Givens Factorization Based Construction of Data-Driven Fast Transforms for Image Coding

by

Dilshan Morawaliyadda

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science

Department of Electrical and Computer Engineering

The University of Manitoba

Winnipeg, Canada

August 2025

Copyright © 2025 by Dilshan Morawaliyadda

Abstract

Adaptive transform coding using data-driven linear transforms has been shown to outperform the widely used separable two-dimensional discrete cosine transform (2D-DCT) in image and video compression. However, while the 2D-DCT benefits from efficient fast algorithms, data-driven transforms are typically unstructured and computationally expensive, limiting their practical adoption in image coding applications. In this thesis, a novel approach to image coding is proposed using low-complexity, data-driven image coding using structured orthonormal transform matrices constructed from approximate Givens factorizations optimized for transform coding.

This thesis first develops a tree-search-based coordinate descent algorithm on the orthonormal matrix manifold for the design of *Givens factorization-based fast transforms* (GFFTs) on minimizing the mean square error of high-rate transform coding, offering improvements over simple coordinate descent. Experimental results demonstrate that the theoretically predicted coding gains of the proposed GFFTs can be up to 7 dB higher than those of the 2D-DCT at comparable complexity, and up to 9 dB higher when operating at twice the complexity of the 2D-DCT.

Next, the design of a data-driven image codec based on the idea of using a predesigned set of fast transform matrices, referred to as a transform matrix codebook, is explored. To this end, a novel method for designing a transform matrix codebook is presented, which is based on the concept of the structure tensor for data-driven classification and the tree-search-based coordinate descent algorithm for designing GFFTs.

Finally, the implementation of a complete data-driven fast transform image codec is presented, which is an adaptation of the baseline JPEG codec integrating a GFFT transform codebook. Experimental results show that, for many images, the proposed GFFT-based codec outperforms the standard JPEG at comparable or lower computational complexity, achieving peak signal-to-noise ratio (PSNR) improvements of up to 6 dB in some cases.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Pradeepa Yahampath, for his unwavering support, insightful guidance, and encouragement throughout my M.Sc. program. It has been both an honor and a privilege to conduct my research under his supervision. I would like to sincerely thank the examination committee for accepting my thesis for review and for their time and consideration.

I would also like to thank the staff of the Department of Electrical and Computer Engineering for their unwavering support throughout my studies. I am deeply grateful to all my instructors for the invaluable knowledge and guidance they provided during the coursework.

A special thank you goes to my wife, Dilushi Weerawardane, for her unwavering support and encouragement throughout my studies. I am deeply grateful to my parents—my father, Nandaweera Morawaliyadda, my mother, Dhammika Padmakumari, and to my sister, Sithara Morawaliyadda, for their constant support, blessings, and encouragement throughout my academic journey. I would like to extend my heartfelt gratitude to my relations and friends for their steadfast support and encouragement throughout this journey.

Table of Contents

Abstract i

Acknowledgment iii

Table of Contents vi

List of Figures viii

List of Tables ix

List of Abbreviations xi

1 Introduction 1
- 1.1 Signal Compression . . . 1
- 1.2 Lossless Compression . . . 2
- 1.3 Lossy Compression . . . 3
- 1.4 Transform Coding . . . 5
- 1.5 Data-driven Transforms . . . 8
- 1.6 Fast Orthonormal Transforms . . . 9
- 1.7 Related Work . . . 10
- 1.8 Motivation, Objectives and Outcomes of the Research . . . 11
- 1.9 Organization of the Thesis . . . 14

2 Theoretical Background 15
- 2.1 Orthogonal Transforms for Coding . . . 16
  - 2.1.1 Optimal transform for coding . . . 18
  - 2.1.2 Energy compaction . . . 18
- 2.2 Orthogonal Matrix Manifold . . . 19
- 2.3 Givens Rotation Matrix . . . 20
- 2.4 Entropy Coding Methods in the JPEG Image Coding Standard . . . 21
  - 2.4.1 Huffman coding . . . 21
  - 2.4.2 Differential Huffman coding . . . 21
  - 2.4.3 Run length coding . . . 22

2.5 Structure Tensor . . . 23
2.5.1 Image gradient . . . 23
2.5.2 Structure tensor definition . . . 24
2.6 Metrics for Evaluating Performance of Image Encoders . . . 26
2.6.1 Coding gain . . . 26
2.6.2 Peak signal-to-noise ratio (PSNR) . . . 26
2.6.3 Structural similarity index measure (SSIM) . . . 27

3 A Method for Fast Transform Design 28
3.1 Problem Formulation . . . 28
3.2 Proposed Solution . . . 30
3.2.1 Coordinate descent on orthogonal matrix manifold . . . 30
3.2.2 Multi-path coordinate descent algorithm . . . 33
3.3 Other GFFT Design Methods . . . 36
3.4 Convergence Analysis of GFFT Design Algorithms . . . 38
3.5 Analysis of Computational Complexity . . . 43
3.5.1 Complexity of transform coding . . . 43
3.5.2 Complexity of GFFT design algorithms . . . 45

4 A Data-driven Method for CO-GFFT Design 46
4.1 Data-driven Transform Design . . . 47
4.1.1 Structure tensors classification . . . 47
4.1.2 GFFT codebook design . . . 52
4.2 Coefficient Ordering for Efficient Entropy Coding . . . 56

5 An Implementation of an Image Codec 59
5.1 JPEG Compression . . . 59
5.2 Proposed Data-driven Fast Transform Encoder . . . 60
5.2.1 Optimal Transform Selection . . . 62
5.2.2 Lossless coding of AC coefficients . . . 66
5.2.3 Transform index signaling . . . 67
5.2.4 Format of the encoded bit stream . . . 68
5.3 Decoder . . . 68
5.4 Computational Complexity of the Codec . . . 69

6 Experimental Results and Discussion 71
6.1 Choice of Transform Matrix Codebook Size . . . 72
6.2 Transform Matrix Index Coding . . . 73
6.3 Performance Comparisons . . . 74
6.3.1 Comparisons with JPEG codec . . . 76
6.3.2 Comparisons with FN-GFFT based codebook . . . 78
6.4 Complexity-Performance Trade-off of the CO-GFFT Codec . . . 83

7 Conclusion and Future Directions 88

A Generalized Lloyd Algorithm 90

B Huffman Table for coding DC Coefficients 91

C Huffman Table for coding AC Coefficients 92

D Huffman Table for coding Transform Indices 96

Bibliography 99

List of Figures

1.1 A signal compression system. 2
1.2 Quantization of a discrete time sinusoidal signal. 3
1.3 Main steps of transform coding. 4
1.4 Transform coding architecture. 5

3.1 Tree representation of coordinate descent paths on orthogonal matrix manifold. 34
3.2 MPCD algorithm convergence pattern with the complexity (L). . . . 39
3.3 MPCD algorithm convergence pattern with different initialization and different M values ($M = 1$ corresponds to the SPCD algorithm). . . . 40
3.4 Transform coding gain with the number of Givens rotation stages (L). 41
3.5 Complexity gain at a given coding gain performance. 42
3.6 Percentage complexity savings of the CO-GFFTs designed with the MPCD algorithm compared to those designed with the SPCD algorithm at different complex levels (both algorithms initiated with identity matrix). 43

4.1 Coherence distribution of the training image block dataset. 48
4.2 Images blocks with various principal gradient directions and coherence values- (a)-(d) are from the high coherence class and (e)-(h) are from the low coherence class. 49
4.3 Gaussian kernel used for calculation of structure tensor matrix. . . . 50
4.4 First 8 dominant basis vectors of data-driven KLTs for 8×8 image block classes, constructed by classifying a training-set of blocks into 63 classes (each row is a transform). 51
4.5 First 8 dominant basis vectors of data-driven CO-GFFTs for 63 classes (each row is a transform). 53
4.6 First 8 dominant basis vectors of data-driven FN-GFFTs for 63 classes (each row is a transform). 55
4.7 Transform coefficient histograms for three different classes. 58

5.1 JPEG image compression framework. 60

5.2 Proposed encoder framework for image compression application. . . . 62
5.3 Rate-distortion optimization for optimal transform selection and entropy coding in the encoder. 63
5.4 Format of the the compressed image bit stream $\boldsymbol{I}_{\text{encode}}$. 68

6.1 Sample images from training and testing dataset. 72
6.2 RD performance of 86-stage CO-GFFT codebooks of various sizes. . . 73
6.3 RD performance of 86-stage CO-GFFT codebooks when different entropy coding methods are used for encoding transform matrix indices. 74
6.4 The percentage of blocks in four coded images for which CO-GFFTs produced better RD performance than the 2D-DCT. *Test images* 1- 4 are 101_0093, 147_0196, 147_0043, and 250_0013 respectively from the Caltech image data set. 75
6.5 8×8 blocks (in blue) for which a CO-GFFT was chosen over the 2D-DCT by the RD optimization procedure. 76
6.6 Average PSNR and SSIM over a test set of 40 images. 77
6.7 Comparison of PSNR (dB) and SSIM performance for three test images. 79
6.8 Comparison of the baseline JPEG encoder and the CO-GFFT encoder for the test image *Chessboard* at 0.25 bpp rate. Values in the parentheses are the PSNRs. 80
6.9 Comparison of the baseline JPEG encoder and the CO-GFFT encoder for the Caltech test image 101_0093 at 1.0 bpp rate. Values in parentheses are the PSNRs. 81
6.10 Comparison of the baseline JPEG encoder and the CO-GFFT encoder for the Caltech test image 147_0443 at 1.0 bpp rate. Values in parentheses are the PSNRs. 82
6.11 Comparison of the baseline JPEG encoder, the FN-GFFT encoder, and the CO-GFFT encoder for test image chess_board at different bitrates.(b)-(d) at 0.15 bpp, (e)-(g) at 0.25 bpp and , (h)-(j) at 0.35 bpp. 84
6.12 Comparison of the percentage of blocks in for test images, for which the 2D-DCT was chosen in the CO-GFFT codec and the FN-GFTT codec. 85
6.13 Histograms of PSNR gains achieved by CO-GFFTs with transform complexities identical to the 2D-DCT (8×8 image blocks) and 50% that of the 2D-DCT. The horizontal axis is the PSNR gain relative to the baseline JPEG codec. The test contained 100 images of sizes ranging from 448×336 to 880×600. 86
6.14 PSNR performance of CO-GFFTs for different transform complexities relative to the 2D-DCT (test images taken from the Kodak data set). 87

List of Tables

3.1 Coding gain (dB) of CO-GFFTs (MPCD algorithm) over the 2D-DCT. 44

B.1 Huffman table for coding DC coefficients. 91

C.1 Huffman table for coding AC Coefficients. 92

D.1 Huffman table for coding optimal transform indices. 96

List of Abbreviations

1D One-dimensional

2D Two-dimensional

ANN Artificial neural network

ANS Asymmetric numeral system

bpp Bits per pixel

CO-GFFT Coding gain based GFFT

dB Deci-Bell

DCT Discrete cosine transform

DDCT Directional discrete cosine transform

DFT Discrete Fourier transform

DST Discrete sine transform

FN-GFFT Frobenius norm-based GFFT

GFFT Givens rotation-based fast transform

HEVC High efficiency video coding

JPEG Joint Photographic Experts Group

KLT Karhunen–Loève transform

MMSE Minimum mean square error

MPCD Multi-path coordinate descent

MSE Mean square error

PCA Principal component analysis

PSNR Peak signal to noise ratio

RD Rate-distortion

RLE Run-length coding

SPCD Single-path coordinate descent

SQ Scalar quantization

SSIM Structural similarity index measure

VQ Vector quantization

ZRL Zero run-length

Chapter 1

Introduction

1.1 Signal Compression

Signal compression (or *signal coding*) refers to the process of reducing the data rate required to represent a signal, such as image, video, or speech [1]. In information theory, this process is also known as *source coding*, as first introduced by Shannon in his foundational work [2]. Signal compression is categorized into either *lossless* or *lossy*. If the input signal can be perfectly reconstructed from the compressed representation, the compression process is considered lossless. Conversely, if perfect reconstruction is not possible, the compression is considered lossy. The primary objective of lossy compression is to reduce the data rate while retaining as much of the original signal information as possible. This thesis specifically focuses on lossy image compression, with the primary objective of maximizing the reconstructed image quality while minimizing the number of bits required for the representation. Given the growing demand for efficient image data handling in today's world, where the transmission bandwidth and storage capacity are critical constraints, signal compression plays a vital role

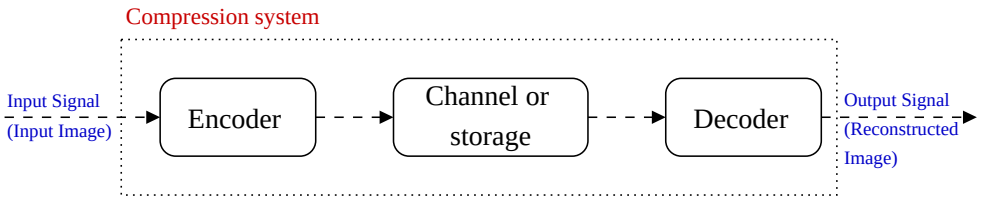


Figure 1.1: A signal compression system.

in addressing these challenges. Fig. 1.1 illustrates the general structure of a typical signal compression system.

1.2 Lossless Compression

When the input signal can be perfectly reconstructed, signal compression is considered as lossless (or *noiseless*) compression or *entropy coding*. Several entropy coding methods are available for practical use in signal compression, including *Huffman coding*, *arithmetic coding*, *asymmetric numeral systems* (ANS), *Lempel-Ziv coding*, and *Golomb coding*, among others. Each entropy coding method exhibits distinct characteristics with respect to compression efficiency, computational complexity, and adaptability to the statistical properties of the input data. Although lossless coding guarantees perfect reconstruction of the original signal, unlike lossy compression, there is a lower bound to the achievable bit rate determined by the entropy of the source [3]. Moreover, it is impossible to store or transmit a continuous (analog) signal in a truly lossless manner.

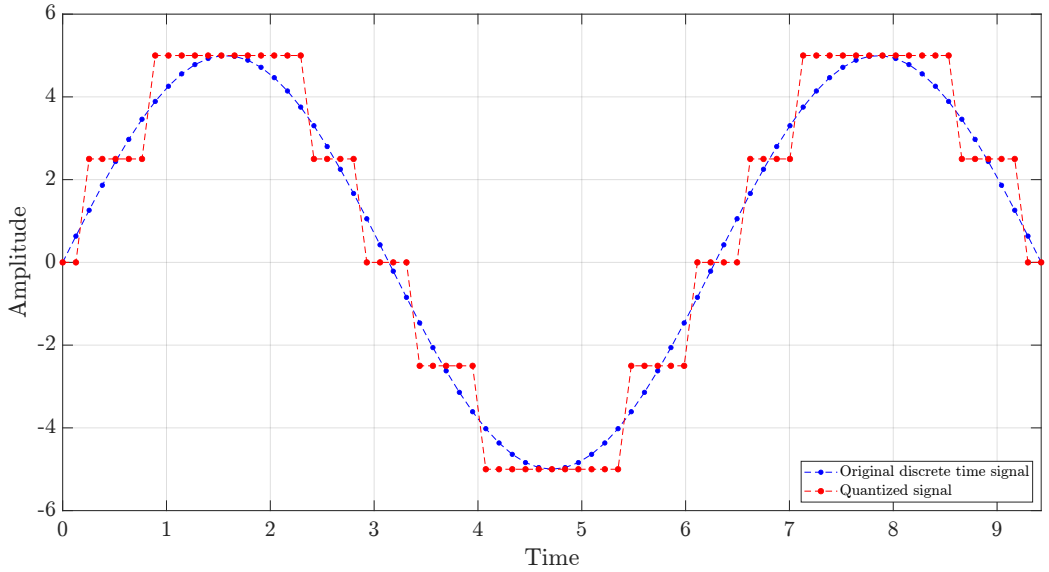
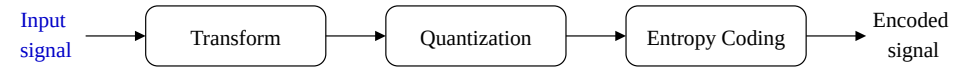


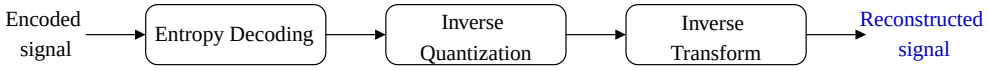
Figure 1.2: Quantization of a discrete time sinusoidal signal.

1.3 Lossy Compression

If the input signal cannot be perfectly reconstructed, the process of signal compression is considered lossy. When the signal is analog, it cannot be represented using discrete data in a fully recoverable way. The simplest lossy compression method is analog-to-digital conversion, which quantizes all signal amplitudes within a given range to a single representation value. However, it is possible to achieve much higher compression ratios for the same approximation error by using optimized quantization methods. The quantization of a sinusoidal signal is illustrated in Fig. 1.2. This introduces a trade-off between the reconstructed signal quality and the data rate. In many applications, such as speech, image, or video compression, perfect signal reconstruction is not necessary, as the human ear and eye cannot perceive very small differences in the signal. Furthermore, if an application does not require the exact signal to function properly, lossy compression can be employed to store or transmit the signal efficiently.



(a) Encoding



(b) Decoding

Figure 1.3: Main steps of transform coding.

Let a N-dimensional signal be represented as a sequence or a vector of scalar samples $\boldsymbol{X} = (x_1, x_2, \ldots, x_N)$. Scalar quantization (SQ) operates on individual samples of the signal independently, making it simple and efficient to implement, and vector quantization (VQ) quantizes the vector $\boldsymbol{X}$ jointly, allowing it to exploit inter-sample correlations. VQ is always better than SQ. According to Shannon's rate-distortion theory [1], the dimension of the VQ increases, its rate-distortion performance improves, and asymptotically approaches a theoretical limit known as the rate-distortion function of the signal [3]. However, in practice, high-dimensional VQ becomes computationally complex. For correlated signals, such as image blocks, a widely adopted and effective alternative for VQ is *transform coding*. Although it is suboptimal compared to high-dimensional VQ, transform coding offers a practical solution by first applying a linear transformation to decorrelate the signal components.

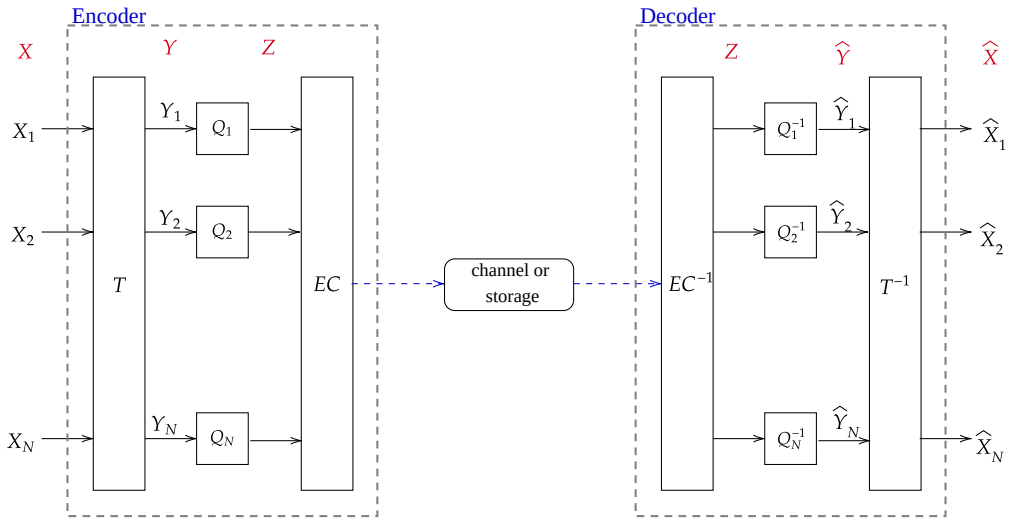


Figure 1.4: Transform coding architecture.

1.4 Transform Coding

The main idea of transform coding is to apply a linear transform to the input vector, followed by scalar quantization and entropy coding [1], as illustrated in Fig. 1.3. Suppose that we have a block of consecutive samples of a stationary random process; these samples may have substantial correlation, and separate quantization (scalar quantization) of each would be inefficient for the correlated samples. Therefore, the main objective of the transformation is to obtain a new vector of components, known as *transform coefficients*, which are uncorrelated and exhibit high energy compaction. Transform coding can compact the energy of the vector into a smaller number of coefficients after the transformation (details in Chapter 2). Then, coefficients related to sufficiently smaller variances are quantized to zero, and the coding of them consumes a very small cost in bits. Let the original signal be the N-dimensional vector $\boldsymbol{X} = (X_1 ... X_N)$ and its reconstruction be the N-dimensional vector $\hat{\boldsymbol{X}} = (\hat{X_1} ... \hat{X_N})$. The structure of the transform coding framework is illustrated in Fig. 1.4, where $\boldsymbol{T}$

and $\boldsymbol{T}^{-1}$ are the $N \times N$ real valued transform and inverse transform matrices, EC and EC^{-1} represent the entropy coding and decoding, and Q_i and Q_i^{-1} represent the quantization and inverse quantization for $i = 1, \ldots N$. The optimal trade-off between the distortion of the reconstructed signal and the bit rate needs to be found in transform coding. The most commonly used distortion measure to assess the performance of a quantizer is the *mean squared error* (MSE) distortion $\boldsymbol{D}_{TC}$, defined as [1],

$$\boldsymbol{D}_{TC} = E[||\boldsymbol{X} - \hat{\boldsymbol{X}}||^2].$$

However, because the quantization error is transformed by $\boldsymbol{T}^{-1}$ during the reconstruction operation, the choice of the transform matrix should avoid magnifying this error. Thus, the mere existence of an inverse and the ability to decorrelate are not sufficient criteria for selecting a transform matrix. Considering an *orthogonal transform* is the most effective approach to prevent the amplification of quantization error, as orthogonal transforms possesses the distance property, $||\boldsymbol{X} - \hat{\boldsymbol{X}}|| = ||\boldsymbol{Y} - \hat{\boldsymbol{Y}}||$. Furthermore, a real-valued orthogonal matrix satisfies the condition $\boldsymbol{T}^{-1} = \boldsymbol{T}^T$.

According to the well-known theory, the optimal transform for a zero-mean, Gaussian input vector $\boldsymbol{X}$ is the *Karhunen-Loeve Transform* (KLT) of the autocorrelation matrix of $\boldsymbol{X}$, which is defined as $\boldsymbol{R}_X = E[\boldsymbol{X}\boldsymbol{X}^T]$ [1]. The KLT is the matrix whose columns are the eigenvectors of $\boldsymbol{R}_X$. In image compression applications, block-based methods are widely employed, where an image is divided into blocks (transform blocks) and the pixel values within each block are encoded using transform coding. The optimal transform (KLT) for each block varies depending on the data content of that block. Consequently, if the KLT is used as the transform, it must also be encoded alongside the transform coefficients, introducing additional overhead, which reduces the overall efficiency of the image compression algorithm.

To avoid the overhead of signaling a separate transform matrix for every image block, practical compression systems often rely on a single, fixed transform. Common choices include the *discrete Fourier transform* (DFT), *discrete cosine transform* (DCT), and *discrete sine transform* (DST). The DCT is the most widely adopted fixed transform in established image compression standards due to its good energy compaction properties and real-valued basis functions. The DCT performs a frequency-domain transformation and is equivalent to the DFT applied to a symmetrically extended input. DCT provides a good approximation to the KLT of the first-order Gauss-Markov process for correlation coefficients close to one. [4]. Therefore, it exhibits good energy compaction properties for certain random processes with high inter-sample correlation. Furthermore, efficient fast algorithms exist for computing the DCT without direct matrix multiplication [4]. Since the 2D-DCT is a separable transform, it can be efficiently computed for an image by applying the 1D-DCT separately along rows and columns (details in Chapter 2). In addition to DCT, the DST is used in video coding applications such as high efficiency video coding (HEVC), along with DCT [5].

Non-linear transform coding has emerged as a prominent area of research in recent years, largely driven by advances in artificial neural networks (ANNs). Several notable works have explored non-linear transform coding [6–10]. Traditional transform coding based on linear transforms is inherently suboptimal for VQ, primarily due to computational constraints. In contrast, non-linear transforms offer greater adaptability to the source signal, enabling more efficient compression. While linear transforms typically lead to lattice-based VQ, non-linear transforms can facilitate more flexible quantization structures that better approximate the performance of optimal VQ schemes. However, high computational complexity remains one of the primary chal-

lenges associated with non-linear transform methods. In addition, their performance is often highly optimized for the training data, which may lead to poor generalization on unseen data due to overfitting. Since these methods are typically based on neural network architectures, it is also difficult to interpret intermediate representations or analyze specific improvements in the learning process.

1.5 Data-driven Transforms

The 2D-DCT is the most widely used transform in image compression. However, image blocks often exhibit diverse texture patterns that may not align well with the fixed DCT basis functions, leading to reduced energy compaction in those regions. As a result, the coding performance can be significantly worse compared to the KLT. Additionally, block-based transform coding methods may produce *block artifacts* due to poor energy compaction in such cases. Therefore, the best solution is to use a transform that is adapted to the local signal properties of the image block, which is referred to as the *data-driven transform coding* or *adaptive transform coding*. In general, transform adaptation methods can be classified into two categories, *backward adaptation* and *forward adaptation* [1]. In the backward adaptive transform coding, the transform applied to each vector is determined using previously decoded data, allowing both the encoder and the decoder to adapt synchronously. This approach uses the decoder's approximate knowledge of the past to improve the accuracy of the coding efficiency without requiring extra side information. While this limits contextual information used for adaptation, it eliminates the need to explicitly signal transform information.

The forward adaptive transform coding is a technique in which the encoder selects a transform based on the input data and explicitly signals this choice to the

decoder to enable accurate reconstruction. In this approach, an optimal transform matrix(e.g., the KLT) can be computed from the data in each block. However, this matrix must also be encoded and transmitted to the decoder along with the transform coefficients. Computing and coding a full, real-valued KLT matrix on a per-block basis is computationally intensive and impractical, especially since the exact KLT must be quantized before transmission. As a result, calculating and coding the exact KLT for each block is both costly and inefficient in practical image compression systems. Therefore, practical systems employ a predefined set of indexed transform matrices, known as a *transform matrix codebook*, from which the encoder selects the best match for an image block. The data-driven approach, which learns or derives transforms from training data, results in *data-driven transforms* used to construct a transform matrix codebook for forward-adaptive transform coding. The design of a transform matrix codebook involves learning a finite set of matrices that serve as effective approximations to the KLTs corresponding to the covariance structures of image blocks likely to occur in the images to be coded.

1.6 Fast Orthonormal Transforms

Transforming the input vector into transform coefficients is computationally expensive in image compression. Direct matrix-vector multiplication involves $O(N^2)$ of complexity for transforming an N-dimensional input vector, while DCT has $O(N \log_2 N)$ of complexity since DCT uses the well-known *butterfly structure* [4]. In image coding applications, the 2D-DCT is the most widely used transform. A transform with computational complexity comparable to the 2D-DCT is typically considered a *fast transform*.

Most data-driven transforms are designed to find a set of transforms that opti-

mally approximate the KLT, to improve energy compaction and coding performance. However, these approaches typically do not consider structured designs that allow for fast computation similar to 2D-DCT. As images used in modern multimedia applications continue to grow in size and resolution, the computational efficiency of both encoding and decoding has become an important issue. This factor is particularly relevant in real-time or resource-constrained applications. While the existing methods for designing data-driven transforms primarily focus on the trade-off between the energy compaction and the additional overhead required for signaling the transform to the decoder, fast orthogonal transforms introduce the computational complexity of the transformation as an additional design constraint.

1.7 Related Work

Previous work on the design of data-driven transforms appears in [11–14]. The method proposed in [14] employs *directional discrete cosine transforms* (DDCT) as a transform codebook. Unlike [14] which constructs generic transforms for a selected set of directional patterns, [11–13] designs more general transforms matched to actual textures appearing in images using a training set of images.

Previous works on the design of data-driven transforms, except [14], are not fast transforms. A direct mathematical approach to determining a fast transformation matrix involves approximating the optimal set of data-driven transforms that possess a structure that enables the fast computation of transform coefficients. Fast transform designs fall into two categories.

1. Optimal approximation of a pre-designed transform by a fast transform.

2. Construction of an optimal fast transform using a training set of source data

(data-driven).

Working in [15–17] belongs to the first category and approximates the KLT by formulating an objective function that minimizes the *Frobenius norm* between the KLT and its approximation. The method presented in [15] uses a row-column transform by minimizing the Frobenius norm between KLT and its approximation using an alternating minimization approach. A commonly used general approach for orthogonal matrix approximation is to represent the orthogonal matrix as a sequence of *Givens rotations*, which is a well-structured, sparse matrix (details in Chapter 2). The method proposed in [16] applies a combination of Givens rotations in a layered sequential pattern. The transform approximation method proposed in [15] is outperformed by the layered Givens rotations approximation method [16]. Another method proposed in [17] was originally developed for principal component analysis (PCA); it can also be used for approximation of a KLT due to the mathematical similarity of PCA and transform coding with the KLT. However, a key limitation of these existing approximation methods do not explicitly optimize the energy compaction of transform coefficients in their objective functions, as required for efficient transform coding.

1.8 Motivation, Objectives and Outcomes of the Research

The main objectives of the thesis are as follows:

1. To develop a principled method for designing data-driven fast transforms for efficient image compression.

2. To develop and evaluate an adaptive image codec incorporating data-driven fast transforms and comparison with the baseline JPEG image codec.

To the best of the author's knowledge, no existing codec design utilizes a data-driven fast transform codebook for adaptive coding. This thesis aims to address the limitations of previous work on data-driven fast transforms, discussed above.

- The 2D-DCT offers lower energy compaction and compression performance than the KLT due to its lack of adaptation to image statistics, but it is computationally more efficient. Therefore, the first goal of this thesis is to design a data-driven fast transform that achieves higher coding gain than the 2D-DCT (closer to KLT performance), while maintaining computational complexity comparable to the 2D-DCT. This is motivated by the fact that no principled method currently exists for designing fast, data-driven transforms that are directly optimized for image compression. The fast transform design is formulated as a problem of finding a sequence of Givens rotations, leveraging their relationship to geodesic movements on the orthogonal matrix manifold. A novel tree-search optimization method is developed to minimize the theoretical MSE of transform coding. This approach allows for explicit control over transform complexity. Proposed Givens factorization-based fast transforms, which are designed using coding gain minimization, are referred to as CO-GFFTs in this thesis. Numerical results demonstrate that the new designs outperform the existing designs in terms of theoretical coding gain for covariance matrices estimated from actual image data.

- To the best of the author's knowledge, no existing method has been proposed for designing data-driven fast transform codebooks. For image compression, to address this issue, a novel approach that uses the structure tensor to classify image

blocks into classes according to the dominant direction of intensity gradient is developed in Chapter 4. A data-driven transform codebook is then constructed by using the method developed in Chapter 3 to design a fast transform for each class. Although previously reported data-driven transform methods rely entirely on a dedicated training dataset and may produce basis vectors that lack interpretable patterns, experimental results reveal that the basis vectors learned with the method proposed in this thesis exhibit patterns consistent with the gradient directions.

- A complete implementation of an adaptive image codec, which employs a CO-GFFT codebook designed using the methodology developed in this thesis, is presented in Chapter 6. The performance of the new image codec has been compared with the baseline JPEG codec. Experimental results are presented that demonstrate that the new codec outperforms the JPEG codec for most of the images in a large test set. At the same computational complexity as the 2D-DCT, the CO-GFFT-based codec demonstrates superior performance over JPEG, with PSNR gains reaching up to 6dB in certain cases. The transform computation complexity in the proposed codec can be adjusted, and even at half the 2D-DCT complexity, some images still exhibit PSNR gains reaching up to 6dB in certain cases over JPEG.

The research presented in this thesis has been published in

- 2024 Data Compression Conference [18],
- IEEE Access [19].

1.9 Organization of the Thesis

- The theoretical background for this thesis is summarized in Chapter 2.
- Chapter 3 presents a new method for designing fast transforms based on approximate Givens factorizations.
- A novel method for designing transform matrix codebooks for image compression based on the concept of structure tensor is presented in Chapter 4.
- A complete implementation of a data-driven image codec based on fast transforms is presented in Chapter 5.
- Experimental results are presented in Chapter 6
- Conclusions and future work are presented in Chapter 7.

Chapter 2

Theoretical Background

This section presents the theoretical foundations of several key topics, beginning with a background on orthogonal transforms and the computational complexity associated with transform coding. The objective function used to minimize the MSE in transform coding is introduced, followed by a discussion of its optimal solution, the KLT. The orthogonal matrix manifold, which constitutes the solution space for this optimization problem, is also described. Entropy coding techniques used in both the JPEG algorithm and the proposed codec are addressed in the following section. Next, the definition and theoretical background of the structure tensor, which is employed for classifying training image data into transform classes for codebook construction, are presented. The performance evaluation metrics used throughout this thesis are discussed in the last section of the chapter.

2.1 Orthogonal Transforms for Coding

Consider the mean-zero random vector $\boldsymbol{X} \in \mathbb{R}^N$ representing pixels from a $W \times W$ image block whose covariance matrix is $\boldsymbol{C} = E\{\boldsymbol{X}\boldsymbol{X}^T\}$, where $N = W^2$. In transform coding, the transform coefficients are first computed as [1],

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{X}, \tag{2.1}$$

using an orthonormal matrix $\boldsymbol{T} \in \mathbb{R}^{N\times N}$ and apply scalar quantization to the elements of $\boldsymbol{Y}$ as despite in the encoder part of the Fig. 1.4. Reconstruction of the vector can be done as the inverse transform given by,

$$\hat{\boldsymbol{X}} = \boldsymbol{T}^{-1}\hat{\boldsymbol{Y}} = \boldsymbol{T}^T\hat{\boldsymbol{Y}}, \tag{2.2}$$

where $\hat{\boldsymbol{X}}$ is the reconstructed vector, $\hat{\boldsymbol{Y}}$ is the entropy decoded vector at the decoder. If $\boldsymbol{T}$ has no structure to support fast computation, the matrix-vector multiplication in the transform step requires $N(2N-1)$ additions and multiplications. If $W = 8$, which is a widely used transform block size, $N(2N-1) = 8128$ additions and multiplications are required to transform the coefficients.

If $\boldsymbol{T}$ is a separable transform then (2.1) can be expressed as,

$$\boldsymbol{Y'} = \boldsymbol{T}_c\boldsymbol{X'}\boldsymbol{T}_r^T, \tag{2.3}$$

where $\boldsymbol{X'} \in \mathbb{R}^{W\times W}$ is matrix representation of image block, $\boldsymbol{Y'} \in \mathbb{R}^{W\times W}$ is matrix representation of transform coefficients, $\boldsymbol{T}_r \in \mathbb{R}^{W\times W}$ and $\boldsymbol{T}_c \in \mathbb{R}^{W\times W}$ are row transform

and column transform respectively. Then the inverse transform is,

$$\hat{\boldsymbol{X}}' = \boldsymbol{T}_c^T \hat{\boldsymbol{Y}}' \boldsymbol{T}_r. \tag{2.4}$$

The computational complexity of a separable 2D transform is given by $2W^2(2W-1)$, where W is the block size. For $W = 8$, this results in a total of 1920 operations. Therefore, separable 2D transforms are more efficient in terms of computational complexity compared to non-separable transforms for the same transform block size. The 2D-DCT, denoted as $\boldsymbol{T}_{dct}$, is the most widely used separable transform in coding applications. It is defined as,

$$[\boldsymbol{T}_{dct}]_{ij} = \begin{cases} \frac{1}{\sqrt{k}} & i = 0 \\ \sqrt{\frac{2}{k}} cos(\frac{i\pi}{2k}(2j+1)) & i = 1, \ldots, W-1, \end{cases}$$

where $j = 0, \ldots, W-1$. Since the DCT has fast computational algorithms, the computational complexity of the 2D-DCT is much less than $2W^2(2W-1)$. The algorithm in [20] is the most computationally efficient method in the literature, requiring only 516 additions and multiplications for the 8×8 transform block size. In addition to [20], several methods have been proposed for computing 2D-DCT coefficients for 8×8 blocks [21–24]. Among these, [23, 24] focuses on pruning techniques, where only a subset of the DCT coefficients is computed. These approaches are approximations, as they discard certain coefficients even before quantization, making them suitable primarily for image coding at lower bit rates. However, since this thesis uses the 2D-DCT performance as a benchmark, an exact computation method has to be used. Therefore, the method proposed in [20] has been adopted, as it achieves the lowest known complexity with only 54 multiplications and 462 additions.

2.1.1 Optimal transform for coding

Let the MSE of transform coding of vector $\boldsymbol{X}$ using a non-separable transform be $D(\boldsymbol{T})$. The optimal $\boldsymbol{T}$ can be found by solving,

$$\boldsymbol{T}^* = \arg \min_{\boldsymbol{T} \in \mathcal{O}(N)} D(\boldsymbol{T}), \tag{2.5}$$

where $\mathcal{O}(N)$ is set of all $N \times N$ real orthonormal matrices. While the general solution to (2.5) depends on the distribution of $\boldsymbol{X}$, it is known that when $\boldsymbol{X}$ is Gaussian, $\boldsymbol{T}^* = \boldsymbol{K}^T$, where $\boldsymbol{K}$ is the KLT of $\boldsymbol{X}$ [25]. In this case, $\boldsymbol{C}' = E\{\boldsymbol{Y}\boldsymbol{Y}^T\}$ is a diagonal matrix whose diagonal elements are the eigenvalues of $\boldsymbol{C}$.

2.1.2 Energy compaction

The energy compaction, which is described as concentrating the total signal energy of input vector $\boldsymbol{X}$ in a smaller number of coefficients in the transform vector $\boldsymbol{Y}$, is an important property of the transform matrix $\boldsymbol{T}$ [26, 27]. Let, energy of the zero mean vector $\boldsymbol{X} = (X_1, X_2, \ldots X_N)^T$ be

$$\sum_{i=1}^{N} E[X_i^2] = \sum_{i=1}^{N} \sigma_{X_i}^2.$$

Since the transform $\boldsymbol{T}$ is orthogonal, $||\mathbf{X}|| = ||\mathbf{Y}||$ and hence,

$$\sum_{i=1}^{N} \sigma_{X_i}^2 = \sum_{i=1}^{N} \sigma_{Y_i}^2$$

where $\sigma_{Y_i}^2 = \lambda_i$ are eigenvalues of the covariance matrix $C = E\{\boldsymbol{X}\boldsymbol{X}^T\}$. If $\boldsymbol{X}$ is uncorrelated, λ_i's are equal. However, if $\boldsymbol{X}$ is correlated, one or more eigenvalues are different. The energy compaction of a transform can be measured by the cumulative

proportion of energy captured in the top k coefficients [26]. Let $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_N$ be the variances (eigenvalues) of the transform coefficients. The energy compaction ratio is given by:

$$\eta_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{N} \lambda_i},$$

where $\eta_k \in [0, 1]$. The KLT maximizes η_k for every $k \leq N$, making it optimal for energy compaction [26]. Since the eigenvectors of KLT are ordered in descending order, $\sigma^2_{Y_1} \geq \sigma^2_{Y_2} \geq \ldots \sigma^2_{Y_N}$, the energy of the transform coefficients is concentrated in transform coefficients with lower indices. Although widely used, the 2D-DCT is generally suboptimal in energy compaction when compared to the KLT.

2.2 Orthogonal Matrix Manifold

The first objective of this thesis is to determine a fast transform solution to (2.5) and the solution space in the problem (2.5) is a manifold, known as the *orthogonal matrix manifold*.

The set of $N \times N$ orthonormal matrices $\mathcal{O}(N) = \{\boldsymbol{T} \in \mathbb{R}^{N \times N} : \boldsymbol{T}^T\boldsymbol{T} = \boldsymbol{I}\}$ is a $N(N-1)/2$ dimensional smooth manifold, which is a subset of the Euclidean space $\mathbb{R}^{N^2}$ [28]. Associated with every point $\boldsymbol{T} \in \mathcal{O}(N)$ is a tangent space, which is an Euclidean space with dimension $N(N-1)/2$. Specifically, the tangent space at $\boldsymbol{T}$ is the set of all $\text{vec}(\boldsymbol{Z}) \in \mathbb{R}^{N(N-1)/2}$ such that $\boldsymbol{Z} = \boldsymbol{T}\boldsymbol{\Omega}$ and $\boldsymbol{\Omega} = -\boldsymbol{\Omega}^T$ (i.e., $\boldsymbol{\Omega}$ is an $N \times N$ skew-symmetric matrix).

The natural generalization of straight lines to manifolds is *geodesic curves*, which is locally the shortest curve between two points on the manifold, or equivalently, a curve with no acceleration tangent to the manifold [28]. Since finding a geodesic

curve is generally hard, it is usually parametrized as $\gamma(\eta) = \boldsymbol{T}Exp(\eta\boldsymbol{\Omega})$, where Exp is the matrix exponential and η is *step size*. The Riemannian equivalent of walking in a straight line is walking along the geodesic curve, taking a step of size $\eta > 0$ from a point $\boldsymbol{T}$, the next point can be find as $\boldsymbol{T}Exp(\eta\boldsymbol{\Omega})$. An orthonormal basis for the tangent space can be obtained by specifying a basis for the space of $N \times N$ skew-symmetric matrices. One such basis is $\text{vec}(\boldsymbol{U}_{ij})$, $0 \leq i < j \leq N-1$, where $\boldsymbol{U}_{ij}$ is the $N \times N$ skew-symmetric matrix whose (i, j) element is 1, (j, i) element is -1, and all other elements are 0's. Therefore, a straightforward calculation for the next point along the geodesic curve is to apply a planar rotation of (i, j) 2D-plane in $\mathbb{R}^{N^2}$ by a *Givens rotation matrix*.

2.3 Givens Rotation Matrix

Givens rotation matrix [29, pp. 74], denoted by $\boldsymbol{G}(i, j, \theta) \in \mathbb{R}^{N \times N}$, is defined as,

$$\boldsymbol{G}(i,j,\theta) = \begin{bmatrix} 1 & & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & & \cos\theta & \cdots & \sin\theta & & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & & -\sin\theta & \cdots & \cos\theta & & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & & 1 \end{bmatrix}$$

where $1 \leq i < j \leq N$, and $-\pi \leq \theta \leq \pi$. $\boldsymbol{G}(i, j, \theta)$, which is a sparse matrix, is identical to the identity matrix, except that G_{ii}, G_{ij}, G_{ji}, and G_{jj} elements. Linear operation $\boldsymbol{G}(i, j, \theta)\boldsymbol{X}$ is a counter-clockwise rotation of the vector $\boldsymbol{X} = (X_1, \ldots, X_N)^T$

by an angle θ about the axis perpendicular to the (X_i, X_j) plane. This operation, which involves only the ith and j^{th} elements, requires 2 additions and 4 multiplications. The identity matrix $\boldsymbol{I_N} \in \mathcal{O}(N)$, can be transformed to any $\boldsymbol{T} \in \mathcal{O}(N)$ using $m = N(N-1)/2$ Givens rotations such that [30],

$$\boldsymbol{T} = \boldsymbol{G}(i_m, j_m, \theta_n)\boldsymbol{G}(i_{m-1}, j_{m-1}, \theta_{m-1}) \ldots \boldsymbol{G}(i_1, j_1, \theta_1)\boldsymbol{I}_N. \tag{2.6}$$

2.4 Entropy Coding Methods in the JPEG Image Coding Standard

2.4.1 Huffman coding

Huffman coding [31], which yields a prefix-free code with the minimum average codeword length for a discrete random source, assigns variable-length binary code words to the input symbols based on their probabilities. By assigning longer codewords to less frequent symbols and shorter codewords to more frequent symbols, the average number of bits per symbol is optimized [32]. The average length of an optimal prefix code is always greater than or equal to the source entropy, but never exceeds it by more than one bit. Consequently, the average codeword length produced by Huffman coding lies within one bit of the source entropy, which serves as the theoretical lower bound [32].

2.4.2 Differential Huffman coding

This method is particularly effective when the neighboring symbols in the input sequence are correlated [32]. An input symbol sequence is first preprocessed by keeping

the initial symbol unchanged, while all subsequent symbols are replaced by their differences from the previous symbol. Due to the inherent correlation, smaller difference values occur more frequently than the larger ones. These different values are then encoded using a Huffman code. The differential Huffman coding process typically used in image compression involves two parts: the binary representation of the *bit length* of each difference symbol and the *value* (magnitude) of the symbol itself. A codeword table for bit length/value pairs is constructed using the Huffman coding method, based on the distribution of the bit lengths of the difference symbols, usually estimated from training data.

2.4.3 Run length coding

Run-length coding (RLE) is another technique that is typically combined with Huffman coding to encode quantized values in image compression [32]. This reduces the inefficiency of directly assigning a codeword to each repeated value in quantized transform coefficients. In this thesis, the resultant vector after quantization is a sparse signal, and repeated values are zero. Following the JPEG standard, in this thesis, RLE is used for the number of consecutive zeros (referred to as the *run*) and the bit length of the non-zero value that follows (referred to as the *length*) [33]. Huffman codewords are first used to binary encode the run/length pairs, and then the binary representation of the non-zero value is appended.

These codewords are generated using the Huffman method based on the probability distribution of the run/length pairs. As a result, each individual symbol (value) in the input sequence does not necessarily correspond to a codeword in the bitstream. A sequence of zeros followed by a non-zero value is represented by a single codeword. In this approach, the run/length pairs serve as the preprocessed symbols. While the

selection of run lengths depends on the probability distribution, the maximum run length can be limited to a value lower than the actual maximum observed in the distribution. In such cases, a special *zero run-length* (ZRL) codeword is used to represent runs of zeros that exceed the defined maximum [33]. Overall, this method results in the number of codewords in the compressed bitstream that is approximately equal to the number of non-zero values in the original signal.

2.5 Structure Tensor

2.5.1 Image gradient

The gradient of an image at a given pixel location (x, y), which is describe the edge strength and direction, can be defined by [34],

$$\nabla \boldsymbol{f}(x, y) = (I_x, I_y)^T ,$$

where, $I_x = \frac{\partial f(x,y)}{\partial x}$ and $I_y = \frac{\partial f(x,y)}{\partial y}$. The magnitude and direction of the gradient are defined as, $\sqrt{(I_x)^2 + (I_y)^2}$ and $\tan^{-1}\left(\frac{I_y}{I_x}\right)$, respectively. Then, the gradient of an image block can be evaluated using intensity variations in the vertical and horizontal directions using the *finite difference method*. However, the gradient strength and direction of an image block, which represents the average variation of the gradient within the block, can be determined as the *weighted average* of I_x and I_y over the entire block. A common choice for the weighting factor is the *Gaussian filter*, which smooths the gradient field and provides an estimate of the overall dominant direction. Nevertheless, this approach has a significant limitation; the weighted averaging process may lead to the cancellation of opposite directional derivatives, thereby reducing

the reliability of the estimated dominant gradient direction. To overcome this issue, *structure tensor* [35–37], which provides a more robust and mathematically convenient method for computing the dominant gradient direction within the block.

2.5.2 Structure tensor definition

The *structure tensor* [35–37], also known as the second-moment matrix, is a mathematical construct derived from the gradients of a function. It characterizes the distribution of gradient orientations within a specified neighborhood around a given point, providing a representation that remains invariant under coordinate transformations. In this application, the 2D structure tensor is utilized, as image data is inherently two-dimensional. For an image block centered at a pixel location $\boldsymbol{p} = (x, y)$, the 2D structure tensor $S(\boldsymbol{p})$ is defined as,

$$S(\boldsymbol{p}) = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{bmatrix}, \tag{2.7}$$

where,

$$\begin{aligned} S_{xx} &= \sum_{\boldsymbol{r}} w[\boldsymbol{r}](I_x[\boldsymbol{p} - \boldsymbol{r}]^2), \\ S_{xy} &= \sum_{\boldsymbol{r}} w[\boldsymbol{r}](I_x[\boldsymbol{p} - \boldsymbol{r}]I_y[\boldsymbol{p} - \boldsymbol{r}]), \\ S_{yy} &= \sum_{\boldsymbol{r}} w[\boldsymbol{r}](I_y[\boldsymbol{p} - \boldsymbol{r}]^2), \end{aligned}$$

and $\boldsymbol{r}$ is the offset from the center pixel $\boldsymbol{p}$ such that the location $\boldsymbol{p}-\boldsymbol{r}$ is inside the block, $w(.)$ is a fixed kernel function defined over the image block such that $\sum_{\boldsymbol{r}} w[\boldsymbol{r}] = 1$, and

the terms $I_x(.)$ and $I_y(.)$ represent the partial derivatives along x and y directions, computed using the finite difference method.

The eigenvalues λ_1, λ_2 ($\lambda_1 \geq \lambda_2 \geq 0$) and the corresponding eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2$ of the *2D structure tensor* $S(\boldsymbol{p})$ provide a summary of the gradient distribution within the selected block centered at $\boldsymbol{p}$. The window function w serves as a weighting factor in the gradient distribution computation, emphasizing contributions relative to the center pixel. The larger eigenvalue, λ_1, and its associated eigenvector $\boldsymbol{u}_1$ (or $-\boldsymbol{u}_1$) represent the *dominant direction* $\phi(\boldsymbol{p})$ given by [38],

$$\phi(\boldsymbol{p}) = \tan^{-1}\left(\frac{2S_{xy}}{S_{yy} - S_{xx}}\right).$$

The relative difference between the two eigenvalues of $S(\boldsymbol{p})$ serves as an indicator of the *gradient anisotropy* within the window. This measure, known as *coherence*, quantifies the degree to which the gradient is preferentially oriented in a specific direction [36]. The coherence $C(\boldsymbol{p})$ can be defined as,

$$C(\boldsymbol{p}) = \frac{(\lambda_1 - \lambda_2)^2}{(\lambda_1 + \lambda_2)^2}. \tag{2.8}$$

The coherence in two-dimensional image data is often characterized using various numerical measures that reflect the conditioning such as the condition number, the eigenvalue difference $(\lambda_2 - \lambda_1)^2$ measure, or any other eigenvalue-based metrics as (2.8) [35]. With this mathematical definition, $C(\boldsymbol{p})$ is taken as a value between 0 and 1, where coherence increases with increasing dominance of the dominant direction.

2.6 Metrics for Evaluating Performance of Image Encoders

2.6.1 Coding gain

Coding gain $\boldsymbol{CG}_c$ is defined as the ratio of the arithmetic mean to the geometric mean of the transform coefficient variances [1]. It can be mathematically defined as,

$$\boldsymbol{CG}_c = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\nu_i^2}{(\prod_{i=0}^{N-1}\nu_i^2)^{1/N}}, \tag{2.9}$$

where ν_i^2 is the variance of the ith transform coefficient of the covariance matrix of the transform coefficient vector. $\boldsymbol{CG}_c$ can take values in the range 1 to ∞. It is typical to express these values in decibels (dB). The coding gain $\boldsymbol{CG}_c = 1$ (0dB) occurs when all variances are equal, indicating no energy compaction. An increase in $\boldsymbol{CG}_c$ corresponds to better energy compaction and, consequently, improved transform coding efficiency.

2.6.2 Peak signal-to-noise ratio (PSNR)

PSNR is defined for a reference image $\boldsymbol{f}$ and compressed image $\boldsymbol{g}$ with size $M \times N$ as [39],

$$PSNR(\boldsymbol{f},\boldsymbol{g}) = 10\log_{10}\frac{MAX_f}{(MSE(\boldsymbol{f},\boldsymbol{g}))}, \tag{2.10}$$

where MAX_f is the maximum value of the reference image (255 for 8-bit images), and $MSE(\boldsymbol{f}, \boldsymbol{g})$ defined as,

$$MSE(\boldsymbol{f}, \boldsymbol{g}) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (f_{ij} - g_{ij})^2.$$

2.6.3 Structural similarity index measure (SSIM)

SSIM is defined for a reference image $\boldsymbol{f}$ and compressed image $\boldsymbol{g}$ with size $M \times N$ as [40],

$$SSIM(\boldsymbol{f}, \boldsymbol{g}) = \frac{(2\mu_f \mu_g + c_1)(2\sigma_{fg} + c_2)}{(\mu_f^2 + \mu_g^2 + c_1)(\sigma_f^2 + \sigma_g^2 + c_2)}, \tag{2.11}$$

where,

- μ_f is the pixel mean of $\boldsymbol{f}$,
- μ_g is the pixel mean of $\boldsymbol{g}$,
- σ_f is the sample variance of $\boldsymbol{f}$,
- σ_g is the sample variance of $\boldsymbol{g}$,
- σ_{fg} isthe sample covariance of $\boldsymbol{f}$ and $\boldsymbol{g}$,
- $c_1 = (0.01L)^2$ and $c_2 = (0.03L)^2$,
- L is the dynamic range of the pixel values.

The SSIM satisfies $SSIM(\boldsymbol{f}, \boldsymbol{g}) \in [0, 1]$, where higher values indicate greater similarity between the two images.

Chapter 3

A Method for Fast Transform Design

This chapter presents the details of the proposed fast transform design method, addressing the first objective of this thesis. Sec. 1.6 introduces the significance of fast transforms in image coding using transform coding and highlights the limitations of existing methods, and Sec. 2.1 provides the necessary mathematical background on orthogonal transforms. By shifting the focus from Frobenius norm–based approximations (Sec. 1.7) to the MSE objective specific to transform coding, the fundamental limitations of existing fast approximation methods can be effectively addressed.

3.1 Problem Formulation

An orthogonal transform $\boldsymbol{T}$ of (2.1) can be expressed as an exact Givens factorization using (2.6). Since the total number of operations required to compute (2.6) $6m = 3N(N-1)$ exceeds the $N(2N-1)$ operations required for direct computation,

the exact Givens factorization does not offer a solution to the complexity problem. Therefore, to reduce the computational complexity, an approximation for $\boldsymbol{T}$ in (2.1) can be constructed using an arbitrary sequence of planar rotations (i_l, j_l, θ_l) where $L < m$, given by,

$$\boldsymbol{T} \approx \prod_{l=1}^{L} G(i_l, j_l, \theta_l). \tag{3.1}$$

The ultimate goal of the fast transform design is to approximate the optimal transform, namely, the KLT of the sample covariance matrix $\boldsymbol{C}$, using a sequence of planar rotations optimized through the minimization of the MSE in transform coding. These sequential *Givens rotation-based fast transforms* are referred to as GFFTs in this thesis.

A tractable expression is required for $D(\boldsymbol{T})$ given in (2.5) to solve this as an optimization problem. A reasonable choice is the widely used high-rate expression [1]: for Gaussian $\boldsymbol{X}$, optimal scalar quantization and entropy coding of transform coefficients at an average bit rate of R bits/coefficient, the minimum achievable MSE for a given $\boldsymbol{T}$ as $R \to \infty$ is

$$D(\boldsymbol{T}) = c \left(\prod_{i=0}^{N-1} \nu_i^2 \right)^{\frac{1}{N}} 2^{-2R}, \tag{3.2}$$

where c is a constant, ν_i^2 is the variance of the ith transform coefficient, i.e., $\nu_i^2 = [\boldsymbol{C'}]_{ii}$ where $\boldsymbol{C'} = \boldsymbol{T}\boldsymbol{C}\boldsymbol{T}^T$ is covariance matrix of the transform coefficient vector. Then the optimization problem in (2.5) can be rewritten as,

$$\boldsymbol{T}^* = \arg \min_{\boldsymbol{T} \in \mathcal{O}(N)} c \left(\prod_{i=0}^{N-1} \nu_i^2 \right)^{\frac{1}{N}} 2^{-2R}. \tag{3.3}$$

3.2 Proposed Solution

3.2.1 Coordinate descent on orthogonal matrix manifold

Based on the theoretical explanation in Sec. 2.2 and the result presented in [41], it can be shown that solving (3.3) using coordinate descent on the orthogonal group $\mathcal{O}(N)$, starting from an initial solution $\boldsymbol{T}_0$, yields a structured orthonormal matrix that can be expressed as a product of Givens rotation matrices. Let the solution at the $(n-1)^{th}$ iteration of the coordinate descent be $\boldsymbol{T}_{n-1} \in \mathcal{O}(N)$. For a step-size $\theta > 0$, a descent step in the direction $\boldsymbol{U}_{ij}$ can be performed as [41],

$$\boldsymbol{T}_n = \boldsymbol{G}(i, j, \theta)\boldsymbol{T}_{n-1}$$

where $\boldsymbol{T}_n \in \mathcal{O}(N)$ is the new solution and $\boldsymbol{G}(i, j, \theta)$ is a Givens rotation matrix. A sequence of coordinate descent steps on $\mathcal{O}(N)$ can be interpreted as a certain sequence of planar rotations $(i_1, j_1, \theta_1), (i_2, j_2, \theta_2), \ldots$. If the coordinate descent is started from the initial solution $\boldsymbol{T}_0 \in \mathcal{O}(N)$, after n iterations, the solution is

$$\boldsymbol{T}_n = \boldsymbol{G}(i_n, j_n, \theta_n)\boldsymbol{G}(i_{n-1}, j_{n-1}, \theta_{n-1}) \ldots \boldsymbol{G}(i_1, j_1, \theta_1)\boldsymbol{T}_0 = \boldsymbol{G}(i_n, j_n, \theta_n)\boldsymbol{T}_{n-1},$$

which holds the required structure of (3.1). Given the solution point $\boldsymbol{T}_{n-1}$ at the iteration $n-1$, the optimal descent direction $\boldsymbol{U}_{i^*,j^*}$ and the corresponding step-size θ_n^* can be found by

$$(i_n^*, j_n^*, \theta_n^*) = \arg\min_{i,j,\theta} D\left(\boldsymbol{G}(i, j, \theta)\boldsymbol{T}_{n-1}\right) \tag{3.4}$$

where $D(\boldsymbol{T})$ is given by (3.2). Let $\boldsymbol{Y}_n = \boldsymbol{T}_n\boldsymbol{X}$, i.e., $\boldsymbol{Y}_n = (Y_{n,0}, \ldots, Y_{n,N-1})^T$ are the transform coefficients after the nth iteration. Then, $\boldsymbol{Y}_n$ can be written as $\boldsymbol{Y}_n = \boldsymbol{G}(i,j,\theta)\boldsymbol{Y}_{n-1}$. Since $\boldsymbol{Y}_{n-1}$ and $\boldsymbol{Y}_n$ defer only in i^{th} and j^{th} elements, $Y_{n,k} = Y_{n-1,k}$ for $k \neq i$ and $k \neq j$, $k = 0, \ldots, N-1$, and

$$\begin{bmatrix} Y_{n,i} \\ Y_{n,j} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{\boldsymbol{R}(\theta)} \begin{bmatrix} Y_{n-1,i} \\ Y_{n-1,j} \end{bmatrix},$$

where $\boldsymbol{R}(\theta)$ is a rotation matrix in $\mathbb{R}^2$. Let σ_k^2 be the variance of $Y_{n-1,k}$. It then it follows that, to minimize $D\left(\boldsymbol{G}(i,j,\theta)\right)$ in coordinate direction $\boldsymbol{U}_{i,j}$, $0 \leq i < j \leq N-1$, one must choose the descent step-size θ such that $\boldsymbol{R}(\theta)$ is a KLT of the covariance matrix of $(Y_{n-1,i}, Y_{n-1,j})^T$ [1]. Without a loss of generality, let the covariance matrix of $(Y_{n-1,i}, Y_{n-1,j})^T$ for $\boldsymbol{Y}_{n-1} = \boldsymbol{T}_{n-1}\boldsymbol{X}$ be

$$\boldsymbol{C}_{ij}(\boldsymbol{T}_{n-1}) = \begin{bmatrix} \sigma_i^2 & \rho_{ij}\sigma_i\sigma_j \\ \rho_{ij}\sigma_i\sigma_j & \sigma_j^2 \end{bmatrix},$$

where ρ_{ij} is the correlation coefficient of $Y_{n-1,i}$ and $Y_{n-1,j}$ and $|\rho_{ij}| < 1$. If $\boldsymbol{R}(\theta)$ is a KLT of $\boldsymbol{C}_{ij}(\boldsymbol{T}_{n-1})$, $(\cos\theta, -\sin\theta)^T$ and $(\sin\theta, \cos\theta)^T$ are the eigenvectors of $\boldsymbol{C}_{ij}(\boldsymbol{T}_{n-1})$, and it is straightforward to verify that optimal θ for a given (i,j) is

$$\theta_{ij}^* = \begin{cases} \alpha_{ij} & \text{if } \alpha_{ij} \geq 0 \\ \frac{\pi}{2} - \alpha_{ij} & \text{otherwise,} \end{cases} \tag{3.5}$$

where

$$\alpha_{ij} = \frac{1}{2}\tan^{-1}\left|\frac{2\rho_{ij}\sigma_i\sigma_j}{\sigma_i^2 - \sigma_j^2}\right|.$$

Algorithm 1: Single-path coordinate descent method

Data: $\boldsymbol{C}$,$\boldsymbol{T}_0$,L
Result: **rotation** sequence($\boldsymbol{S}$)
$\boldsymbol{C}_0 \leftarrow \boldsymbol{T}_0\boldsymbol{C}\boldsymbol{T}_0^T$;
while $n \leq L$ **do**
  find optimal direction $\boldsymbol{U}_n$ using (3.7) and $\boldsymbol{C}_{n-1}$;
  calculate θ_n using (3.5) and $\boldsymbol{C}_{n-1}$;
  calculate $\boldsymbol{G}_n$ using $\boldsymbol{U}_n$ and θ_n ;
  $\boldsymbol{C}_n \leftarrow \boldsymbol{G}_n\boldsymbol{C}_{n-1}\boldsymbol{G}_n^T$;
  $\boldsymbol{S}(n) \leftarrow (\boldsymbol{U}_n, \theta_n)$;
end

The product of the eigenvalues of $\boldsymbol{C}_{ij}(\boldsymbol{T}_{n-1})$ is $\det\left[\boldsymbol{C}_{ij}(\boldsymbol{T}_{n-1})\right] = \sigma_i^2\sigma_j^2(1-\rho_{ij}^2)$ which is also the product of the variances of $Y_{n,i}$ and $Y_{n,j}$, and hence

$$D\left(\boldsymbol{G}(i,j,\theta^*)\boldsymbol{T}_{n-1})\right) = c\left((1-\rho_{ij}^2)\prod_{k=0}^{N-1}\sigma_k^2\right)^{\frac{1}{N}}2^{-2R}.$$

Noting that $D(\boldsymbol{T}_{n-1}) = c\left(\prod_{k=0}^{N-1}\sigma_k^2\right)^{\frac{1}{N}}2^{-2R}$, the new transform coding MSE as a result of taking the optimal descent step θ^* starting from $\boldsymbol{T}_{n-1}$ in some $\boldsymbol{U}_{ij}$ direction, $0 \leq i < j \leq N-1$, is given by

$$D(\boldsymbol{T}_n) = (1-\rho_{ij}^2)^{\frac{1}{N}}D(\boldsymbol{T}_{n-1}). \tag{3.6}$$

Clearly, the direction $\boldsymbol{U}_{ij}$ which minimizes $D\left(\boldsymbol{G}(i,j,\theta_{ij}^*)\right)$ is (i,j) such that,

$$(i^*,j^*) = \max_{0\leq i<j\leq N-1}\rho_{ij}. \tag{3.7}$$

Therefore, optimal direction $\boldsymbol{U}_{ij}$ at the nth step is the maximum correlation coefficient in $\boldsymbol{C}(\boldsymbol{T}_{n-1})$. Algorithm 1 shows the greedy coordinate descent algorithm for finding the sequence of Givens rotations using the closed form solution (3.5) and the exhaus-

tive search-based solution (3.7). This solution method is referred to as *single-path coordinate descent method* (SPCD).

3.2.2 Multi-path coordinate descent algorithm

The SPCD algorithm described in Algorithm 1 has two main limitations.

- It performs descent along only a single coordinate direction at each iteration, which can lead to slow convergence compared to full steepest descent methods. Since each iteration introduces one Givens rotation factor into the approximation in (3.1), the resulting transform may be an inefficient approximation of the optimal transform.

- Coordinate descent is inherently a greedy algorithm, as it selects the best direction $\boldsymbol{U}_{i,j}$ from among the $N(N-1)/2$ possible coordinate directions in each iteration, without considering long-term optimization effects.

Therefore, the SPCD algorithm does not necessarily ensure that the resulting sequence of Givens rotations obtained after L iterations has the lowest $D(\boldsymbol{T}_L)$ among all $(N(N-1)/2)^L$ possibilities for a sequence of L Givens rotations. This can be visualized by interpreting each possible sequence of Givens rotations in (3.1) as a path through a tree, where each node branches into $(N(N-1)/2)$ possible directions, each corresponding to a coordinate direction as illustrated in Fig. 3.1. A node at depth n in the tree corresponds to an orthonormal matrix $\boldsymbol{T}_n$ with an associated $D(\boldsymbol{T}_n)$. When the maximum tree depth is constrained to L, the tree will have $(N(N-1)/2)^L$ terminal nodes, which correspond to all possible solutions for $\boldsymbol{T}_L$. The greedy solution found by the SPCD algorithm is one of the terminal nodes, and clearly, it is not necessary that this node has the lowest $D(\boldsymbol{T}_L)$ among all terminal nodes at depth L.

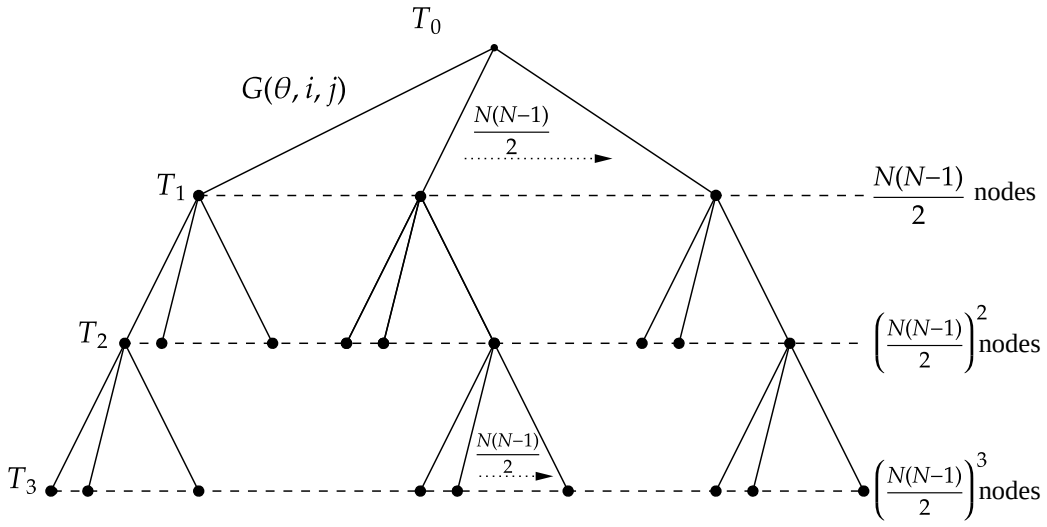


Figure 3.1: Tree representation of coordinate descent paths on orthogonal matrix manifold.

The optimal solution can be found by searching the tree for the *minimum mean square error* (MMSE) path. Each of the $N(N-1)/2$ branches originating from a node $\boldsymbol{T}_n$ corresponds to a coordinate direction $\boldsymbol{U}_{ij}$, $0 \leq i < j \leq N-1$. An additive path metric can be obtained by considering the logarithmic form of (3.6)

$$\log D(\boldsymbol{T}_n) = \frac{1}{N} \log(1 - \rho_{ij}^2) + \log D(\boldsymbol{T}_{n-1}), \tag{3.8}$$

where the path metric $d_{ij}(\boldsymbol{T}_{n-1}) = \frac{1}{N} \log(1 - \rho_{ij}^2)$ is the change in transform coding MSE when a descent step as given by (3.5) is taken from $\boldsymbol{T}_{n-1}$ in $\boldsymbol{U}_{ij}$ direction on $\mathcal{O}(N)$. Thus, the best sequence of Givens factors can be determined by searching the tree for the path which has the minimum $J(\boldsymbol{T}_L) = \sum_{n=1}^{L} d_{i_n j_n}(\boldsymbol{T}_{n-1})$, where (i_n, j_n), $n = 1, \ldots, L$ represent a path. The number of paths in the tree grows exponentially with the path length L, and an exhaustive tree search is clearly impractical for any useful value of L. However, a constrained-complexity tree search algorithm, such as the M-algorithm, can be used to find the optimal path with a high probability in a

Algorithm 2: Multi-path coordinate descent algorithm

Data: $\boldsymbol{C}$,$\boldsymbol{T}_0$,L,M
Result: Givens rotation sequence($\boldsymbol{S}_n(1)$)
Initialize arrays for Givens rotation sequence matrix $(n-1)$ and n steps as $\boldsymbol{S}_{n-1}(M)$ and $\boldsymbol{S}_n(M)$, MSE sequence as $\boldsymbol{D}(M)$, covariance matrix sequence of $(n-1)$ and n steps as $\boldsymbol{C}_{n-1}(M)$, $\boldsymbol{C}_n(M)$
for $m = 1$ *to* M **do**
  $\boldsymbol{C}_{n-1}(m) \leftarrow \boldsymbol{T}_0 \boldsymbol{C} \boldsymbol{T}_0^T$;
end
while $l \leq L$ **do**
  for $m = 1$ *to* M **do**
    $\boldsymbol{C}_{temp} \leftarrow \boldsymbol{C}_{n-1}(m)$;
    for $\boldsymbol{U}p = 1$ *to* $N(N-1)/2$ **do**
      calculate θ_{temp} using (3.5)and $\boldsymbol{C}_{temp}$;
      calculate $\boldsymbol{G}_{temp}$ using θ_{temp} ;
      calculate MSE (D_{temp}) using $\boldsymbol{G}_{temp}$ and $\boldsymbol{C}_{temp}$;
      if $D_{temp} < \boldsymbol{D}(M)$ **then**
        $\boldsymbol{D}(M) \leftarrow D_{temp}$;
        $\boldsymbol{S}_n(M) \leftarrow [\boldsymbol{S}_{n-1}(m), (\boldsymbol{U}_p, \theta_n)]$;
        $\boldsymbol{C}_n(M) \leftarrow \boldsymbol{G}_n \boldsymbol{C}_{n-1}(m) \boldsymbol{G}_n^T$;
        sort $\boldsymbol{D}(M)$ ascending order and rearrange $\boldsymbol{S}_n$ and $\boldsymbol{C}_n$ as same.;
      end
    end
  end
  $\boldsymbol{C}_{n-1} \leftarrow \boldsymbol{C}_n$;
  $\boldsymbol{S}_{n-1} \leftarrow \boldsymbol{S}_n$;
end

time that grows only linearly with L [42]. For $M = 1$, the M-algorithm reduces to the greedy SPCD algorithm. Algorithm 2 shows the *multi-path coordinate descent algorithm* (MPCD) which incorporates with the M-algorithm. The MPCD is a method to construct a GFFT that minimizes the high-rate MMSE of transform coding. This is also equivalent to minimizing the theoretical transform coding gain. Therefore, a fast transform found in this manner is referred to as *coding gain-based GFFT* (CO-GFFT) in the thesis. In order to ensure that the solution found by the MPCD algorithm is never worse than that found by the SPCD algorithm, the steepest descent direction

is included as one of the paths in the M-paths retained by the MPCD algorithm.

One Givens rotation operation requires only 6 multiplications and additions to transform the coefficients. Therefore, if the MPCD algorithm converges in $n = L$ iterations, the product of Givens factors in (3.1) requires $6L$ computations. If $\boldsymbol{T}_0$ is the identity matrix, then the computational complexity of (2.1) using $\boldsymbol{T}_L$ (CO-GFFT) in (3.1) is also the same. In comparison, the direct computation of an arbitrary $\boldsymbol{T}$ (e.g., KLT) requires $N(2N-1)$ multiplications and additions for an $N \times N$ transform. Thus, if $L << N(2N-1)/6$, the obtained solution is a fast transform. Furthermore, if $\boldsymbol{T}_0$ is the 2D-DCT, the convergence of the MPCD algorithm can be faster, and L can be even lower. However, the computational complexity then includes that of the 2D-DCT as well.

In this thesis, γ_{DCT}, γ_{KLT}, γ_{SP}, and γ_{MP} are respectively used as the notation for operational complexity of the 2D-DCT, KLT, and the CO-GFFTs obtained by SPCD and MPCD algorithms. For 8×8 transform block size, $\gamma_{\text{KLT}} = 8128$ multiply/add and $\gamma_{\text{DCT}} = 516$ multiply/add [20]. Furthermore, for L-stage approximations $\gamma_{\text{SP}} = \gamma_{\text{MP}} = 6L$.

3.3 Other GFFT Design Methods

An alternative approach to designing Givens rotation-based transforms is proposed in [16], where an orthonormal transform matrix $\boldsymbol{H}$ is approximated by a structured matrix $\boldsymbol{T}$, consisting of multiple layers of permutation matrices and Givens rotations. This method is referred to as *Layered*-GFFT in this thesis. In this case, the objective function that is minimized is $\|\boldsymbol{H} - \boldsymbol{T}\|_F$, where $\boldsymbol{H}$ denotes the optimal transform (e.g., the KLT), and $\boldsymbol{T}$ is its approximation. In [16], this problem is solved by using

an alternating optimization algorithm, which gives a solution of the form:

$$\boldsymbol{T}^* = \boldsymbol{G}_M \dots \boldsymbol{G}_1 \boldsymbol{G}_0 \boldsymbol{P}_0, \tag{3.9}$$

where $\boldsymbol{G}_l$, $l = 1, 2, \dots L$ are Givens rotation matrices of each layer l and $\boldsymbol{P}_0$ is a permutation matrix which is used to rearrange the columns. Each $\boldsymbol{G}_l$ consists of $N/2$ Givens rotations, and it can be further represented as,

$$\boldsymbol{G}_l = \boldsymbol{G}_{l,1} \boldsymbol{G}_{l,2} \dots \boldsymbol{G}_{i,N/2}. \tag{3.10}$$

A key limitation of minimizing a matrix distance, such as the Frobenius norm, without incorporating additional structural constraints is that the resulting approximation $\boldsymbol{T}$ may fail to preserve the original eigen-structure of $\boldsymbol{T}$, which is critical for achieving high performance in transform coding. This issue becomes particularly critical when only a limited number of Givens rotations (or stages) are used. For $\boldsymbol{T}$ to effectively approximate the KLT $\boldsymbol{H}$, it is not sufficient for it to be merely close to $\boldsymbol{T}$ in terms of matrix norm; rather, the columns of $\boldsymbol{T}$ must be well-aligned with those of $\boldsymbol{H}$. Then, the directional characteristics and energy compaction properties, which are fundamental to the efficacy of the transform, are effectively preserved. The method proposed in [17] addresses this limitation by explicitly considering the eigen-structure of the optimal transform. Originally developed in the context of PCA, the objective function for the fast approximation of the principal component matrix $\boldsymbol{U}_p$ is formulated as,

$$\min_{\boldsymbol{T} \in \mathcal{O}(N)} \left\| \boldsymbol{U}_p \boldsymbol{\Sigma}_p - \bar{\boldsymbol{U}}_p \bar{\boldsymbol{\Sigma}}_p \right\|_F, \tag{3.11}$$

where $\bar{\boldsymbol{U}}_p$ is the approximation using givens rotations, $\boldsymbol{\Sigma}_p$ is the diagonal matrix of eigenvalues, and $\bar{\boldsymbol{\Sigma}}_p$ is a diagonal matrix that is optimized as part of the solution. The inclusion of weighting by eigenvalues in the objective function enables this approach to more effectively preserve the eigenstructure during the approximation process. Due to the strong connection between PCA and transform coding, this method can be naturally extended to approximate a data-driven KLT using Givens rotations. The final solution takes the form of a sequential product of Givens rotations across L stages, expressed as,

$$\boldsymbol{U_p}^* = \boldsymbol{U}_L \boldsymbol{U}_{L-1} \dots \boldsymbol{U}_1. \tag{3.12}$$

where $\boldsymbol{U}_l$ corresponds to Givens rotation (i_l, j_l, θ_l). In this thesis, the fast transforms constructed method in [17] is referred to as FN-GFFTs.

3.4 Convergence Analysis of GFFT Design Algorithms

Since the complexity of a CO-GFFT depends on the number of Givens rotation steps L, the convergence of the theoretical coding gain to its maximum as a function of L is a desirable property of a GFFT design algorithm. This section compares the convergence of the MPCD algorithm with the design methods in [16,17] and [43]. The fast transform design method in [43] is a similar solution to the SPCD algorithm, and the authors of [43] arrive at their algorithm based on heuristic arguments regarding Givens rotations, without recognizing it as manifold coordinate descent.

The proposed MPCD algorithm's input is the covariance matrix $\boldsymbol{C}$, and the output is the Givens rotation sequence $\boldsymbol{S}$. The initial solution $\boldsymbol{T}_0$ and the tree-search

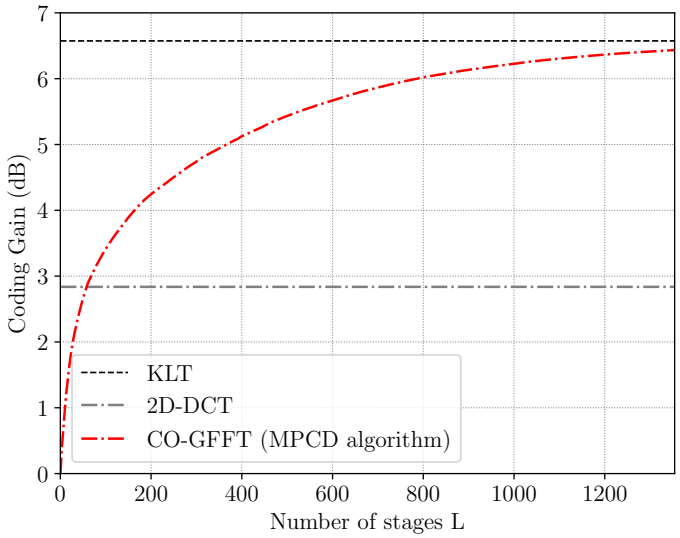
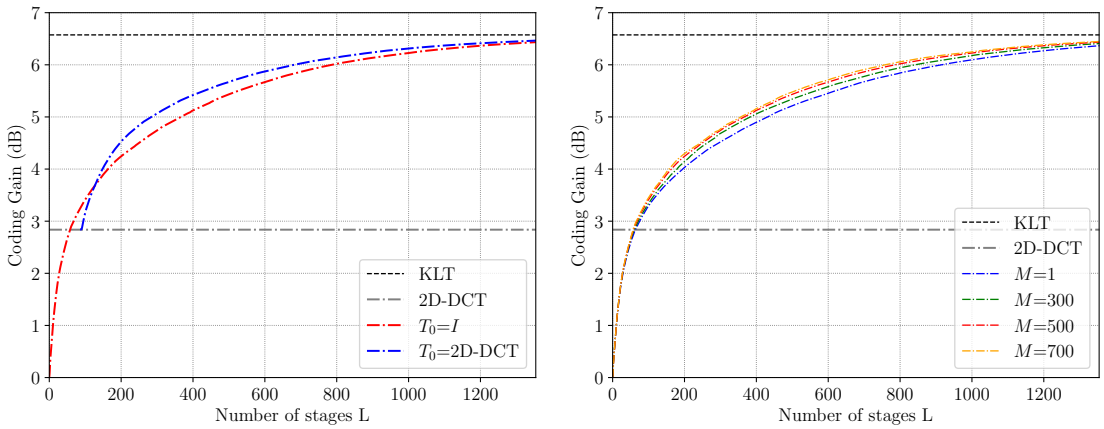


Figure 3.2: MPCD algorithm convergence pattern with the complexity (L).

parameter M are two hyperparameters of the method. Convergence analysis is performed using a set of covariance matrices estimated from standard 512×512 grayscale images (*pirate*, *livingroom*, and *lena*). 64×64 covariance matrices are estimated using non-overlapping 16×16 macroblocks such that all possible 8×8 transform blocks within a macroblock are used to estimate the covariance matrix of the macroblock. Fig. 3.2 shows the typical convergence pattern of the MPCD algorithm initialized with the identity matrix and $M = 500$, for a given covariance matrix. When $L = 1354$ (the number of stages for which $\gamma_{MP} = \gamma_{KLT}$), the solution of the MPCD algorithm converges to the KLT. Furthermore, faster convergence can be observed when L is lower, while the convergence rate decreases as L is increased..

Fig. 3.3a, which shows the convergence of the MPCD algorithm when T_0 is the 2D-DCT. At the 2D-DCT complexity level, the MPCD algorithm initialized with the identity matrix provides a better final solution than 2D-DCT for the given covariance matrix. However, for a given covariance matrix, if the coding gain obtained from the 86-stage (equivalent to γ_{DCT}) solution found by the MPCD algorithm initialized with



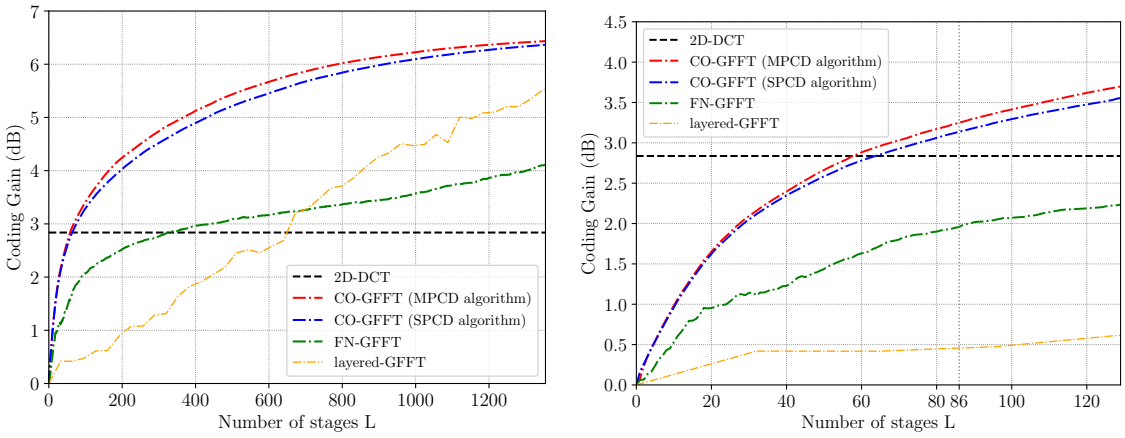
(a) Different T_0 (b) Different M

Figure 3.3: MPCD algorithm convergence pattern with different initialization and different M values ($M = 1$ corresponds to the SPCD algorithm).

the identity matrix is worse than that of the 2D-DCT, the MPCD algorithm can be initialized with the 2D-DCT to find a better solution—at the cost of an additional complexity beyond γ_{DCT}.

The purpose of using the M-algorithm is to reduce tree-search complexity. While a lower M helps reduce the tree-search complexity, as discussed in the design complexity analysis section (Sec. 3.5), a higher M increases the likelihood of finding the optimal solution. Therefore, the trade-off between the goodness of the final solution and the search complexity is an important parameter in the MPCD algorithm. Fig. 3.3b shows the effect of M on the convergence of the MPCD algorithm. $M = 500$ has been used to obtain the results presented in the rest of the thesis since increasing M beyond 500 only produced marginal improvements. $M = 500$ corresponds to approximately 25% of the total number of branches, $N(N-1)/2$, in the search tree of the MPCD algorithm for $N = 64$.

Fig. 3.4 illustrates the convergence patterns of GFFTs designed using the aforementioned design methods. The results show that CO-GFFTs (both MPCD and



(a) Complexity up to γ_{KLT} (b) Complexity up to $1.5 \times \gamma_{DCT}$

Figure 3.4: Transform coding gain with the number of Givens rotation stages (L).

SPCD algorithms) outperform the 2D-DCT, whereas the fast transforms obtained via matrix-distance minimization methods (Layered-GFFT and FN-GFFT) require a much larger number of stages to surpass the 2D-DCT. Among the matrix-distance minimization methods, the FN-GFFT performs better than the Layered-GFFT at γ_{DCT}. As can be seen in Fig. 3.4, even though the MPCD and SPCD algorithms have similar convergence rates, the MPCD algorithm consistently outperforms the SPCD. As shown in the Fig. 3.5, for a given covariance matrix, while the SPCD algorithm requires $L_{SP} = 600$ stages to achieve the 5.44dB coding gain, the MPCD algorithm requires only $L_{MP} = 509$ stages to achieve the same coding gain. To further investigate the advantage of using the MPCD algorithm over the SPCD algorithm, the percentage *complexity saving* ($\frac{\gamma_{SP}-\gamma_{MP}}{\gamma_{SP}}\%$) was computed for a set of covariance matrices estimated from image data. As stated in the introduction of the thesis, a transform is considered fast if the computational complexity is comparable to that of the 2D-DCT. Since $\gamma_{KLT} = 16\gamma_{DCT}$ for 8×8 transform blocks above investigation was done up to $2\gamma_{DCT}$ complexity. The complexity saving values are calculated as per the given definition at the coding gain values of SPCD at the $\gamma_{\mathrm{SP}} = \alpha\gamma_{\mathrm{DCT}}$, where

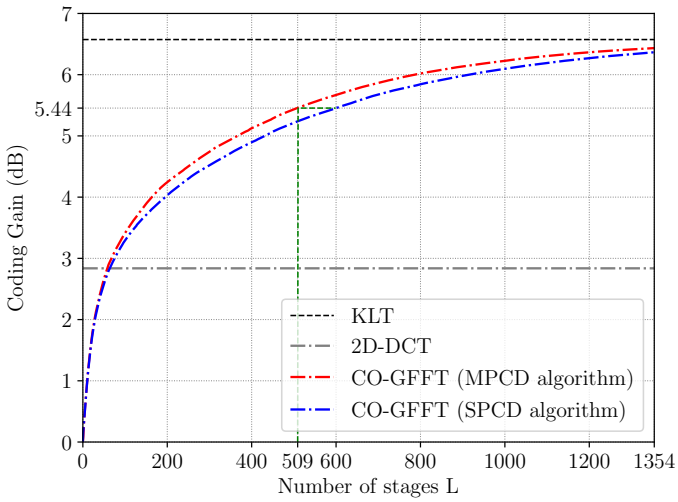
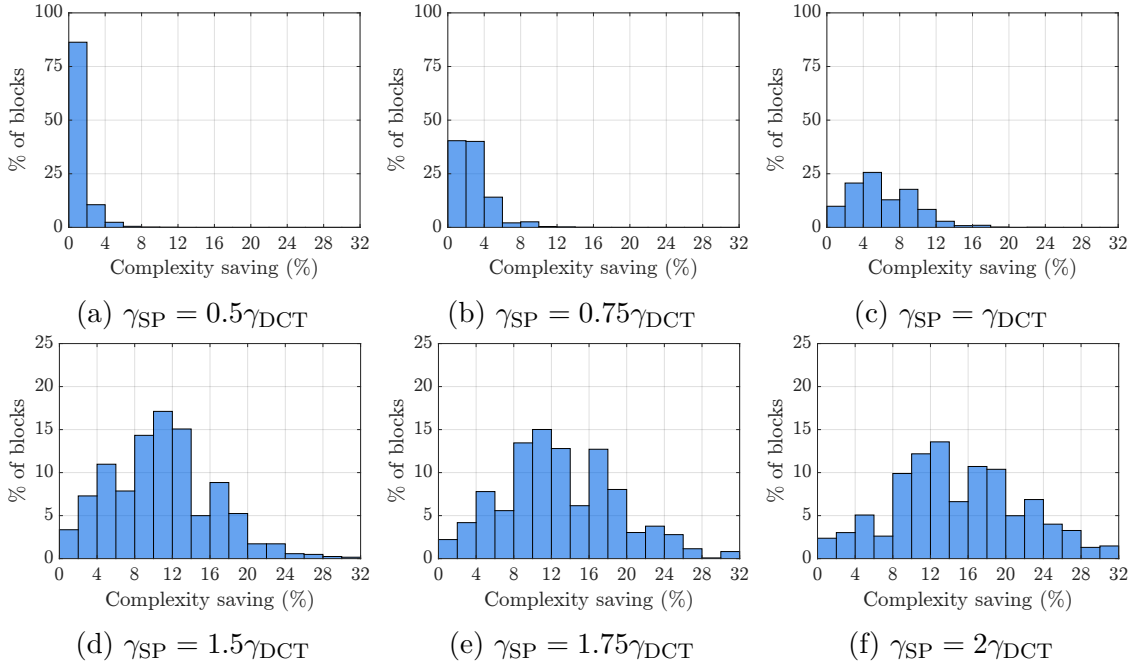


Figure 3.5: Complexity gain at a given coding gain performance.

$\alpha = 0.5,\ 0.75,\ 1,\ 1.5,\ 1.75,\ 2$. Fig. 3.6 shows the histograms of macroblocks for various levels of complexity savings of the MPCD algorithm over the SPCD algorithm. These histograms indicate that $\gamma_{\text{SP}} = \gamma_{\text{DCT}}$, the MPCD algorithm can yield up to a 28% complexity saving. Note that the percentage of blocks for which the MPCD algorithm yields transform matrices with significant complexity savings increases as the allowed complexity relative to the 2D-DCT is increased.

The first column of Table 3.1 presents the observed range of coding gains achieved by transform matrices found with the MPCD algorithm over the 2D-DCT at each complexity level, when the optimization is initialized with the identity matrix. While this initialization yielded fast transforms that outperformed the 2D-DCT for the majority of image blocks after 86 iterations, there remained some blocks for which the 2D-DCT achieved higher coding gain. For these cases, the MPCD algorithm can be initialized with the 2D-DCT itself, allowing the resulting transform to surpass the coding gain of the original 2D-DCT, as demonstrated in the last column of Table 3.1. The above convergence analysis of various design algorithms shows that GFFTs designed with the MPCD algorithm, which minimizes the transform coding MMSE,



(a) $\gamma_{\text{SP}} = 0.5\gamma_{\text{DCT}}$ (b) $\gamma_{\text{SP}} = 0.75\gamma_{\text{DCT}}$ (c) $\gamma_{\text{SP}} = \gamma_{\text{DCT}}$

(d) $\gamma_{\text{SP}} = 1.5\gamma_{\text{DCT}}$ (e) $\gamma_{\text{SP}} = 1.75\gamma_{\text{DCT}}$ (f) $\gamma_{\text{SP}} = 2\gamma_{\text{DCT}}$

Figure 3.6: Percentage complexity savings of the CO-GFFTs designed with the MPCD algorithm compared to those designed with the SPCD algorithm at different complex levels (both algorithms initiated with identity matrix).

achieve the highest theoretical coding gain over the 2D-DCT at the same complexity. Furthermore, an even better coding gain can be achieved with the CO-GFFTs if an additional complexity range of up to $2\gamma_{\text{DCT}}$ is allowed. Among the two algorithms, the MPCD algorithm produces CO-GFFTs with a higher coding gain than those produced by the SPCD algorithm. However, the MPCD algorithm has a higher design complexity than the SPCD algorithm.

3.5 Analysis of Computational Complexity

3.5.1 Complexity of transform coding

An L-stage CO-GFFT or FN-GFFT is a sequence of L Givens rotations. An L-stage layered GFFT is a sequence of layers, each consisting of Givens rotations. The num-

Table 3.1: Coding gain (dB) of CO-GFFTs (MPCD algorithm) over the 2D-DCT.

$\frac{\gamma_{SP}}{\gamma_{DCT}}$	Range of coding gain (dB)	
	$\boldsymbol{T}_0 = \boldsymbol{I}$	$\boldsymbol{T}_0$=2D-DCT
0.50	-5.27 to 5.74	-
0.75	-1.63 to 6.36	-
1.00	-0.31 to 7.38	-
1.50	0.36 to 8.48	0.66 to 2.85
1.75	0.50 to 8.79	0.81 to 3.19
2.00	0.62 to 9.03	0.94 to 3.47

ber of stages L determines the complexity- each stage requires 6 multiplication and addition operations. The computational complexity of (2.1) is directly proportional to the number of Givens rotations L.

Among GFFTs produced by different algorithms, the layered-GFFT [16] has the highest complexity, as this design algorithm has the slowest convergence. Each layer of a layered GFFT involves $N/2$ Givens rotations, where all indices are employed to generate $N/2$ planar rotations, ensuring that the same index is not reused for planar rotations in a single layer. Since the design parameter that controls the complexity is the number of layers, the transform complexity of the layered-GFFT is a multiple of $N/2$ planar rotations [16]. In addition to the Givens rotations, this method incorporates a rearrangement, which can be mathematically represented using a permutation matrix to reorder the Givens rotation index pairs. Each layer involves two forward and backward permutations to ensure that the coefficient order remains fixed.

An FN-GFFT [17] employs sequential Givens rotations, similar to the CO-GFFTs, and requires more rotations than a CO-GFFT designed using the MPCD algorithm to obtain the same coding gain performance, since the objective function used in the FN-GFFT algorithm does not directly maximize the transform coding gain. The FN-GFFT design method prioritizes approximating the eigenvectors corresponding

to higher eigenvalues, which enhances the energy compaction and the coding gain compared to the layered-GFFT design method. In contrast, a significant complexity saving can be achieved at the same coding gain performance with a CO-GFFT design using the MPCD algorithm, whose solution is lower-bounded by that of the SPCD algorithm. The MPCD algorithm directly maximizes the coding gain and uses a tree search to increase the chances of finding the optimum CO-GFFT. Therefore, a CO-GFFT designed with the MPCD algorithm achieves the lowest transform complexity among the discussed methods.

3.5.2 Complexity of GFFT design algorithms

The complexity of the algorithm that is used to design a fast transform has no effect on the complexity of encoding an image with the transform. All the design algorithms discussed in this chapter aim to find a sequence of Givens rotations, which are then applied to compute the transform coefficients during image coding. Among these methods, the SPCD algorithm and the design algorithm of the FN-GFFT design algorithm have a comparatively lower complexity, as they rely on closed-form solutions to determine the next best plane and the corresponding rotation angle. The proposed MPCD algorithm employs the M-algorithm, and its complexity increases with the value of M. Therefore the this algorithm has a higher complexity SPCD algorithm and the FN-GFFT design algorithm, and is directly proportional to M and N.

The complexity of the layered-GFFT design algorithm is relatively higher than all other methods, as it requires the Hungarian method [44] and the blossom algorithm [45] to determine the optimal permutation matrix and optimal directions, respectively. If the algorithm converges in A iterations, these two algorithms have to run A times.

Chapter 4

A Data-driven Method for CO-GFFT Design

Data-driven transforms, introduced in Sec. 1.5, are particularly important due to their ability to adapt to the content of each block in an image. Even though the optimal transform may differ for each block, the use of a transform codebook helps to summarize common patterns in image blocks using a finite number of fixed transforms to reduce the transform matrix signalling overhead, thereby making forward adaptive coding practical. This chapter presents a data-driven image block classification approach using the concept of structure tensors, followed by a fast data-driven transform codebook design method that incorporates the MPCD algorithm introduced in Chapter 3.

The main idea behind data-driven transform design is to capture common image properties through a form of classification, enabling the construction of a suitable set of candidate KLTs. One possible approach to constructing a data-driven transform codebook is to generate a large set of candidate transforms and reduce this set to a

suitably small number based on their coding efficiency. Although this method may produce optimal transforms for a given training set, it does not guarantee optimality for a new image. Another method is to estimate the covariance matrices of classified image block classes from a training set of image blocks and apply the MPCD algorithm to these covariance matrices. The objective is to classify transform blocks with similar structural patterns, and an image gradient-based method is employed for identifying similar structural patterns. For image blocks exhibiting similar structural patterns, it is expected that a single transform matrix will result in high energy compaction in the transform domain, and consequently, a high coding gain. The structure tensor is used to estimate the dominant gradient directions ϕ and the gradient anisotropy through a coherence measure C_h. An advantage of using the gradient pattern–based method is its interpretability; it allows for visual inspection of the diversity among the basis vectors in the final transform set, making it easier to identify missing or underrepresented patterns. In contrast, randomly generated classes typically do not offer this level of interpretability of basis vectors.

4.1 Data-driven Transform Design

4.1.1 Structure tensors classification

The theoretical background and mathematical definitions of the structure tensors [35, 36] are presented in the Sec. 2.5. According to that, the directional properties of texture within an image block are characterized by the features (C_h, ϕ). Unless $\lambda_1 = \lambda_2$, there exists a dominant direction such that $\lambda_1 \geq \lambda_2 \geq 0$. In a prior work [37], the structure tensor has been utilized for classification, but only a single strong dominant direction is used as the classification criterion. Fig. 4.1 illustrates the

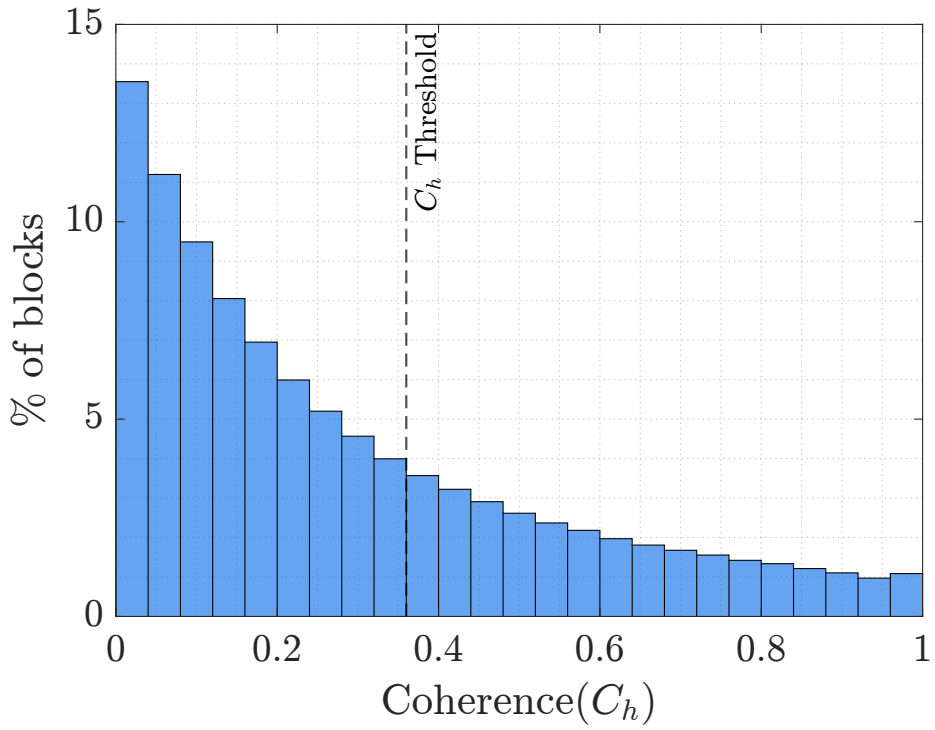
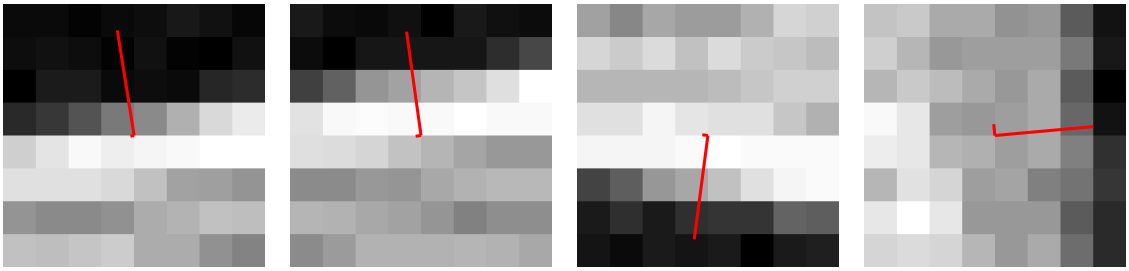


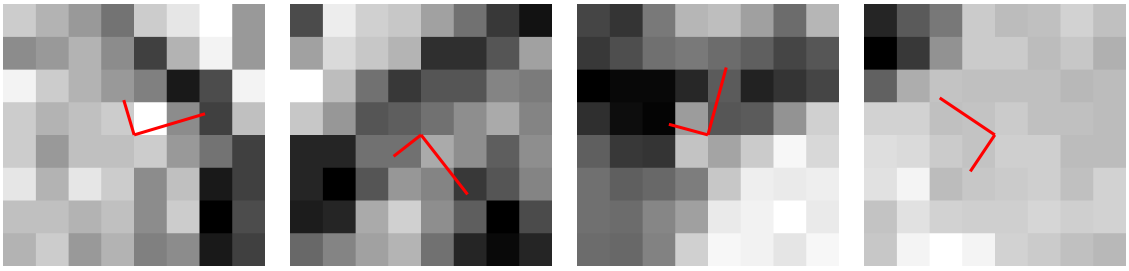
Figure 4.1: Coherence distribution of the training image block dataset.

distribution of coherence values among 8×8 blocks in a training set of images from *Kodak*[1],*Caltech* [46] databases, as well as commonly used images *Livingroom, Pirate, Mandrill, Walkbridge* and *Blonde*[2]. Since a substantial portion of the blocks exhibit lower coherence values, it would be unwise to disregard them in the design process. Therefore, the proposed classification approach considers all the image blocks, and the coherence value is included as a parameter in the classification process. To estimate a suitable set of candidate KLTs from a given training set of image blocks, the blocks are first classified into a finite number of classes, based on their (C_h, ϕ) features. Subsequently, a covariance matrix is estimated for each class to find a CO-GFFT using the MPCD algorithm. A common approach to classifying a set of vectors into a given number of classes is the K-means algorithm [47]. However, this does not

[1]https://r0k.us/graphics/kodak/
[2]https://www.kaggle.com/datasets/saeedehkamjoo/standard-test-images



(a) $C_h = 0.87$ (b) $C_h = 0.81$ (c) $C_h = 0.81$ (d) $C_h = 0.63$



(e) $C_h = 0.12$ (f) $C_h = 0.14$ (g) $C_h = 0.07$ (h) $C_h = 0.04$

Figure 4.2: Images blocks with various principal gradient directions and coherence values- (a)-(d) are from the high coherence class and (e)-(h) are from the low coherence class.

guarantee that each class will be homogeneous with respect to the KLT. As a useful heuristic, the following hierarchical two-stage classification procedure is employed for the training dataset.

1. The training set of 8×8 image blocks was first partitioned into two classes based on the coherence measure C_h. The blocks were classified as high coherence if the condition $\lambda_1 \geq 4\lambda_2$ (or equivalently $C_h \geq 0.36$) was satisfied. Blocks with $C_h < 0.36$ were assigned to the low-coherence class C_1, while those with $C_h \geq 0.36$ were assigned to the high-coherence class C_2.

2. The image blocks within the C_i coherence class were subsequently classified into K_i subclasses based on the texture orientation ϕ for $i = 1, 2$. To this end, a sample set of $\phi \in (0, \pi)$ values was generated from the image blocks in each coherence class, and this set was used to design a non-uniform scalar

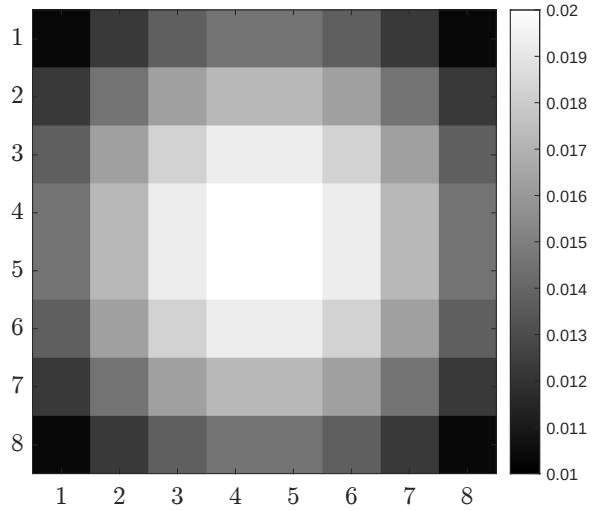
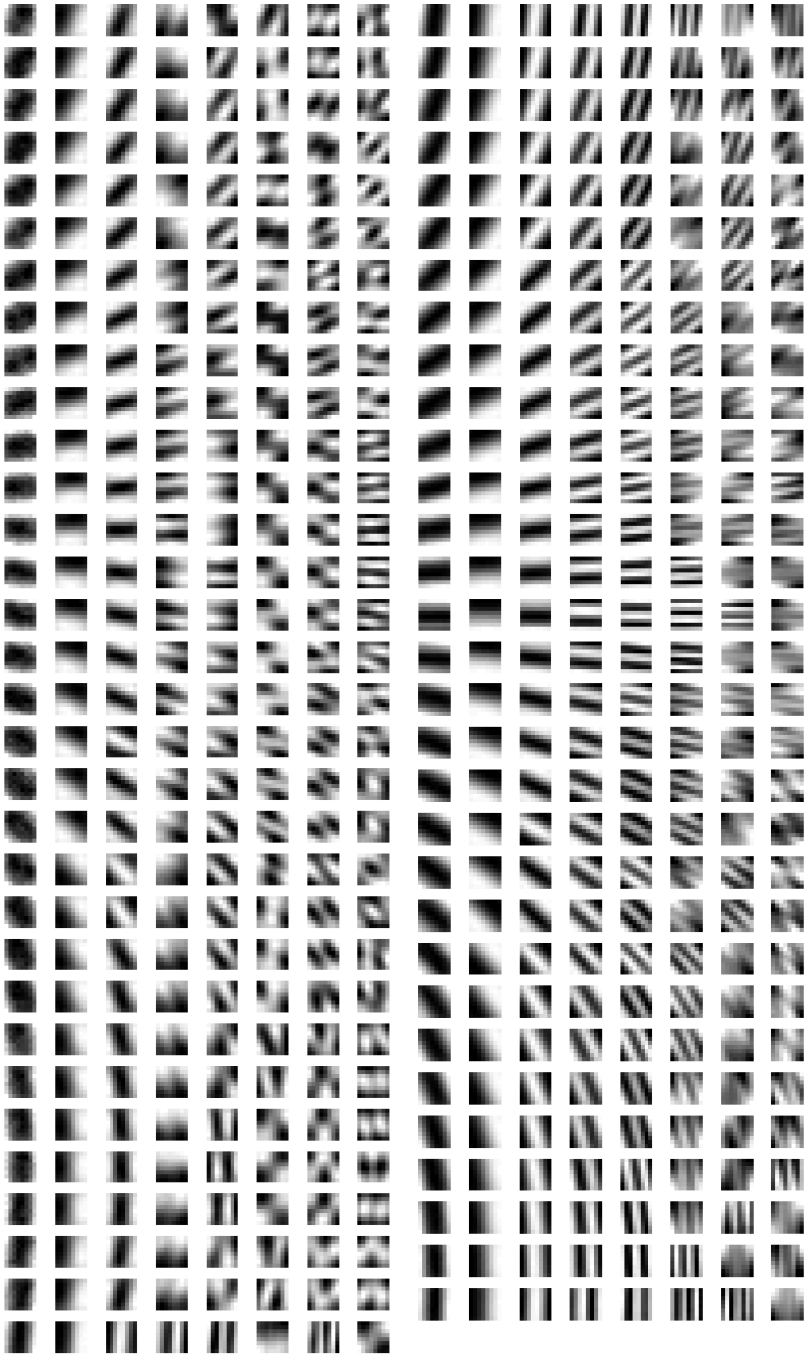


Figure 4.3: Gaussian kernel used for calculation of structure tensor matrix.

quantizer with K_i intervals, employing the Lloyd algorithm [1] (algorithm is in Appendix A) . The resulting quantizer was then applied to assign the image blocks to orientation-based subclasses. As a result, the training block set was partitioned into a total of $K = K_1 + K_2$ image block classes.

Fig. 4.2 shows sample image blocks along with their coherence values for the two classes, C_1 and C_2. When the coherence value is high, a dominant gradient direction is clearly visible. In contrast, when the coherence is low, gradient patterns are observed along both the $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ directions as described in Sec. 2.5.2. A Gaussian kernel with a standard deviation of 4.2, shown in Fig. 4.3, was used as the weighting function $\boldsymbol{w}$ in the structure tensor calculation using the (2.7). The training set of image blocks was then classified into 63 image block classes (reasons for which to be explained in Sec. 6.1) following the two-stage hierarchical procedure described above, with $K_1 = 32$ subclasses for C_1 and $K_2 = 31$ subclasses for C_2. The covariance matrix of each of the resulting K image block classes was estimated and, given by, $\boldsymbol{\mathcal{C}} = \{\boldsymbol{\mathcal{C}}_1, \boldsymbol{\mathcal{C}}_2 \ldots \boldsymbol{\mathcal{C}}_K\}$. Then the KLT of each $\boldsymbol{\mathcal{C}}_i$ is calculated and the first 8 basis vectors of each KLT are shown in Fig. 4.4. Note that the blocks with similar gradient directions can have very



(a) $C_h < 0.36$ (b) $C_h \geq 0.36$

Figure 4.4: First 8 dominant basis vectors of data-driven KLTs for 8×8 image block classes, constructed by classifying a training-set of blocks into 63 classes (each row is a transform).

different textures depending on the degree of anisotropy. For high-coherence classes, clearly defined directional patterns are exhibited in the first few basis vectors. For low-coherence classes, the basis vectors display more diverse and less distinct structures, with directional patterns appearing in a more diffuse and shadowed manner.

4.1.2 GFFT codebook design

Fast transform codebook design primarily involves two key stages.

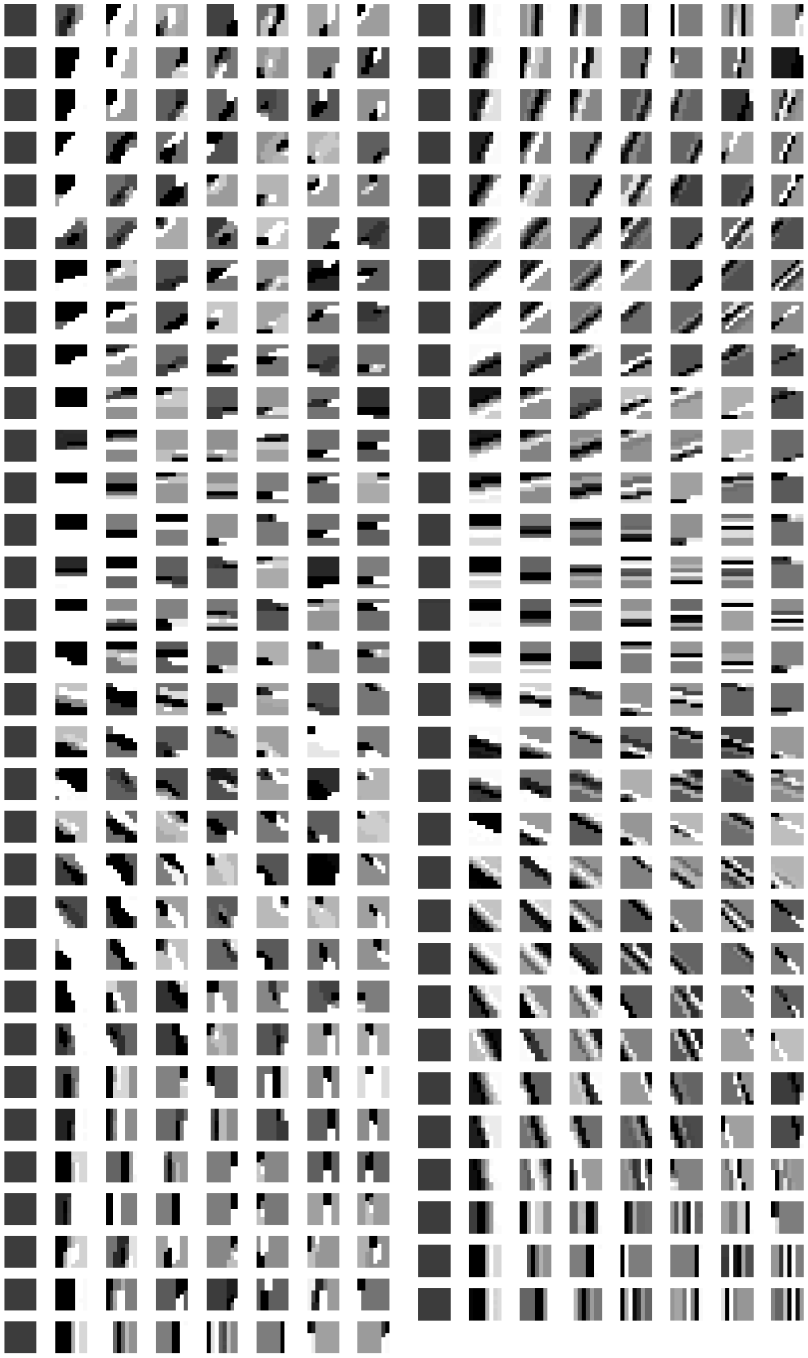
1. The first stage involves classifying the image blocks into K classes using structure tensor parameters, as described in Sec. 4.1.1. Based on this classification, a set of covariance matrices denoted by $\boldsymbol{\mathcal{C}}$ is estimated.

2. The second stage is to compute the CO-GFFT codebook by applying the MPCD algorithm described in algorithm (2).

The MPCD algorithm was executed using parameter settings $L = 86$ and $M = 500$. The value of L was selected so that the computational complexity of the transform coding is equal to γ_{DCT}. The MPCD algorithm produces an L-stage Givens rotations sequence for each class-specific covariance matrix, denoted by $\boldsymbol{q}_i$, where $i = 1, \ldots, K$. These sequences define a set of data-driven fast transforms that form a CO-GFFT codebook,

$$\mathcal{Q}_{\text{CO-GFFT}} = \{\boldsymbol{q}_1,\ \boldsymbol{q}_2 \ldots \boldsymbol{q}_K\}.$$

Each sequence $\boldsymbol{q}_i$ corresponds to a fast transform as in (3.1), and Fig. 4.5 illustrates the first 8 basis vectors of the 63 CO-GFFTs produced by the MPCD for the 63 image block classes in Fig. 4.4. It is worth noting that for a block size of 8×8, the exact representation of a full 64×64 KLT would require 2016 Givens rotation stages.



(a) $C_h < 0.36$ (b) $C_h \geq 0.36$

Figure 4.5: First 8 dominant basis vectors of data-driven CO-GFFTs for 63 classes (each row is a transform).

Even with the approximation using only 86 stages, the resulting basis vectors shown in Fig. 4.5 still preserve texture characteristics similar to those of the eigenvectors shown in Fig. 4.4.

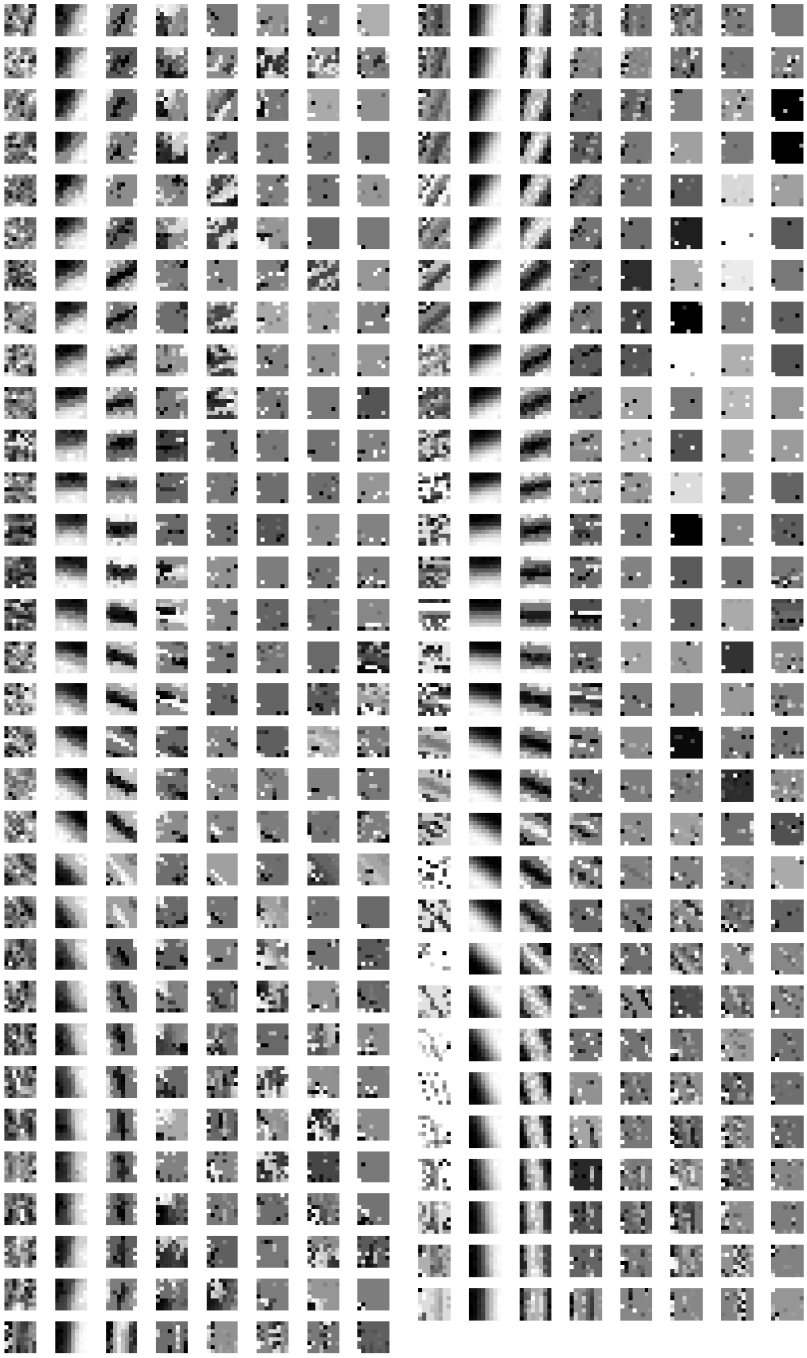
Given that the FN-GFFT design method is the best for designing GFFTs based on the matrix-distance minimization at the 2D-DCT complexity, a CO-GFFT codebook was experimentally compared with an FN-GFFT codebook (results presented in Chapter 6). The step in the FN-GFFT codebook design was to construct a codebook of the KLT matrices. This codebook is denoted by,

$$\mathcal{Q}_{\text{KLT}} = \{\boldsymbol{T}_1,\ \boldsymbol{T}_2 \ldots \boldsymbol{T}_K\},$$

where $\boldsymbol{T}_i$ is the KLT corresponding to $\boldsymbol{\mathcal{C}}_i$. A codebook of the FN-GFFTs was then constructed by approximating the KLT transforms in $\mathcal{Q}_{\text{KLT}}$. This codebook is denoted by,

$$\mathcal{Q}_{\text{FN-GFFT}} = \{\boldsymbol{r}_1,\ \boldsymbol{r}_2 \ldots \boldsymbol{r}_K\},$$

where $\boldsymbol{r}_i$ is the L-stage Givens rotation sequence of the approximation for the $\boldsymbol{T}_i$ with $L = 86$ using the algorithm in [17]. The first 8 basis vectors of the 63 FN-GFFTs in $\mathcal{Q}_{\text{FN-GFFT}}$ are shown in Fig. 4.6, illustrating that the approximation is significantly less accurate compared to the corresponding CO-GFFT basis vectors.



(a) $C_h < 0.36$ (b) $C_h \geq 0.36$

Figure 4.6: First 8 dominant basis vectors of data-driven FN-GFFTs for 63 classes (each row is a transform).

4.2 Coefficient Ordering for Efficient Entropy Coding

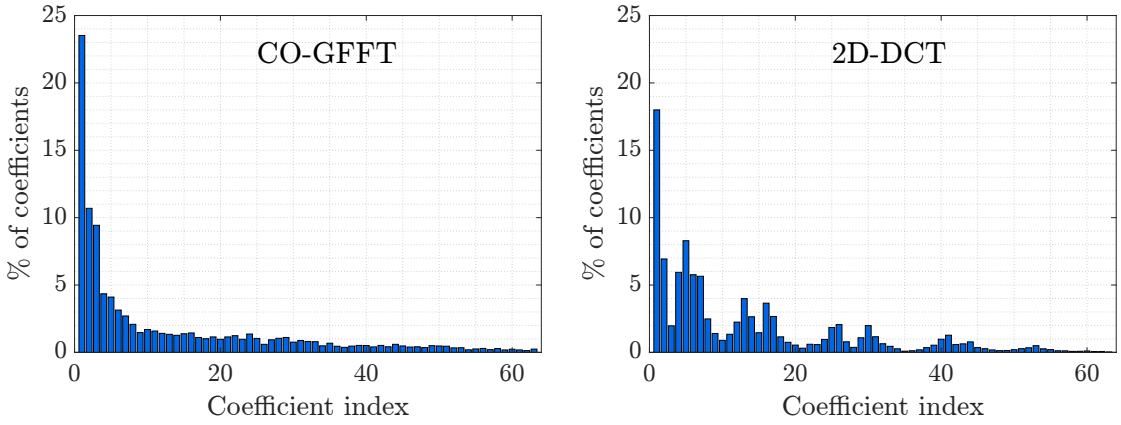
The rate-distortion advantage of transform coding arises from the property that, with well-designed transforms, only a small number of transform coefficients in an image block carry significant values due to the energy compaction property [26, 27]. As a result, most coefficients become zero after quantization, enabling efficient compression. To exploit the energy compaction property in entropy coding, the quantized transform coefficients must be ordered such that long runs of consecutive zeros appear in the quantized sequence. To achieve this goal permuting the rows of the KLT are usually permuted such that the diagonal elements of the transformed covariance matrix are in descending order. In the JPEG algorithm a zig-zag scan of 2D-DCT coefficients is applied to achieve the same goal [33]. Given that a CO-GFFT serves as an approximation to the KLT, the resulting transform coefficients must also be ordered according to their variances to facilitate effective energy compaction and entropy coding. However, since a CO-GFFT is expressed as a product of an ordered sequence of Givens rotations, the rows of the overall transform matrix are defined only implicitly and cannot be permuted a priori. To address this, a separate permutation matrix is computed for each CO-GFFT in the codebook to reorder the transform coefficients appropriately, as described below.

- Let the set of covariance matrices corresponding to the K classes be denoted by $\boldsymbol{C}_K$, where $\boldsymbol{C}_{(k)}$ is the covariance matrix of the kth class. Givens rotation sequence of the kth class is denoted by $\boldsymbol{q}_{(k)}$, and the resulting CO-GFFT transform matrix is represented by $\boldsymbol{T}_k$. The covariance matrix of the transform coefficient vector, denoted $\tilde{\boldsymbol{C}}_{(k)}$, is then given by, $\tilde{\boldsymbol{C}}_{(k)} = \boldsymbol{T}_k \boldsymbol{C}_{(k)} \boldsymbol{T}_k^T$.

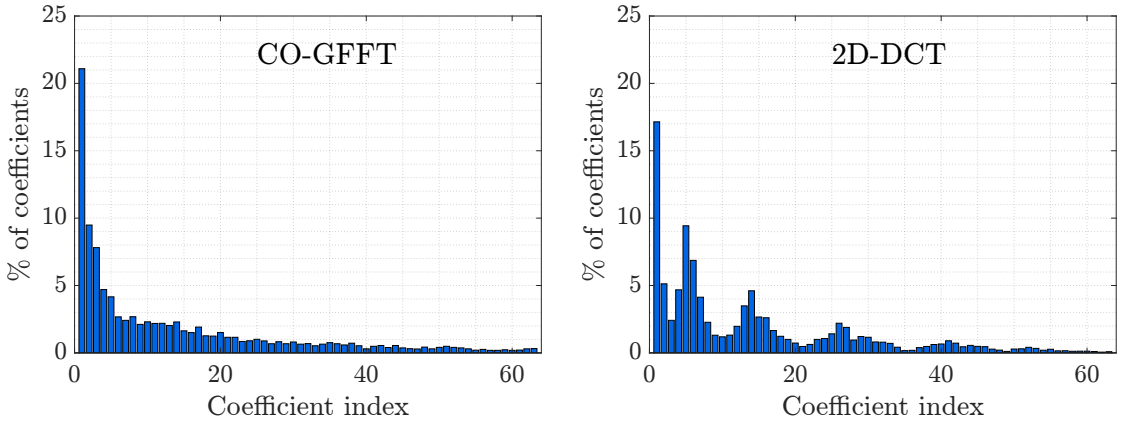
- The permutation matrix used to order the transform coefficients is a symmetric permutation matrix $\boldsymbol{P}_{(k)}$, chosen such that the diagonal elements $\boldsymbol{P}_{(k)}\tilde{\boldsymbol{C}}_{(k)}\boldsymbol{P}_{(k)}^{T}$ are arranged in descending order.

The set of permutation matrices $\mathcal{P}_{\text{CO-GFFT}} = \{\boldsymbol{P}_{(1)}, \ldots \boldsymbol{P}_{(K)}\}$ is stored with the CO-GFFT codebook. Thus, the properly ordered transform coefficient vector for the CO-GFFT for a given image-block vector $\boldsymbol{X}$ is given by $\boldsymbol{Y} = \boldsymbol{P}_{(K)}\boldsymbol{T}_k\boldsymbol{X}$,where $\boldsymbol{T}_k$ has the factorized form (3.1).

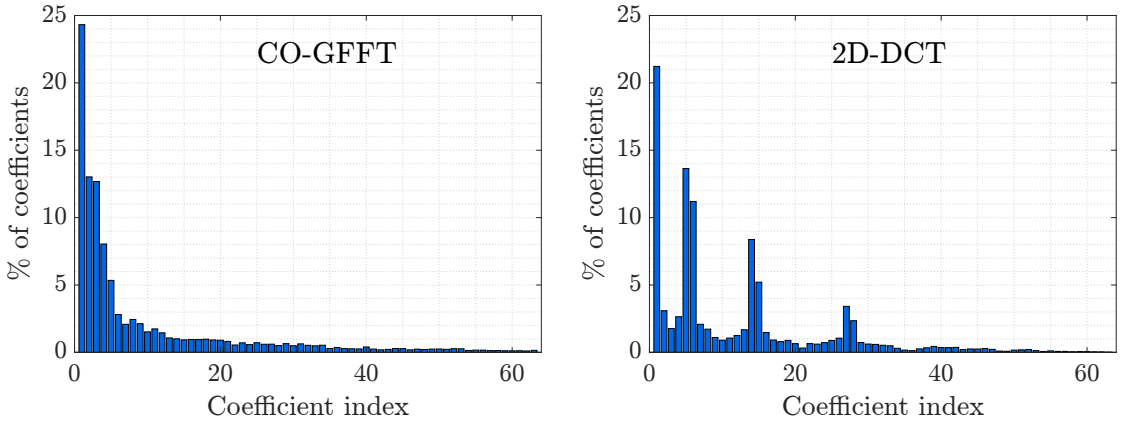
Despite the histograms of the transform coefficients produced by the CO-GFFTs in the codebook and the 2D-DCT for the three classes shown in Fig. 4.7. In these figures, CO-GFFT coefficient indexing is based on the aforementioned permutation matrix for each class, and that for 2D-DCT is based on the zig-zag scan used in the JPEG algorithm. When properly ordered, the decreasing indices correspond to KLT basis vectors with decreasing spatial frequencies (see Fig. 4.5). Although the GFFTs are low-complexity approximations to the KLTs, they still produce coefficient distributions where larger coefficients are more concentrated in the low-frequency basis vectors compared to the 2D-DCT.



(a) Class-35



(b) Class-60



(c) Class-63

Figure 4.7: Transform coefficient histograms for three different classes.

Chapter 5

An Implementation of an Image Codec

An image codec has been developed that employs data-driven fast transforms described in Chapter 4 [1]. While the architecture of the codec closely follows the baseline JPEG encoder [33], it has several notable differences to accommodate the data-driven fast transforms codebook.

5.1 JPEG Compression

The basic steps of the JPEG compression method are introduced before discussing the details of the proposed architecture. Fig. 5.1 illustrates the core components of the JPEG compression algorithm. The image is first divided into non-overlapping 8×8 blocks, which are then forward-transformed using the 2D-DCT. The transform coefficients of each block are subsequently quantized using a standard quantization

[1] Matlab code here: `https://github.com/dilshanmora/CO-GFFT-codec`

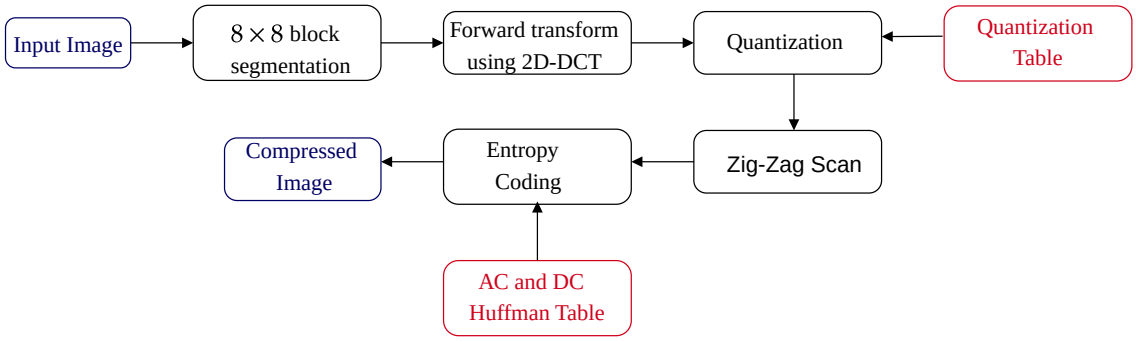


Figure 5.1: JPEG image compression framework.

table [33, Table K.1]. The DC coefficients (zero-frequency components) are separated from the AC coefficients (non-zero frequency components), and the AC coefficients are reordered using a zig-zag scan [33] to order low-frequency components at lower indices. Then, entropy coding is applied separately to the DC and the AC coefficients. While the DC coefficients are encoded using differential Huffman coding using a separate Huffman table [33, Table K.3], the AC coefficients are encoded using RLE and a Huffman table [33, Table K.5]. The header information is added, including the quantization table, image width, image height, start of image data marker, and other relevant metadata to form the final bit stream [33].

5.2 Proposed Data-driven Fast Transform Encoder

Fig. 5.2 illustrates the main steps of the proposed encoder. An image is first segmented into 8×8 non-overlapping blocks, similar to the JPEG standard. The segmented image blocks are individually vectorized. Let a vectorized image block be represented as $\boldsymbol{X}_l = \{X_1, \ldots, X_{64}\}$. The DC values of all L image blocks in the image are collected into a vector, $\boldsymbol{X}_{DC} = \{\bar{X}_1, \ldots \bar{X}_L\}$, where $\bar{X}_l$ is the DC value of lth image block that

calculates as,

$$\bar{X}_l = \frac{1}{64} \sum_{i=1}^{64} X_i.$$

The elements of the vector $\boldsymbol{X}_{DC}$ are quantized using a uniform scalar quantizer and the quantized values are individually entropy coded, using differential Huffman coding as described in Sec. 2.4.2 with the Huffman coding table K.3 defined in the JPEG standards [33] (table is included in Appendix B). The encoded binary vector is referred to as $\boldsymbol{H}_{DC}$. For transform coding, DC values are removed from the image block vectors to produce the vector,

$$\tilde{\boldsymbol{X}}_l = \boldsymbol{X}_l - \bar{X}_l \boldsymbol{b},$$

where $\tilde{\boldsymbol{X}}_l$ is the mean removed version corresponding to the lth image block and, $\boldsymbol{b}$ is 64×1 vector of all ones. The zero-mean vectors are sequentially processed in the rate-distortion optimization step shown in Fig. 5.2, where the optimal transform is selected from the transform matrix codebook $\mathcal{Q}_{\text{CO-GFFT}}$ and entropy coding is performed using RLE. An optimal transform selection through rate-distortion optimization is further illustrated in the Fig. 5.3, and details are discussed in the Sec. 5.2.1. After determining the optimal transform for each vector (image block), the corresponding transform indices of these matrices in the codebook $\mathcal{Q}_{\text{CO-GFFT}}$ are collected into a single vector and losslessly encoded using Huffman coding (details in Sec. 5.2.3). The details of the coded bitstream $\boldsymbol{I}_{\text{encode}}$ are provided in Sec. 5.2.4.

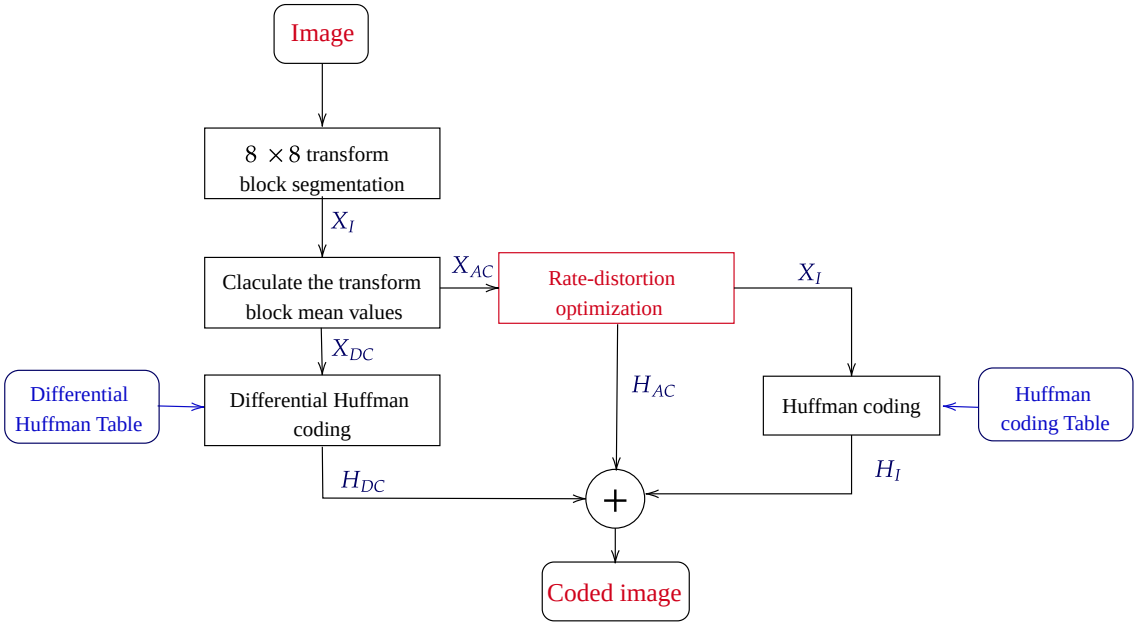


Figure 5.2: Proposed encoder framework for image compression application.

5.2.1 Optimal Transform Selection

The optimal transform selection and entropy coding of the AC coefficients of the mean-removed vector $\tilde{\boldsymbol{X}}_l$ are performed simultaneously, as illustrated in Fig. 5.3. AC transform coefficient vector of the $\tilde{\boldsymbol{X}}_l$ is calculated as,

$$\tilde{\boldsymbol{Y}}_l = \boldsymbol{P}_k \boldsymbol{T}_k \tilde{\boldsymbol{X}}_l, \tag{5.1}$$

where $\boldsymbol{T}_k$ is k^{th} CO-GFFT (in the $\mathcal{Q}_{\text{CO-GFFT}}$ codebook) and $\boldsymbol{P}_k$ is k^{th} permutation matrix corresponding to $\boldsymbol{T}_k$. Coefficients in $\tilde{\boldsymbol{Y}}_l$ are scalar quantized using a uniform quantizer with step size Δ and entropy coded as described in Sec. 5.2.2. The optimal

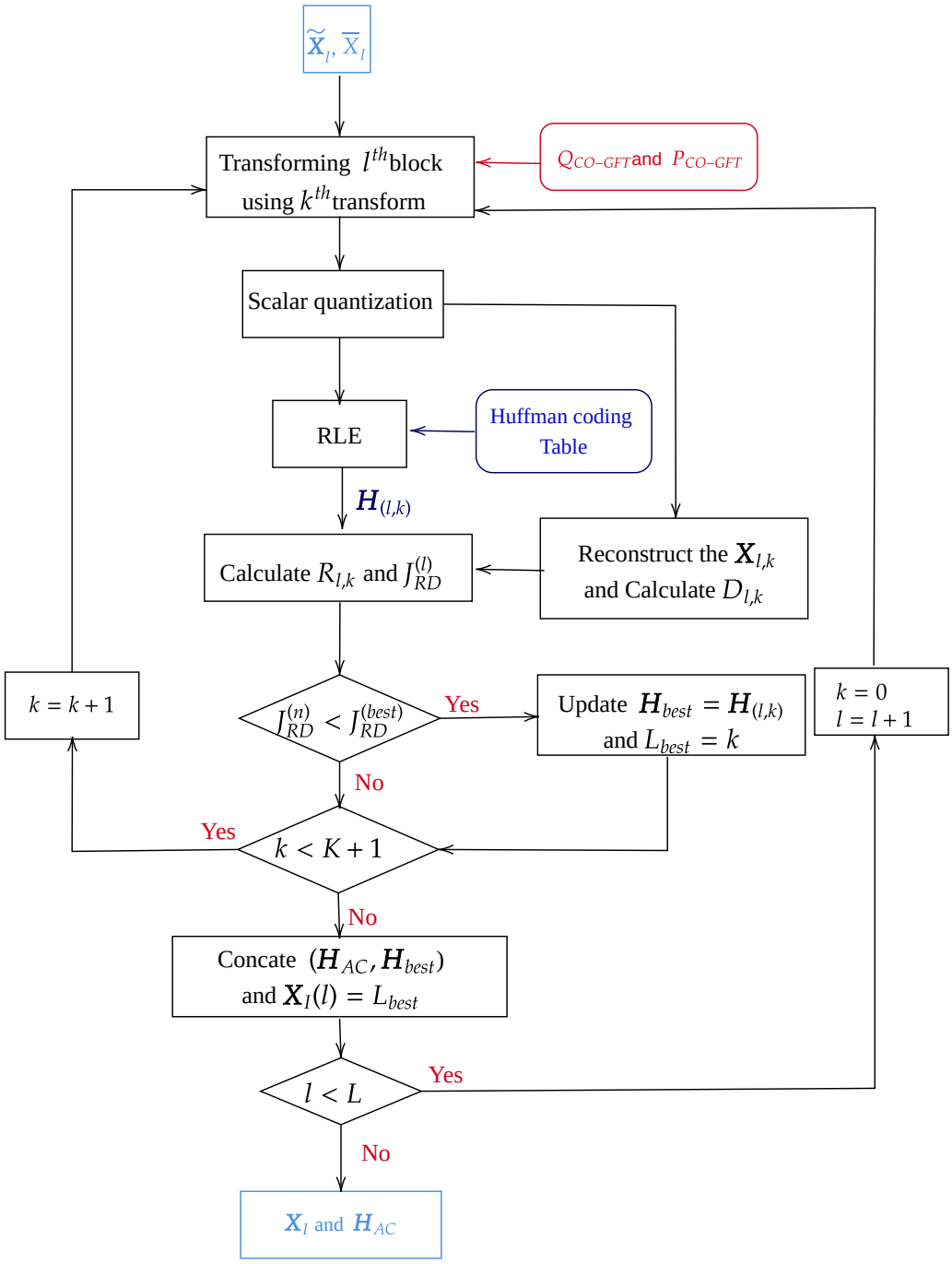


Figure 5.3: Rate-distortion optimization for optimal transform selection and entropy coding in the encoder.

transform $\boldsymbol{T}_k^*$ for a given image block is found by,

$$\boldsymbol{T}_k^* = \arg \min_{\boldsymbol{T}_k \in \mathcal{Q}_{\text{CO-GFFT}}} D(\boldsymbol{T}_k), \tag{5.2}$$

$$\text{subject to} \quad R(\boldsymbol{T}_k) \leq R_T, \tag{5.3}$$

where $D(\boldsymbol{T}_k)$ is the MSE given by $D(\boldsymbol{T}_k) = E[||\boldsymbol{X}_l - \hat{\boldsymbol{X}}_l||^2]$, $\hat{\boldsymbol{X}}_l$ is the reconstructed image block in vector form which is a function of $\boldsymbol{T}_k$, and $R(\boldsymbol{T}_k)$ is bit rate measure by the entropy of the quantized transform coefficients. The parameter optimization procedure described in [48] is adapted in this work for selecting the optimal transform. The Lagrangian cost function J_{RD} for rate-distortion optimization problem in (5.2) is defined by,

$$J_{RD}(D, R) = D + \lambda R,$$

where $\lambda(> 0)$ is the Lagrangian multiplier. Then the optimal λ can be found by,

$$\begin{aligned} \frac{\partial J_{RD}}{\partial R} &= 0 \\ \frac{\partial D}{\partial R} + \lambda^* &= 0 \\ \lambda^* &= -\frac{\partial D}{\partial R} \end{aligned}$$

By assuming a high-rate model for uniform quantization, the distortion D is given by,

$$D = \frac{\Delta^2}{12}, \tag{5.4}$$

where Δ is the quantization step-size. Furthermore, for a Gaussian distributed transform coefficients distortion D can be expressed as [3],

$$D = c2^{-2R}, \tag{5.5}$$

where $c = \sigma^2$, σ^2 is the variance of the Gaussian distribution. taking the logarithm, (5.5) can be written as,

$$\begin{aligned} \ln D &= \ln c - 2R \ln (2) \\ \frac{1}{D}\frac{\partial D}{\partial R} &= -2 \ln (2) \\ \frac{\partial D}{\partial R} &= -2 \ln (2) D, \end{aligned}$$

and using (5.4),

$$\frac{\partial D}{\partial R} = -2 \ln (2) \frac{\Delta^2}{12}.$$

Therefore, the optimal λ is given by,

$$\lambda^* = \frac{\ln(2)\Delta^2}{6},$$

and the optimal Lagrangian cost function is given by,

$$J^*_{RD} = D + \frac{\ln(2)\Delta^2 R}{6}. \tag{5.6}$$

Let the bit rate and distortion associated with the kth transform for l^{th} image blocks be $R_{l,k}$ and $D_{l,k}$. Then, the optimal Lagrangian cost function can be given as $J^*_{RD}(D_{l,k}, R_{l,k})$. In actual implementation, the MSE distortion for an 8×8 image

block can be approximated by,

$$D_{l,k} \approx \frac{1}{64} \sum_{i=1}^{64} [X_i - \hat{X}_i]^2.$$

Although the entropy of the quantized transform coefficients is used to derive the theoretical solution in (5.6), 64 values of quantized transform coefficients in an 8×8 block are not sufficient to estimate their probabilities accurately. Therefore, $R_{l,k}$ is replaced by the actual number of bits produced by the lossless encoding used in the codec. The design of the lossless encoding used in this codec is described in Sec. 5.2.2.

In addition to K CO-GFFTs, the 2D-DCT is also included as an option in the codebook since it has been experimentally observed that, for some image blocks, the 2D-DCT provides better rate-distortion performance than any of the CO-GFFTs (see Sec. 6.3). Therefore, $K + 1$ transforms are available to select the optimal transform for each image block. The optimal solution $T^*_{k_l}$ for (5.2) can be found for the lth image block using (5.6), where k_l is the index of the optimal transfrom for the lth image block. Transform indices of all blocks in the image being coded, k_l, $l = 1, \ldots L$ are collected into the vector $\boldsymbol{X}_I$, which is the output of the rate-distortion optimization step in Fig. 5.3.

5.2.2 Lossless coding of AC coefficients

The quantized AC transform coefficient vector $\tilde{\boldsymbol{Y}}_l$ of the image block l is entropy coded using RLE and a Huffman table as in the JPEG codec. However, in this case, the Huffman table has been specifically optimized for CO-GFTTs in the codebook. To enable this, an initial Huffman table was designed with a two-step procedure, using a training set of image blocks as follows. In the first step, the Huffman table used in the JPEG coded [33, Table K.3] was used to encode the training image blocks for optimal

transform selection, and the probabilities of quantized transform coefficients were estimated according to the selected transforms. In the next step, these probabilities are used to design a new Huffman table. This table is shown in Appendix C.

During encoding an image, the bit rate R for a given block is computed by dividing the entropy-coded bit stream by the number of pixels in an image block (64 in an 8×8 image block), which is used in rate-distortion optimal transform selection. The optimal transform selection is applied to all blocks in an image, and the vectors $\boldsymbol{H}_{k_l} = \boldsymbol{H}_{l,k}$ of coded bits are collected in a vector $\boldsymbol{H}_{AC}$ which is the output of the transform coding step.

5.2.3 Transform index signaling

For an encoded image, the optimal transform matrix of each transform block needs to be signaled to the decoder by specifying its index in the codebook. The index vector $\boldsymbol{X}_I$ is encoded using Huffman coding utilizing the Huffman table shown in Appendix D. This Huffman codeword table has been designed based on the probability distribution of transform indices observed in the training data. As another method, differential Huffman encoding (described in Sec. 2.4.2) was also considered as a way of exploiting the possible correlation in the index sequence. Experimental results for both these methods are shown in Sec. 6.2. These results show that differential Huffman coding does not provide a significant advantage. This is due to a lack of correlation in transform index sequences, which can be explained by observing the basis vectors in the transform codebook in Fig. 4.5. Note that the codebook includes transforms for very slight changes in the gradient direction.



Figure 5.4: Format of the the compressed image bit stream $\boldsymbol{I}_{\text{encode}}$.

5.2.4 Format of the encoded bit stream

The coded bit sequences $\boldsymbol{H}_{DC}$, $\boldsymbol{H}_{AC}$, and $\boldsymbol{H}_{I}$ must be accompanied by a header that includes the image width, height, and quantization step sizes. Subsequently, the header and the coded data are concatenated with a *marker code* that identifies the beginning of each bit sequence. The format of the complete encoder output, referred to as $\boldsymbol{I}_{\text{encode}}$, is illustrated in Fig. 5.4. Since the coded image needs to be stored as a binary file, the bit stream is converted to bytes using *byte packing*, which is the process of combining multiple bits into a sequence of bytes. This step is essential to ensure that the compressed data can be efficiently stored.

5.3 Decoder

The decoder uses the information in the encoded bit stream to reconstruct the image. Image metadata, such as image size and quantization step sizes, is encoded in the header of the coded image. During the decoding process, the $\boldsymbol{I}_{\text{encode}}$ is first parsed to separate the header, $\boldsymbol{H}_{DC}$, $\boldsymbol{H}_{AC}$, and $\boldsymbol{H}_{I}$ using the marker code. The header information allows the decoder to identify the number of 8×8 blocks, the image size, and the quantization step sizes at the start of the decoding process. Then $\boldsymbol{H}_{DC}$, $\boldsymbol{H}_{AC}$, and $\boldsymbol{H}_{I}$ are decoded using the corresponding Huffman tables, and inverse quantization is applied to the decoded DC and AC coefficient vectors. The decoded vectors DC coefficients, AC coefficients, and the optimal transform indices vectors are referred to

as $\boldsymbol{Y}_{DC}$, $\boldsymbol{Y}_{AC}$, and $\boldsymbol{Y}_{I}$. As the EOB codeword serves as a marker within each $\boldsymbol{H}_{k_l}$ across all L image blocks, the AC coefficients corresponding to each block can be accurately extracted from the $\boldsymbol{Y}_{AC}$.

Let the decoded AC coefficients for the lth image block be $\boldsymbol{Y}_{AC,l}$. These coefficients are then inverse transformed, and the image block is reconstructed as,

$$\hat{\boldsymbol{X}}_l = \boldsymbol{T}_k^T \boldsymbol{P}_k^T \boldsymbol{Y}_{AC,l} + Y_{DC,l}\boldsymbol{b}, \tag{5.7}$$

where $\boldsymbol{P}_k$ is the kth permutation matrix, $\boldsymbol{T}_k$ is the kth transform matrix in $\mathcal{Q}_{\text{CO-GFFT}}$, $Y_{DC,l}$ is the mean value of the lth vector, and $\boldsymbol{b}$ is a 64×1 vector of all ones. Using $\hat{\boldsymbol{X}}_l$ for $l = 1, \ldots, L$, the image blocks are reconstructed. The decoder identifies the value of k corresponding to the lth vector using $\boldsymbol{Y}_I$, such that $\boldsymbol{Y}_{I,l} = k$. Once all image block vectors $\hat{\boldsymbol{X}}_l$, $l = 1, \ldots L$, have been regenerated, the image is assembled based on the image size information.

5.4 Computational Complexity of the Codec

The primary difference between the encoder and decoder complexities in the proposed codec comes from the rate-distortion optimization step, which is carried out only in the encoder. Each transform in the matrix codebook is implemented as an 86-stage GFFT, with computational complexity comparable to that of the 2D-DCT. However, unlike the decoder, the encoder must perform the transformation and RLE step $K+1$ times per block, once for each of the K candidate transforms and 2D-DCT in the codebook, in order to select the optimal transform. As a result, the encoder is approximately $K+1$ times more complex than the decoder, which only needs to apply a single inverse transform per transform coefficient vector of an image block using the

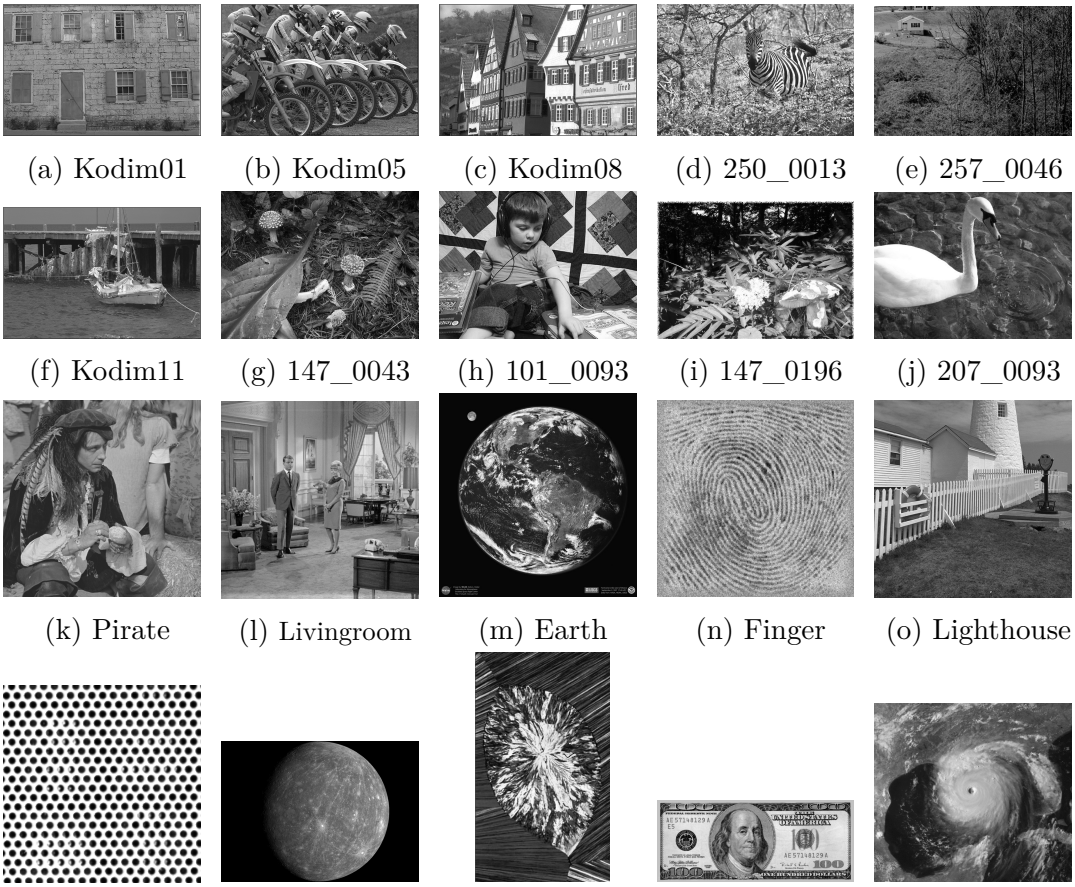
known transform index. In contrast, JPEG uses a fixed 2D-DCT for all blocks, and therefore its complexity is approximately equivalent to that of the proposed decoder. The proposed codec includes an optimal transform index encoding and decoding step, where the selected optimal transform index per block is encoded and decoded using Huffman coding. This step introduces a complexity overhead for both the encoder and decoder compared to JPEG.

Chapter 6

Experimental Results and Discussion

In this section, experimental results produced with the image codec described in the Chapter 5 are presented and discussed. Comparisons are made with the standard baseline JPEG codec. To ensure fair training and evaluation, separate training and testing image sets consisting of images from the *Caltech* [46], *Kodak*[1], and *ImageNet* [2] datasets, as well as widely used test images such as Livingroom, Pirate, Mandrill, Walkbridge, and Blonde[3]. A sample set of images from both the training and testing datasets is shown in Fig. 6.1. The experimental rate-distortion (RD) performance of the proposed image codec is evaluated using commonly used performance metrics *PSNR* and *SSIM* (Sec. 2.6). The bit rate (R) is measured in bits per pixel (bpp).

[1] `https://r0k.us/graphics/kodak/`
[2] `https://www.image-net.org/challenges/LSVRC/2017/index.php`
[3] `https://www.kaggle.com/datasets/saeedehkamjoo/standard-test-images`



(a) Kodim01 (b) Kodim05 (c) Kodim08 (d) 250_0013 (e) 257_0046

(f) Kodim11 (g) 147_0043 (h) 101_0093 (i) 147_0196 (j) 207_0093

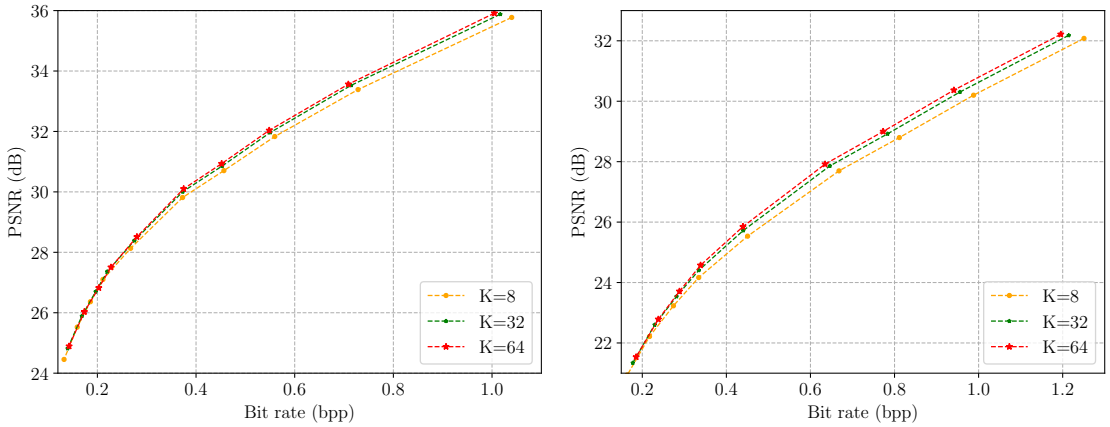
(k) Pirate (l) Livingroom (m) Earth (n) Finger (o) Lighthouse

(p) TextureB (q) Mercury (r) Cholesterol (s) Dollar-100 (t) Katrina

Figure 6.1: Sample images from training and testing dataset.

6.1 Choice of Transform Matrix Codebook Size

In Chapter 4, a method for designing a GFFT codebook was presented where sample image blocks were classified into a predetermined number of classes, and the number of classes K determined the size of the final codebook. The design of the transform matrix codebook ($\mathcal{Q}_{\text{CO-GFFT}}$), as presented in Chapter 4, includes $K = 63$ transforms, whose number was selected based on experimental analysis. While increasing the number of image block classes allows better adaptation of the transform matrices to diverse textures in image blocks, the improvement shows diminishing returns as



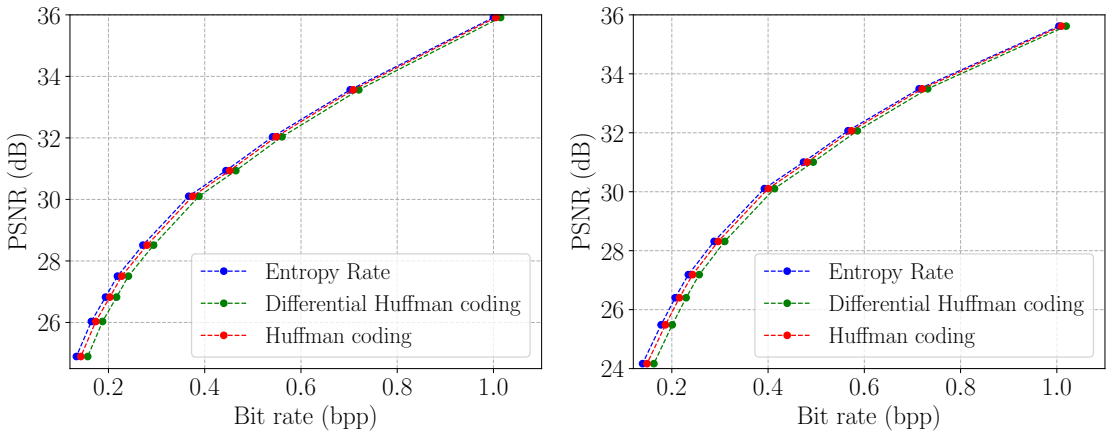
(a) Image-pirate (b) Image-Kodim05

Figure 6.2: RD performance of 86-stage CO-GFFT codebooks of various sizes.

K increases. This is because the additional bit rate required to signal the optimal transform index for each block also increases, which may negatively impact the overall RD performance. This trade-off is illustrated in Fig. 6.2, which shows RD performance comparisons for the encoder using CO-GFFT codebooks of sizes 8, 32, and 64 across two test images. After testing on a wide range of images, it was observed that using more than 63 KLT classes did not provide any significant performance gain. Therefore, all subsequent experimental results are reported using the $\mathcal{Q}_{\text{CO-GFFT}}$ of size 64 (Sec. 4.1.2). Subsequently, comparisons will be made with codebooks based on the class-KLTs and FN-GFFTs, constructed using the same set of 63 image block classes in the following sections. These two codebooks will be denoted as $\mathcal{Q}_{\text{KLT}}$ and $\mathcal{Q}_{\text{FN-GFFT}}$, respectively.

6.2 Transform Matrix Index Coding

The encoder described in Chapter 5 employs Huffman coding for signalling the transform matrix index vector. Additionally, differential Huffman coding was explored as



(a) Image-pirate (b) Image-livingroom

Figure 6.3: RD performance of 86-stage CO-GFFT codebooks when different entropy coding methods are used for encoding transform matrix indices.

an alternative method. Differential Huffman coding is applied to the index vector $\boldsymbol{X}_I$ in a manner similar to that used for the DC coefficient vector, utilizing the Huffman table specified in Appendix B. Fig. 6.3 demonstrates the effectiveness of the Huffman coding of the transform matrix index vector $\boldsymbol{X}_I$ with the Huffman codeword table in Appendix D, which achieves performance close to the theoretical entropy bound which is despite the curve entropy rate. In contrast, differential Huffman encoding performs poorly due to the weak correlation between the transform matrix indices of neighboring blocks.

6.3 Performance Comparisons

As described in Chapter 5, the image codec is based on a codebook of the 2D-DCT and 63 CO-GFFTs, which are designed as described in Chapter 3. The 2D-DCT was also included as a candidate transform in the codebook, as it was observed to be highly effective for certain image blocks. During the experiments with the test images,

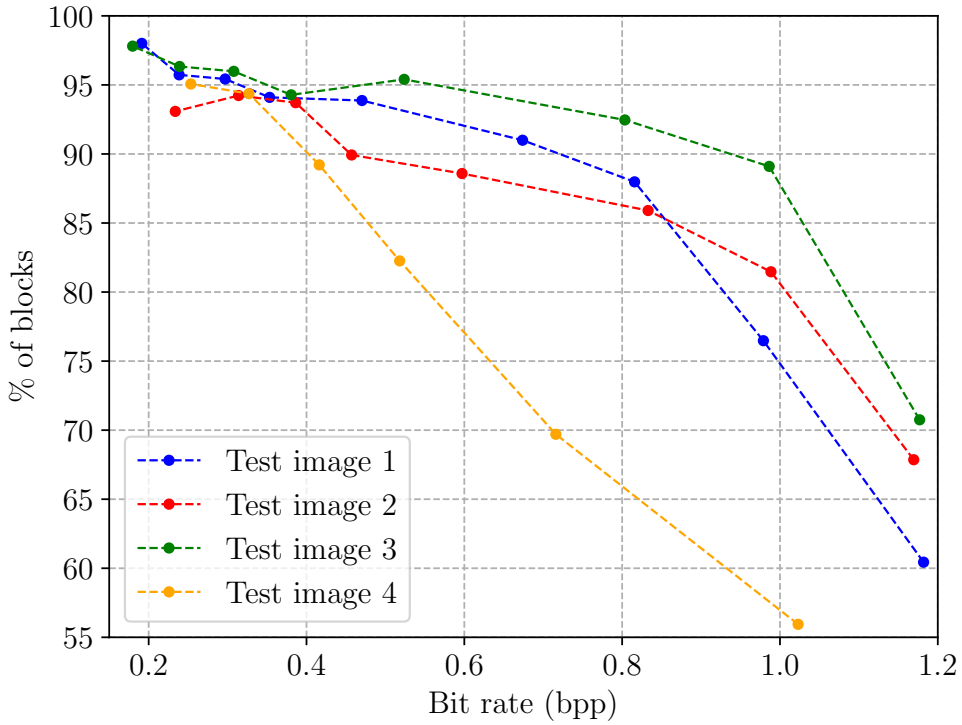
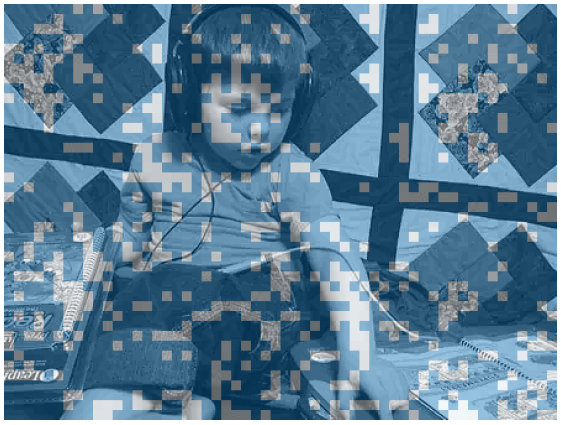
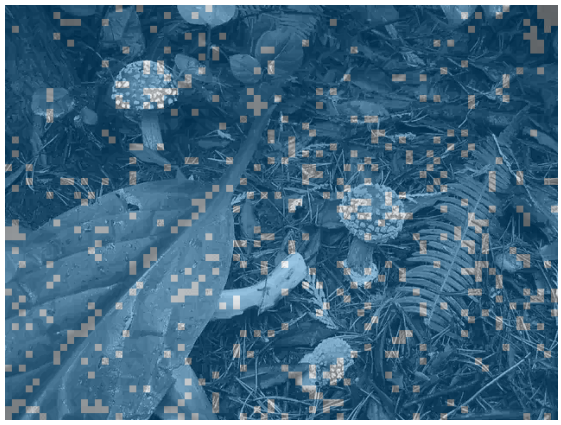


Figure 6.4: The percentage of blocks in four coded images for which CO-GFFTs produced better RD performance than the 2D-DCT. *Test images* 1- 4 are 101_0093, 147_0196, 147_0043, and 250_0013 respectively from the Caltech image data set.

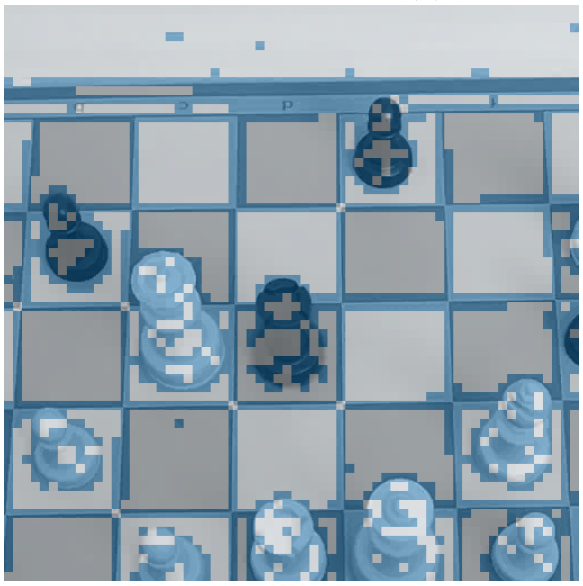
it was observed that, although CO-GFFTs were selected more frequently overall, the 2D-DCT was chosen most often among all individual transforms. Fig. 6.4 shows the fraction of blocks in four test images for which a higher PSNR was achieved by a CO-GFFT compared to the 2D-DCT. Fig. 6.5 shows the blocks in three images for which the RD-optimal selection procedure preferred a CO-GFFT over the 2D-DCT. While CO-GFFTs are selected for most blocks in the texture-rich images 101_0093 and 147_0043, they are primarily used on the chess pieces and the edges of the squares in the Chessboard image.



(a) Image 101_0093 (1 bpp)



(b) Image 147_0043 (1 bpp)

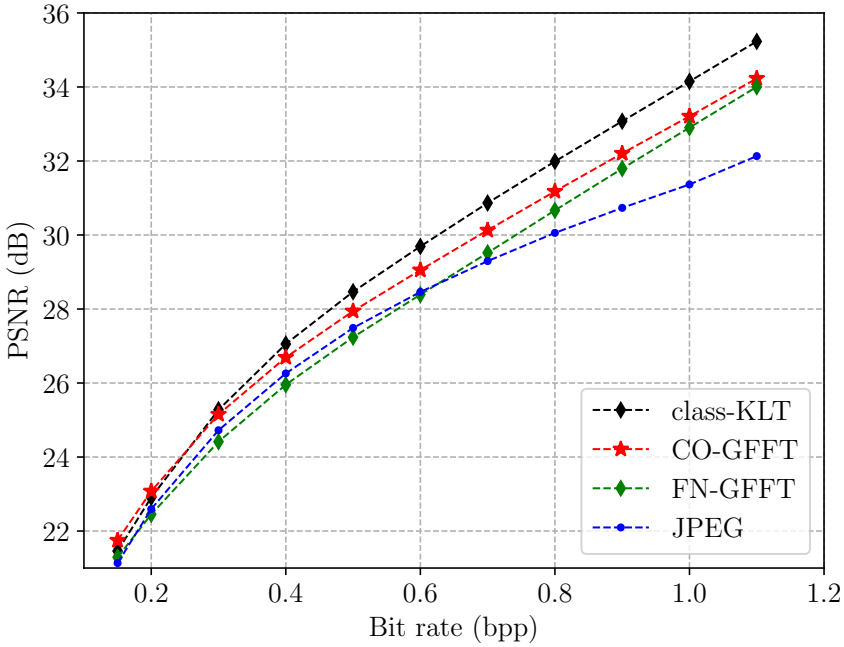


(c) Image Chessboard (0.25 bpp)

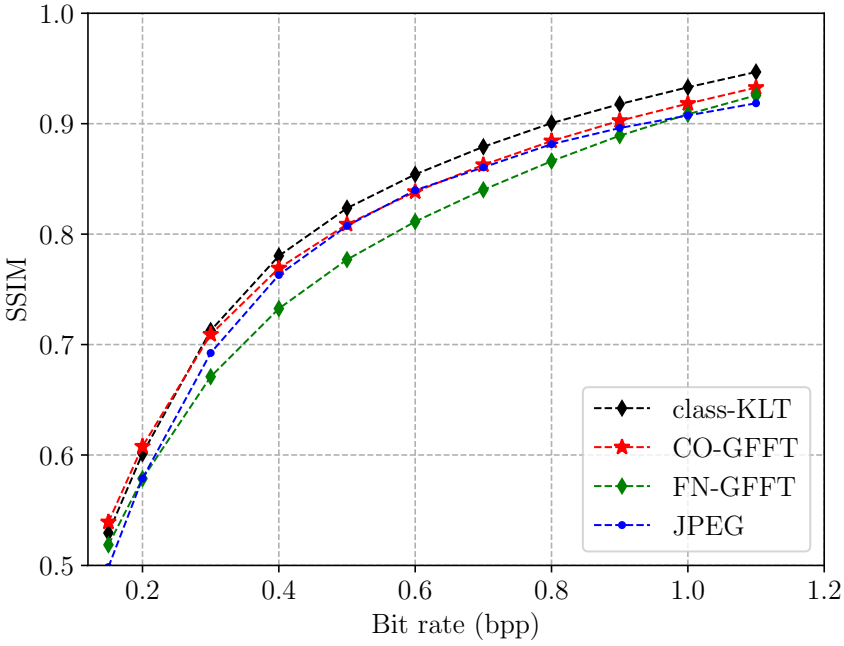
Figure 6.5: 8×8 blocks (in blue) for which a CO-GFFT was chosen over the 2D-DCT by the RD optimization procedure.

6.3.1 Comparisons with JPEG codec

RD performance of the image codec with $\mathcal{Q}_{\text{CO-GFFT}}$ is compared with the JPEG codec in this section. To maintain a complexity comparable to that of the 2D-DCT, the proposed data-driven fast transform employs an 86-stage Givens factorization, and its coding performance in the $\mathcal{Q}_{\text{CO-GFFT}}$ codec is evaluated relative to the JPEG method.



(a) Average PSNR



(b) Average SSIM

Figure 6.6: Average PSNR and SSIM over a test set of 40 images.

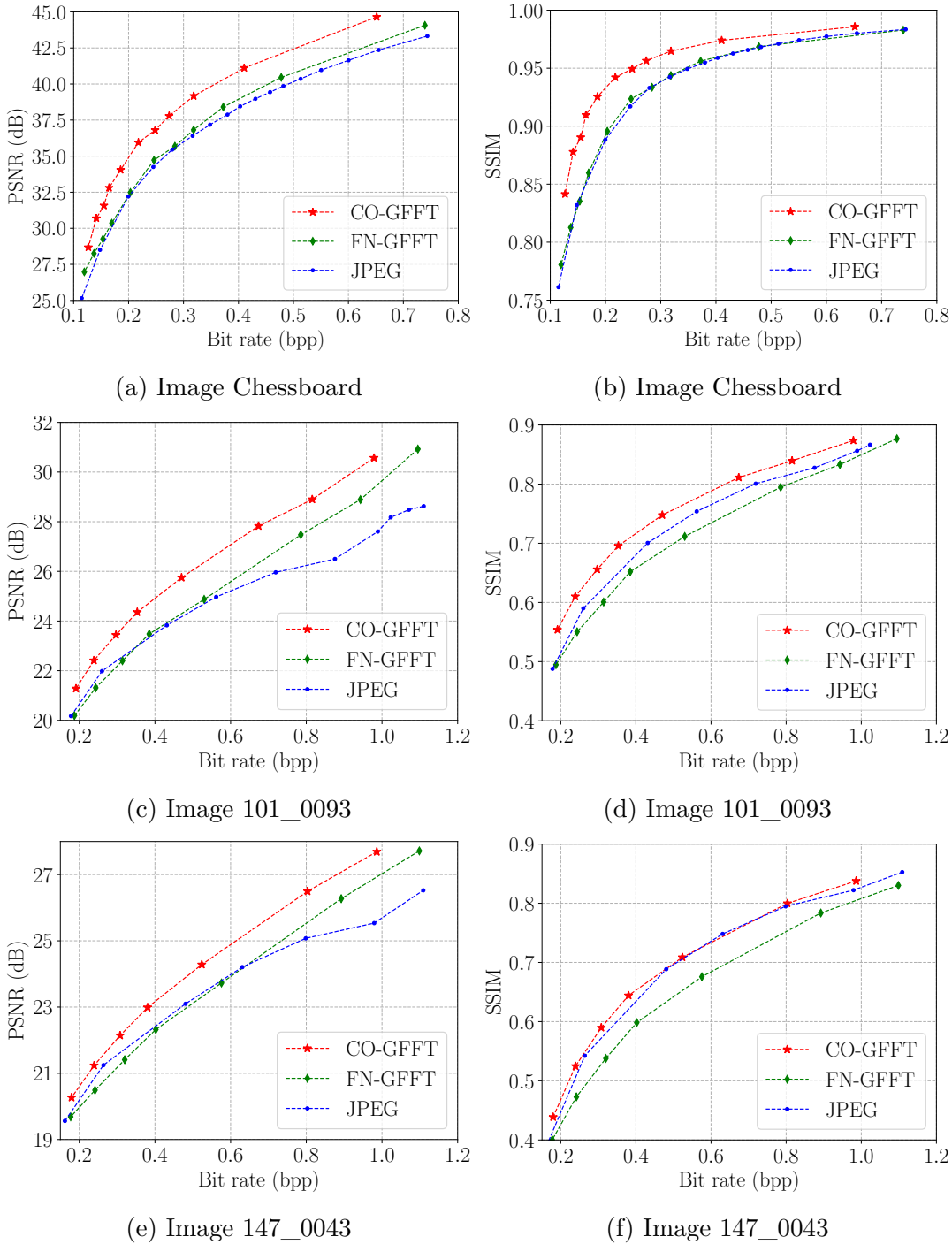
Fig. 6.6 illustrates the average RD performance of the GFFT-based encoders along with that of the baseline JPEG encoder,[4] averaged over the entire image test dataset. For all the bit rates CO-GFFT based codec outperforms the JPEG codec. The curves labeled class-KLT represent the performance of the adaptive encoder utilizing the $\mathcal{Q}_{\text{KLT}}$, which served as the upper bound for the CO-GFFT based codec. The performance gap observed between the GFFT-based codebooks and the class-KLT codebook stems from the approximation of the latter by computationally efficient 86-stage GFFTs. Specifically, the computational complexity of an 86-stage CO-GFFT matrix of size 64×64 is 516 arithmetic operations, equivalent to that of the 2D-DCT, whereas a full 64×64 class-KLT matrix requires 8128 operations.

Experimental results across a broad set of images indicate that the CO-GFFTs based adaptive encoder consistently outperforms the 2D-DCT baseline at all bit rates for most test cases, although the extent of the performance gain varies depending on image content. For instance, Fig. 6.7 compares the coding performance of the evaluated methods on three representative test images, and decoded examples from the CO-GFFT and JPEG encoders are shown in Figs. 6.8, 6.9, and 6.10. Notably, the CO-GFFT approach more accurately preserves details in regions with complex textures and randomly oriented edges. This is evident in the sharpness of the chessboard squares and chess pieces in Fig. 6.8, the clarity of the letters and book spiral in Fig. 6.9, and the mushroom surface in Fig. 6.10.

6.3.2 Comparisons with FN-GFFT based codebook

The FN-GFFT method is adapted for image coding as a Frobenius norm–based approximation to the KLT, and similarly uses an 86-stage Givens factorization to con-

[4]JPEG encoding is implemented using MATLAB's `imwrite()` function.

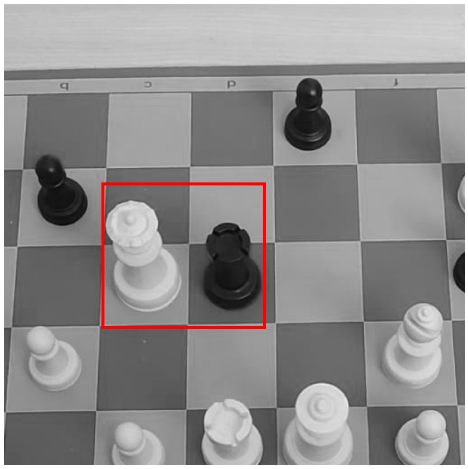


(a) Image Chessboard (b) Image Chessboard

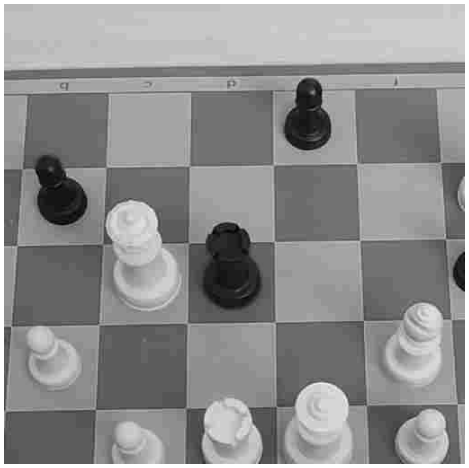
(c) Image 101_0093 (d) Image 101_0093

(e) Image 147_0043 (f) Image 147_0043

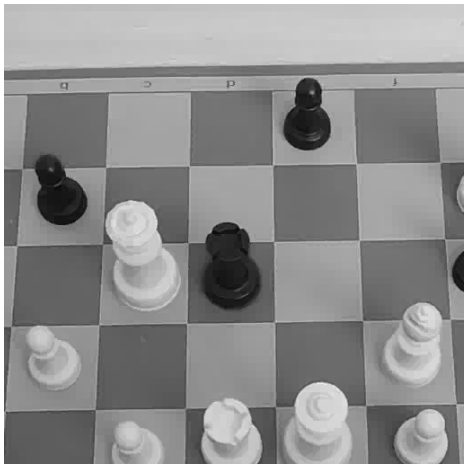
Figure 6.7: Comparison of PSNR (dB) and SSIM performance for three test images.



(a) Original image



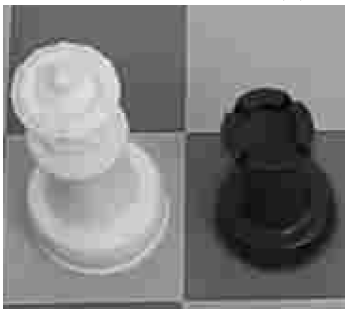
(b) JPEG (33.9 dB)



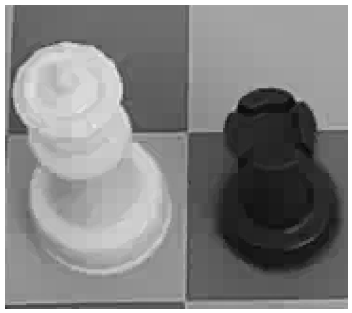
(c) CO-GFFT (36.7 dB)



(d) Original



(e) JPEG (32.9 dB)



(f) CO-GFFT (35.3 dB)

Figure 6.8: Comparison of the baseline JPEG encoder and the CO-GFFT encoder for the test image *Chessboard* at 0.25 bpp rate. Values in the parentheses are the PSNRs.



(a) Original image



(b) JPEG (27.4 dB)



(c) CO-GFFT (30.1 dB)



(d) Original



(e) JPEG (24.1 dB)



(f) CO-GFFT (28.3 dB)

Figure 6.9: Comparison of the baseline JPEG encoder and the CO-GFFT encoder for the Caltech test image 101_0093 at 1.0 bpp rate. Values in parentheses are the PSNRs.



(a) Original image



(b) JPEG (25.5 dB)



(c) CO-GFFT (27.3 dB)



(d) Original



(e) JPEG (23.2 dB)



(f) CO-GFFT (26.4 dB)

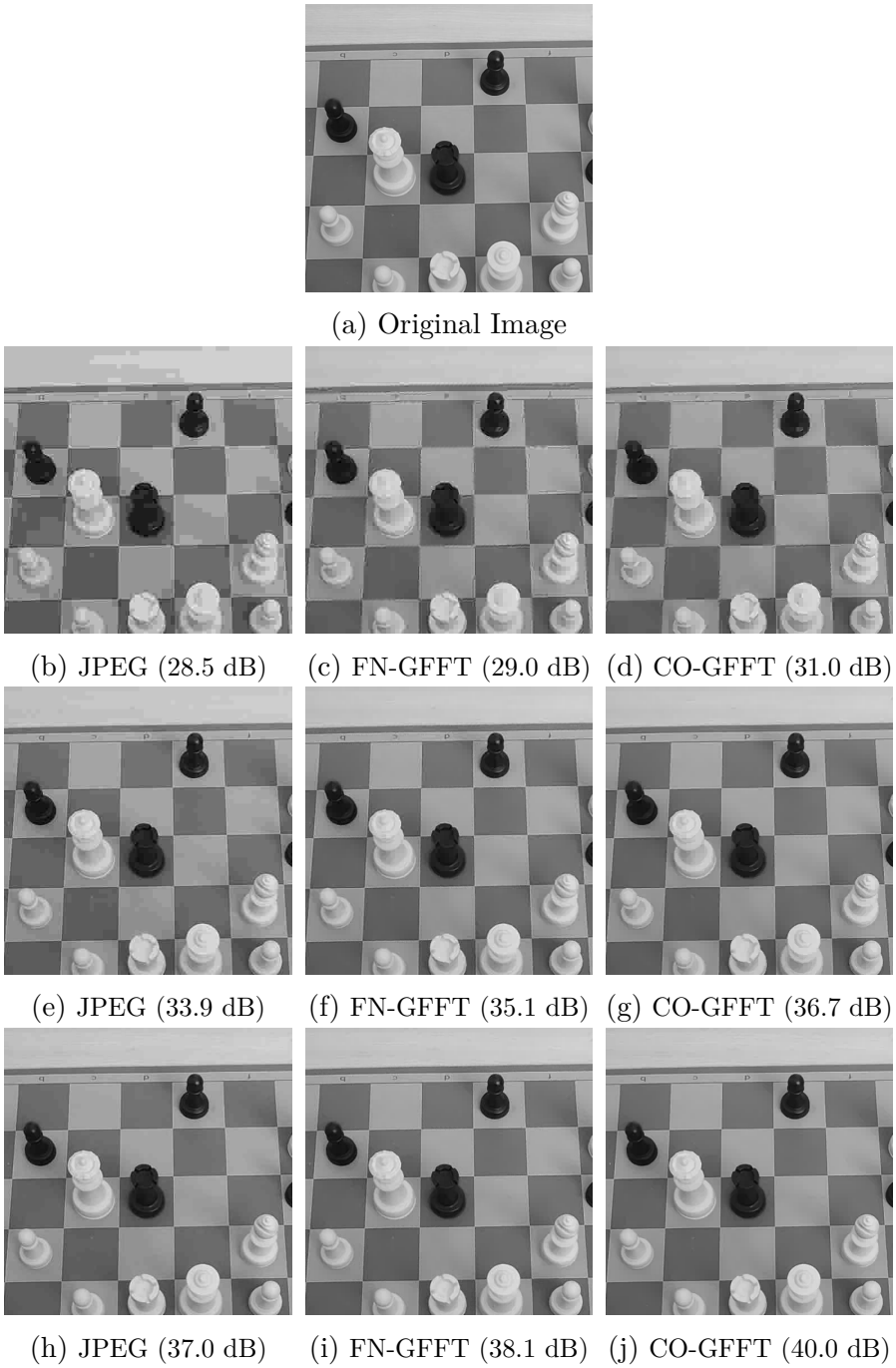
Figure 6.10: Comparison of the baseline JPEG encoder and the CO-GFFT encoder for the Caltech test image 147_0443 at 1.0 bpp rate. Values in parentheses are the PSNRs.

struct FN-GFFTs in $\mathcal{Q}_{\text{FN-GFFT}}$ as described in Chapter 4. Then $\mathcal{Q}_{\text{FN-GFFT}}$ is adapted to the codec described in Chapter 5, and average RD curves of the test image set are shown in the Fig. 6.6. According to the figure, CO-GFFTs provide better coding performance than the FN-GFFTs for all the bit rates. Fig. 6.11 illustrates examples of the chessboard image using CO-GFFTs, FN-GFFTs, and JPEG encoders at three different bit rates (0.15, 0.25, and 0.35 bpp). These results demonstrate the superiority of data-driven transform coding over the JPEG codec, which uses a fixed transform (2D-DCT). Data-driven GFFTs perform better on images with rich edges and textures, as these transforms are designed using gradient-based method.

Fig. 6.12 shows the percentage of blocks that select the 2D-DCT transform over the GFFTs for four test images, using both the CO-GFFT and FN-GFFT codebooks. At all bit rates, the 2D-DCT is chosen more frequently in the FN-GFFT codec than in the CO-GFFT codec. Owing to its better approximation capability relative to the FN-GFFT, the CO-GFFT achieves improved energy compaction. Consequently, a larger fraction of image blocks is adaptively coded using CO-GFFTs rather than the 2D-DCT at an equivalent bit rate.

6.4 Complexity-Performance Trade-off of the CO-GFFT Codec

The experimental results demonstrate that, for the same transform complexity of $O(N^2 \log_2 N)$ on $N \times N$ image blocks, the CO-GFFT-based adaptive codec outperforms JPEG and introduces fewer visual coding artifacts. Note that the decoders of both the adaptive codec and the JPEG codec have the same order of complexity. However, the adaptive encoder requires block-wise RD optimal transform selection,



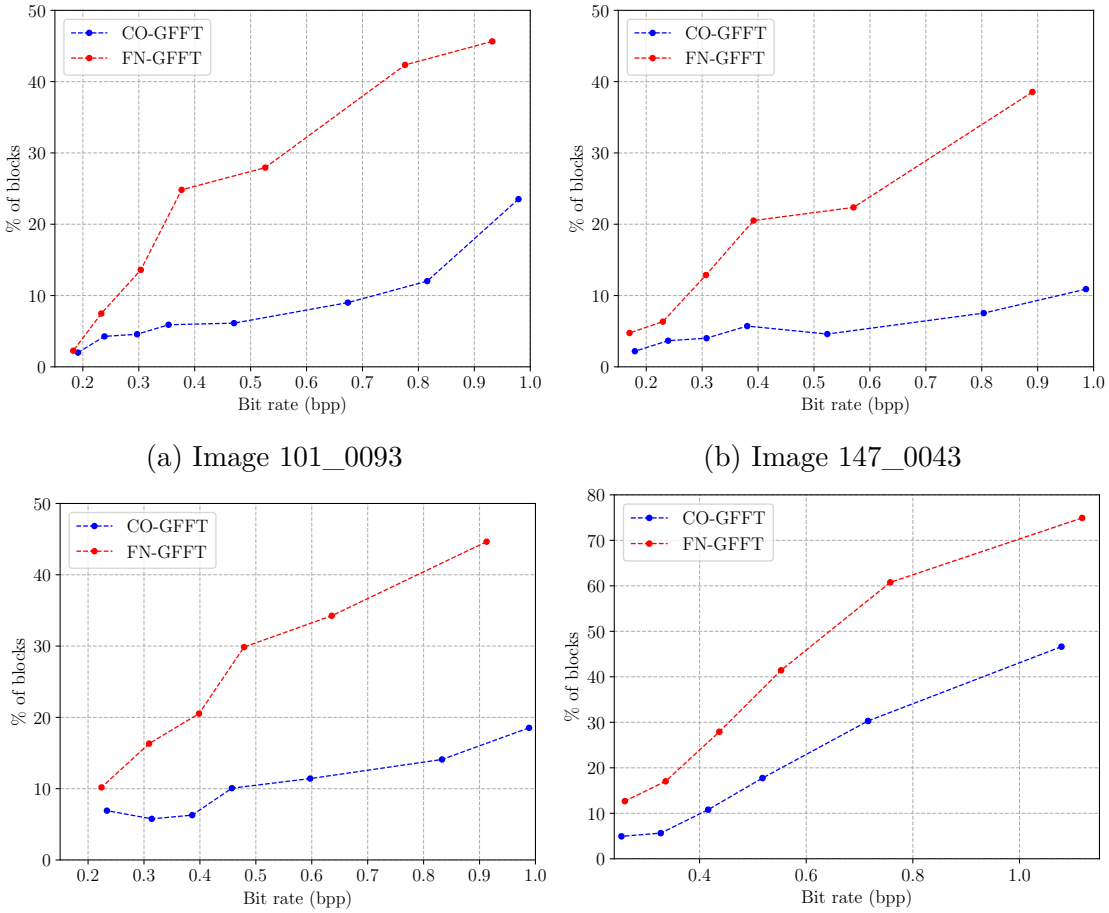
(a) Original Image

(b) JPEG (28.5 dB) (c) FN-GFFT (29.0 dB) (d) CO-GFFT (31.0 dB)

(e) JPEG (33.9 dB) (f) FN-GFFT (35.1 dB) (g) CO-GFFT (36.7 dB)

(h) JPEG (37.0 dB) (i) FN-GFFT (38.1 dB) (j) CO-GFFT (40.0 dB)

Figure 6.11: Comparison of the baseline JPEG encoder, the FN-GFFT encoder, and the CO-GFFT encoder for test image chess_board at different bitrates.(b)-(d) at 0.15 bpp, (e)-(g) at 0.25 bpp and , (h)-(j) at 0.35 bpp.



(a) Image 101_0093

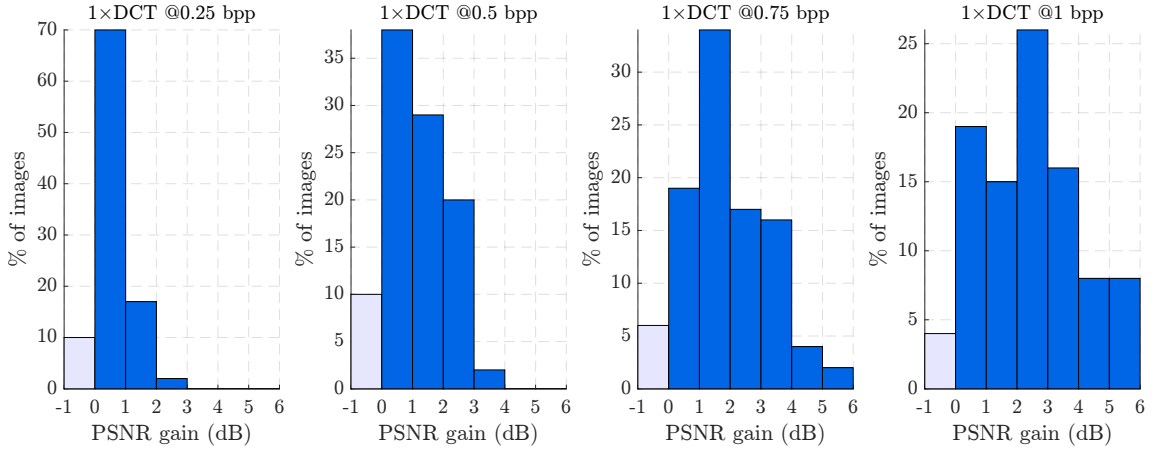
(b) Image 147_0043

(c) Image 147_0196

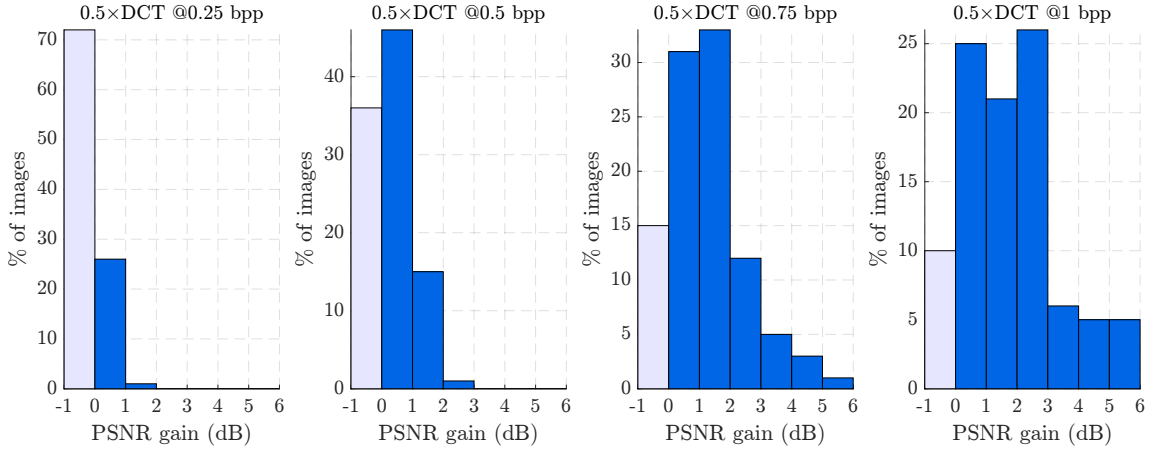
(d) Image 250_0013

Figure 6.12: Comparison of the percentage of blocks in for test images, for which the 2D-DCT was chosen in the CO-GFFT codec and the FN-GFTT codec.

resulting in a complexity of $O(KN^2 \log_2 N)$ for a GFFT codebook of K matrices. This remains significantly more efficient than using data-driven KLTs directly for adaptation. A key advantage of CO-GFFTs is the flexibility to control the transform complexity. As illustrated in Fig. 6.13(b), PSNR-gain histograms for CO-GFFT codecs with reduced complexity, specifically, 0.5× that of the 2D-DCT (with $L = 43$ and codebook size $K = 64$), show that the codec can still outperform JPEG for certain images. Furthermore, the fraction of such cases increases with bit-rate. This trend is consistent across all complexity levels and is attributed to the relatively large bit



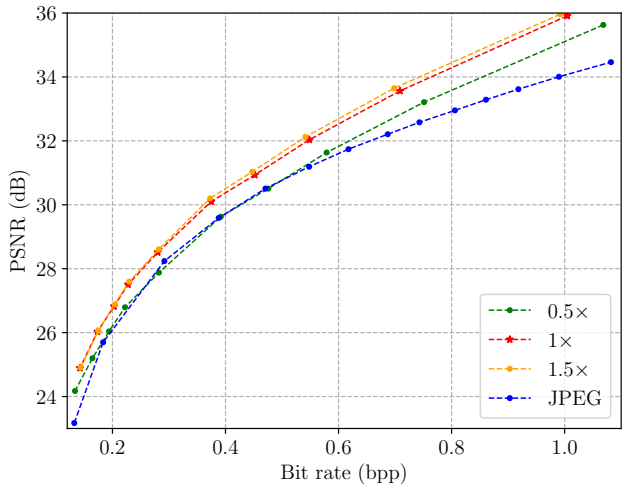
(a) 2D-DCT complexity



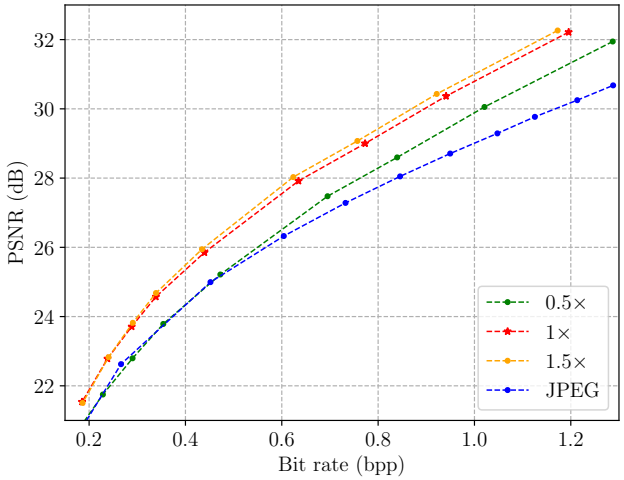
(b) 50% 2D-DCT complexity

Figure 6.13: Histograms of PSNR gains achieved by CO-GFFTs with transform complexities identical to the 2D-DCT (8×8 image blocks) and 50% that of the 2D-DCT. The horizontal axis is the PSNR gain relative to the baseline JPEG codec. The test contained 100 images of sizes ranging from 448×336 to 880×600.

overhead of signalling transform indices at lower bit rates. Fig. 6.14 further highlights this observation using two example images where, even at 50% transform complexity, the CO-GFFT codec significantly outperforms JPEG at higher bit rates.



(a) Test image *Pirate*



(b) Test image *Kodim05*

Figure 6.14: PSNR performance of CO-GFFTs for different transform complexities relative to the 2D-DCT (test images taken from the Kodak data set).

Chapter 7

Conclusion and Future Directions

Although data-driven transform coding can yield better image quality compared to the 2D-DCT, encoding with data-driven transforms often incurs high computational complexity. To address this challenge, data-driven fast transforms with computational complexity comparable to that of the 2D-DCT were investigated for image coding. These fast transforms are based on structured constructions using approximate Givens factorizations of orthonormal matrices. It was observed that GFFTs specifically optimized for the expected MSE of transform coding using a multi-path coordinate descent algorithm achieved noticeably better compression performance than those optimized for indirect criteria such as the Frobenius distance to the sample KLT. Notably, at equivalent transform computation complexity of the 2D-DCT, the codec employing MSE-minimization based GFFTs was found to outperform the 2D-DCT-based JPEG codec for the majority of test images, and in some cases, even when at half of the 2D-DCT complexity. This observation is significant, as data-driven transform encoders inherently involve higher computational complexity due to the rate-distortion optimal selection of transforms on a per-block basis. Without the

availability of structured, only data-driven fast transforms, such as the encoder may become impractical for real-world applications. Moreover, since the decoder complexity remains comparable to that of the 2D-DCT, GFFT-based codecs can provide improved image quality while maintaining the same decoding complexity as JPEG. This makes them particularly well-suited for scenarios in which encoding is performed once, or on a high-performance device, while decoding is executed repeatedly on devices with limited computational resources.

A small fraction of images exhibited slight degradation in quality when encoded using the CO-GFFT-based codec compared to the JPEG codec. This was found to be due to the use of a uniform quantization step size across all transform coefficients within an image block, whereas the JPEG encoder employs an optimized quantization table tailored to the characteristics of the 2D-DCT. Therefore, as part of the future work, further improvements over JPEG may be achieved by optimizing a quantization table according to the statistical properties of the GFFT coefficients. The proposed multi-path coordinate descent algorithm used in the design of CO-GFFTs is currently based on a single covariance matrix. A promising direction for future research is to extend this method to identify an optimal GFFT for a set of covariance matrices by jointly optimizing the coding gain across the entire set. Such an approach could enable the construction of transforms that are more broadly effective across diverse signal classes, potentially improving compression performance in more complex or varied image sources.

Appendix A

Generalized Lloyd Algorithm

The generalized Lloyd algorithm is given in Algorithm 3.

Algorithm 3: Generalized Lloyd algorithm

Data: Dominant directions of training image block dataset τ, Initial centroids $K^{(0)} = \{y_1, \ldots, y_n\}$, average distortion threshold δ

Result: n-image block classes

$\Delta D_c \leftarrow$ inf;

repeat

  Partition the training dataset into cluster sets R_i using the nearest neighbor condition: $R_i = \{x \in \tau : d(x, y_i) \leq d(x, y_j);\ \text{all}\ \ j \neq i\}$.

  Using the centroid condition, compute the centroids for the cluster set.

  Compute the average distortion ΔD_c for the dataset.

  $\Delta D \leftarrow \Delta D_{i-1} - \Delta D_i$

  $\Delta D_c \leftarrow \Delta D_c$;

until $\Delta D \leq \delta$;

Appendix B

Huffman Table for coding DC Coefficients

Table B.1: Huffman table for coding DC coefficients.

Bit-length of difference	Code word
0	00
1	010
2	011
3	100
4	101
5	110
6	1110
7	11110
8	111110
9	1111110
10	11111110
11	111111110

Appendix C

Huffman Table for coding AC Coefficients

Table C.1: Huffman table for coding AC Coefficients.

Run-size	Code word
EOB	110
0/1	00
0/2	100
0/3	01000
0/4	111010
0/5	010110001
0/6	010110110111
0/7	0100100010101010110
1/1	011
1/2	1111
1/3	101100
1/4	01001001
1/5	101110101
1/6	010010001110
1/7	0100100011000010111

Run-size	Code word
2/1	1010
2/2	101101
2/3	01011111
2/4	01001000100
2/5	0101111010110
2/6	010010001100010100
2/7	0101111010100100101111
3/1	01010
3/2	1110111
3/3	0101101111
3/4	0100100011110
3/5	0101111010100101
3/6	01001000110001010111
3/7	0101111010100100101110
4/1	11100
4/2	010110000
4/3	010010001011
4/4	010010001010100
4/5	010010001100010110
4/6	01011110101001001010
4/7	010111101010010011001
5/1	010011
5/2	010111100
5/3	010010001101
5/4	010111101010000
5/5	01001000101010100110
5/6	01011110101000111101
5/7	010111101010010011011000
6/1	101111
6/2	0101101110
6/3	0101111010101
6/4	0101111010100010
6/5	0100100010101010010
6/6	01011110101000111100
6/7	010111101010010011011011
7/1	0100101
7/2	0101111011

Run-size	Code word
7/3	0100100011111
7/4	01001000101010110
7/5	01001000101010101111
7/6	01011110101000111111
7/7	0101111010100100101101 0
8/1	0101110
8/2	01011110100
8/3	010010001100000
8/4	010010001010101010
8/5	01011110101000111110
8/6	01011110101000111001
8/7	0100100011000101010010 1
9/1	1011100
9/2	01011011010
9/3	01001000101011
9/4	01011110101000 1101
9/5	01001000110001010110
9/6	01011110101000111000
9/7	0100100011000101010010 0
10/1	01011001
10/2	0100100010100
10/3	0100100011000100
10/4	0100100010101010001
10/5	01011110101000111011
10/6	01011110101000111010
10/7	0100100011000101010011 1
11/1	01011010
11/2	010110110110
11/3	0100100011000010
11/4	01011110101000 11001
11/5	01011110101001000101
11/6	01011110101001000100
11/7	0100100011000101010011 0
12/1	10111011
12/2	0101111010111
12/3	01001000101010111
12/4	0100100010101010111 0

Run-size	Code word
12/5	01011110101001000111
12/6	01011110101001000110
12/7	0100100011000101010000l
13/1	010010000
13/2	0100100011001
13/3	010111101010011
13/4	0101111010100011000
13/5	01011110101001000001
13/6	01011110101001000000
13/7	010010001100010101000000
14/1	101110100
14/2	010010001100011
14/3	0100100010101010000
14/4	0100100010101010011l
14/5	01011110101001000011
14/6	01011110101001000010
14/7	010010001100010101000011
15/0	1110110
15/1	0101101100
15/2	0100100011000011
15/3	010111101010010011
15/4	0100100011000101010l
15/5	01011110101001001001
15/6	0101111010100100l000
15/7	010010001100010101000l0

Appendix D

Huffman Table for coding Transform Indices

Table D.1: Huffman table for coding optimal transform indices.

Bit-length of difference	Code word
1	0011001
2	0011000
3	0011011
4	0011010
5	110101
6	110100
7	110111
8	111110
9	110001
10	110000
11	110011
12	110010
13	111101
14	111100

Bit-length of difference	Code word
15	111111
16	111110
17	111001
18	111000
19	111011
20	111010
21	100101
22	100100
23	100111
24	100110
25	100001
26	100000
27	100011
28	100010
29	101101
30	101100
31	101111
32	101110
33	101001
34	101000
35	101011
36	101010
37	010101
38	010100
39	010111
40	010110
41	010001
42	010000
43	010011
44	010010
45	011101
46	011100
47	011111
48	011110
49	011001
50	011000

Bit-length of difference	Code word
51	011011
52	011010
53	0010101
54	0010100
55	0010111
56	0010110
57	0010001
58	0010000
59	0010011
60	0010010
61	0011101
62	0011100
63	001111
64	000

Bibliography

[1] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[2] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*, Wiley-Interscience, July 2006.

[4] K. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, 1990.

[5] "ITU-R rec. H.265 ISO/IEC 23008-2: High efficiency video coding," 2013.

[6] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," in *Picture Coding Symposium (PCS)*, 2016, pp. 1–5.

[7] P. Balle, J.and Chou, D. Minnen, N. Singh, S.and Johnston, S. Agustsson, E.and Hwang, and G. Toderici, "Nonlinear transform coding," *IEEE J. Sel. Topics Signal Process.*, vol. PP, pp. 1–1, 10 2020.

[8] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *International Conference on Learning Representations (ICLR)*, 2017.

[9] G. Toderici, D. Vincent, N. Johnston, D. Hwang, Sung J.and Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5435–5443.

[10] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[11] X. Zhang, C. Yang, X. Li, S. Liu, H. Yang, I. Katsavounidis, S. Lei, and C.-C. J. Kuo, "Image coding with data-driven transforms: Methodology, performance and potential," *IEEE Trans. Image Process.*, vol. 29, pp. 9292–9304, 2020.

[12] R. Boragolla and P. Yahampath, "An algorithm for learning orthonormal matrix codebooks for adaptive transform coding," *IEEE Trans. Image Process.*, vol. 32, pp. 3650–3663, 2023.

[13] O. G. Sezer, R. Cohen, and A. Vetro, "Robust learning of 2-D separable transforms for next-generation video coding," in *Data Compression Conference*, 2011, pp. 63–72.

[14] B. Zeng and J. Fu, "Directional discrete cosine transforms—a new framework for image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 3, pp. 305–313, 2008.

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

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

[17] C. Rusu and L. Rosasco, "Fast approximation of orthogonal matrices and application to PCA," *Signal Processing*, vol. 194, 5 2022.

[18] D. Morawaliyadda and P. Yahampath, "Construction of fast data-driven transforms for image compression via multipath coordinate descent on orthogonal matrix manifold," in *Data Compression Conference (DCC)*, 2024, pp. 452–461.

[19] D. Morawaliyadda and P. Yahampath, "Image coding with data-driven fast transforms based on approximate givens factorizations," *IEEE Access*, 2025, Accepted for publication.

[20] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Trans. Signal Process.*, vol. 40, no. 9, pp. 2174–2193, 1992.

[21] W. H. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. 25, no. 9, pp. 1004–1009, 1977.

[22] H. Lee, H.-J. Ahn, K.-R. Kim, P. Kim, and J.-Y. Koo, "Geodesic clustering for covariance matrices," *Communications for Statistical Applications and Methods*, vol. 22, pp. 321–331, 07 2015.

[23] A. Skodras, "Fast discrete cosine transform pruning," *IEEE Trans. Signal Process.*, vol. 42, pp. 1833 – 1837, 08 1994.

[24] X. Ji, C. Zhang, J. Wang, and S. Boey, "Fast 2-D 8×8 discrete cosine transform algorithm for image coding," *Science in China Series F: Information Sciences*, vol. 52, pp. 215–225, 02 2009.

[25] V.K. Goyal, J. Zhuang, and M. Veiterli, "Transform coding with backward adaptive updates," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1623–1633, 2000.

[26] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[27] R. J. Clarke, "Relation between the Karhunen Loève and cosine transforms," *IEEE Proceedings on (Communications, Radar and Signal Processing)*, vol. 128, pp. 359–360, 1981.

[28] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, 2008.

[29] R. A. Horn and C. R. Johnson, *Matrix Analysis.*, Cambridge University Press, 1985.

[30] R. Shepard, S. Brozell, and G. Gidofalvi, "The representation and parametrization of orthogonal matrices," *The journal of physical chemistry A*, vol. 119, 05 2015.

[31] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[32] W. A. Pearlman and Amir Said, *Digital Signal Compression: Principles and Practice*, Cambridge University Press, 2011.

[33] "ISO/IEC 10918-1 information technology — digital compression and coding of continuous-tone still images: Requirements and guidelines," 1994.

[34] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Prentice Hall, 2008.

[35] T. Brox, J. Weickert, B. Burgeth, and P. Mrázek, "Nonlinear structure tensors," *Image and Vision Computing*, vol. 24, no. 1, pp. 41–55, 2006.

[36] J. Bigun, G.H. Granlund, and J. Wiklund, "Multidimensional orientation estimation with applications to texture analysis and optical flow," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 8, pp. 775–790, 1991.

[37] I. Rotondo, G. Cheung, A. Ortega, and H. E. Egilmez, "Designing sparse graphs via structure tensor for block transform coding of images," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2015, pp. 571–574.

[38] J. Bigun and G. H. Granlund, "Optimal orientation detection of linear symmetry," in *IEEE International Conference on Computer Vision*, 1987.

[39] A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369.

[40] Z. Wang, E.P. Simoncelli, and A.C. Bovik, "Multiscale structural similarity for image quality assessment," in *37th Asilomar Conference on Signals, Systems Computers*, 2003, vol. 2, pp. 1398–1402 Vol.2.

[41] U. Shalit and G. Chechik, "Coordinate-descent for learning orthogonal matrices through Givens rotations," in *31st International Conference on Machine Learning*, Jun. 2014, pp. 548–556.

[42] J. B. Anderson and S. Mohan, *Source and Channel Coding: An Algorithmic Approach*, Springer New York NY, 1991.

[43] H. Chen and B. Zeng, "New transforms tightly bounded by DCT and KLT," *IEEE Signal Process. Lett.*, vol. 19, pp. 344–347, Jun. 2012.

[44] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics (NRL)*, vol. 52, 1955.

[45] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of Mathematics*, vol. 17, pp. 449–467, 1965.

[46] G. Griffin, A. Holub, and P Perona, "Caltech 256 (1.0) [data set]," Apr 2022, https://doi.org/10.22002/D1.20087.

[47] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *5th Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967, vol. 1, pp. 281–297.

[48] T. Wiegand and B. Girod, "Lagrange multiplier selection in hybrid video coder control," in *IEEE International Conference on Image Processing*, 2001, vol. 3, pp. 542–545 vol.3.