-loop design for predictor optimization 57
4.3 Predictor optimization with asymptotic closed-loop design 57
4.4 Flow chart of the proposed DA-ACL-TDTP design algorithm 69
4.5 Illustration of extended block transform domain temporal prediction . . . 70
4.6 First-order scalar predictive coding. Reconstructed SNR vs average bits per sample for the test set of speech files 74
4.7 Comparison of performance of EB-TDTP modes designed by ACL and DA-ACL methods over HEVC for test sequences of (a) Coastguard, (b) Kimono, (c) Soccer and (d) BasketballDrive sequence. 77

5.1 Closed-loop design of transforms 83
5.2 Asymptotic closed-loop design of transforms 88
5.3 Illustration of design instability in closed-loop design and ACL as a stable platform design 90
5.4 Bases images for 8X8 block for first transform mode initialized to DCT-DCT obtained for HD sequence training at 700Kbps 93
5.5 Bases images for 8X8 block for first transform mode initialized to DCT-DCT obtained for EAC sequence training at 1000Kbps 93

List of Tables

2.1 BD-rate gains in % for Y-component by employing rotation motion model over HEVC using translation motion model in projected domain 20

3.1 BD-rate gains in % for Y component over HEVC for rotation motion model and the geodesic translation motion model 44

4.1 Temporal correlation coefficients, along motion trajectory, of DCT coefficients in the block 65
4.2 The training set of video sequences for EB-TDTP design 78
4.3 Performance over the test set: bit-rate savings over HEVC (in %) for the Y component by employing EB-TDTP filters designed from CL, ACL and DA-ACL design paradigms 78

5.1 The training set of video sequences for transform design 92
5.2 % bit-rate savings on test set for YUV-PSNR over AV1 for cif sequences (uses trigonometric transforms). 94
5.3 % bit-rate savings on test set for YUV-PSNR over AV1 for HD sequences. 94
5.4 % bit-rate savings on test set for YUV-PSNR over AV1 for EAC sequences. 94

Chapter 1

Introduction

Video forms an important component of the data consumption and constitutes a major part of the internet traffic. Evidently, there is a pressing need for better compression algorithms. A recent challenge to the research community is the compression of spherical videos that forms the backbone of many virtual reality related applications. In contrast to the regular videos that have limited field of view, spherical videos have a 360° field of view that enables the user to view in any desired direction. There is also a growing interest in augmented reality related applications that gives six degrees of freedom that enables the user to navigate in a virtual environment. Here too, spherical video forms the key factor in realising such applications. Due to increased field of view, spherical videos generate enormous amounts of data. Thus, there is a clear need for efficient compression tools tailored to spherical videos.

Most of the video codecs including the recent HEVC [1] and AV1 [2] divide the frame into blocks and compress them by four fundamental steps, namely, prediction, transform, quantization and entropy coding. The first step of prediction involves predicting the current block of pixels from previous reconstructed pixels. It can either be i) inter-prediction: wherein the best matching block from a reconstructed frame is used as the

prediction signal and the offset to the matching block (often called a motion vector) is sent to the decoder or ii) intra-prediction: wherein the current block is predicted from the reconstructions of the adjacent blocks in the current frame. The resulting prediction residual is transformed by different kernels such as discrete cosine transform (DCT), asymmetric discrete sine transform (ADST) etc. The key purpose of the transform block is to decorrelate the prediction residual and achieve energy compaction with as few transform coefficients as possible. These transform coefficients are then quantized and entropy coded. Quantizer decides the information to remove (usually high frequency signals), and optimizes the bit allocation based on the bit-rate requirement. Entropy coding optimizes the codebook based on the prior probability information to minimize the average number of bits needed to get the resulting bit-stream.

The thesis focuses on optimization of prediction and transform modules. The first line of research considers optimization of prediction module. Initial focus is on motion compensated prediction for spherical videos. As mentioned earlier, spherical videos generate enormous amounts of data and demand for efficient compression tools. Existing methods simply project spherical videos onto planes using various geometries and use 2D codecs for compression. The projection introduces warping, resulting in complex non-linear motion of objects in the projected domain. Traditional codecs use block based translational motion model which is highly ineffective in capturing the complex non-linear motion in the projected domain. Further, during motion estimation at the encoder, the fixed search pattern in the projected plane results in spatially varying pattern on the sphere which severely compromises the compression efficiency. To address these shortcomings, the chapter 2 of the thesis introduces a rotational motion model that performs motion compensation on the sphere in the natural way. Specifically, given a vector v corresponding to the center of the block mapped to the sphere and a new vector v' to which v is to be motion compensated, the model proposes to rotate v to v' along the geodesic

connecting them. The same rotation operation is applied for the rest of the pixels in the block. Complementary to the motion model, a search pattern is defined on the sphere that is agnostic of the location of the block on the sphere and the projection geometry. Only for the purpose of defining the search pattern, the block is treated as though it is on the equator. The search pattern thus defined will be the same for all the blocks irrespective of its location on the sphere. The proposed motion model and the search pattern are defined on the sphere and can thus be easily extended to any arbitrary projection format. Experimental results show huge bit-rate savings over the standard methods demonstrating the efficacy of the proposed method.

Chapter 3 focuses on an important class of spherical videos with camera motion. The chapter introduces a geodesic translation motion model to capture the perceived motion of the surrounding objects on the sphere due to camera motion. The core observation is that, with camera motion, all the surrounding static objects are perceived to move along their respective geodesics that all intersect at the points where the camera velocity vector intersects the sphere. The model perfectly accounts for the perspective distortions. Specifically, as the objects approach the camera, it is perceived to get magnified and vice versa, which is perfectly captured by geodesic translation. The motion vectors in the model are largely 1-D, resulting in huge bit-rate savings in conveying the motion vectors to the decoder. The chapter then analyses the rate of displacement of pixels on the sphere as related to depth of the object and the camera velocity. The analysis yields a relationship that opens door for a motion vector modulation scheme (under the mild assumption of constant depth for a block). A motion vector is sent to the decoder that corresponds to the amount of geodesic translation of the center of the block. The modulation scheme is then applied to derive the motion vectors for other pixels in the block. Finally, the chapter introduces a motion search grid that adapts to the direction of the camera motion. The grid is defined such that the first component

of the motion vector captures the geodesic translation of the center of the block and the second component captures the actual motion of object that is independent of the camera motion. For capturing the actual object motion, the grid leverages a novel equal-area projection based motion vector distribution consistent with expectation of actual intrinsic object motion. Experimental results show that the proposed method outperforms the standard approaches and brings huge bit-rate savings.

The next focus in the prediction optimization is the design of prediction filters for compression of non-stationary signals with emphasis on video coding. Most of the real-world signals are non-stationary and effective compression of such signals should involve adaptivity. Chapter 4 considers adaptive prediction, wherein, adaptivity is achieved by dividing the signal into blocks of signals and choosing the best prediction filter from a set of offline-trained prediction filers. The design poses several challenges including i) design instability due to closed-loop nature of the coder ii) non-convexity of the cost function that traps the design in poor local-minima. The chapter discusses a general design framework to overcome these challenges that are in fact inherent to almost all compression scenarios. The design instability is tackled by an asymptotic closed-loop (ACL) framework. ACL is an iterative design framework in which, predictors are optimized for a given set of reconstructions and the reconstructions are updated in an *open-loop* fashion with the new predictors. Open-loop updates ensure design stability, with the subterfuge that the design on convergence optimizes predictors for closed-loop operation of the coder. The non-convexity of the cost function is handled by re-deriving for this problem setting the powerful non-convex optimization tool of deterministic annealing. DA solves the initialization problem inherent to all greedy approaches. Its careful annealing schedule avoids poor local minima. The chapter thus gives a near-optimal solution for a fundamental problem of prediction filter design. The chapter then considers an important application of the proposed design in video compression. For a novel temporal

prediction paradigm of transform domain temporal prediction (TDTP) earlier proposed in our lab, prediction modes are derived in the light of the proposed design strategy. In TDTP, the prediction filters are in high-dimensional space and thus a DA based solution is very effective in avoiding local minima. The framework is then broadened to design extended block TDTP filters, wherein the prediction filters are obtained as least square estimates. This paves way to apply the design to a rich class of problems whose optimal solutions are obtained as least-square estimates.

The second line of research focuses on offline learning of transform kernels for inter-prediction residuals in video coding. Traditionally, DCT is used as the transform kernel for inter-prediction residual. To enable better adaptation to residue statistic, recent codecs allow other transforms including ADST, flip-ADST etc. However, the optimality of such transforms for inter-prediction residual is highly questionable. Moreover, for projected spherical videos, the residue statistic is expected to be very different from the residue statistic in regular 2D videos, and further expected to vary for different geometries. Due to non-uniform sampling induced on the sphere, an analytical solution to derive the optimal transform (by working in spherical domain) is nearly impossible. Thus, in chapter 5, we pursue a data driven approach to learn the transform kernels. The design is plagued by instability issue due to closed-loop nature of the codec that is exacerbated compared to predictor design, since the design involves a larger set of parameters to be optimized. We thus propose an ACL based transform optimization strategy that efficiently handles the stability issues. The proposed design yields transforms that outperform the transforms designed by standard methods as demonstrated in the experimental results.

Chapter 2

Rotational Motion Model for Spherical Video Coding

2.1 Introduction

An immersive experience for users is enabled by capturing video with 360° view of the world on a sphere, allowing end users to dynamically control the viewing direction. To simplify storage, transmission and efficient access to desired portions of the 360° video, the data are projected onto planes via one of several possible geometries, e.g., equirectangular, cubemap, octahedron or icosaherdon [3]. In each case a uniform sampling of the plane induces a variable sampling density on the sphere which, in turn, introduces significant warping that varies in magnitude depending on location.

With its increased field of view, 360° video represents a considerably larger volume of data than that of standard 2D video, and hence the practicality of applications using such video critically depends on powerful compression algorithms that are tailored to this signal characteristics. A central component in modern video codecs such as H.264 [4] and HEVC [1] is motion compensated prediction, often referred to as "inter-prediction", which

is tasked with exploiting temporal redundancies. Standard video codecs use a (piecewise) translational motion model for inter prediction, while some non-standard approaches considered extensions to affine motion models that may be able to handle more complex motion, at a potentially significant cost in side information (see recent approaches in [5, 6]). Still, in 360° video, the amount of warping induced by the projection varies for different regions of the sphere, and yields complex non-linear motion in the projected plane, for which both the translation motion model and its affine motion extension are ineffective. Note that even a simple translation of an object on the unit sphere leads to complex non-linear motion in the projected domain. Thus, a new motion compensated prediction technique that is tailored to the setting of 360° video signals is needed.

Since 360° video represents the scene captured on the unit sphere, it is most natural to characterize motion on that sphere. We thus propose a rotational model to characterize angular motion on the sphere. In the proposed framework, we define motion as rotation of a block of pixels on the surface of the sphere along geodesics and transmit information specifying this rotation as "motion vector" in lieu of the block displacement in the 2D projected geometry. Since rotations are unitary transformations, the proposed motion model preserves the shape and area of the objects on the sphere. This model also ensures that given a motion vector, regardless of a block's location on the sphere, it is rotated to the same extent. This feature addresses the motion search suboptimalities of current approaches, by allowing the search pattern, range and precision to be independent of the position of the block on the sphere. Complementary to the motion model, we propose employing location invariant search grid around the center of the coding block on the sphere for further performance improvement. Performing motion compensation on the sphere and having a fixed motion search pattern makes the proposed approach agnostic of the projection geometry and hence universally applicable to all projection geometries. Substantial gains in experiments validate the efficacy of the proposed approach.

2.2 Background

2.2.1 Overview of Common Projections

Sphere to plane mappings have been studied extensively, and a plethora of such projections are covered, e.g., in [3]. In this section, we briefly review a few important projection geometries. Perhaps the most popular projection, with extensive historical use is equi-rectangular projection (ERP), which is also commonly encountered in many virtual reality applications to this day. Beside ERP, we also review equatorial cylindrical projection (ECP), which is among the best known projection formats for video compression applications. Each projection format is presented concisely. For detailed mappings, please refer to [7].

Equirectangular Projection

The sampling pattern induced on the sphere by ERP, and the corresponding projection to 2-D, are shown in Fig. 2.1.

Let (ϕ, θ) be the spherical coordinates of a point on the unit sphere and (x_p, y_p) be the coordinates in the projected domain. ERP can be represented mathematically as:

$$\begin{aligned} x_p &= (1 + \frac{\phi}{\pi})\frac{W}{2} \\ y_p &= (1 - \frac{2\theta}{\pi})\frac{H}{2} \end{aligned} \tag{2.1}$$

where W and H are respectively the width and height of the 2-D projection. Mapping a point in ERP back to sphere is straightforward from (2.1). ERP induces on the sphere a vertical (along longitude) sampling density that is constant. However, the horizontal (along latitude) sampling density increases as we move towards the poles.

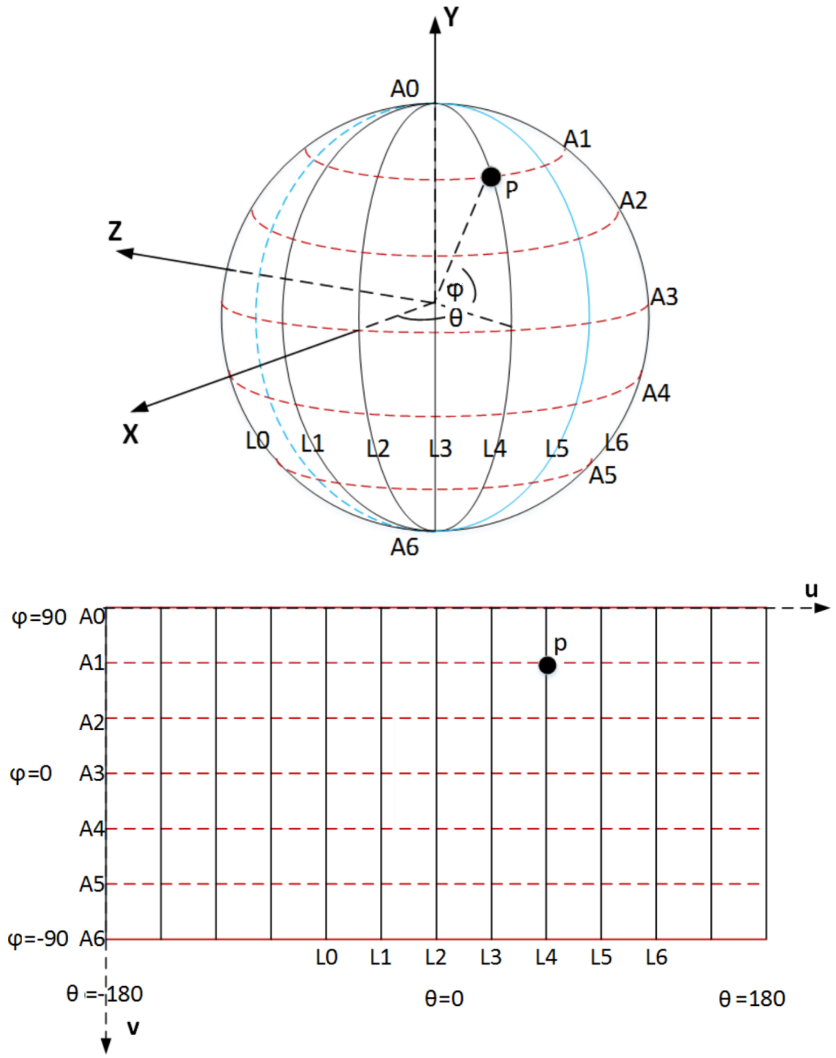


Figure 2.1: Sampling pattern induced on sphere due to equirectangular projection (top) and corresponding 2-D projection (bottom)

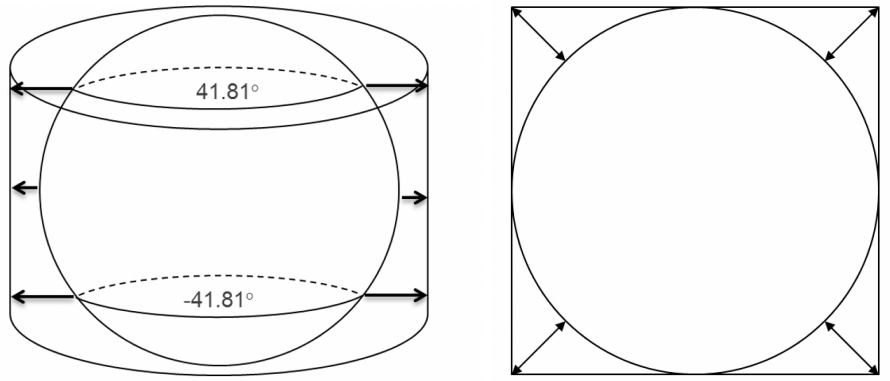


Figure 2.2: Equatorial Cylindrical Cube-map: Mapping of equatorial sphere region onto cylinder (left) and mapping of polar discs to square faces (right)

Equatorial Cylindrical Projection

Equatorial cylindical projection (ECP) was proposed in [8] and is shown in Fig. 2.2. In ECP, the equatorial region corresponding to $\{-sin^{-1}\frac{2}{3} \leq \phi \leq sin^{-1}\frac{2}{3}\}$ is mapped to four faces of the cube via Lambert equi-area sampling [3]. The remaining two faces correspond to the polar caps which are first mapped to planar discs and then stretched to fit the square faces.

2.2.2 Relevant Work

In this section, we introduce the standard encoding pipeline for spherical videos, and summarize various recent motion-compensated prediction techniques for spherical video compression.

Standard Spherical Video Coding Pipeline

The standard spherical video coding pipeline is shown in Fig. 2.3 and discussed in detail in [9]. The original spherical video is projected onto a plane (or planes) via a projection format such as ERP, cubemap, etc. The projected video is then encoded

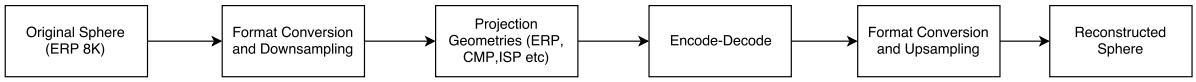


Figure 2.3: Standard spherical video coding pipeline

using a standard video coder. At the decoder, the projected video is decoded, and mapped back to the sphere to obtain the reconstructed spherical video. As previously explained, employing a standard video coder in this manner is highly suboptimal, as it fails to characterize accurately natural motion in spherical video, due to the warping introduced by the geometric projection.

Spherical video coding techniques

A few approaches have been proposed recently to either model motion in 3-D space or to manage discontinuities between cube faces. Li et al. proposed a 3-D translational motion model [10, 11] in which a block of pixels is mapped to the sphere and then linearly translated in 3-D space. The translated pixels must then be re-projected onto the sphere, causing distortion. The approach in [12] models motion of objects on a plane tangential to the sphere. The approaches in [13] and [14] consider projection onto multiple cube faces, and try to minimize errors due to discontinuities across face boundaries. A motion vector scaling approach is proposed in [15] to reduce the cost of motion vectors.

A closely related problem is that of motion compensated prediction in video captured with fish-eye cameras, where projection to a plane also leads to significant warping. A few interesting approaches have been proposed to address this problem in [16, 17], but these are not applicable to motion under different projection geometries for 360° videos.

A spherical rotation approach was proposed recently in [18] for encoding ERP. Their motivation was to identify so-called static points in an ERP video and then rotate the sphere such that the static points are the new poles of the rotated ERP. The algorithm

in [18] relies on local statistics to find a rotation angle that would help standard 2-D motion compensated prediction in the ERP domain. This method is not extendable to other projection formats.

2.3 Prediction Framework with a Rotational Motion Model

Since motion compensation in the projected domain lacks a precise physical meaning, we propose to perform motion compensation directly on the sphere. Let us consider a block of pixels in the current frame in the projected domain, which we seek to predict from the reference frame. An example of such a block in the ERP domain is illustrated in Fig. 3.2(a). We first map the block of pixels in the current frame to the sphere using the inverse projection mapping. The example block in Fig. 3.2(a) mapped back to the sphere is illustrated in Fig. 3.2(b). Let the center of this coding block in the projected domain correspond to vector $\mathbf{v}$ on the sphere. Our proposed motion search grid around the vector $\mathbf{v}$ is described next.

2.3.1 Proposed Motion Search

As previously mentioned, one of the main shortcomings of performing motion search in the projected domain is that the corresponding (on the sphere) search range, pattern and precision vary across the sphere. Since we propose to perform motion compensation directly on the sphere we overcome such arbitrary variations and employ the same search pattern for blocks everywhere on the sphere, agnostic of the projection geometry.

Let $\{(m, n)\}$ be the set of integer motion vectors and let R be the predefined search range, i.e., $-R \leq \{m, n\} \leq R$. We treat $\mathbf{v}$, the vector on the sphere corresponding to the

center of the current prediction unit, as if it were the vector on the sphere corresponding to zero yaw and pitch. An integer motion vector (m, n) then defines the rotation of $\mathbf{v}$ to a new point $\mathbf{v}'$ whose spherical coordinates (ϕ', θ') are given by:

$$\phi' = m\Delta\phi, \qquad \theta' = n\Delta\theta \tag{2.2}$$

where, $\Delta\phi$ and $\Delta\theta$ are predefined step sizes. Let H denote the height of the ERP frame, then $\Delta\theta$ is chosen to be $\frac{\pi}{H}$ as it corresponds to the change in the pitch (elevation) when we move by a single integer pixel in vertical direction. Similarly, $\Delta\phi$ is chosen to be $\frac{2\pi}{W}$, where W denotes the width of the ERP frame. For cube projections, the step sizes are chosen to be $\frac{\pi}{2W}$ where W corresponds to the width of the cube face. For each successive stage of motion vector refinement, the step sizes are correspondingly halved, leading to a direct correspondence with the hierarchical motion vector refinement in HEVC.

2.3.2 Proposed Rotation of the Block

Once we have the new vector $\mathbf{v}'$ corresponding to a motion vector (m, n), we rotate $\mathbf{v}$ to $\mathbf{v}'$ along the geodesic from $\mathbf{v}$ to $\mathbf{v}'$, via the Rodrigues' rotation formula [19]. This formula gives an efficient method for rotating a vector $\mathbf{v}$ in 3D space about an axis defined by unit vector $\mathbf{k}$, by an angle α. Let (x, y, z) and (u, v, w) be the coordinates of the vectors $\mathbf{v}$ and $\mathbf{k}$ respectively. The coordinates of the rotated vector $\mathbf{v}'$ will be:

$$\begin{aligned}
x' &= u(\mathbf{k}\cdot\mathbf{v})(1-\cos\alpha) + x\cos\alpha + (-wy+vz)\sin\alpha, \\
y' &= v(\mathbf{k}\cdot\mathbf{v})(1-\cos\alpha) + y\cos\alpha + (wx-uz)\sin\alpha, \\
z' &= w(\mathbf{k}\cdot\mathbf{v})(1-\cos\alpha) + z\cos\alpha + (-vx+uy)\sin\alpha
\end{aligned} \tag{2.3}$$

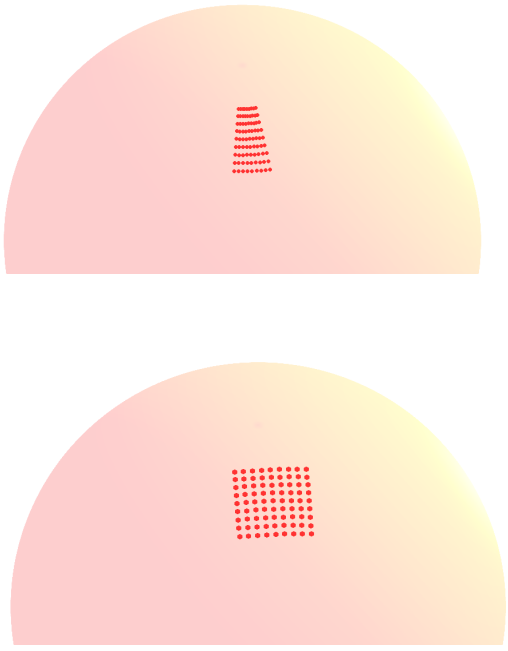


Figure 2.4: Comparison of motion search grids: Search pattern defined in projected domain in HEVC mapped to sphere (top) and proposed location invariant search pattern (bottom)

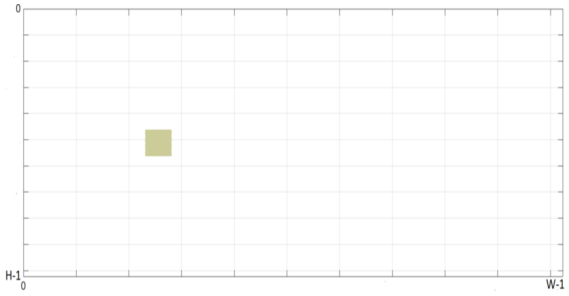
where $\mathbf{k} \cdot \mathbf{v}$ is the dot product of vectors $\mathbf{k}$ and $\mathbf{v}$. Since we want to rotate $\mathbf{v}$ to $\mathbf{v}'$ along the geodesic from $\mathbf{v}$ to $\mathbf{v}'$, we calculate the corresponding axis of rotation $\mathbf{k}$ and angle of rotation α, to employ Rodrigues' rotation formula. The axis of rotation $\mathbf{k}$ is the vector perpendicular to the plane defined by the origin, $\mathbf{v}$ and $\mathbf{v}'$ and is obtained by taking the cross product of vectors $\mathbf{v}$ and $\mathbf{v}'$, i.e,

$$\mathbf{k} = \frac{\mathbf{v} \times \mathbf{v}'}{|\mathbf{v} \times \mathbf{v}'|}. \tag{2.4}$$

The angle of rotation is given by,

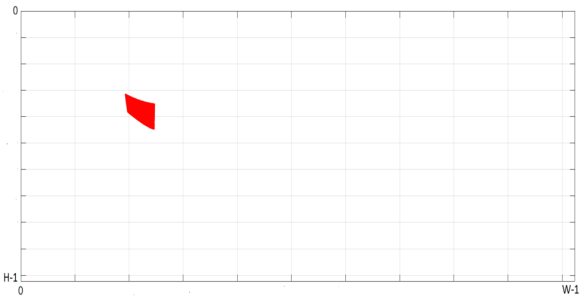
$$\alpha = \cos^{-1}(\mathbf{v} \cdot \mathbf{v}'). \tag{2.5}$$

Given this axis and angle, we rotate all the points in the current block with same rotation operation. Rotation of block in Fig. 3.2(b) along the geodesic from $\mathbf{v}$ to $\mathbf{v}'$ is illustrated in Fig. 3.2(c). After rotation, we map the rotated block to the reference frame using the forward projection. An illustration of rotated block mapped back to ERP domain is shown in Fig. 3.2(d). Since the projected location might not be on the sampling grid of the reference frame, we perform interpolation in the reference frame to get the pixel value at the projected coordinate. The proposed motion compensation technique is summarized in Algorithm 1.

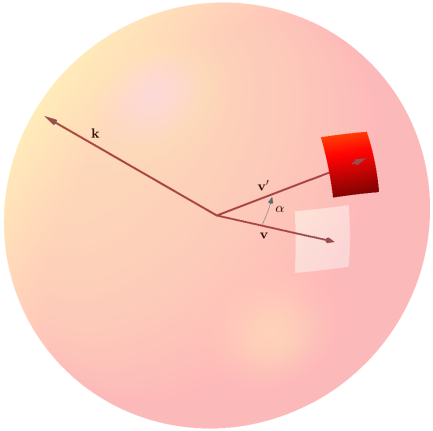


(a) A block in current ERP frame

(b) Block mapped to sphere



(d) Rotated block mapped to reference ERP frame



(c) Geodesic rotation of block on sphere

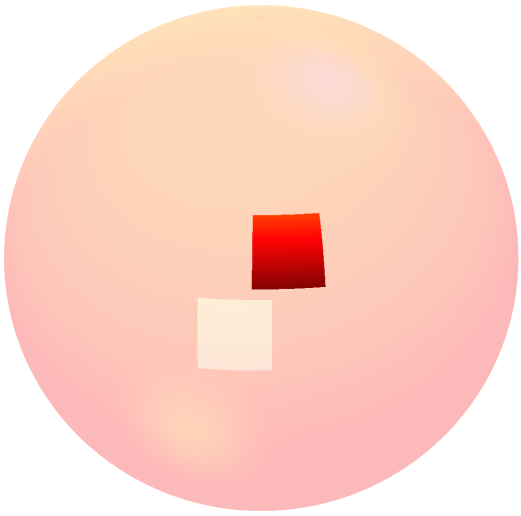
Figure 2.5: Steps in motion compensation with rotational motion model

Algorithm 1: Proposed motion compensation technique

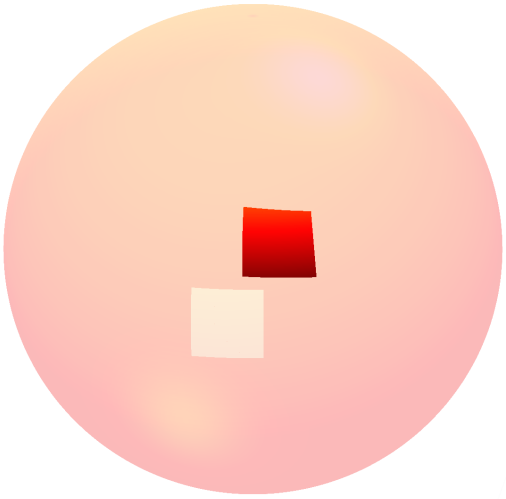
1: Map the block of pixels in the current coding unit on to the sphere.
2: Define a search pattern around the center of the block $\mathbf{v}$ as discussed in 2.3.1, to get the possible set of reference locations $\{ \mathbf{v}' \}$.
3: Define a rotation operation which rotates $\mathbf{v}$ to $\mathbf{v}'$ along the geodesic from $\mathbf{v}$ to $\mathbf{v}'$.
4: Rotate all the pixels in the block with the rotation operation defined in Step 3.
5: Map the rotated coordinates on the sphere to the reference frame in projected geometry.
6: Perform interpolation in the reference frame to get the required prediction.

2.3.3 Comparison of Motion Models

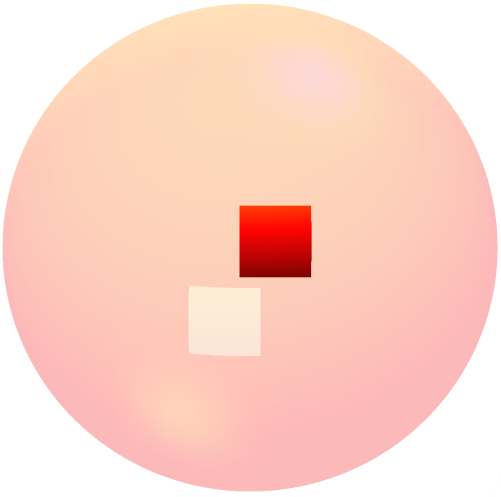
Different motion compensation techniques lead to different shape changes of the object on the sphere. Fig. 2.6 illustrates the differences in the proposed approach, the motion model proposed in [10], and the motion compensation in HEVC. Marked in yellow is the block of pixels in ERP projected on to the sphere. The pixel locations in the reference frame derived based on different motion models are marked in red. Translation in ERP leads to a shrinkage of the block as we move away from the equator and is clearly seen in Fig. 2.6(a). As discussed earlier, 3D translation followed by projection on to sphere leads to change in shape and size of the block which is clearly seen in Fig. 2.6(b). The proposed approach preserves the shape and size of the block which is illustrated in Fig.2.6(c). While both our approach and the approach in [10] perform motion estimation on the sphere, our approach significantly differentiates in that the motion model is in terms of rotations on the sphere instead of translation in 3D space. Moreover, the search pattern in [10] depends on the projection geometry and varies across the sphere in contrast to the fixed search pattern employed in the proposed approach.



(a) Motion Compensation in HEVC



(b) 3D translation motion model



(c) Proposed rotational motion model

Figure 2.6: Comparison of motion models: Model effect on block shape as seen on sphere

2.4 Experimental Results

2.4.1 Simulation Settings

The proposed encoding procedure was implemented with HM-16.15 [20] as the video codec. Geometry conversion and the sample rate conversion were performed using the projection conversion tool 360Lib-3.0 [21]. The proposed method was tested over five video sequences [22], [23] and [24]. The first one second of these videos were encoded at four QP values of 22, 27, 32 and 37 in random access profile. We provide results with ERP and ECP as the low resolution projection formats. ERP is encoded at 2K resolution. The face width for ECP is chosen to be 576 so that the total number of samples is approximately the same as ERP, namely, 2K. We use sinc interpolation at $\frac{1}{64}^{th}$ pixel accuracy to derive prediction signal from the reference frame.

2.4.2 Objective results

For objective results, bit-rate reduction is calculated as per [25] over standard HEVC encoding technique for all the approaches. We measured the distortion in terms of end-to-end weighted spherical PSNR [26], as recommended in [9] and [27]. Our nearest competitor is the 3-D translation model [11]. Table 2.1 compares the 3-D translation model and rotational motion model in terms of bit-rate reduction over HEVC, for the Y component, and provide results in conjunction with projections ERP and ECP. Note that 3-D translation model gives worse results than HEVC for ECP and thus has not been included for comparison. It is clear from the table that rotational motion model gives significant gains when compared to 3-D translational model and 2-D translation model employed in projected domain by HEVC. The rate-distortion (RD) curves for the *bicyclist* and *balboa* sequences for different projection formats are shown in Fig. 4.7-3.7.

Overall, the results demonstrate consistent performance gains at all bit-rates and across different geometries.

Table 2.1: BD-rate gains in % for Y-component by employing rotation motion model over HEVC using translation motion model in projected domain

Geometry	Sequence	3-D Translation model in [11]	Rotational model
ERP	Glacier	13.9	16.9
	Chairlift	3.5	7.5
	Bicyclist	2.4	12.7
	Driving in Country	1.7	12.4
	Kiteflite	2.1	5.1
	Average	**4.7**	**10.9**
ECP	Glacier		6.0
	Chairlift		2.5
	Bicyclist		4.2
	Driving in Country		5.1
	Kiteflite		1.5
	Average		**3.9**

2.4.3 Subjective results

To get subjective results, we compressed videos with specified target bit-rate with HEVC anchor and the proposed method. Fig. 2.9 shows significant improvement in the visual quality for example frames from the "chairlift" sequence with the proposed motion model as compared to HEVC based encoding at the same bit rate. Specifically, HEVC has a 'washed out' appearance since many details of the texture are lost while the proposed method retains much of the visual textures.

2.5 Conclusions

This chapter proposes a novel rotational motion model for 360^o video coding, which effectively captures the motion of the objects directly on the sphere. The chapter also proposes a motion search pattern that is independent of the position of the block on the sphere. The proposed framework retains the shape and size of the object after motion, while being agnostic of the projection geometry. The substantial gains compared to standard HEVC and other motion models, demonstrate the effectiveness of the proposed technique.

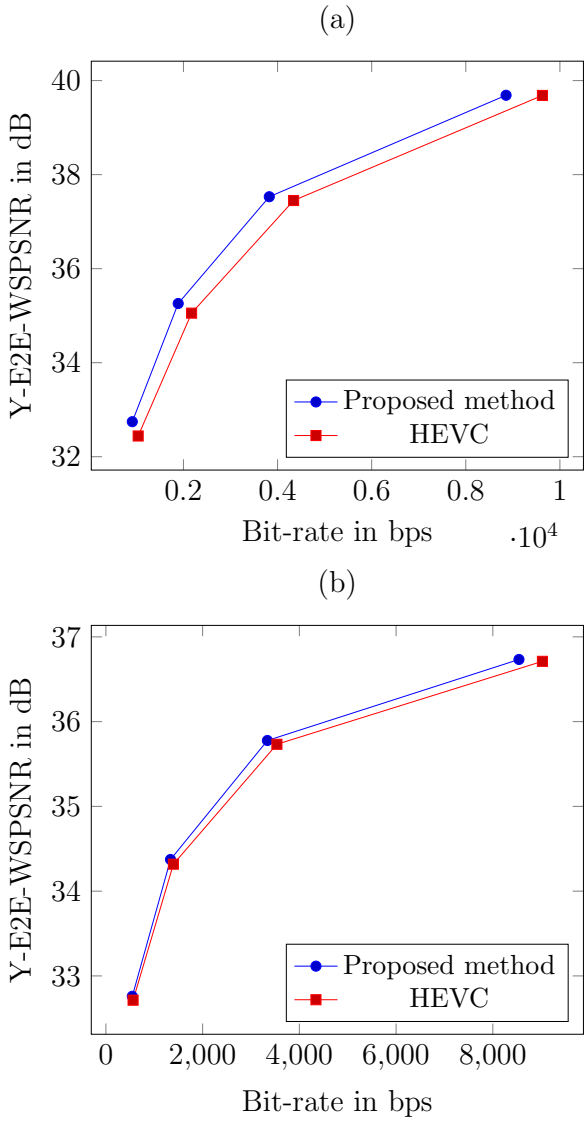


Figure 2.7: RD curves for (a) *glacier* and (b) *chair* sequences with ERP as the projection format

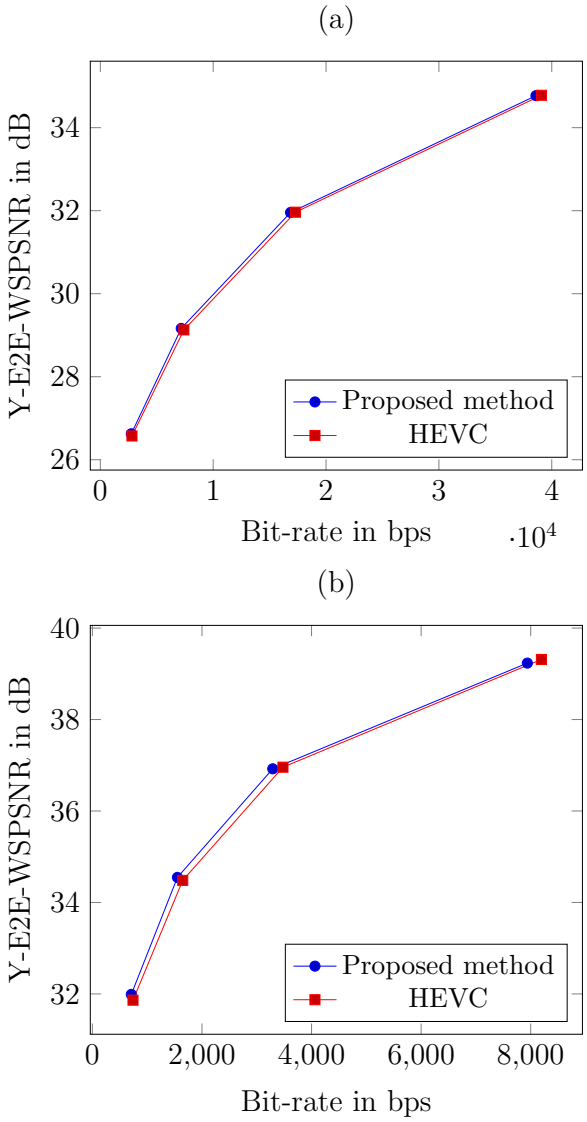


Figure 2.8: RD curves for (a) *bicyclist* and (b) *glacier* sequences with ECP as the projection format

Figure 2.9: Subjective comparison for chairlift sequence encoded at constant bit-rate with ERP as projection format: HEVC (top) and proposed method (bottom)

Chapter 3

Geodesic Translation Motion Model for Spherical Video Coding

3.1 Introduction

In this chapter, we consider an important class of spherical video signals that involves motion dominated by camera translation. Such signals are frequently encountered in numerous applications, including robotics and navigation, sports and outdoor activities, etc. The prevalence of this class of video signals, and the enormous amount of data generated, necessitate the design of efficient compression tools that are tailored to this scenario. None of the recent work discussed in the previous chapter account for camera motion during motion compensation. In this chapter, we propose a motion compensation procedure to capture *on the sphere* the accurate motion field that is due to camera translation. An important basic observation is that straight lines in 3-D space map to geodesics on the sphere. Thus, in the case of camera translation, all surrounding static objects exhibit relative motion along straight lines in 3-D space (parallel to the camera velocity vector), which is mapped to perceived motion along geodesics on the sphere.

More specifically, the proposed method builds on the core realization that all static points in the environment are perceived to move on the sphere along their respective geodesics, namely, geodesics that intersect at the two points where an axis aligned with the camera velocity vector "pierces" the sphere. This characterization of the motion on the sphere also accounts for the perspective deformations that are due to camera motion. Specifically, it captures the magnification effects as objects approach the camera, and vice versa.

Having established the nature of perceived motion trajectories of surrounding objects on the sphere, the approach is further refined to characterize the rate of translation of pixels along their geodesics. A mathematical analysis sheds light on the rate of geodesic translation of pixels as related to the corresponding elevation of the pixels on the sphere with respect to the camera velocity axis. Based on this realization, we propose a motion vector modulation scheme, wherein, the geodesic translation prescribed for the center of a block of pixels, is modulated to extract refined individual motion vectors for the pixels in the block, which account for their respective degrees of elevation. The motion vector modulation scheme captures the variations in perceived motion across pixels in the block to yield significantly improved prediction and consequently additional coding gains. Moreover, since a 1-D motion vector is (largely) sufficient to capture the motion that is mostly along the geodesics, unlike the general 2-D motion vector required by all existing approaches, the proposed approach achieves significant savings in side information bit rate to convey motion vectors to the decoder. Nevertheless, to correct for the possibility of non-stationary objects whose motion is independent of the camera motion, we allow for a second motion component to capture lateral displacement (away from the geodesic). Overall, pixels in a prediction unit are mapped to the sphere, moved along the geodesics defined by the camera motion, where the rate of translation of each pixel along its geodesic is determined by the proposed modulation scheme, and finally mapped back

to the reference frame in the projected geometry to derive the ultimate prediction signal.

Another focus of this chapter is on the motion search module. The motion model efficacy in video coders largely depends on an effective motion search procedure. In the context of spherical videos, a fixed search pattern in the projected plane induces a spatially varying search pattern on the sphere, which is unnatural and undesirable. To overcome this shortcoming, this chapter proposes to define search grid on the sphere making it agnostic of the geometry. Further, the grid reflects the expected geodesic motion of the objects due to camera motion.

Thus, in contrast with standard spherical video coders that perform their motion analysis and compensation in the "warped" projected domain, the approach proposed herein effectively conducts its analysis in the natural domain of the sphere. It is important to emphasize that the proposed motion estimation and compensation is hence independent of the projection format.

3.2 Geodesic Motion Compensated Prediction

This section presents the proposed geodesic translation motion model. We first illustrate, in a simple setting, the perceived motion of objects on the sphere, due to underlying camera motion. Based on these observations, we introduce a geodesic-based framework for motion compensated prediction. We then focus on the precise rate of translation of pixels along geodesics, and refine the standard motion vector definition for a block, by proposing a motion vector modulation framework to capture the exact motion of each pixel in the block. Finally, we introduce a location invariant motion search grid that effectively circumvents shortcomings of standard motion estimation in the projected domain.

3.2.1 Motion Compensation by Geodesic Translation

Perceived Motion on the Sphere

In order to illustrate the perceived motion on the sphere, resulting from translational motion of the camera, consider a viewer at the origin, enclosed by a sphere, as shown in Fig. 3.1. The viewer sees point P, in the 3-D environment, through its projection point S on the sphere. As the camera moves forward according to its velocity vector v, the stationary point P is perceived as displaced to point P$'$ relative to the viewer. Clearly, its corresponding projection on the sphere advances along the arc S-S$'$. It is important to note that the arc S-S$'$ is a segment of a geodesic that connects the two points where the camera velocity axis intersects the sphere. Building on this observation, we see that given constant translational motion of the camera, static surrounding points are perceived as moving on the sphere along their *respective* geodesics, which all intersect at the poles of the camera motion axis. It follows from these observations that the most natural way to capture the perceived motion of objects is by characterizing their geodesic translation on the sphere, in sharp contrast with the complex non-linear characterisation that would be necessary in the projected domain. Thus, based on the above observations, we propose a motion compensation procedure on the sphere, as discussed next.

Geodesic-based Motion Compensation

As we observed, in cases of video dominated by camera motion, it is most natural to capture the perceived motion directly on the sphere. Specifically, motion compensation on the sphere will be performed as translation of a block along appropriate geodesics. The proposed method assumes that the direction of camera motion is known, as most smart phones and 360 cameras include sensors such as accelerometer, gyroscope, etc., to detect and estimate motion, which can be fed to the video encoder. However, when

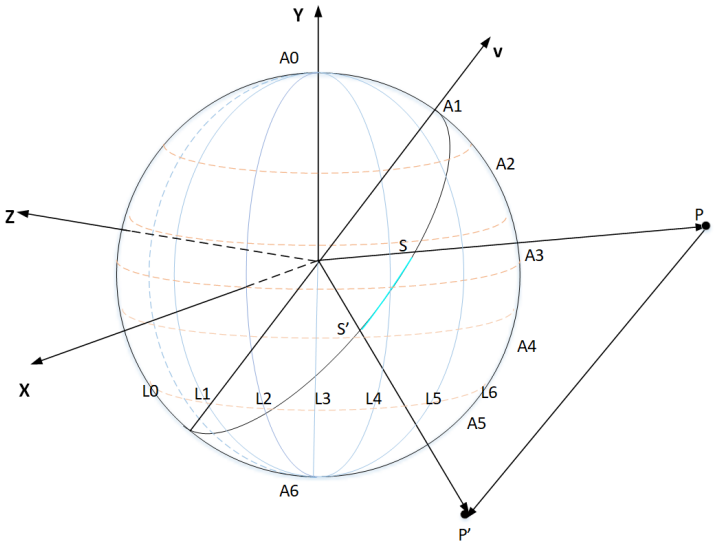
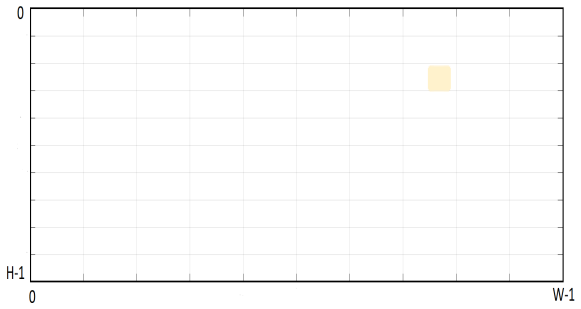


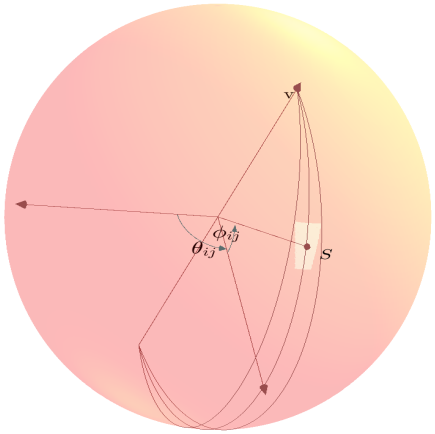
Figure 3.1: A static point's perceived trajectory on the sphere due to camera motion

such information is not available, it can be estimated directly from the video (see, e.g., [28]). Given the camera velocity vector, we define geodesics that intersect at the two points where an axis aligned with this vector pierces the sphere. In other words, these two points are the "camera motion poles". With this setup in place, the specific three steps are specified and explained next:

a) Sphere mapping: Consider a block of pixels in the current frame, which needs to be predicted with motion compensation. Fig. 3.2a illustrates one such block in an ERP frame. We first project the block onto the sphere. For simplicity of presentation, let us define spherical coordinates with respect to the camera motion vector. Specifically, for pixel (i, j) in the prediction block, let (θ_{ij}, ϕ_{ij}) be the spherical coordinates relative to the polar axis defined by the camera velocity vector. A block of pixels mapped to the sphere and the spherical coordinate system with respect to camera translation vector is shown in Fig. 3.2b. This step facilitates work on the sphere in a manner that is entirely agnostic of the projection format.

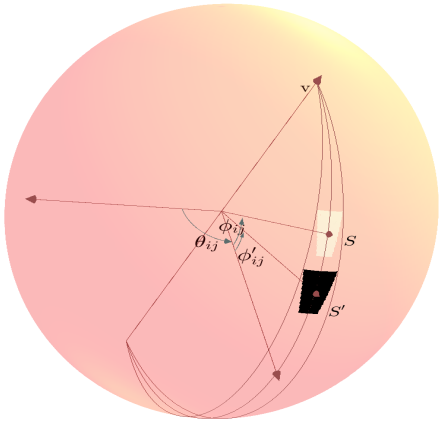


(a) Block of pixels in the projected domain



(b) Block mapped to the sphere

(d) Block mapped back to the projected domain



(c) Translation along geodesics

Figure 3.2: The geodesic translation motion model

b) Geodesic translation: Given a motion vector (m, n), we move a pixel on the sphere along its geodesic to arrive at the spherical coordinates of the reference pixel as,

$$\theta'_{ij} = \theta_{ij} + m\Delta\theta_s, \phi'_{ij} = \phi_{ij} + n\Delta\phi_s \tag{3.1}$$

where $\Delta\theta_s$ and $\Delta\phi_s$ are predefined step sizes (more on the design choices in the experimental section). Note that if the video motion field is entirely determined by translational motion of the camera, we only expect motion along the geodesics with no "lateral" motion, i.e., $\theta'_{ij} = \theta_{ij}$. From the compression perspective, this results in notable bit-rate savings in terms of significant reduction in the side information allocated to motion vectors. Nevertheless, we allow for 2-D motion vectors to account for actual object motion, unrelated to camera translation. Fig. 2.3b illustrates the geodesic translation of the block for a static object. It is evident from the figure that the proposed approach accounts for perspective distortions. Specifically, in the illustration, the object appears magnified as the camera approaches the object. Moreover, the motion vectors convey the amount of geodesic translation on the sphere, unlike motion vectors in the projected domain of standard techniques which afford little physical meaning or interpretation.

c) Projection and interpolation: The reference frame is still in the 2-D projection format. Thus, after geodesic translation of the block on the sphere, the translated pixels on sphere are projected to the reference frame. The projected coordinates may not be on the sampling grid of the reference frame. We thus perform interpolation in the projected domain to obtain the value of the prediction signal at the projected coordinate. Fig. 3.2d illustrates the reference block obtained by geodesic translation and mapping back to the projected domain.

3.2.2 Rate of Displacement: Motion Vector Modulation

The motion compensation so far exploits the nature of perceived motion on the sphere, due to camera motion, and translates all pixels in a block by the *same* distance on their respective geodesics. In this section, we further examine the rate of displacement of these pixels. Intuitively, it is easy to see that the rate of displacement of static surrounding points along their geodesics is inversely related to their depth, thereby reflecting the well known parallax effect, albeit in the context of spherical video. Moreover, even for objects at constant depth, the rate of translation depends on their position on the sphere. Mathematical analysis sheds light on the exact relationship of the rate of displacement with object depth and the elevation of the block on the sphere. This analysis leads to a motion vector modulation scheme that captures the exact motion of each pixel in a block.

Geodesic Displacement Analysis

In order to analyze the exact motion of each pixel along its geodesic, let us focus on the plane defined by P, P′ and the origin O, as shown in Fig. 3.3. Let ϕ be the elevation of point P with respect to the camera motion axis (i.e., relative to the corresponding "equator"), and let $\Delta\phi$ be the change in elevation due to camera translation. Applying the law of sines to triangle POP′ we get,

$$\frac{|\mathrm{OP}|}{\sin(\angle \mathrm{OP'P})} = \frac{|\mathrm{PP'}|}{\sin(\angle \mathrm{P'OP})} \tag{3.2}$$

It is easily seen that $\angle \mathrm{OP'P} = \frac{\pi}{2} - (\phi + \Delta\phi)$. OP is the depth of the point, denoted as d, and PP′ is the amount of camera translation denoted as t. We thus have the following

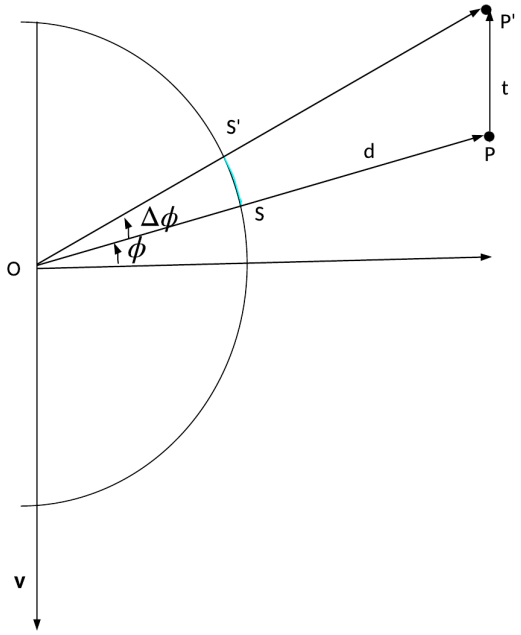


Figure 3.3: Relation between geodesic displacement, elevation, depth and camera translation

relation,

$$\frac{d}{t} = \frac{\cos(\phi + \Delta\phi)}{\sin(\Delta\phi)} \tag{3.3}$$

To motion-compensate a block of pixels, we make the simplifying assumption that all pixels in the block are approximately at the same depth from the origin. In case the pixels do not have any constant depth, the encoder can always split the block via quad-tree partitioning and get blocks of approximately constant depth. Thus, the ratio $\frac{d}{t}$ remains constant for all pixels in the block. This yields a relationship between the elevation of a pixel ϕ and the corresponding elevation change $\Delta\phi$. Armed with this observation, we extend the motion compensation procedure in 3.2.1 to account for the actual rate of translation of pixels.

Motion Compensation with Modulated Motion Vectors

Similar to motion compensation summarized in sub-section 3.2.1, a block of pixels that needs to be inter-predicted is mapped to the sphere. Let (θ_{ij}, ϕ_{ij}) be the corresponding spherical coordinates with respect to the camera translation vector. Let (θ_c, ϕ_c) be the spherical coordinates of the center of the block after mapping to the sphere. Given a motion vector (m, n), the center of the block is translated along its geodesic as,

$$\theta'_c = \theta_c + m\Delta\theta_s, \phi'_c = \phi_c + n\Delta\phi_s \tag{3.4}$$

where $\Delta\theta_s, \Delta\phi_s$ are predefined step-sizes similar to (3.1). Let us specifically denote the change in elevation by $\Delta\phi_c$, i.e, $\Delta\phi_c = n\Delta\phi_s$. Now, for a pixel P_{ij} in the block, under the assumption of constant depth across pixels in a block, we obtain from (3.3):

$$\frac{\cos(\phi_{ij} + \Delta\phi_{ij})}{\sin(\Delta\phi_{ij})} = \frac{\cos(\phi_c + \Delta\phi_c)}{\sin(\Delta\phi_c)} = \frac{d}{t} = k \tag{3.5}$$

where $\Delta\phi_{ij}$ is the change in elevation of P_{ij} on the sphere and k is a constant. By simple trigonometry we obtain the relationship,

$$\Delta\phi_{ij} = \tan^{-1}(\frac{\cos\phi_{ij}}{k + \sin\phi_{ij}}) \tag{3.6}$$

Thus, given the change in elevation of the center of the block, the elevation change for each individual pixel, or the amount of translation along its respective geodesic, is modulated according to (3.6). The pixels are thus translated to points with spherical coordinates given by,

$$\theta'_{ij} = \theta_{ij} + m\Delta\theta_s, \phi'_{ij} = \phi_{ij} + \Delta\phi ij \tag{3.7}$$

The translated pixels are then mapped to the reference frame to derive the prediction signal. Here too, the mapped pixel will often not fall on the sampling grid of the reference frame. We thus perform interpolation as needed to obtain the properly sampled prediction signal. The extended motion compensated prediction that accounts for the rate of displacement of individual pixels can thus be summarized as:

- A block of pixels is mapped to the sphere and the spherical coordinates (θ_{ij}, ϕ_{ij}) are derived with respect to the camera translation vector
- For a given motion vector (m, n), the block center on the sphere is translated according to (3.4).
- The change in elevation for each pixel in the block is calculated according to (3.6) and they are translated according to (3.7).
- The translated pixels are mapped to the reference frame to derive the prediction signal.

Note that motion vector modulation proves particularly effective in low bit-rate coding, since the encoder tends to use larger blocks, where motion vector modulation has significant impact in accurately capturing motion variations within the block.

3.2.3 Motion Search Grid Adaptation

To gain the full benefit of the proposed motion model, we rely on efficient motion estimation procedures, the efficacy of which critically depends on the motion search grid. We first consider the shortcomings of the standard search grid, and then propose means to overcome these shortcomings, as well as to adapt the grid to account for camera motion.

Shortcomings of the Standard Search Grid

As observed in the discussion of projection formats, uniform sampling in the projection plane induces non-uniform sampling on the sphere. Thus, employing a fixed search pattern in the projection plane leads to spatially varying search patterns on the sphere. This observation is illustrated for the case where ERP is employed, in Fig. 3.2.3(a). Observe how the search pattern varies spatially on the sphere, in a way that depends on the arbitrary North-South pole. For the current scenario with camera motion, motion of the center of the block is expected to be along its respective geodesic. However, the search grid doesn't exploit this observation and need not have grid points along the expected geodesic. Moreover, for a mere 'approximation' of the motion of the center of the block, we need a 2-D motion vector in the projected domain. Thus, there is a clear motivation for the optimization of the search grid.

Proposed Search Grid Adaptations

To address the above mentioned shortcomings, we define a search grid on the sphere that directly captures the expected motion of the center of the block due to camera motion. Given a motion vector (m, n), the first component is used to capture change in yaw and the second component to capture change in elevation *with respect to the camera velocity vector*. This interpretation of motion vectors leads to the search pattern illustrated in Fig. 3.2.3(b). The proposed approach offers two benefits: It eliminates dependence on the arbitrary parameters of the projection (e.g., ERP's dependence on the North-South pole), and it explicitly captures the expected geodesic translation of the center of the block. Further, all the grid points on this geodesic corresponds to the case where $m = 0$, i.e, there is no change in yaw. Thus, for static objects, we have motion vectors that are 1-D, leading to bit-rate savings in side-information.

In case of object motion that is independent of camera motion, we use the component m to capture any "lateral" motion. Specifically, the component of the motion vector 'm' captures the change in azimuth as $m\Delta\theta$. It is important to note that, as we move away from the equator towards a camera motion pole, $m\Delta\theta$ corresponds to smaller lateral displacement. In terms of motion search pattern during motion estimation, this corresponds to a "shrinking" search grid, as illustrated in Fig. 3.2.3(b). This leads to suboptimality in estimating object motion that is independent of camera motion. It also results in excess penalty in side information needed to convey the lateral motion to the decoder, since small lateral displacement for blocks closer to the pole translate to large values of m. Thus, we need further search grid optimization to account for object motion with the following desired characteristics: i) The grid range should be agnostic of the elevation of the block with respect to the camera velocity vector. ii) For the scenario with dominant camera motion, we expect less 'lateral displacement', which motivates denser grid points near the center of the block, to capture the change in azimuth. However, we still must preserve the search range to handle occasional large object motions. To achieve the first desired characteristic, *only for the purpose of defining the search grid*, we proceed as if the block were at the equator, thereby eliminating the dependence of the search grid on the elevation of the block. To achieve dense grid close to the center of the block along the azimuth and yet not compromising on the search range, we define grid points in the spirit of equal area projection (EAP). In EAP, the longitudes are sampled such that, the sampling density decays as the cosine of the elevation, resulting in dense sampling close to the equator and spare sampling as we move towards the pole. We exploit this observation to have non-uniform density of motion vectors *along the azimuth* such that, we have dense sampling close the center of the block and spare sampling as we move away from center. Specifically, given a motion vector (m, n), the component m now represents

the change is azimuth as,

$$\Delta\theta_c = Ksin^{-1}(\frac{m}{R}), -R \leq m \leq R \tag{3.8}$$

where R is the search range. The choice of K determines the search range on the sphere, since $m = \pm R$ corresponds to $\Delta\theta_c = \pm K\frac{\pi}{2}$. For ERP, the width W corresponds to field of view of 2π, so K is chosen to get a search range of $\frac{2\pi R}{W}$ rad on the sphere. This yields $K = \frac{4R}{W}$. Similarly, for cube projections, the face-width W corresponds to field of view of $\frac{\pi}{2}$. Thus, $K = \frac{R}{W}$. The proposed distribution of grid points along the azimuth is illustrated in Fig. 3.4, in comparison with the uniform distribution. Note how the proposed grid is denser near the center and becomes sparser as we move away, while maintaining the same search range. The proposed sampling pattern in conjunction with its agnostic nature with respect to the elevation of the block on the sphere, is illustrated in Fig. 3.2.3(c).

Combining all the coding tools that we have discussed, the overall motion estimation algorithm at the encoder is summarized in Algorithm 4. The decoder essentially does this operation only for the best motion vector.

3.3 On efficient implementation of the geodesic model

The proposed method consists of mappings between projection format and the sphere and interpolations in reference frame that can be computationally very expensive. In this section, we explain our efforts to reduce the complexity. We consider each step in the proposed method and discuss the relevant optimizations:

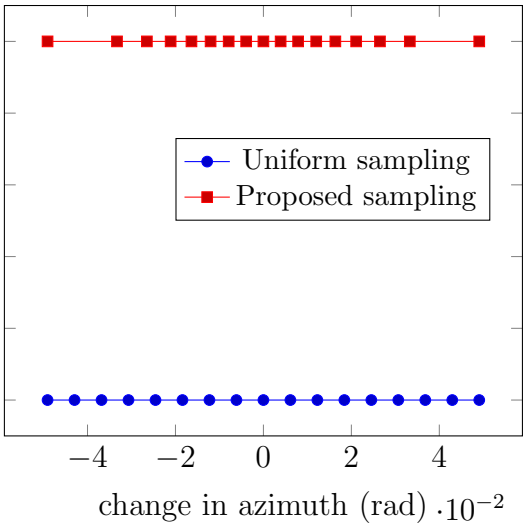


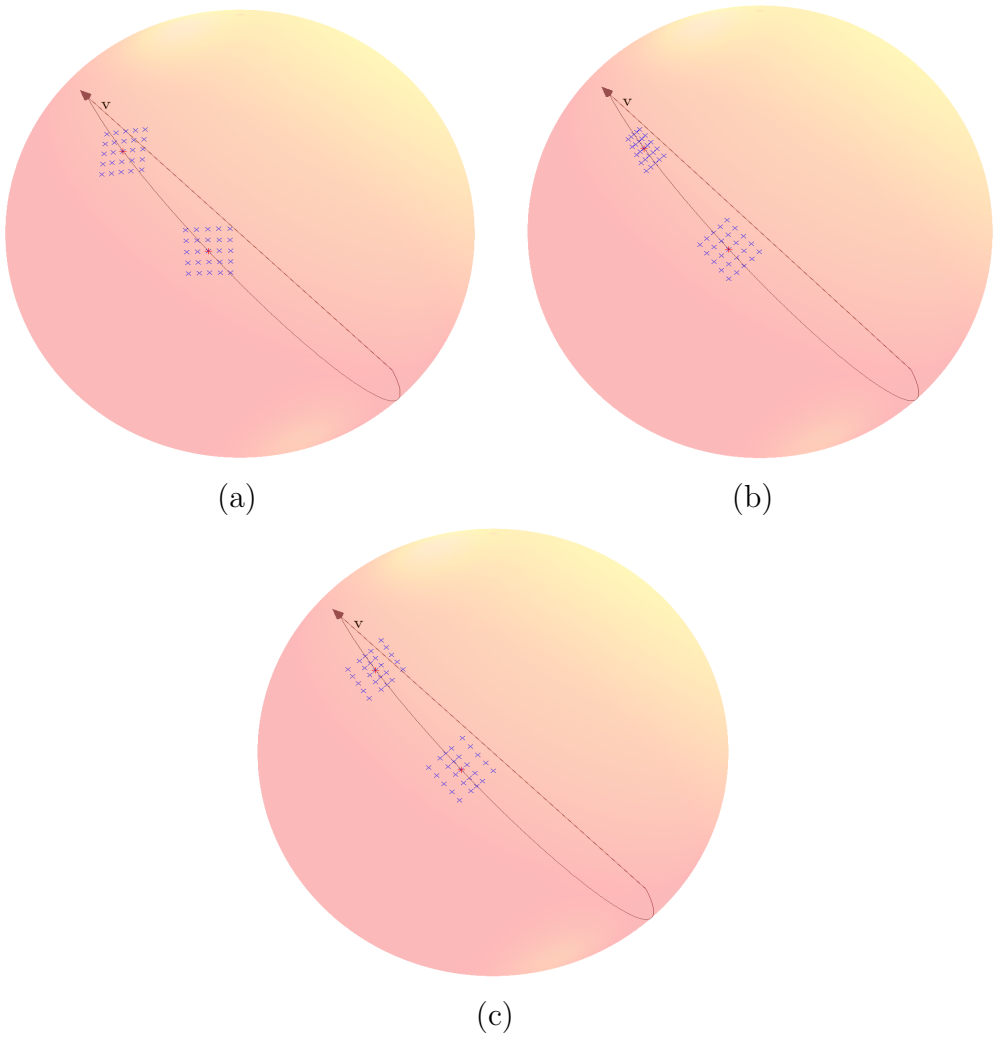
Figure 3.4: Comparison of the proposed EAP based distribution of motion vectors along the azimuth and uniform distribution of motion vectors

Sphere Mapping

The first step in the proposed method involves mapping a block from a plane onto the sphere and computing the spherical coordinates with respect to the camera velocity vector. Given the projection format, the set of samples on the sphere are fixed. Thus, we create a look-up table of the spherical coordinates for all the pixels in the projected domain. This is a one-time computation whose results can then be reused for all frames, during motion estimation and compensation. The created look-up table greatly alleviates the burden of mapping from projection format to the sphere for a block in a given frame.

Geodesic translation

For geodesic translation without modulated motion vectors, this step simply involves adding $(\Delta\theta, \Delta\phi)$ for all pixels in the block and doesn't call for much optimization. However, with motion vector modulation, we need trigonometric functions, for which we use look-up tables to reduce complexity.



(a) (b)

(c)

Figure 3.5: Comparison of motion search patterns: (a) HEVC search pattern defined in projection format, (b) Search pattern aligned with the expected geodesic motion of the center of the block and (c) Geodesic aligned search pattern optimized to account for object motion

Inverse projection and interpolation

After geodesic translation, the proposed method involves mapping pixels back to the reference frame and performing interpolation in the reference frame (in the projected domain). Mapping between pixels on the sphere to the projection plane often involves complex trigonometric operations, where again we employ look-up tables to minimize the computational burden. During motion estimation, it would be computationally expensive to perform higher order interpolation for every pixel in the block for each choice of motion vector. In order to mitigate this concern, for the initial integer motion estimation stage, we up-sample the reference frames and use nearest neighbor interpolation in the up-sampled reference frame to derive prediction signal. An up-sampling factor of four was seen to be a good trade-off between memory and performance. For successive motion vector refinements we use sinc interpolation in the reference frame at $\frac{1}{64}$ pixel precision. We note that the central focus of this chapter is to demonstrate the potential of the geodesic motion model and the above mentioned optimizations enumerate our initial efforts to reduce complexity.

3.4 Experimental Results

3.4.1 Simulation Settings

The proposed encoding procedure was implemented with HM-16.15 [20] as the video codec. Geometry conversion and the sample rate conversion were performed using the projection conversion tool 360Lib-3.0 [21]. The proposed method was tested over five video sequences [22], [23] and [24], which are dominated by translational motion of the camera. The first one second of these videos were encoded at four QP values of 22, 27, 32 and 37 in random access profile. We provide results with ERP and ECP as the low

Algorithm 2: Overall motion estimation algorithm

Map the block onto the sphere;
Let $(\theta_c.\phi_c)$ be the spherical coordinates of the center with respect to camera velocity vector;
Define search pattern as discussed in 3.2.3 ;
for *each point on sphere in search pattern* **do**
  (a) The center of the block is moved to (θ'_c, ϕ'_c) given by the search pattern;
  (b) Set $\Delta\theta_c = \theta'_c - \theta_c$ and $\Delta\phi_c = \phi'_c - \phi_c$;
  (c) Define $k = \frac{\cos(\phi_c + \Delta\phi_c)}{\sin(\Delta\phi_c)}$;
  for *each pixel in the block* **do**
    i) New azimuth is $(\theta_{i,j} + \Delta\theta_c)$;
    ii) New elevation is calculated as $\phi_{i,j} + \tan^{-1}(\frac{\cos\phi_{ij}}{k + \sin\phi_{ij}})$;
    iii) Map the new pixel on sphere to the projected domain;
    iv) Perform interpolation in the projected domain to get the prediction signal;
  end
  (d) Calculate the error between the original block and the predicted block;
end
Choose the best motion vector that minimizes prediction error.

resolution projection formats. ERP is encoded at 2K resolution. The face width for ECP is chosen to be 576 so that the total number of samples is approximately the same as ERP, namely, 2K. The step sizes $\Delta\theta_s$ and $\Delta\phi_s$ are chosen to be $\frac{\pi}{H}$, where H is the height of the ERP video. The corresponding step sizes for ECP with face-width W is chosen to be $\frac{\pi}{2W}$ since a face-width of W corresponds to a field of view of $\frac{\pi}{2}$rad. We use sinc interpolation at $\frac{1}{64}^{th}$ pixel accuracy to derive prediction signal from the reference frame.

3.4.2 Objective results

For objective results, bit-rate reduction is calculated as per [25] over standard HEVC encoding technique for all the approaches. We measured the distortion in terms of end-to-end weighted spherical PSNR [26], as recommended in [9] and [27]. In [29], we had already shown that the rotational model outperforms other existing approaches, and this is the reason it was selected here as leading (nearest) competitor. Table 3.1 compares the proposed method and rotational motion model [29] in terms of bit-rate reduction over HEVC, for the Y component, and provide results in conjunction with projections ERP, EAC and ECP, respectively. It is clear from the table that the new motion model tailored to the translation motion of camera gives significant gains when compared to models that do not properly account for camera motion. The rate-distortion (RD) curves for the *bicyclist* and *balboa* sequences for different projection formats are shown in Fig. 4.7-3.7. Overall, the results demonstrate consistent performance gains at all bit-rates and across different geometries. As regards the complexity, unoptimized encoder and decoder has complexity over 40x and 8x respectively compared to HEVC anchor. The proposed optimizations in section 3.3 cuts down the complexity to 10x-15x for the encoder and 3x for the decoder.

Table 3.1: BD-rate gains in % for Y component over HEVC for rotation motion model and the geodesic translation motion model

Geometry	Sequence	Rotational motion model in [29]	Proposed Method
ERP	Bicyclist	12.7	24.8
	Chairlift	7.5	14.5
	Broadway	1.8	22.6
	Balboa	3.2	29.7
	Harbor	4.6	35.9
	Average	**5.9**	**25.5**
ECP	Bicyclist	4.2	15.7
	Chairlift	2.5	6.8
	Broadway	0.6	6.7
	Balboa	1.6	8.8
	Harbor	0.9	2.2
	Average	**2.0**	**8.0**

3.4.3 Subjective results

To get subjective results, we compressed videos with specified target bit-rate with HEVC anchor and the proposed method. Fig. 3.8 shows significant improvement in the visual quality for example frames from the "balboa" sequence with the proposed motion model as compared to HEVC based encoding at the same bit rate. Specifically, we draw attention to the edges of the building where the proposed method has a crisp reconstruction as against to the highly distorted reconstruction of HEVC.

3.5 Conclusions

This chapter proposes a novel encoding technique for spherical videos with dynamics dominated by camera motion. The proposed approach leverages insights into the perceived motion of static objects on the sphere, and the perspective distortion due to camera motion. The motion model is agnostic of the projection format and the approach

is extendable to other geometries in a straightforward manner. Experimental results yield substantial bit rate reduction and demonstrate the effectiveness of the proposed framework.

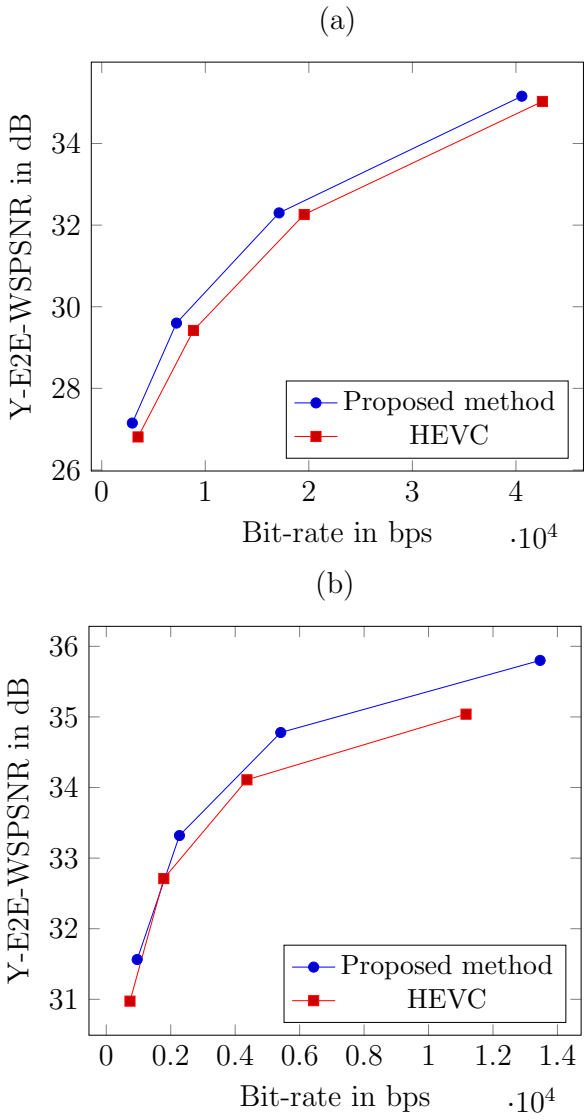


Figure 3.6: RD curves for (a) *bicyclist* and (b) *balboa* sequences with ERP as the projection format

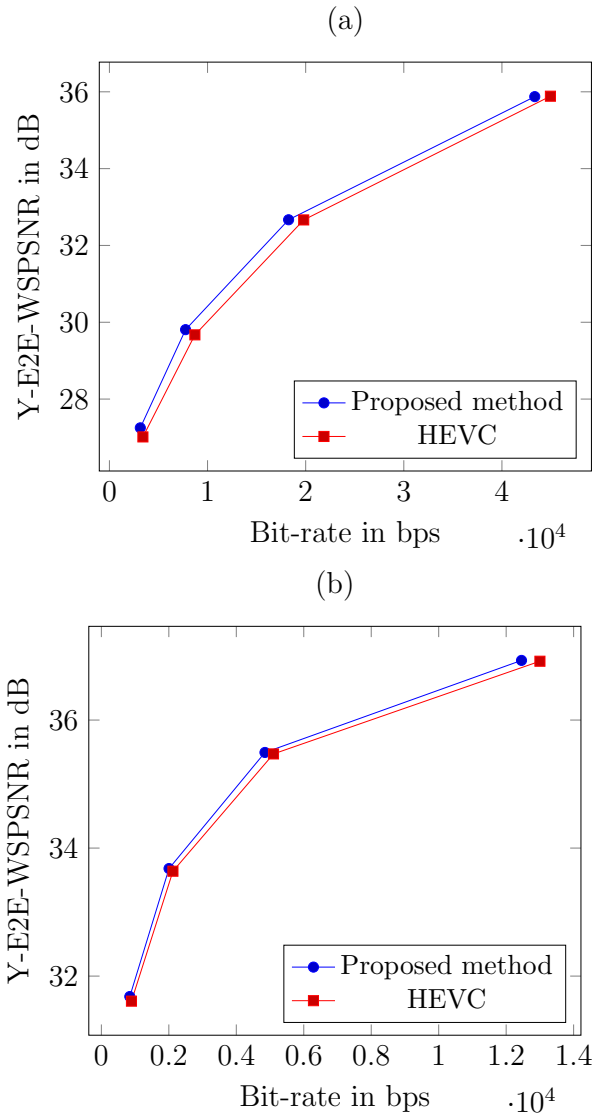


Figure 3.7: RD curves for (a) *bicyclist* and (b) *balboa* sequences with ECP as the projection format

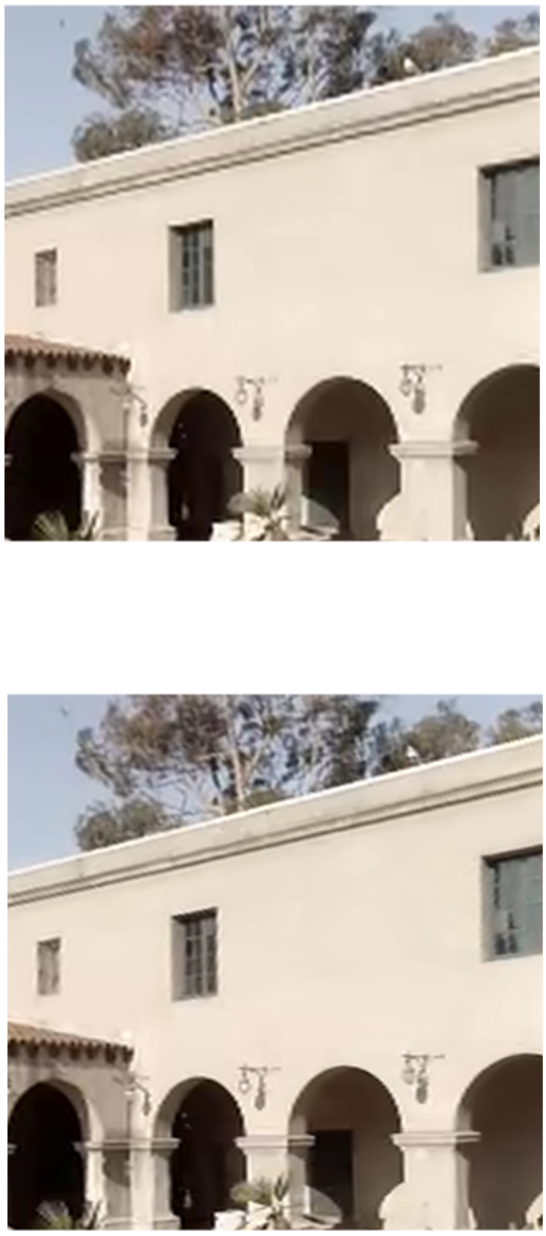


Figure 3.8: Subjective comparison for balboa sequence encoded at constant bit-rate with ERP as projection format. anchor (top) and proposed method (bottom)

Chapter 4

Efficient Predictor Mode Design for Adaptive Prediction

4.1 Introduction

Linear prediction is an integral part of most modern compression systems [1, 30, 31], tasked with removing temporal or spatial redundancies in signals. Often, the design of prediction filters assumes that the signal is stationary. However, most real-world signals are non-stationary and naturally call for adaptive compression systems. A common paradigm to achieve adaptivity involves block-based encoding, wherein the source signal is partitioned into blocks and the prediction filters can be adapted per block. However, sending per-block prediction filter specification would incur considerable overhead. Instead, a common, cost-effective approach is to design a 'codebook' of predictors, which is also available to the decoder, and have the encoder convey the index of the predictor (mode) used to predict a given block. The performance gains of such an adaptive system critically depend on efficient design of all prediction modes.

The design of a codebook of prediction modes poses several challenges. The design

can be viewed as 'quantization' of the prediction filter parameter space. The cost-function depends on both the codebook and the encoder decisions (assigning codebook entries to individual signal blocks). Note that the cost is piece-wise constant with respect to the encoder decisions (the encoder does not modify a decision until the block content changes sufficiently) which implies that the corresponding derivatives vanish almost everywhere. This makes it impossible to employ standard gradient-based algorithms. A common remedy is to design predictors in a "K-means" clustering fashion [32], wherein the design iterates between choosing the best prediction modes for the blocks (i.e., nearest neighbor rule for the encoding decisions) and then updating the prediction modes (centroid rule). It is well known that the performance of greedy approaches depends on initialization, and there is substantial risk of getting trapped in poor local optima. The prevalence of local optima, coupled with the piecewise constant property of the cost function, make the design of a codebook of prediction filters a highly challenging, non-convex optimization problem.

The problem is further exacerbated by stability issues that arise due to the coder's prediction loop. Specifically, note that the optimal set of prediction filters depends on the reconstructed signal from which predictions are made. But the reconstructed signal itself depends on the prediction filters in use. Clearly, we have a "chicken and egg" problem here, and this complex interplay between predictors and reconstructions makes codebook design a challenging problem. The dependency between predictors and reconstructions calls for an iterative design technique, wherein optimal predictors are designed for the given reconstruction statistics, and then the reconstructions are updated with the designed predictors. In the standard closed-loop technique (see for e.g., [33] for quantizer design and [34] for a stochastic gradient version), the predictors designed in a given design iteration, i.e., given a training set of reconstructed blocks, are then plugged into the encoder and applied to a newly reconstructed signal in the next iteration,

which will likely exhibit different statistics. This statistical mismatch can (and often does) grow as the encoder proceeds down the sequence, due to propagation through the prediction loop, causing severe design instability. As an effective remedy, the asymptotic closed-loop (ACL) design paradigm was proposed in [35]. ACL operates in an open-loop fashion by predicting from the (now fixed) reconstructed samples in the previous iteration. Thus, the predictors are applied to the same reconstruction statistics they were designed for, thereby eliminating statistical mismatch and ensuring better reconstructions over iterations. Nevertheless, as will be explained in Section II, on convergence, the design effectively operates in closed-loop fashion, and optimizes the predictors for closed-loop operation.

ACL provides a stable design platform. However, the design is still plagued by many poor local minima of the cost function. To address this, we propose a deterministic annealing (DA) approach to design prediction modes. DA [36] is a powerful non-convex optimization tool, inspired by principles of statistical physics and information theory. The probabilistic nature of DA yields an effective cost function via expectation, which is differentiable with respect to the prediction modes. At high temperature (maximum randomness), at the early stage of the algorithm, all the prediction modes are shown to coincide (at convergence all modes are identical), regardless of initialization, and they will only separate (through a sequence of phase transitions in the physical analogy) as the temperature is lowered. In other words, DA is independent of the initialization. Its annealing schedule gradually reduces the randomness of the solution so as to avoid poor local minima. The overall method proposed herein embeds ACL within the DA framework. The benefits of DA are complemented by the stable design platform of ACL, effectively addressing the central design challenges enumerated above.

The design of prediction modes has many practical applications involving adaptive prediction. In this chapter, we consider an important application, namely, predictor de-

sign in video coding. Modern video coders exploit temporal correlations by employing motion compensated prediction [1]. Simple pixel copying of the best (motion-compensated and possibly interpolated) block from the reference frame is used to obtain the prediction signal. The resulting prediction error is then decorrelated by a transform, typically the discrete cosine transform (DCT), and the transform coefficients are quantized and sent to the decoder. Such pixel-to-pixel temporal prediction is suboptimal in that it ignores significant spatial correlations in the video signal. Several approaches that account for spatial correlations include multi-tap filtering [37, 38] and three-dimensional subband coding [39, 40], which incur high encoder complexity. An earlier work from our lab [41] proposed an effective way to account for complex spatio-temporal correlations by first applying the transform to spatially decorrelate a block, and subsequently performing temporal prediction of the resulting uncorrelated transform coefficients. The temporal evolution of each transform coefficient in a block, along its motion trajectory, is modelled as a first order auto-regressive process. Thus, we have a set of uncorrelated temporal processes, each representing the temporal evolution of a given coefficient (or "frequency") in the block. Moreover, transform domain temporal prediction (TDTP) perfectly captures and exploits the variations in temporal correlations across frequencies, which are otherwise masked in the pixel domain.

Modern video coders employ sub-pixel motion compensation for improved prediction, by interpolating the reference blocks to fractional pixel accuracy. Interpolation filters also use information from outside the block boundary, a fact that must be accounted for when optimizing prediction modes. Moreover, sub-pixel interpolation filters, when considered in the transform domain, interfere with the operation of TDTP filters. Thus, to completely disentangle the effect of interpolation filters and to account for boundary information, extended block TDTP (EB-TDTP) was proposed in [42]. With EB-TDTP, an extended reference block is first spatially decorrelated via DCT. Temporal prediction

filters are then applied for the extended transform blocks. This is followed by inverse-DCT and interpolation to obtain the prediction signal. The optimal EB-TDTP filters were shown to be least square estimates [42], which enhances the performance beyond that offered by the standard correlation coefficient formulation.

Video signals exhibit significant variations in local statistics. This requires the coder to adapt to local statistics, and an effective approach involves a set of trained prediction modes for the encoder to choose from. The EB-TDTP filter is a high-dimensional vector and the problem at hand effectively corresponds to vector quantizer design, a notorious non-convex optimization problem. Here too, standard closed-loop design suffers from significant instability issues. We thus propose a DA-ACL framework to learn prediction filters to address these challenges.

4.2 Background

4.2.1 Linear prediction

Fig. 4.1 shows a predictive compression system. Let x_n, $0 \leq n \leq N$ be the input samples. The signal is modelled as a first-order auto-regressive process. The current sample x_n is predicted from the previous reconstructed sample as,

$$\tilde{x}_n = \alpha \hat{x}_{n-1} \tag{4.1}$$

where $\hat{x}_n$ represents the reconstruction. The resulting prediction error, $x_n - \tilde{x}_n$, is quantized and sent to the decoder. The predictor is designed to minimize the sum of squared

prediction errors given by

$$E = \sum_{n=1}^{N} (x_n - \alpha \hat{x}_{n-1})^2 \tag{4.2}$$

The optimal predictor, obtained by basic linear estimation derivation, is

$$\alpha = \frac{\sum_n x_n \hat{x}_{n-1}}{\sum_n \hat{x}_{n-1}^2} \tag{4.3}$$

In order to adapt the predictor to variations in signal statistics, let the input be partitioned into groups or blocks of samples $\{g\}$. Let N_g be the set of samples belonging to a particular block g. The encoder is given a choice of K prediction filters $\{\alpha_k, k = 1, 2, \ldots, K\}$. The encoder chooses the best prediction mode for each block of samples. Let the best prediction mode for a given block g be $\hat{\alpha}_g \in \{\alpha_k\}$. The problem at hand is to design the prediction filters $\{\alpha_k\}$ such that the overall sum of squared prediction error

$$E' = \sum_g \sum_{n \in N_g} (x_n - \hat{\alpha}_g \hat{x}_{n-1})^2, \tag{4.4}$$

is minimized.

The piecewise constant nature of the cost function, with respect to the encoder's mode decisions, renders standard convex optimization algorithms inapplicable to the current scenario. A common, suboptimal remedy is the "*K-modes*" predictor design which we discuss next.

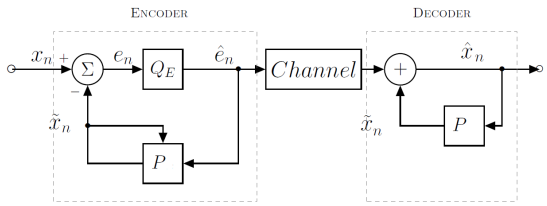


Figure 4.1: A simple first order predictive compression system

4.2.2 Iterative K-mode predictor design

Let us assume for the moment that we have a set of reconstructed samples $\hat{x}_n$ at the encoder. Given these reconstructions, we can design prediction modes in a way similar to "K-means" clustering. With an initialization of the prediction modes, the following steps are performed iteratively:

- Mode assignment: For a given block g, assign the best mode from the set of prediction modes which minimizes the squared prediction error for the block.

- Prediction modes update: Let N_k be the union of samples from blocks that share the same prediction mode. Similar to (4.3), the optimal prediction mode α_k for this cluster is given by,

$$\alpha_k = \frac{\sum_{n \in N_k} x_n \hat{x}_{n-1}}{\sum_{n \in N_k} \hat{x}_{n-1}^2} \tag{4.5}$$

Such *"K-modes"* predictor design optimizes the predictors for a given fixed set of reconstructions. However, in practice, these reconstructions will themselves depend on the predictors in use. This necessitates a two-fold optimization strategy, wherein, reconstructions and predictors are optimized iteratively. Given an updated set of prediction modes, there are several optional ways to update the reconstructions, leading to the following design paradigms.

4.2.3 Open-loop, closed-loop and asymptotic closed-loop design

Various techniques have been proposed in the context of joint design of predictors and quantizers. Since in most of modern video codecs the quantizer is fixed (up to scaling), our focus here is on predictor design given fixed quantizers, while noting that the same principles are also applicable to other predictive coder modules such as quantizers. In open-loop predictor design (see e.g., [33]), the predictor is designed using original samples, which do not depend on the predictors and the design is inherently stable. However, since the predictor must ultimately be applied to reconstructed samples, to avoid decoder drift, it will in fact operate on statistics mismatched with the design phase. In closed-loop design, predictors are designed iteratively. Let $\hat{\alpha}_g^i$ be the predictor for block g in iteration i. The reconstructed samples for the corresponding block in iteration $i+1$ is updated as,

$$\hat{x}_n^{i+1} = \hat{\alpha}_g^i \hat{x}_{n-1}^{i+1} + \hat{e}_n^{i+1} \tag{4.6}$$

where $\hat{e}_n^{i+1}$ is the quantized prediction error $e_n = x_n - \hat{\alpha}_g^i \hat{x}_{n-1}^{i+1}$. Predictor $\hat{\alpha}_g^i$ was designed for reconstruction in iteration i: $\{\hat{x}_n^i\}$. However, it is applied to the reconstructed samples of iteration $i+1$: $\{\hat{x}_n^{i+1}\}$. This mismatch results in design instability, which is exacerbated due to feedback through the prediction loop, and often proves catastrophic at low rates. To tackle this issue, ACL was proposed in [35]. ACL enjoys the best of both worlds. At each iteration, the samples are predicted and reconstructed in open loop fashion as,

$$\hat{x}_n^{i+1} = \hat{\alpha}_g^i \hat{x}_{n-1}^{i} + \hat{e}_n^{i+1} \tag{4.7}$$

where $\hat{e}_n^{i+1}$ is the quantized prediction error $e_n^{i+1} = x_n - \hat{\alpha}_g^i \hat{x}_{n-1}^{i}$. The predictor $\hat{\alpha}_g^i$ is used with reconstructed samples $\hat{x}_n^i$, the same set of samples that it was designed for, thereby eliminating statistical mismatch and the resulting design instability. The new set of

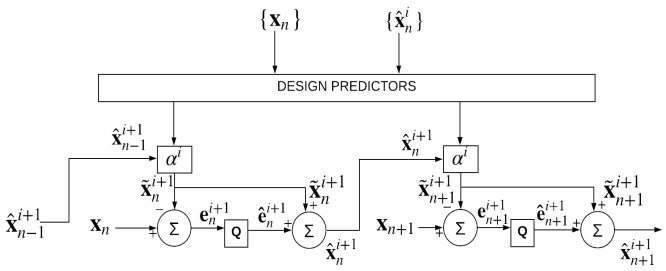


Figure 4.2: Standard closed-loop design for predictor optimization

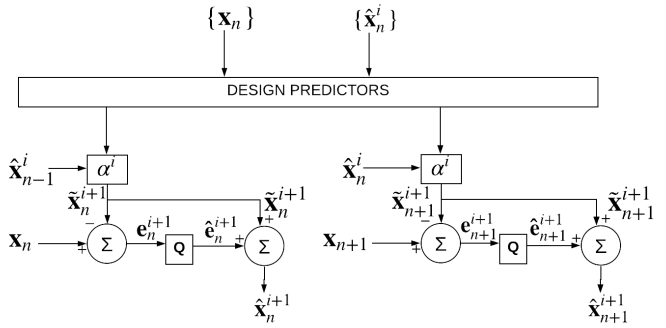


Figure 4.3: Predictor optimization with asymptotic closed-loop design

reconstructed samples are then used to design prediction modes α_k^{i+1}. Upon convergence, the reconstructed samples remain the same over iterations. Thus, predicting from $\hat{x}_n^i$ is same as predicting from $\hat{x}_n^{i+1}$, which is essentially closed-loop operation. The predictors designed are thus optimal for closed-loop operation. Fig. 4.2 depicts closed-loop design and Fig. 4.3 illustrates ACL design. Note that the prediction loop of CL is open in ACL which disallows propagation through the loop and hence avoids change in statistics.

With this background, we next introduce the proposed paradigm for predictor modes design.

4.3 Deterministic Annealing Based Predictor Design

The hard prediction mode assignment to every signal block makes it difficult to optimize the system with respect to the prediction modes, as the derivatives with respect to mode decisions vanish almost everywhere. Hence an iterative K-mode design, a variant of "K-means" clustering was proposed in [43]. However, this only ensures convergence to a local minimum and renders the system highly sensitive to initialization. A related problem is encountered in quantizer design, where the piecewise constant nature of the quantizer makes it a challenging optimization problem. In order to jointly overcome the fundamental challenges of non-convexity and design instability, we propose to embed the ACL based minimization of the overall prediction error within the DA framework. The proposed approach is inspired by, and builds on the deterministic annealing (DA) framework of [36]. DA is motivated by the intuition gained from annealing process in physical chemistry, where certain systems are driven to their low energy states by gradual cooling of the system. Analogously, we introduce controlled randomness in the prediction mode assignment for the blocks, but deterministically minimize the overall prediction error, thereby avoiding many poor local minima. The inherent probabilistic nature of DA allows us to deterministically optimize the effective cost function, an appropriate expectation function that efficiently accounts for and replaces the stochastic wandering on the cost surface of the classical method of simulated annealing [44]. The amount of randomness is measured by the Shannon entropy and is essentially controlled by the "temperature" of the system. The prediction mode assignment is no longer piecewise constant, and is differentiable everywhere, thus paving the way to effective optimization of prediction modes.

4.3.1 Prediction Mode Derivation

We consider a random setting wherein for each block, a prediction mode is chosen *in probability*. Thus, the mean squared prediction error to minimize in ACL iteration i is taken as the expectation,

$$J = \sum_{g} \sum_{k} \sum_{n \in N_g} P_g P^i_{k|g} (x_n - \alpha^i_k \hat{x}^i_{n-1})^2 \tag{4.8}$$

where P_g is the probability assigned to input data block g which is assumed to be uniform over all signal blocks. Association probability $P^i_{k|g}$ is the probability that prediction mode α_k is selected for input block g. The degree of randomness in the system is naturally measured by the Shannon entropy:

$$H = -\sum_{g} \sum_{k} P^i_{gk} \log(P^i_{gk}), \tag{4.9}$$

where $P^i_{gk} = P_g P^i_{k|g}$ is the joint probability distribution over prediction modes and training data blocks. The optimization problem is naturally restated as the minimization of the Lagrangian cost function, directly analogous to the Helmholtz free energy of statistical physics:

$$\mathcal{L} = J - TH, \tag{4.10}$$

where Lagrange parameter T controls the randomness of the solution. As an aside, there is an alternative (equivalent) way of formulating the problem, which is to maximize the Shannon entropy under a constraint of a given level of expected distortion, i.e, to maximize the Lagrangian given by,

$$\mathcal{L}' = H - \beta J \tag{4.11}$$

The motivation to maximize entropy stems from Jaynes's celebrated maximum entropy principle [45] which states that of all the probability distributions that satisfy a given set of constraints, it is beneficial to choose the one that maximizes the entropy, thereby avoiding the implicit imposition of any restrictive assumptions. It is obvious that the solution that minimizes the Lagrangian in (4.10) also maximizes the Lagrangian in (4.11). Returning to the formulation of (4.10), note that the degree of randomness is controlled by Lagrangian parameter T, which corresponds to temperature in the physical analogy. As we lower T, we trade entropy for prediction error. At the limit of zero randomness, we in fact directly minimize the overall prediction error.

A notable benefit of randomization is that the expected distortion is now differentiable with respect to the mode decisions (now association probabilities rather than binary decisions). Minimizing the Lagrangian cost with respect to the association probabilities $P^i_{k|g}$, while additionally imposing the obvious constraint $\sum_k P^i_{k|g} = 1$ (legitimate probabilities), yields the Gibbs distribution:

$$P^i_{k|g} = \frac{e^{\frac{-\sum_{n\in N_g}(x_n-\alpha^i_k\hat{x}^i_{n-1})^2}{T}}}{\sum_j e^{\frac{-\sum_{n\in N_g}(x_n-\alpha^i_j\hat{x}^i_{n-1})^2}{T}}} \tag{4.12}$$

Note that at high temperatures, we in fact maximize the system entropy and the association probabilities are indeed uniform.

The optimal prediction modes satisfy,

$$\begin{aligned}\frac{\partial J}{\partial \alpha^i_k} &= \sum_g \sum_{n\in N_g} 2P_g P^i_{k|g}(x_n - \alpha^i_k\hat{x}^i_{n-1})(-\hat{x}^i_{n-1}) \\ &= 0\end{aligned} \tag{4.13}$$

Thus, the optimal prediction modes are given by,

$$\alpha_k^i = \frac{\sum_g \sum_{n \in N_g} P_{k|g}^i x_n \hat{x}_{n-1}^i}{\sum_g \sum_{n \in N_g} P_{k|g}^i (\hat{x}_{n-1}^i)^2} \tag{4.14}$$

At this point, it is instructive to pause and compare the solution from DA (4.14) to the prediction modes in standard K-modes design (4.5). As we see in (4.5), the cross correlations and auto-correlations are taken as expectations over samples classified to a particular mode. However, in (4.14), the expectations are taken, with respect to the association probabilities, over the entire training set. Thus, in standard K-modes design in (4.5), the samples have highly localized influence, as they only impact the "nearest mode", thus blinding the system to possible better solutions further away. In other words, it is easy to get trapped in poor local optima. In contrast, in a DA based solution, each sample influences all the prediction modes through their association probabilities and the degree of influence varies with temperature. Specifically, at high temperature, all the association probabilities are uniform. Thus, the optimal prediction modes converge to and coincide at the correlation coefficient of the entire training set, the globally optimal single prediction mode. As the temperature is lowered, the degree of influence decreases and at the limit of zero temperature, the design is similar to the standard K-modes design. From this viewpoint, the standard K-modes design is a hard, zero-temperature design.

4.3.2 Overall design

The design starts with a closed-loop initialization of the reconstructions and at a high temperature T_0. As observed earlier, at high temperatures, given the uniform association probabilities, all the prediction modes coincide at the optimal single prediction mode of (4.3), *regardless of initialization*. Thus, DA is effectively independent of ini-

tialization. As the temperature is lowered, the association probabilities become more "discriminating" and the solution less random. As the system is cooled it reaches certain temperatures called "critical temperatures", where the existing solution with its set of prediction modes is no longer stable. Thus, with slight perturbations, the number of distinct modes increases as new prediction modes emerge through cluster splits. This phenomenon corresponds to "phase transitions" in the physical analogy.

At each temperature, the design iterates between predictor design and reconstruction update. For a given set of reconstruction statistics, the predictor design iterates between:

- a) Computing association probabilities for the prediction modes (4.12)
- b) Updating prediction modes (4.14)

These monotonically non-increasing steps minimize the Lagrangian $\mathcal{L}$. Upon convergence, the reconstructed samples $\hat{x}_n^{i+1}$ in a block g are updated in ACL fashion as,

$$\hat{x}_n^{i+1} = \sum_k P_{k|g}^i (\alpha_k^i \hat{x}_{n-1}^i + \hat{e}_{n,k}^{i+1}) \tag{4.15}$$

where, $\hat{e}_{n,k}^{i+1}$ is the quantized prediction error. Open-loop update ensures better reconstructions. Thus, ACL iterations are also monotonically non-increasing, ensuring the convergence of reconstructions. Upon convergence in reconstructions, the system is cooled and the process is repeated. Once the cooling is complete, the system gives prediction modes that minimize the overall prediction error (4.4) and that are optimal for closed-loop operation. The overall design procedure is summarized in Algorithm 3.

Having introduced a general framework for predictor design, we next consider an important application in the context of video coding.

Algorithm 3: Proposed DA-ACL predictor design

initialize: closed-loop reconstructions, T=T_0;
while $T < T\_min$ **do**
  while $ACL\_iter < max\_ACL\_iter$ **do**
    (a) Predictor design:
    do
      (i) Optimize association probabilities;
      (ii) Optimize prediction modes;
    while association probabilities converge
    (b) ACL update of reconstructions;
    break on convergence;
  end
  Cool system: T = bT;
end

4.4 Predictor Design in Video coding

Motion compensated prediction is a central component in modern video coders, tasked with removing temporal redundancies, which is critical to the overall compression efficiency of the coder. The best matching block from the reference frame is used as the prediction for the current block. Simple pixel copying, however, largely ignores the spatial correlations between pixels, and renders the prediction suboptimal. Moreover, pixel copying implicitly assumes that the temporal correlation coefficient is one at all frequencies. The invalidity of this implicit assumption is illustrated by the temporal correlation coefficients evaluated for various DCT coefficients in Table 4.1, over a sample sequence. Note how the correlation varies with frequency. Thus, to completely disentangle spatial and temporal correlations and to exploit the true frequency dependent nature of temporal correlations, transform domain temporal prediction (TDTP) was proposed in [41] which we briefly discuss next.

4.4.1 Transform Domain Temporal Prediction

TDTP models the temporal evolution of each transform coefficient as a first order AR process. In other words, we have parallel, uncorrelated AR processes, one per frequency (transform coefficient). Let x_n be a particular transform (say, DCT) coefficient in a given block in frame n, along a motion trajectory. The evolution of x_n is thus modeled as,

$$x_n = \alpha \hat{x}_{n-1} + e_n \tag{4.16}$$

where $\hat{x}_{n-1}$ is the corresponding DCT coefficient of the block in reconstructed frame $n-1$, along the motion trajectory, and e_n is the innovation sequence. The optimal prediction coefficient that minimizes the mean square prediction error is given by (4.3). By performing temporal prediction in the transform domain, TDTP effectively achieves both temporal and spatial decorrelation. Further, TDTP captures the variation of temporal correlation with spatial frequency, by optimizing the predictor for each transform coefficient.

We observe that if one were to use the entries of Table 4.1 as predictor coefficients, the effect would be coincidentally similar to that of a low-pass filter. Video coders use sub-pixel motion compensation which employs low-pass filters for interpolation. These interpolation filters interfere with TDTP, and may compromise its performance. Thus, to completely disentangle the effects of interpolation filters and TDTP filters, extended-block transform domain temporal prediction (EB-TDTP) was proposed in [42]. For simplicity and clarity of presentation, we first consider the basic design of TDTP modes, and then extend it to EB-TDTP.

Table 4.1: Temporal correlation coefficients, along motion trajectory, of DCT coefficients in the block

0.99	0.96	0.92	0.91	0.89	0.84	0.79	0.67
0.97	0.95	0.91	0.87	0.83	0.78	0.73	0.58
0.96	0.93	0.88	0.86	0.84	0.75	0.69	0.6
0.93	0.88	0.88	0.84	0.79	0.72	0.64	0.58
0.89	0.90	0.90	0.84	0.75	0.66	0.62	0.46
0.83	0.89	0.84	0.83	0.70	0.58	0.54	0.44
0.83	0.81	0.82	0.74	0.62	0.53	0.49	0.4
0.77	0.71	0.62	0.66	0.58	0.45	0.39	0.38

4.4.2 Problem Formulation

Let us consider an input training set which is partitioned into segments, each of which is called a group of pictures (GOP). Let the set of frames in a GOP be denoted by N_g. At the GOP level, the encoder switches between prediction modes $\{\boldsymbol{\alpha}_k\}$, where each $\boldsymbol{\alpha}_k$ is a matrix of prediction coefficients. Let us consider a block b in frame n that is inter-predicted from a reference block in frame $n-1$. Let the prediction mode chosen for the GOP g be $\hat{\boldsymbol{\alpha}}_g$. We spatially decorrelate the block and perform prediction in the DCT domain. The temporal prediction of the p^{th} DCT coefficient, $x_{n,b,p}$, is thus,

$$\tilde{x}_{n,b,p} = \hat{\alpha}_{g,p}\hat{x}_{n-1,b,p} \tag{4.17}$$

where $\hat{x}_{n-1,b,p}$ is the corresponding DCT coefficient of the reference block. The problem at hand is to design the prediction modes to minimize the overall prediction error of the training set. Thus, the cost function is,

$$E_{TDTP} = \sum_{g}\sum_{n\in N_g}\sum_{b}\sum_{p\in b}(x_{n,b,p} - \hat{\alpha}_{g,p}\hat{x}_{n-1,b,p})^2 \tag{4.18}$$

4.4.3 Deterministic Annealing Based TDTP Mode Design

Similar to the previous scenario, the proposed TDTP mode design enjoys the complementary benefits of ACL and DA. We randomize the association of TDTP modes to the GOPs. In an ACL iteration i, let $P^i_{k|g}$ denote the probability of assigning TDTP mode $\boldsymbol{\alpha}^i_k$ to GOP g. The prediction error to be minimized is thus given by the expectation,

$$J_{TDTP} = \sum_g \sum_k \sum_{n \in N_g} \sum_b \sum_p P_g P^i_{k|g} (x_{n,b,p} - \alpha^i_{k,p} \hat{x}^i_{n-1,b,p})^2 \tag{4.19}$$

where P_g denotes the probability of the input GOPs (assumed uniform). The degree of randomness is measured by the Shannon entropy,

$$H_{TDTP} = -\sum_g \sum_k P^i_{gk} \log(P^i_{gk}), \tag{4.20}$$

where $P^i_{gk} = P_g P^i_{k|g}$ is the joint distribution over TDTP modes and input GOPs. The cost function to be minimized is the Lagrangian,

$$\mathcal{L}_{TDTP} = J_{TDTP} - T H_{TDTP} \tag{4.21}$$

Assuming uniform distribution over the training set, the association probabilities that minimize the Lagrangian cost (subject to the additional constraint $\sum_k P^i_{k|g} = 1$), are given by the Gibbs distribution:

$$P^i_{k|g} = \frac{e^{\frac{-\sum_{n \in N_g} \sum_b \sum_p (x_{n,b} - \alpha^i_{k,p} \hat{x}^i_{n-1,b,p})^2}{T}}}{\sum_j e^{\frac{-\sum_{n \in N_g} \sum_b \sum_p (x_{n,b,p} - \alpha^i_{j,p} \hat{x}^i_{n-1,b,p})^2}{T}}} \tag{4.22}$$

Next, we note that the Lagrangian depends on the prediction modes only through the expected prediction error J, as all other terms only depend on the association probabili-

ties. Moreover, since the transform coefficients are uncorrelated, we obtain the following partial derivative with respect to each component of the prediction mode, which we then set to zero to obtain the necessary condition for optimality:

$$\begin{aligned}\frac{\partial J}{\partial \alpha^i_{k,p}} &= \sum_g \sum_{n\in N_g} \sum_b \{2P_g P^i_{k|g}(x_{b,n,p} - \alpha^i_{k,p}\hat{x}^i_{b,n-1,p}) \\ &\qquad (-\hat{x}^i_{b,n-1,p})\} \\ &= 0 \end{aligned} \tag{4.23}$$

Thus, the p^{th} component of the optimal prediction modes is given by,

$$\alpha^i_{k,p} = \frac{\sum_g \sum_{n\in N_g} \sum_b P^i_{k|g} x_{b,n,p}\hat{x}^i_{b,n-1,p}}{\sum_g \sum_{n\in N_g} \sum_b P^i_{k|g}(\hat{x}^i_{b,n-1,p})^2} \tag{4.24}$$

The optimal predictor for a particular DCT coefficient is clearly a variant of the standard optimal linear estimator for that particular coefficient, computed over the entire training set, while accounting for the association probabilities $P^i_{k|g}$ with respect to which the expectation is defined.

The overall design is similar to 4.3.2, where the design starts from a high temperature and is gradually cooled. At high temperatures, the association probabilities are uniform as is obvious from (4.22) and all the TDTP modes given by (4.24) are coincidental. At a given temperature T, the design iterates between optimizing predictors and updating reconstructions in ACL way. Optimizing predictors for a given reconstruction set involves computing association probabilities as (4.22) and updating updating prediction modes according to (4.24). Upon convergence, the reconstructed samples in GOP g are updated

via ACL as,

$$\hat{x}^{i+1}_{n,b,p} = \sum_{k} P^{i}_{k|g}(\alpha^{i}_{k,p}\hat{x}^{i}_{n-1,b,p} + \hat{e}^{i+1}_{k,n,b,p}) \tag{4.25}$$

where, $\hat{e}^{i+1}_{k,n,b,p}$ is the quantized prediction error $e^{i+1}_{k,n,b,p} = x_{n,b,p} - \alpha^{i}_{k,p}\hat{x}^{i}_{n-1,b,p}$. Upon convergence in reconstructions, the system is cooled and the process is repeated. As the temperature is lowered, the system becomes more deterministic with the emergence of more TDTP modes through a sequence of phase transitions. At the limit of zero temperature, the prediction modes directly minimize the squared distortion and the prediction modes designed are optimal for the closed-loop operation.

Having introduced the DA based TDTP mode design, we next consider predictor design with extended block TDTP which disentangles the effects of sub-pixel interpolation filters and transform domain prediction filters.

4.4.4 Transform Domain Temporal Prediction with Extended Blocks

Video coders employ sub-pixel motion compensation in which interpolated reference blocks are used as prediction signals. To obtain the interpolated signal, the coder makes use of boundary samples outside the reference block. Thus, applying TDTP on the interpolated signal is effectively considering spatial and temporal decorrelations in the subspace of the interpolated signal, in contrast with decorrelating in the actual space of the boundary-extended reference block. Moreover, as observed earlier, interpolation filters interfere with TDTP filters. EB-TDTP effectively addresses these challenges by first scaling the extended block pixels according to the temporal prediction coefficients in the transform domain, and then applying the interpolation filters. To formulate this mathematically, let $Y_{n,b}$ be the block b of dimensions $B_1 \times B_1$ in frame n, which is to be

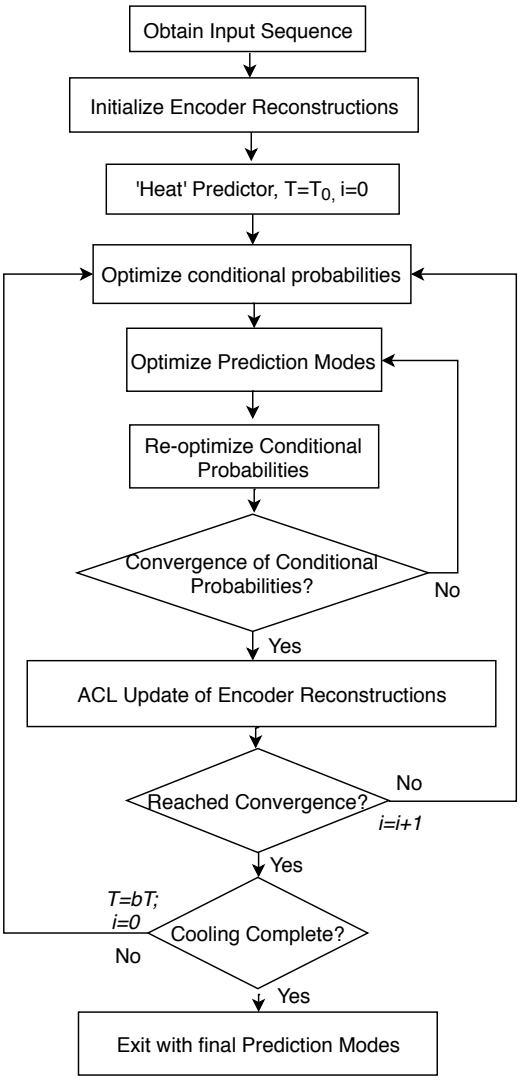


Figure 4.4: Flow chart of the proposed DA-ACL-TDTP design algorithm

predicted. Let $\hat{Y}^{mv}_{n-1,b}$ be the reference block in frame $n-1$, of dimensions $B_2 \times B_2 (B_2 > B_1)$, to which the video coder applies interpolation to obtain the prediction signal. Let the vertical and horizontal interpolation filters be denoted as I_v and I_h, which are matrices of sizes $B_1 \times B_2$ and $B_2 \times B_1$, respectively. Thus the interpolated prediction signal of standard coders is,

$$\tilde{Y} = I_v \hat{Y}^{mv}_{n-1,b} I_h \tag{4.26}$$

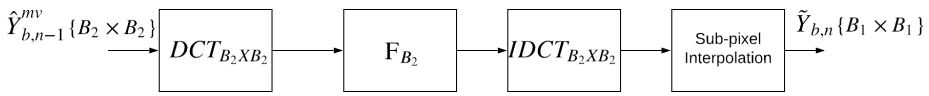


Figure 4.5: Illustration of extended block transform domain temporal prediction

and the prediction signal from "plain" (i.e., without "extended block") TDTP is,

$$\tilde{Y}_{TDTP} = D'_{B_1}[\{D_{B_1}(I_v \hat{Y}^{mv}_{n-1,b} I_h) D'_{B_1}\} \circ F_{B_1}] D_{B_1} \tag{4.27}$$

where D_{B1} is the DCT matrix, F_{B1} is the TDTP filter and $\circ$ denotes component-wise matrix multiplication.

In EB-TDTP, we first spatially decorrelate the extended block $\hat{Y}^{mv}_{n-1,b}$ by a separable DCT. Then, the extended block TDTP filter F_{B_2} is applied to the transformed extended block. This is followed by inverse transform and interpolation to derive the prediction signal. Thus, EB-TDTP of $Y_{n,b}$, as illustrated in Fig. 4.5 can be formulated as,

$$\tilde{Y}_{EB-TDTP} = I_v D'_{B_2}\{\{D_{B_2} \hat{Y}^{mv}_{n-1,b} D'_{B_2}\} \circ F_{B_2}\} D_{B_2} I_h \tag{4.28}$$

To derive the predictor F_{B_2}, let $K_1 = I_v D'_{B_2}$, $K_2 = D_{B_2} I_h$, and $\hat{X}^{mv}_{n-1,b} = D_{B_2} \hat{Y}^{mv}_{n-1,b} D'_{B_2}$. The prediction error can thus be written as,

$$
\begin{aligned}
E_{EB-TDTP} &= \sum_{n}\sum_{b}\left\|Y_{n,b}-\tilde{Y}_{n,b}\right\|^2 \\
&= \sum_{n}\sum_{b}\left\|Y_{n,b}-K_1(\hat{X}^{mv}_{n-1,b}\circ F_{B_2})K_2\right\|^2 \\
&= \sum_{n}\sum_{b}[\sum_{r=1}^{B_1}\sum_{s=1}^{B_1}\{Y_{n,b}(r,s)- \\
&\quad \sum_{i=1}^{B_2}\sum_{j=1}^{B_2}F_{B_2}(i,j)\hat{X}^{mv}_{n-1,b}(i,j)K_1(r,i)K_2(j,s)\}^2]
\end{aligned}
\tag{4.29}
$$

This is essentially a least-squares estimation problem of minimizing,

$$
E_{EB-TDTP} = \sum_{n}\sum_{b}\left\|A_{n,b}\boldsymbol{f}_{B_2}-\boldsymbol{t}_{n,b}\right\|^2, \tag{4.30}
$$

where $\boldsymbol{f}_{B_2}$ is a vector representation (containing all elements) of matrix F_{B_2}. $A_{n,b}$ and $\boldsymbol{t}_{n,b}$ are derived as,

$$
A_{n,b}(u,v) = \hat{X}^{mv}_{n-1,b}(i,j)K_1(r,i)K_2(j,s) \tag{4.31}
$$

$$
\boldsymbol{t}_{n,b}(u) = Y_{n,b}(r,s) \tag{4.32}
$$

where, $u = rB_1 + s$ and $v = iB_2 + j$. The optimal predictor is given by,

$$
\boldsymbol{f}_{B_2} = (\sum_{n}\sum_{b}A^T_{n,b}A_{n,b})^{-1}(\sum_{n}\sum_{b}A^T_{n,b}\boldsymbol{t}_{n,b}) \tag{4.33}
$$

As seen from (4.33), the optimal predictor computation is essentially posed as a classic least-squares problem. The discussion so far involved a single predictor. To

realize the full potential of EB-TDTP, we need a set of EB-TDTP filters to achieve adaptivity, which implies the design of an efficient set of prediction modes. We note that multiple prediction modes introduce optimization in a higher dimensional parameter space, making the design more prone to be trapped in local poor minima. Thus, there is strong motivation to pursue a DA based solution.

4.4.5 EB-TDTP Mode Derivation

The design involves randomization of the prediction mode assignment to GOPs. At a given temperature T and ACL iteration i, let conditional probability $P^i_{k|g}$ denote the probability of assigning EB-TDTP mode F^i_k to GOP g. The prediction error to be minimized is given by the expectation:

$$J_{EB-TDTP} = \sum_g \sum_k \sum_{n \in g} \sum_b P_g P^i_{k|g} \left\| A^i_{g,n,b} \boldsymbol{f}^i_k - \boldsymbol{t}_{n,b} \right\|^2 \tag{4.34}$$

where P_g denotes the probability of the input GOPs (assumed uniform). The degree of randomness is measured by the Shannon entropy,

$$H_{EB-TDTP} = -\sum_g \sum_k P^i_{gk} \log(P^i_{gk}), \tag{4.35}$$

where $P^i_{gk} = P_g P^i_{k|g}$ is the joint distribution over EB-TDTP modes and input GOPs. The cost function to be minimized is the Lagrangian,

$$\mathcal{L}_{EB-TDTP} = J_{EB-TDTP} - T H_{EB-TDTP} \tag{4.36}$$

The problem at hand is posed as the minimization of the entropy-constrained Lagrangian $J_{EB-TDTP}$. The association probabilities that minimize the Lagrangian cost

subject to the standard normalization constraint (adding up to 1), are given by:

$$P^i_{k|g} = \frac{e^{-\frac{\sum_{n\in g}\sum_b \left\| A^i_{g,n,b}\boldsymbol{f}^i_k - \boldsymbol{t}_{n,b}\right\|^2}{T}}}{\sum_j e^{-\frac{\sum_{n\in g}\sum_b \left\| A^i_{g,n,b}\boldsymbol{f}^i_j - \boldsymbol{t}_{n,b}\right\|^2}{T}}} \tag{4.37}$$

Minimizing the expected distortion with respect to the prediction modes yields,

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{f}^i_k} &= \sum_g \sum_{n\in g} \sum_b P^i_{k|g}((A^i_{g,n,b})^T A^i_{g,n,b}\boldsymbol{f}_k - (A^i_{g,n,b})^T \boldsymbol{t}_{n,b}) \\ &= \mathbf{0} \end{aligned} \tag{4.38}$$

Thus, the optimal prediction modes are given by,

$$\begin{aligned} \boldsymbol{f}^i_k &= \{\sum_g \sum_{n\in g} \sum_b P^i_{k|g}((A^i_{g,n,b})^T A^i_{g,n,b})\}^{-1} \\ &\quad \{\sum_g \sum_{n\in g} \sum_b P^i_{k|g}((A^i_{g,n,b})^T \boldsymbol{t}_{n,b})\} \end{aligned} \tag{4.39}$$

The overall design is similar to the design of TDTP modes.

4.5 Experimental Results

4.5.1 A simple First Order Predictive Encoder

The first experiment considers the simple setting of scalar, first order predictive coding. We chose speech signals as a real-world source data. A set of six speech files from the EBU SQAM database were chosen for simulations [46]. Half of the speech files were used as the training set for designing prediction modes and the remaining half as the test set. A set of six prediction modes were designed. A fixed dead-zone quantizer was

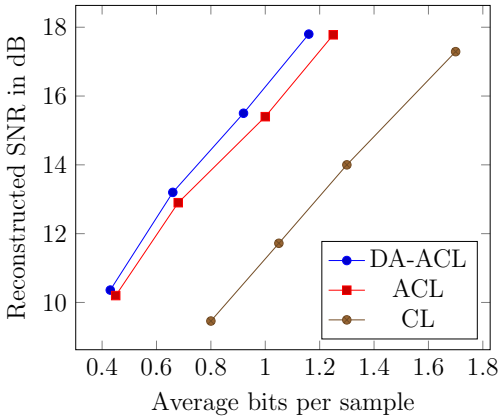


Figure 4.6: First-order scalar predictive coding. Reconstructed SNR vs average bits per sample for the test set of speech files

employed for quantization. Different R-D points were obtained by varying Lagrange multiplier of entropy constrained quantization. The 3 competitors were: closed-loop (CL), "plain ACL", and the proposed method (DA-ACL). While DA-ACL is independent of initialization, CL and ACL designs were repeated with multiple initializations and the best results were selected. Fig. 4.6 shows the reconstructed SNR versus bit rate. It is evident from the results that the proposed DA-ACL method gives significant 0.4dB and 5dB gains over competitors ACL and CL, respectively.

4.5.2 Video Coding Results

The proposed method was implemented in HM 14.0. We chose low-delay P or LDP profile for our experiments. For all the experiments, the encoder only uses the previous frame as reference. The anchor is the HEVC codec which performs conventional pixel domain prediction, i.e, either simple pixel copying or an interpolated block from the reference frame. The competing codec uses EB-TDTP prediction modes. A EB-TDTP mode is a collection of filters for all the block sizes. Thus, specifying a EB-TDTP mode for a GOP completely specifies the prediction filters for all the block sizes. To minimize the

encoding complexity, EB-TDTP filters are used only during motion compensation and are not used during motion estimation. In other words, during motion search, conventional pixel domain prediction is used to decide the best motion vector and only during motion compensation, we perform transform domain prediction with EB-TDTP filters. We consider four QP values of 22, 27, 32 and 37 for our simulations. The implementation details for training EB-TDTP modes are discussed next.

Training EB-TDTP filters

EB-TDTP filters depend on reconstruction statistic which varies with QP value. Thus, EB-TDTP modes are trained conditioned on QP value. For each QP value, we design four EB-TDTP modes by each of the following design methods:

i) Standard closed-loop design denoted CL: predictors are optimized in 'K-modes' clustering method and the reconstructions are updated in standard closed-loop method. This is the traditional approach to predictor design and suffers from both initialization due to 'K-modes' style design of predictors and the design instability due to closed-loop update of reconstructions.

ii) K-mode design with "plain ACL" denoted ACL: predictors are still optimized in 'K-modes' clustering method, but reconstructions are updated in ACL way. This design enjoys stability due to ACL but still suffers from initialization problem.

iii) The proposed method denoted DA-ACL: predictors are optimized by DA and the reconstructions are updated in ACL fashion. This solves both the initialization and the design instability issues.

The aforementioned methods perform iterative optimization of predictors and reconstructions. Given a set of predictors, reconstructions are updated by using HEVC codec. During reconstruction update, reconstruction statistic for different block sizes are collected for predictor optimization in the next design iteration. The predictor optimization

is done in a separate module outside the codec. The training set sequences are listed in Table 4.2.

4.5.3 Testing

The trained prediction modes are stored at both encoder and decoder. The encoder does a brute-force search over all prediction modes for each GOP and selects the best mode. The average bit-rate reduction by using EB-TBTP modes over HEVC performing conventional pixel domain prediction is calculated as per [25]. The bit-rate savings achieved by using EB-TDTP modes from CL, ACL and DA-ACL design methods for the test set sequences are tabulated in Table 4.3. The rate-distortion (RD) curves for some example test sequences are shown in Fig. 4.7 (note that, we have only plotted three R-D points for better visualization). Significant bit-rate reduction over the test set provides clear evidence for the utility of proposed approach. The seemingly less gains at low bit-rates can be explained from the fact that at low bit-rate, a large number of transform coefficients are quantized to zero, leaving us less parameters to optimize. Moreover, a certain PSNR improvement, say 0.1dB, bears more significance in low bit-rate regime than it does in the high bit-rate regime. As regards the complexity, employing transform domain prediction with a particular EB-TDTP mode, increases the encoder complexity by a factor of two. Further, since the encoder does brute-force search over four modes, the overall encoding complexity is 8x compared to anchor. The decoder just uses the best mode and thus has a complexity increase of 2x compared to the anchor. Various approaches can be explored for fast selection of prediction modes and fast implementation of transform domain prediction.

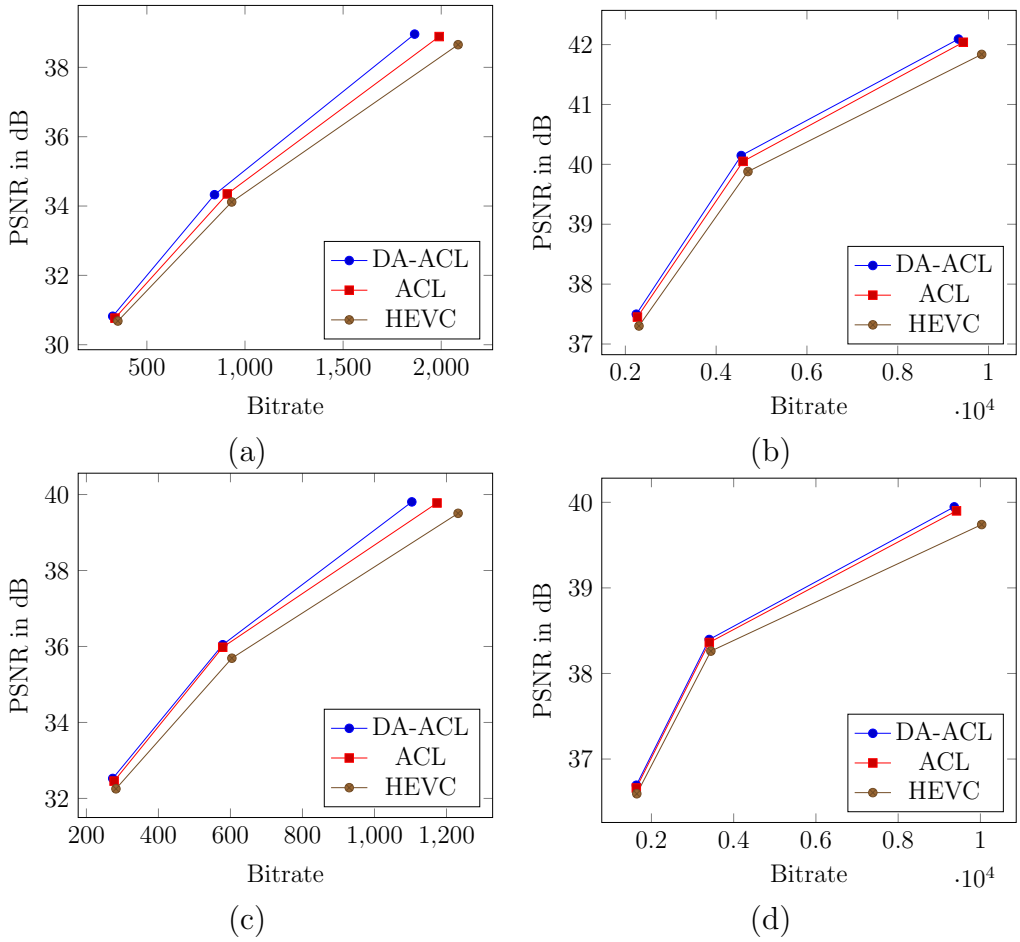


Figure 4.7: Comparison of performance of EB-TDTP modes designed by ACL and DA-ACL methods over HEVC for test sequences of (a) Coastguard, (b) Kimono, (c) Soccer and (d) BasketballDrive sequence.

4.6 Conclusions

This chapter presents a novel near-optimal procedure for designing prediction modes for adaptive compression systems. It effectively resolves significant shortcomings due to statistical mismatch and design instability of standard approaches. The deterministic annealing-based framework enables direct optimization of the overall cost with respect to prediction mode decisions, and avoids many poor local minima that trap its competitors. Substantial gains in the experiments demonstrate the efficacy of the proposed approach.

Table 4.2: The training set of video sequences for EB-TDTP design

Sequence
Tennis (1080p)
Pedestrian (1080p)
Parkjoy (720p)
Vidyo3 (720p)
BQMall (720p)
Racehorses (240p)
Paris (cif)
Waterfall (cif)
City (cif)
Stefan (cif)
Highway (cif)
Mobile (cif)

Table 4.3: Performance over the test set: bit-rate savings over HEVC (in %) for the Y component by employing EB-TDTP filters designed from CL, ACL and DA-ACL design paradigms

Sequence	CL	ACL	DA-ACL
Kimono (1080p)	7.2	8.5	11.3
Tractor (1080p)	12.8	14.4	15.7
Parkscene (1080p)	5.9	6.0	6.5
BasketballDrive (1080p)	7.5	7.6	10.3
KristenAndSara (720p)	8.0	8.1	8.3
vidyo4 (720p)	14.3	16.1	16.5
Ducks (720p)	12.4	13.6	14.2
Mobisode2 (480p)	3.8	3.8	4.2
BQTerrace (480p)	0.4	1.8	2.8
Keiba (480p)	6.1	6.6	7.5
BasketballPass (240p)	-1.2	-0.1	0.2
Mobisode2 (240p)	-1.8	0.4	1.1
Coastguard (cif)	7.4	7.6	11.2
Bridge-far (cif)	3.9	5.3	6.3
Soccer (cif)	7.9	8.7	9.7
Salesman (qcif)	-3.7	-0.2	0.9
Average	**5.7**	**6.6**	**7.9**

Chapter 5

Transform Design for the Inter-Prediction Residual in Video Coding

5.1 Introduction

Transform coding is an essential component in video compression, wherein it is applied to the prediction residual after a block of pixels has undergone intra or inter-prediction. The goal is to achieve energy compaction in as little transform coefficients as possible. For a given stationary signal statistics, it is well known that the Karhunen- Loève transform (KLT) is the optimal decorrelating transform. However, its dependency on signal statistics and its high computational complexity compromise its practicality. Instead, the discrete cosine transform (DCT) has been the most widely adopted transform due to its fast implementation and good energy compaction property, as well as the theoretical justification provided to its ability to approximate performance of KLT on certain Gauss-Markov processes [47]. Inter-prediction residue exhibits a wide range of statis-

tic. Thus, to achieve better compression, there has been growing interest in employing switched transforms that adapt to variations in signal statistics. The latest open source codec AV1 [2] allows switching within a set of known trigonometric transforms such as DCT and ADST in order to capture some additional gains. The authors in [48] also propose to use known trigonometric transforms for inter-prediction residuals which was later adopted in JEM codec [49]. The trigonometric transforms are optimal only under certain model assumptions. The validity of such assumptions and the optimality of these transforms is highly questionable for inter-prediction residual. Thus, there is a strong motivation for a data-driven approach to learn these transforms. For projected spherical videos, the residue statistic is expected to be very different from the residue statistic in regular 2D videos, and further expected to vary for different geometries. Due to non-uniform sampling induced on the sphere, an analytical solution to derive the optimal transform (by working in spherical domain) is nearly impossible. Again, there is a strong motivation to pursue a data driven approach to learn the transform kernels.

A major challenge in the joint design of multiple transform modes is due to the instability inherent to the closed-loop design of the coder. Updated transforms are applied to prediction residuals to obtain new reconstructions, which in turn affect the prediction residual statistics. This complex interplay between the transforms and reconstructions makes effective transform design quite elusive. In standard closed-loop design, the transforms and reconstructions are optimized in an iterative design procedure. For a given fixed set of reconstructions in a certain design iteration, the residue signal and the corresponding optimal transforms are computed. The transforms are used in the next iteration to update reconstructions. The transforms are designed for a given residue statistic and are applied with a different residue statistic in the next design iteration. The resulting statistical mismatch generates error that propagates through the prediction loop causing design instability. Transform design involves design of a large set of parameters and

the design suffers severely from this instability causing severe hindrance in transform optimization. In the previous chapter, we used ACL to address design instability in predictor design. In this chapter, we extend the ACL paradigm to effective transform design. Specifically, transforms are designed iteratively, in an open loop that ensures design stability, but with a subterfuge that guarantees that upon convergence, the transforms are optimal for closed loop operation. Thus, in this chapter, we use ACL as a stable hence effective platform for designing multi-mode transforms. Note that, while the focus is on the design of separable transforms which are preferred due to their lower complexity, the proposed design paradigm is general and applicable to non-separable transforms.

5.2 Background

5.2.1 Separable KLT

Let $\mathbf{e}$ be a random vector of (say, prediction residual) samples, whose covariance matrix is $\mathbf{C_e}$. Let $\mathbf{T}$ be the transform matrix. The transform-domain signal vector $\mathbf{y}$ is given by

$$\mathbf{y} = \mathbf{Te}, \tag{5.1}$$

and its covariance matrix $\mathbf{C_y}$ is

$$\mathbf{C_y} = \mathbf{TC_eT'} \tag{5.2}$$

The optimal transform that diagonalizes $\mathbf{C_y}$, i.e., decorrelates the components of $\mathbf{y}$ is precisely KLT, whose basis vectors are the eigenvectors of $\mathbf{C_e}$.

In the context of video coding, let $\mathbf{E}$ be the random prediction residual block. Let $\mathbf{T_r}$ and $\mathbf{T_c}$ be the respective KLTs for the row covariance $\mathbf{C_r}$ and column covariance $\mathbf{C_c}$,

of $\mathbf{E}$. The transform-domain signal can be written as,

$$\mathbf{Y} = \mathbf{T_c} \mathbf{E} \mathbf{T}'_\mathbf{r} \tag{5.3}$$

KLTs are optimal for given statistics of the prediction residual signal. But updating the transforms changes the reconstructions and hence also the residual signal statistics, requiring a new KLT calculation. Thus, transform design requires an iterative procedure. Next we summarize the standard iterative approach.

5.2.2 Closed-Loop Design

Standard closed-loop techniques (see e.g., [34]), when applied to transform design, employ transforms trained on the residual sequence of iteration i to transform the residue in the next iteration $i+1$, i.e,

$$\mathbf{y}^{i+1} = \mathbf{T}^i \mathbf{e}^{i+1} \tag{5.4}$$

where the residual $\mathbf{e}^{i+1} = \mathbf{x}_n - \hat{\mathbf{x}}_{n-1}^{i+1}$ is the prediction error in iteration $i+1$ (assuming prediction coefficient of one, as is common practice in video coding). Thus transform $\mathbf{T}^i$, optimal for the previous residual sequence $\{\mathbf{e}^i\}$, is in fact applied to a potentially very different residual sequence in iteration $i+1$. This results in statistical mismatch which tends to grow as errors propagate in the prediction loop, and the resulting instability may prove catastrophic at low rates. Fig. 5.1 illustrates closed-loop design.

5.3 Relevant Work

Much of the work on transform design focused on the intra-prediction residual, including the derivation of asymmetric trigonometric transforms to leverage the directionality of intra-prediction, which were further shown to approach KLT optimality under mild

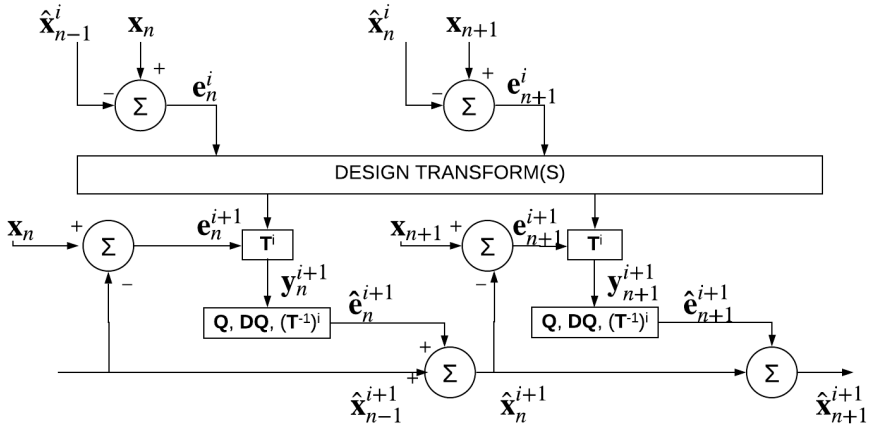


Figure 5.1: Closed-loop design of transforms

Markovian assumption [50] as well as several other approaches to mode-dependent transforms (e.g., [51, 52, 53]). The design of transforms for inter-prediction residuals, however, attracted significantly less attention, perhaps due to the fact such transforms do not exhibit as "obvious" properties such as the directionality inherent to intra prediction modes, and are hence more challenging to design. As mentioned earlier, recent codecss switch between known trigonometric transforms. However, to realize the full potential of multimodal transforms, it is necessary to look beyond the known trigonometric transforms, and employ a data-driven approach that statistically learns the optimal set of transforms. Recently efforts in data-driven approach can be classified into following categories:

Online learning of transforms

As the name suggests, the transforms are learnt on the fly during the encoding and decoding process (see for e.g, [54]). Computation of covariance matrix and its eigen vectors is highly complex. Thus, online learning of transforms is not generally preferred. We will focus on the practical alternative of an offline design paradigm. Some of the works in offline design is discussed next.

Offline learning of transforms

A notable work in this vein includes [55], where residue statistics are collected from a training set and the resulting KLT is given as an option during encoding. Another interesting work is the graph based transforms in [56], in which, along with the computation of covariance matrix, additional constraints about the model assumptions for video signal are imposed while computing transforms. Other relevant approaches include the 1-D transforms developed in [57], directional DCTs in [58], row column transforms in [59] and layered-Givens transforms in [60]. All these approaches largely ignore what is a critical difficulty of closed-loop iterative design of modules of a predictive coding system. Specifically, in the case of transforms, an updated transform changes the reconstructions, which in turn modify the prediction residual statistics on which the transform update was premised. Authors in [48] realize this dependency in designing secondary transform kernels for intra-prediction residuals and pursue a closed-loop design of transforms. For, intra-prediction, the design instability doesn't post a challenge. However, for inter-prediction residue, the instability is a critical issue. Addressing the design instability forms the central focus of the current chapter.

Most of the work related to spherical video coding is focused on projection geometry optimization (see for e.g, [61]) or prediction optimization (including our own work and other efforts summarized in [62]). Some of the recent work including [63] optimize transforms for spherical image compression. However, none of these efforts consider transform optimization for inter-prediction residue in projected spherical videos. Transform optimization for spherical videos forms another core contribution of the current chapter.

5.4 Proposed Method

In this section, we propose a stable design paradigm for learning the transforms. The design considers transform design in variable block setting. We leverage the fact that block size implicitly conveys some information about nature of residue statistic and this information is available to the decoder as well. We exploit this fact and for each block size or partition size indexed by s, we design M_s separable transforms, which we also refer to as transform modes. Thus, the problem at hand is to design a set of transform modes for different block sizes $\{\mathbf{T_s}\}$, where $\mathbf{T_s}$ corresponds to a set of M_s pairs of row and column transforms denoted $\{\{\mathbf{T_{s,m,r}}, \mathbf{T_{s,m,c}}\}, m = 1, 2..M_s$. An iterative design technique is needed to optimize transforms and update reconstructions. Given a training set of residual sequences, a clustering based framework is presented first to enable transform modes design.

5.4.1 Clustering

Let $\{\mathbf{E}^i_{b,n}\}$ be the training sequence of prediction residual where $\mathbf{E}^i_{b,n}$ is block b in frame n, obtained by subtracting from source block $\mathbf{X}_{b,n}$ its motion-compensated prediction $\hat{\mathbf{X}}^i_{b^{mv},n-1}$

$$\mathbf{E}^i_{b,n} = \mathbf{X}_{b,n} - \hat{\mathbf{X}}^i_{b^{mv},n-1} \tag{5.5}$$

To design the transform-modes, we employ an algorithm in the spirit of "K-means clustering", which iterates between following steps :

(a) Nearest neighbor step: This corresponds to assigning the best transform for each block. The mode assignment decisions need to be RD-optimal and take into account the total cost of coding the transform coefficients and signaling these modes to the decoder. Thus, given a set of transforms, we get the block partitions and

mode assignments by plugging these transforms in the codec. Note that the reconstructions here are held fixed and thus the codec is run in an open-loop fashion rather than a closed-loop fashion. This differentiation will be more clear after the discussion of reconstruction update that follows shortly.

(b) Centroid step: This step corresponds to design of optimal row and column transforms design for each mode. In other words, separable KLT is designed for each cluster of residue corresponding to a particular block size s and mode m.

We next consider how to embed within the approach an ACL paradigm for transform design so as to avoid the notorious instability of closed-loop design.

5.4.2 Asymptotic Closed Loop Design

As discussed in 5.2.2, the main shortcoming of the closed-loop approach is the design instability due to error propagation in the prediction loop. ACL design effectively resolves the stability issue by updating the reconstructions in an open-loop fashion as illustrated in Fig. 5.2. The updated transforms are used with the same set of residual sequences for which they were designed. This ensures increasingly better reconstructions over the iterations. However, on convergence, the reconstructed sequence remains essentially unchanged. Therefore, predicting from the previous iteration's reconstructions approaches equivalence with predicting from the current iteration, i.e., it effectively operates in closed-loop. Thus, ACL asymptotically optimizes transforms for closed-loop operation. For the problem at hand, given optimal transform-modes from a design iteration i, the transform signal is obtained as,

$$\mathbf{Y}^i_{b,n} = \mathbf{T}^i_{\mathbf{c},\text{best}} \mathbf{E}^i_{b,n} \mathbf{T}^{i'}_{\mathbf{r},\text{best}} \tag{5.6}$$

where $\mathbf{T}^i_{\mathbf{c},\text{best}}, \mathbf{T}^i_{\mathbf{r},\text{best}}$ are the best row and column transforms chosen by the encoder from the transforms designed in iteration i. This is followed by, quantization, de-quantization and inverse transform to obtain the block $\hat{\mathbf{E}}^i_{b,n}$. Note how the transforms designed in iteration i is used on the same set of residuals $\mathbf{E}^i_{b,n}$. The reconstructions are updated as,

$$\hat{\mathbf{X}}^{i+1}_{b,n} = \hat{\mathbf{X}}^{i}_{b^{mv},n-1} + \hat{\mathbf{E}}^i_{b,n} \tag{5.7}$$

The overall design procedure has been illustrated in Algorithm 4. First, a closed-loop initialization is performed to obtain a reconstructions sequence. Standard trigonometric transforms are used as initialization for the transform modes in each block size. The algorithm then iterates between "K-means" style transform-modes design for a given residual statistic and the reconstruction update in ACL fashion. Note that, after a reconstruction update, we update the encoder decisions including the motion vectors, ensuring optimal encoder decisions for the new reconstructions. We use these decisions to generate prediction residue for the next iteration. Upon convergence, both the reconstructions and the encoder decisions remain the same, and hence the system effectively operates in closed-loop fashion.

We note that in our earlier work [64], we had fixed block setting. This resulted in wide range of residue statistic and prompted us to design super-modes of transforms to cover such a statistic. Each super-mode is a collection of collection of set of separable transforms or the transform-modes. The adaptivity was such that the encoder could switch between super-modes at the GOP level and further switch between the transform modes of the chosen super-mode at the GOP level. In the current variable block setting, we exploit the partition of the residue statistic in terms of block sizes and achieve significant gains by designing different transform modes for different block sizes and overcome the need

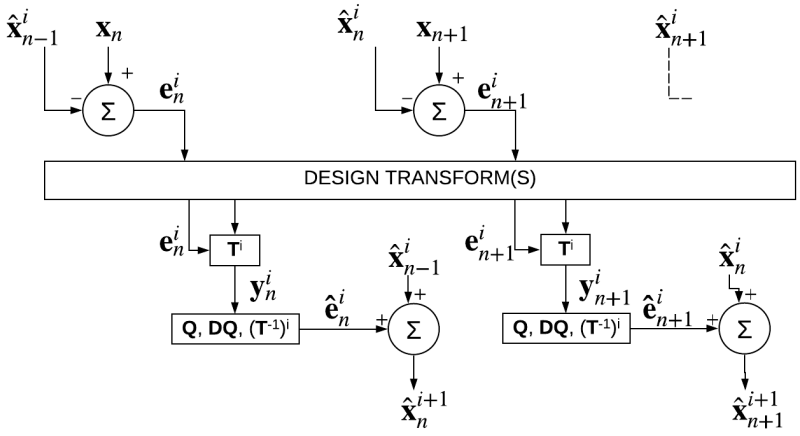


Figure 5.2: Asymptotic closed-loop design of transforms

of super-modes to effectively cover the residue statistic.

Algorithm 4: Overall design approach

initialize: reconstructed sequence from closed loop encoder, standard trigonometric transform modes;
while *ACL_iter* < *max_ACL_iter* **do**
  Generate residue statistics;
  Design transform-modes:
  while *M_mode_iter*< *Max_M_mode_iter* **do**
    (i) Assign best transform-mode to each block ;
    (ii) Design KLTs for each mode in each block size;
    break on convergence;
  end
  Update reconstructions in ACL fashion ;
  Update encoder decisions with new transforms;
  break on convergence ;
end

5.5 Experimental Results

5.5.1 Simulation Setting

We designed transforms for the AV1 codec, which includes a set of sixteen separable transform modes for block sizes of 4X4 and 8X8 and a set of twelve separable transforms for block size 16X16. We optimize these transforms from our ACL based design. We note that both in AV1 anchor and the codec that uses transforms obtained from proposed design method has rectangular partitioning of residue. For rectangular partitions default transforms from AV1 are used. We use real time encoding configuration of AV1 and perform design at fixed target bit-rate. We first illustrate the critical design instability in closed-loop design and illustrate how ACL gives a stable platform.

5.5.2 ACL as a stable design platform

In order to illustrate the stability issues with the standard closed-loop design, we consider optimizing transforms for *mobile_cif* sequence at a target bit-rate of 100Kbps. (More specific implementation details pertaining to training will be presented shortly). The YUV-PSNR of the reconstructed sequence for closed-loop design iterations are plotted in Fig.5.3. Note how in many design iterations, employing the updated transforms in-fact kills the performance compared to the previous iteration. In contrast, ACL ensures better reconstructions by using transforms on the same set of residue statistics it was designed. As can be seen from the plot, the reconstructed sequence converges after certain design iterations ensuring that the transforms obtained are optimal for closed-loop operation of the codec.

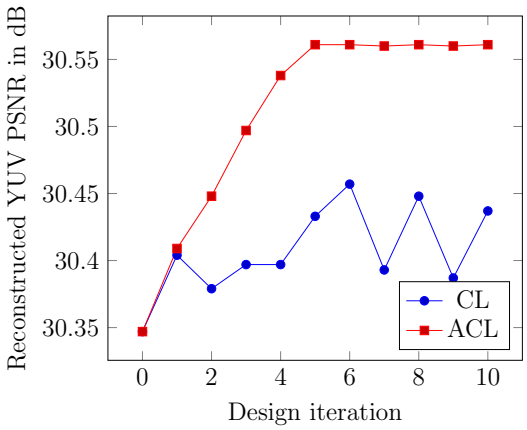


Figure 5.3: Illustration of design instability in closed-loop design and ACL as a stable platform design

5.5.3 Training

We design transforms from both the standard closed-loop design approach and the proposed ACL approach. In both these approaches, given a fixed set of reconstructions, the transforms are optimized in a "K-means" clustering approach. For the nearest neighbor step in the clustering, we plug the transforms in the codec and get the R-D optimal classification of residue to the transform modes. Once we obtain the classification of residue of each block size into different modes, we optimize the transforms in each cluster in each block size by computing the KLT for that cluster. Computation of KLT is done outside the codec and corresponds to the centroid step. The algorithm iterates between these two steps till convergence. Upon convergence, in closed-loop design, the reconstructions are updated by plugging these transforms in the codec and performing a closed-loop run of the codec. In ACL, the reconstructions are updated in open-loop fashion. To achieve this, the codec is given the reconstructions from the previous design iteration and the prediction is performed from these reconstructions instead of the reconstructions of the current iteration. The above process of transform optimization and reconstruction update are performed iteratively. For closed-loop design, design almost

always never converges. Thus, we run the design for a maximum of ten iterations. For ACL, we run the design till we hit convergence (typically achieved in 7-12 iterations). For regular 2D videos, we design transforms for cif and HD resolution sequences. For spherical videos, we consider EAC whose face width is 512. The training set is listed in Table 5.1. As mentioned earlier, the training was done at constant bit-rate configuration of AV1. Target bit-rate was varied to get different RD points. Since the residue statistic changes with bit-rate, we design transforms for each target bit-rate. The bases images obtained for the first transform mode in 8X8 blocks initialized to DC-DCT is shown in Fig.5.4 and Fig.5.5 for the case of HD sequence training at target rate of 1000 Kbps and the EAC training at target rate of 2000 kbps respectively. Note how the obtained transforms are very different from the bases images of any trigonometric transform.

5.5.4 Testing

Bit-rate reduction over the baseline (which uses trigonometric and identity transforms) is calculated as per [25]. The results for the test set sequences are shown in Tables 5.2-5.4. It is evident that the proposed method brings significant gains over AV1 for all categories of sequences.

5.6 Conclusions

This chapter presents an efficient offline-design procedure to learn transforms for inter-prediction residuals. Critical design instability was circumvented by deriving the method within the asymptotic-closed loop framework. Significant bit-rate reduction substantiates the potential of this data-driven approach to effectively learn transforms and outperform standard trigonometric transforms.

Table 5.1: The training set of video sequences for transform design

cif	HD	EAC
bridge-close	basketballdrive	broadway
bridge-far	cactus	chair
mobile	parkscene	glacier
highway	station	
foreman	tennis	
tempete		
flower		
bus		
city		

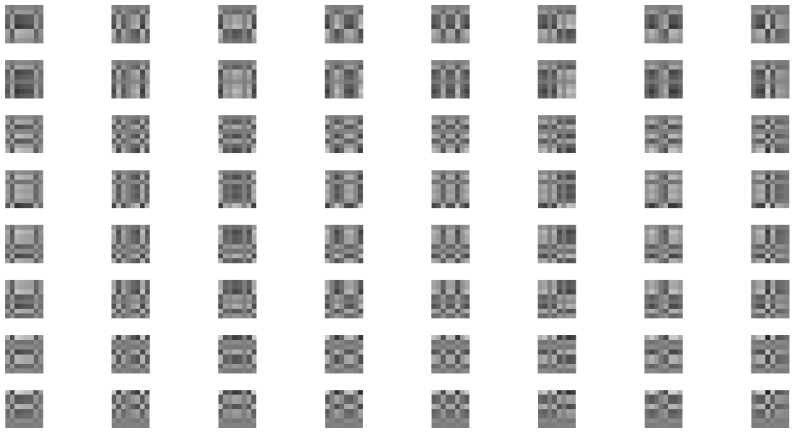


Figure 5.4: Bases images for 8X8 block for first transform mode initialized to DC-T-DCT obtained for HD sequence training at 700Kbps

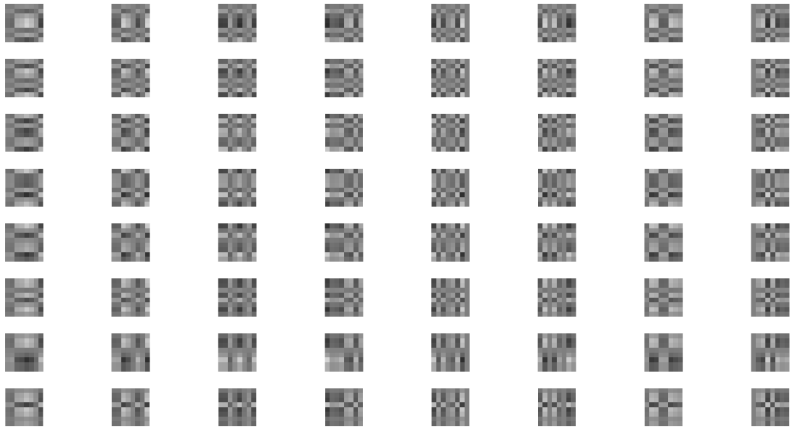


Figure 5.5: Bases images for 8X8 block for first transform mode initialized to DC-T-DCT obtained for EAC sequence training at 1000Kbps

Table 5.2: % bit-rate savings on test set for YUV-PSNR over AV1 for cif sequences (uses trigonometric transforms).

Test Sequence	Bit-rate Savings over AV1
silent	2.4
soccer	1.1
akiyo	2.7
bowing	3.3
hall	1.0
mother-daughter	4.2
paris	1.3
coastguard	2.6
stefan	2.4
ice	0.8
Average	**2.2**

Table 5.3: % bit-rate savings on test set for YUV-PSNR over AV1 for HD sequences.

Test Sequence	Bit-rate Savings over AV1
bqterrace	3.2
kimono	1.9
pedestrian	2.9
sunflower	2.3
tractor	2.0
Average	**2.4**

Table 5.4: % bit-rate savings on test set for YUV-PSNR over AV1 for EAC sequences.

Test Sequence	Bit-rate Savings over AV1
skate	3.6
driving	0.7
kite	1.3
harbor	3.4
balboa	0.8
Average	**1.9**

Bibliography

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, *et. al.*, *Overview of the high efficiency video coding (HEVC) standard*, *IEEE Transactions on Circuits and Systems for Video Technology* **22** (2012), no. 12 1649–1668.

[2] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, *et. al.*, *An overview of core coding tools in the AV1 video codec*, in *2018 Picture Coding Symposium (PCS)*, pp. 41–45, IEEE, 2018.

[3] J. P. Snyder, *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.

[4] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, *Overview of the H.264/AVC video coding standard*, *IEEE Transactions on Circuits and Systems for Video Technology* **13** (2003), no. 7 560–576.

[5] M. Narroschke and R. Swoboda, *Extending HEVC by an affine motion model*, in *Picture Coding Symposium (PCS)*, pp. 321–324, 2013.

[6] H. Huang, J. W. Woods, Y. Zhao, and H. Bai, *Control-point representation and differential coding affine-motion compensation*, *IEEE Transactions on Circuits and Systems for Video Technology* **23** (2013), no. 10 1651–1660.

[7] Y. He, B. Vishwanath, X. Xiu, and Y. Ye, *AHG8: Algorithm description of InterDigital's projection format conversion tool (PCT360)*, *Document JVET-D0021* (2016).

[8] G. Van der Auwera, M. Coban, and M. Karczewicz, *AHG8: Equatorial cylindrical projection for 360-degree video*, *Document JVET-F0026* (2017).

[9] J. Boyce, E. Alshina, A. Abbas, and Y. Ye, *Jvet common test conditions and evaluation procedures for 360◦ video*, *JVET-F1030,* (April 2017).

[10] L. Li, Z. Li, M. Budagavi, and H. Li, *Projection based advanced motion model for cubic mapping for 360-degree video*, *arXiv preprint arXiv:1702.06277* (2017).

[11] L. Li, Z. Li, X. Ma, H. Yang, and H. Li, *Advanced spherical motion model and local padding for 360° video compression*, *IEEE Transactions on Image Processing* **28** (2019), no. 5 2342–2356.

[12] F. De Simone, N. Birkbeck, B. Adsumilli, and P. Frossard, *Deformable block based motion estimation in omnidirectional image sequences*, in *IEEE 19th International Workshop on Multimedia Signal Processing*, no. EPFL-CONF-229997, 2017.

[13] J. Sauer, J. Schneider, and M. Wien, *Improved motion compensation for 360 video projected to polytopes*, in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 61–66, IEEE, 2017.

[14] Y. He, Y. Ye, P. Hanhart, and X. Xiu, *Motion compensated prediction with geometry padding for 360 video coding*, in *2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, IEEE, 2017.

[15] R. Ghaznavi Y. and A. Aminlou, *Geometry-based motion vector scaling for omnidirectional video coding*, in *2018 IEEE International Symposium on Multimedia (ISM)*, pp. 127–130, IEEE, 2018.

[16] A. Ahmmed, M. M. Hannuksela, and M. Gabbouj, *Fisheye video coding using elastic motion compensated reference frames*, in *IEEE International Conference on Image Processing (ICIP)*, 2016.

[17] G. Jin, A. Saxena, and M. Budagavi, *Motion estimation and compensation for fisheye warped video*, in *IEEE International Conference on Image Processing (ICIP)*, pp. 2751–2755, 2015.

[18] J. Boyce and Q. Xu, *Spherical rotation orientation indication for HEVC and JEM coding of 360 degree video*, in *Applications of Digital Image Processing XL*, International Society for Optics and Photonics, 2017.

[19] O. Rodrigues, *Des lois géométriques qui regissent les déplacements d'un systéme solide dans l'espace et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire*, *Journal de Mathématiques Pures et Appliquées* **5** (1840) 380–440.

[20] *High efficiency video coding test model, HM-16.15*, *https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/ tags/* (2016).

[21] Y. He, B. Vishwanath, X. Xiu, and Y. Ye, *AHG8: InterDigital's projection format conversion tool*, *Document JVET-D0021* (2016).

[22] A. Abbas, *Gopro test sequences for virtual reality video coding*, *Document JVET-C0021* (2016).

[23] A. Abbas and B. Adsumilli, *New Gopro test sequences for virtual reality video coding*, *Document JVET-D0026* (2016).

[24] *Test sequences for virtual reality video coding from Interdigital, author=Asbun, E. and Ye, Y. and Hanhart, P and He, Y. and Ye, Y., journal=Document JVET-G0055, year=2017*, .

[25] G. Bjontegaard, *Calculation of average PSNR differences between RD-curves*, *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April* (2001).

[26] B. Vishwanath, Y. He, and Y. Ye, *AHG8: Area weighted spherical PSNR for 360 video quality evaluation*, *JVET-D0072, Chengdu, CN* (2016).

[27] X. Xiu, Y. He, Y. Ye, and B. Vishwanath, *An evaluation framework for 360-degree video compression*, in *2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, IEEE, 2017.

[28] R. C. Nelson and J. Aloimonos, *Finding motion parameters from spherical motion fields (or the advantages of having eyes in the back of your head)*, *Biological cybernetics* **58** (1988), no. 4 261–273.

[29] B. Vishwanath, T. Nanjundaswamy, and K. Rose, *Rotational motion model for temporal prediction in 360 video coding*, in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2017.

[30] A. S. Spanias, *Speech coding: A tutorial review*, *Proceedings of the IEEE* **82** (1994), no. 10 1541–1582.

[31] T. Painter and A. Spanias, *Perceptual coding of digital audio*, *Proceedings of the IEEE* **88** (2000), no. 4 451–515.

[32] J. A. Hartigan and M. A. Wong, *Algorithm as 136: A K-means clustering algorithm*, *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **28** (1979), no. 1 100–108.

[33] V. Cuperman and A. Gersho, *Vector predictive coding of speech at 16 kbits/s*, *IEEE Transactions on Communications* **33** (1985), no. 7 685–696.

[34] P.-C. Chang and R. Gray, *Gradient algorithms for designing predictive vector quantizers*, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **34** (1986), no. 4 679–690.

[35] H. Khalil, K. Rose, and S. L. Regunathan, *The asymptotic closed-loop approach to predictive vector quantizer design with application in video coding*, *IEEE transactions on Image Processing* **10** (2001), no. 1 15–23.

[36] K. Rose, *Deterministic annealing for clustering, compression, classification, regression, and related optimization problems*, *Proceedings of the IEEE* **86** (1998), no. 11 2210–2239.

[37] J. Kim and J. W. Woods, *Spatiotemporal adaptive 3-D Kalman filter for video*, *IEEE Transactions on Image Processing* **6** (1997), no. 3 414–424.

[38] T. Wedi, *Adaptive interpolation filter for motion and aliasing compensated prediction*, in *Visual Communications and Image Processing 2002*, vol. 4671, pp. 415–423, International Society for Optics and Photonics, 2002.

[39] S.-J. Choi and J. W. Woods, *Motion-compensated 3-D subband coding of video*, *IEEE Transactions on Image Processing* **8** (1999), no. 2 155–167.

[40] J.-R. Ohm, *Three-dimensional subband coding with motion compensation*, *IEEE Transactions on Image Processing* **3** (1994), no. 5 559–571.

[41] J. Han, V. Melkote, and K. Rose, *Transform-domain temporal prediction in video coding: exploiting correlation variation across coefficients*, in *IEEE International Conference on Image Processing (ICIP)*, pp. 953–956, 2010.

[42] S. Li, T. Nanjundaswamy, and K. Rose, *Transform domain temporal prediction with extended blocks*, in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1476–1480, 2016.

[43] S. Li, Y. Chen, J. Han, T. Nanjundaswamy, and K. Rose, *Rate-distortion optimization and adaptation of intra prediction filter parameters*, in *IEEE International Conference on Image Processing ((ICIP)*, pp. 3146–3150, 2014.

[44] P. J. Laarhoven and E. H. Aarts, *Simulated annealing*, in *Simulated annealing: Theory and Applications*, pp. 7–15. Springer, 1987.

[45] R. D. Rosenkrantz, *ET Jaynes: Papers on probability, statistics and statistical physics*, vol. 158. Springer Science & Business Media, 2012.

[46] G. Waters, *Sound quality assessment material recordings for subjective tests*, *Users' Handbook for the EBQ-SQAM Compact Disc, European Broadcasting Union, Avenue Albert Lancaster* **32** 1180.

[47] K. R. Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014.

[48] X. Zhao, J. Chen, M. Karczewicz, A. Said, and V. Seregin, *Joint separable and non-separable transforms for next-generation video coding*, *IEEE Transactions on Image Processing* **27** (2018), no. 5 2514–2525.

[49] J. Chen, M. Karczewicz, Y. Huang, K. Choi, J.-R. Ohm, and G. J. Sullivan, *The joint exploration model (JEM) for video compression with capability beyond HEVC*, *IEEE Transactions on Circuits and Systems for Video Technology* (2019).

[50] J. Han, A. Saxena, V. Melkote, and K. Rose, *Jointly optimized spatial prediction and block transform for video and image coding*, *IEEE Transactions on Image Processing* **21** (2011), no. 4 1874–1884.

[51] C. Yeo, Y. H. Tan, and Z. Li, *Low-complexity mode-dependent KLT for block-based intra coding*, in *2011 18th IEEE International Conference on Image Processing*, pp. 3685–3688, IEEE, 2011.

[52] A. Arrufat, P. Philippe, and O. Déforges, *Non-separable mode dependent transforms for intra coding in HEVC*, in *2014 IEEE Visual Communications and Image Processing Conference*, pp. 61–64, IEEE, 2014.

[53] X. Zhao and S. Liu, *Unified secondary transform for intra coding beyond AV1*, in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3393–3397, IEEE, 2020.

[54] C. Lan, J. Xu, W. Zeng, G. Shi, and F. Wu, *Variable block-sized signal-dependent transform for video coding*, *IEEE Transactions on Circuits and Systems for Video Technology* **28** (2017), no. 8 1920–1933.

[55] K. Fan, R. Wang, W. Lin, L. Y. Duan, and W. Gao, *Signal-independent separable KLT by offline training for video coding*, *IEEE Access* **7** (2019) 33087–33093.

[56] H. E. Egilmez, Y.-H. Chao, and A. Ortega, *Graph-based transforms for video coding*, *IEEE Transactions on Image Processing* **29** (2020) 9330–9344.

[57] F. Kamisli and J. Lim, *1-D transforms for the motion compensation residual*, *IEEE Transactions on Image Processing* **20** (2010), no. 4 1036–1046.

[58] B. Zeng and J. Fu, *Directional discrete cosine transforms—A new framework for image coding*, *IEEE transactions on circuits and systems for video technology* **18** (2008), no. 3 305–313.

[59] H. E. Egilmez, O. G. Guleryuz, J. Ehmann, and S. Yea, *Row-column transforms: Low-complexity approximation of optimal non-separable transforms*, in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 2385–2389, IEEE, 2016.

[60] B. Li, O. G. Guleryuz, J. Ehmann, and A. Vosoughi, *Layered-Givens transforms: Tunable complexity, high-performance approximation of optimal non-separable transforms*, in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 1687–1691, IEEE, 2017.

[61] Y. He, X. Xiu, P. Hanhart, Y. Ye, F. Duanmu, and Y. Wang, *Content-adaptive 360-degree video coding using hybrid cubemap projection*, in *2018 Picture Coding Symposium (PCS)*, pp. 313–317, IEEE, 2018.

[62] M. Xu, C. Li, S. Zhang, and P. Le Callet, *State-of-the-art in 360 video/image processing: Perception, assessment and compression*, *IEEE Journal of Selected Topics in Signal Processing* **14** (2020), no. 1 5–26.

[63] M. Rizkallah, *Graph based transforms for compression of new imaging modalities*. PhD thesis, Université Rennes 1, 2019.

[64] B. Vishwanath, S. Li, and K. Rose, *Asymptotic closed-loop design of transform modes for the inter-prediction residual in video coding*, in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3403–3407, IEEE, 2020.