People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

$N°$ d'ordre : ....
Série: ...

Mohammed Seddik BENYAHIA University - Jijel

Faculty of Sciences and Technology
Department of Electronic

Thesis Submitted
In Partial Fulfillment of the Requirements for the Degree of Science
Doctorate in: **Electronic**

Presented by: BRAHIMI Nabila epouse DETTOUCHE

Subject

Contribution to Data Compression

The thesis defense will take place on: 17/12/2023 from 10h00

Board of Examiners:

Dr.	Mourad GRIMES	U.M.S.B. Jijel	Chairman
Pr.	Toufik BOUDEN	U.M.S.B. Jijel	Supervisor
Pr.	Larbi BOUBCHIR	U.Paris-8, LIASD, France	Co-supervisor
Pr.	Noureddine DOGHMANE	U.Annaba	Examiner
Pr.	Abderrazak LACHOURI	U.Skikda	Examiner
Dr.	Nasreddine KOUADRIA	U.Annaba	Examiner

To my parents
To my husband and to my daughter . . .

Acknowledgements

I would like to express my gratitude to my PhD supervisor, Pr. BOUDEN Toufik, from NDT Laboratory at university of Jijel, who has inspired me with his enthusiasm and provided me with new ideas and challenges every day. It has been an honour and a privilege to work with him and his team, and I hope our paths will cross again in the future. An especially grateful to my so-supervisor Pr. Larbi BOUBCHIR, who welcomed me and provided me with moral support during my thesis work at the LIASD Laboratory, University of Paris 8 Saint-Denis, France.

I am also thankful for the valuable assistance of Pr. Tahar BRAHIMI from L2EI Laboratory at university of Jijel.

I would like to extend my appreciation to the jury members, including Dr. Mourad GRIMES from the University of Jijel, Pr. Noureddine DOGHMANE and Dr. Nasreddine KOUADRIA from the University of Annaba, and Pr. Abderrazak LACHOURI from the University of Skikda who accepted to review my thesis, attended my PhD defense and provided me with constructive feedback.

I must also acknowledge my colleagues at the Department of Electronic, my friends, and the internet for the free and open-source software that was used in all the experiments.

I would like to extend my sincere gratitude to the members of the LEND laboratory, with special appreciation to Director Nabil Mahamdioua, for their invaluable support and guidance throughout my research journey.

Finally, I am grateful to all those who have contributed to this thesis in various ways, including the technical staff, fellow researchers, and participants in my study. Your contributions have enriched the findings and strengthened the validity of this work.

≪ Science does not aim at establishing immutable truths and eternal dogmas:
its aim is to approach the truth by successive approximations,
without claiming that at any stage
final and complete accuracy has been achieved ≫.
Bertrand Arthur William RUSSELL
The ABC of Relativity, 1925.

Abstract

In this thesis, two novel contributions for the enhancement of JPEG-based integer transforms are introduced. The first contribution consists of the development of two methods to generate integer Discrete Cosine Transform (DCT) approximations. The first proposed method is based on 16-points DCT and rounding-off operations leads to two new low-complexity transforms. The second one consists of an introduction of null elements into a specified integer DCT leads to two other new low-complexity, faster, and more efficient transforms. Therefore, the capability of proposed transform matrices is improved in image compression applications.The second contribution in this thesis consists of the development of a novel method for enhancement of integer-based DCT and integer-based DTT JPEG by generating a multiplier-less approximate Joint Photographic Expert Group (JPEG) quantisation matrix. The two proposed quantisation matrices have been successfully evaluated against the conventional JPEG quantisation matrix for all different type test images. Experimental results show that the compression of greyscale images by the JPEG standard based on the integer Discrete Tchebichev Transform (DTT) and combined with our proposed quantisation matrix leads to a performance improvement when compared to the JPEG conventional luminance quantisation matrix while maintaining high visual quality of the reconstructed images.

Keywords:
Image compression, JPEG, DCT , DTT, integer approximations.

Résumé

Dans cette thèse, deux nouvelles approches pour l'amélioration de standard JPEG basées sur les transformées entières sont introduites. La première approche consiste à développer deux méthodes pour générer des approximations DCT entières. La première méthode proposée est basée sur la DCT de 16-points et les opérations d'arrondi. La deuxième méthode proposée consiste à introduire des éléments nuls dans une DCT entière spécifiée, ce qui conduit à deux nouvelles transformations peu complexes, plus rapides et plus efficaces. Par conséquent, la capacité des matrices de transformation proposées est améliorée dans les applications de compression d'images. La deuxième approche de cette thèse consiste à développer une nouvelle méthode pour générer une matrice de quantification JPEG approximative sans multiplication. La matrice de quantification proposée a été évaluée avec succès par rapport à la matrice de quantification JPEG conventionnelle pour tous les différents types d'images de test. Une nouvelle matrice de quantification efficace a été générée aussi par cette approche en utilisant la DTT entière. Les résultats expérimentaux montrent que la compression d'images en niveaux de gris par la norme JPEG basée sur la TCD entière et combinée avec notre matrice de quantification proposée conduit à une amélioration des performances par rapport à la matrice de quantification de luminance conventionnelle du JPEG, tout en maintenant une haute qualité visuelle des images reconstruites.

Mots clés :

Compression d'images, JPEG, DCT, DTT, approximations entières.

ملخص

في هذه الأطروحة، تم تقديم طريقتين جديدتين لتحسين تحويلات الأعداد الصحيحة المستندة إلى JPEG. النهج الأول هو تطوير طريقتين لتوليد تقريبا للتحويل DCT باستعمال الاعداد الصحيحة. تعتمد الطريقة الأولى المقترحة على عمليات التقريب DCT المكونة من 16 نقطة. تتمثل الطريقة الثانية المقترحة في إدخال عناصر معدومة في مصفوفة DCT المقترحة من قبل مما يؤدي إلى تحولين جديدين منخفضي التعقيد وأسرع وأكثر كفاءة. لذلك، يتم تحسين قدرة مصفوفات التحويل المقترحة في تطبيقات ضغط الصور. النهج الثاني لهذه الأطروحة هو تطوير طريقة جديدة لتوليد مصفوفة تكميم JPEG خالية من التقريبات. تم تقييم مصفوفة التكميم المقترحة بنجاح مقابل مصفوفة تكميم JPEG التقليدية لأنواع مختلفة من صور الاختبار. كما تم إنشاء مصفوفة تكميم جديدة فعالة باستعمال هذا النهج للتحويل DTT. تظهر النتائج التجريبية أن ضغط الصور الرمادية بمعيارJPEG بناءًا على DTT بالكامل ودمجها مع مصفوفة التكميم المقترحة لدينا يؤدي إلى تحسين الخواص مقارنةً بمصفوفة تكميم التقليدية لـ JPEG ، مع الحفاظ على جودة بصرية عالية للصور.

الكلمات المفتاحية:

ضغط الصور، JPEG، DCT، DTT، تقريب الاعداد الصحيحة.

List of Abreviations

DCT Discrete Cosine Transform

DST Discrete Sine Transform

DTT Discrete Tchebichev Transform

JPEG Joint Photographic Expert Group

PSNR Peak Signal to Noise Ratio

dB Decibel

SDCT Signed Discrete Cosine Transform

GA Genetic Algorithm

NSGAII Non-dominated Sorting Genetic Algorithm II

APID Average Pixel Intensity Distance

VLSI Very Large-Scale Integrated

HDTV High-Definition TV

HDDs Hard Disk Drives

SHV Super Hi-Vision

RLC Run Length Coding

LZW Lempel-Ziv-Welch

VQ Vector Quantisation

MCU Minimum Coded Unit

DU Data Unit

DCPM Differential Predictive Coding Modulation

AC Alternating Current

DWT Discrete Wavelet Transform

SSIM Structural SIMilarity

PEEN Percentage Energy Error Norm

MSE Mean Square Error

APE Absolute percentage error

KLT Karhunen–Loève transform

DFT Discrete Fourier Transform

BDCT Binary Discrete Cosine Transform

WHT Walsh Hadamard Transform

SFG Signal Flow Graphs

InDCT Integer Discrete Cosine Transform

SFM patial Frequency Measure

SAM Spectral Activity Measure

DTP Discrete Tchebichef Polynomials

RCT reversible color transform

ICT Irreversible Color Transform

List of symbols

ϵ_{total}	Total error energy
$\| \, . \, \|_F$	Frobenius norm
$\eta(C)$	Efficiency of C Transform
A	Square matrix
A^t	Matrix transpose
A^{-1}	Inverse matrix
I	Matrix identique
A_h	Horizontal transform matrix
A_v	Vertical transform matrix
x	Input vector
X	Input matrix
y	ouput vector
Y	ouput matrix
C	Transform Matrix
C_T	Tchebichef transform matrix

List of figures

List of tables

Author works

0.1 Conference papers

1. **Designing Multiplier-Less JPEG Luminance Quantisation Matrix**

   ICONIP 2021, CCIS 1517, pp. 683–691, 2021.
   https://doi.org/10.1007/978-3-030-92310-5_79
   Authors:**Nabila Brahimi**, Toufik Bouden, Larbi Boubchir, and Tahar Brahimi

   Abstract. A novel approach for generating a multiplier-less approximate JPEG quantisation matrix has been proposed in this paper. It is based on energy distribution of integer DCTs coefficients and rounding to integer power-of-two matrix. No multiplications required during the encode and decode stages using the combination between an integer DCT and the proposed rounded quantisation matrix. An arithmetic operation savings about 44.8% using the proposed quantisation matrix with integer DCT against the JPEG. The proposed quantisation matrix has been successfully evaluated against the conventional JPEG quantisation matrix for all different type test images. The experimental results reveal that JPEG compression scheme base on integer DCT combined with our proposed quantisation matrix can provide significant improvement in PSNR values compared with other quantisation matrices. Keywords: DCT energy distribution, Integer DCT approximation, Integer DTT, JPEG quantisation matrix.

2. **Novel Quantisation Table for Lossy Image Compression**

   SSD 2022, pp. 1381-1386, 2022
   doi:10.1109/SSD54932.2022.9955865.
   Authors: **Nabila Brahimi**, Toufik Bouden, Tahar Brahimi

Abstract: The Discrete Tchebichef Transform (DTT) has been considered comparable to the discrete cosine transform (DCT) in terms of data compression performance. In this paper, we propose an integer DTT-based image compression scheme. For several standard test images with varying statistical features, a new efficient quantisation matrix can be generated using the energy distribution of the approximated integer coefficients of the DTT. Furthermore, as the majority of integer DTT matrices are not normal; their diagonal matrices have to be merged into the quantisation process. Experimental results show that the compression of greyscale images by the JPEG standard based on the integer DTT and combined with our proposed quantisation matrix leads to a performance improvement when compared to the JPEG conventional luminance quantisation matrix, while maintaining high visual quality of the reconstructed images.

Keywords—DTT, JPEG standard, image compression, quantisation matrix, Tchebichef transform.

0.2 Journals papers

1. **Brahimi, N.**, Bouden, T., Brahimi, T., Boubchir, L. (2020). A novel and efficient 8-point DCT approximation for image compression. Multimedia Tools and Applications, 79(11-12), 7615-7631. (IF : 2.577 (2021))

2. **Brahimi, N.**, Bouden, T., Brahimi, T., Boubchir, L. (2022). Lossy image compression based on efficient multiplier-less 8-points DCT. Multimedia Systems, 28(1), 171-182. (IF : 2.603 (2021))

3. **Brahimi, N.**, Bouden, T., Brahimi, T., Boubchir, L. (2022). Efficient multiplier-less parametric integer approximate transform based on 16-points DCT for image compression. Multimedia Tools and Applications, 81(26), 37723-37746. (IF : 2.577 (2021))

Table of contents

Introduction

Data compression schemes aim to reduce the amount of data required to represent any digital data, including images and signals [8–10]. Indeed, data compression is the art of information representation in a compact form [11]. Transform coding is a well-established and commonly used technique for image compression.

Today, JPEG [12] is a widely accepted format for image compression, largely due to its simplicity of implementation and its rather trad-off between complexity and compressed image quality. The Peak Signal to Noise Ratio (PSNR) of an JPEG image is just a few Decibel (dB) less than that of the same JPEG 2000 image, whereas the compression time of the former image is seven times smaller than that of the latter image [13]. In comparison, the noise resilience of a JPEG image is, in many cases, equivalent to that of a JPEG 2000 image [13], and the modern systems specified by the JPEG 2000 standard (i.e., scalability and region-of-interest) are also defined by the JPEG standard and its extensions. Although these extensions to the primary core of JPEG may be less successful, the low complexity of the JPEG can effectively offset this lower performance in many applications (especially battery-driven ones) compared to those of the JPEG 2000.

The standard JPEG is, despite its age and multiple attempts to replace it by more modern technology, still the most prominent still image codec used today [13]. Most digital cameras export JPEG format, and most image editing software supports JPEG compression operation. Hence, JPEG images are involved in many forensics issues, such as authenticity of JPEG compression history [14], image forgery detection [15, 16] or steganalysis application [17, 18] and the JPEG standard is also applicable to computer generated holography [19]. The JPEG standard uses the DCT-II as the core of the standard and three standard quantisation tables [20].

However, in image and video compression application, the 2-D DCT transformation stage is one of the most computationally intensive steps; it possesses the more significant computational complexity in terms of arithmetic operations. It consumes alone more than 60% of the total computation requirement of the encoder [21].

As with the 2-D separability property of 2-D DCT in its matrix form, it can be implemented by the row–column application of the 1-D DCT; therefore, for 8 elements vectors, the 1-D DCT of order 8 implementation required 64 floating-points multiplications and 56 additions. This number means rather important, thus, the development of fast DCT transformation algorithms is, therefore, an essential task for a successful hardware implementation. Thus, during decades, to make the DCT faster, several prominent algorithms have been proposed in the scientific literature including [22, 23].

The more efficient algorithm required 5 multiplications and 29 additions [24]. However, the problem of the floating-point number is still present. For an even faster realization of DCT transform, without multiplications, the implementation of many integer approximations has been investigated for more than a decade. The Signed Discrete Cosine Transform (SDCT) introduced in [25] is among the most recent of these integer transforms; it is obtained by applying the Signum function on the conventional DCT. Although it requires 24 additions only, it is non-orthogonal and its performance in image compression is still very low. Then, Bouguezel–Ahmad–Swamy (BAS) introduced a series of algorithms [7, 26, 27, 5, 28], whose they improve the performance of the SDCT approximation by introducing some zeros in it. In 2014, Cintra et al. introduced in [29] a collection of DCT approximations (12 transforms) based on applying integer functions to 8-floating-point DCT, the resulting transforms are orthogonal or quasi-orthogonal with a computational complexity ranging from 18 to 24 additions. Many other researchers have developed the low complexity integer approximations for the DCT; transforms requiring 14 additions only are introduced in [30–32] and another approach for reducing the arithmetic complexity of the DCT aims to compute only a subset of all the DCT coefficients called pruned DCT-like transformations are proposed in [33, 34] requiring 10 additions only, but a remarkable degradation in image compression quality has been observed.

One of the recent works of low complexity DCT transforms is proposed by Oliveira et al. in [35]. Two low complexity 8-point DCT approximations are obtained according to greedy heuristic, which minimised the angle between the rows of the approximation and exact DCT matrices, the two resulting transforms are orthogonal involving 24 additions and 6 bit-shift operations. Apart from this, the transform in [6] (PADCT) results from applying the Signum function operator to the transform in [36], this transform is quasi-orthogonal, requiring 17 additions. Another transform with only 10 additions is described in [33], while others transform are equally developed in [37, 31, 35, 38]. All the approximate transforms mentioned above are characterised

by a reduction in the number of operations required, while their image compression performance is relatively low.

The JPEG standard uses three standard quantisation tables [20]. For grayscale images, only one quantisation matrix is used, while in color images, separate quantisation matrices are allowed for two-color components. It has been recognised in the field of image compression and mentioned in the CCITT recommendation documents that the JPEG quantisation matrix is strongly linked to image quality and compression rate. To this end, in recent decades, many researchers in the literature have carried out different approaches to generate image-dependent/independent quantisation matrices. Wu in [39], based on Genetic Algorithm (GA), proposed a quantisation matrix referred to a slight improvement in quality of decoded image compared with the conventional quantisation matrix of the baseline JPEG. Better image compression and quality trade-off technique is developed by Lazzerini et al. [40] utilizing Non-dominated Sorting Genetic Algorithm II (NSGAII). A more efficient firefly algorithm has been proposed by Tuba and Bacanin in [41] using the Average Pixel Intensity Distance (APID) between the original and compressed image as an objective function. Noted that all these algorithms have a parameter-dependent nature, and they are intended for specific types of images. Hence, we show that the generation of quantisation matrices for the based integer transforms JPEG baseline is a necessary task. To our best knowledge, the only approximate JPEG quantisation matrix required bit-shift operations only is introduced in 2017 by Oliveira et al. [42]. The reason is to reduce the complexity of the transform-quantisation pair of JPEG algorithm. The elements of the approximate quantisation matrix are power of two numbers, and it provided almost the same performances compared it with its exact counterparts, with arithmetic operation savings of 24.1% in additions.

This thesis proposes an enhancement of baseline JPEG compression standard based on integer DCT approximations. Two contributions have been introduced.

The first one consists of:

• Introducing new multiplier-less integer approximations for the floating 8-point DCT-II which are a solution of DCT-II floating points multiplications problem; no multiplication operations are required.

• The proposed transforms approximate the conventional DCT-II very well by reducing the arithmetical operations, and it has good performance conservation.

• A fast algorithm is derived for each proposed transform.

• All the new transforms show their ability in image compression applications compared to existing transforms having the same number of arithmetic operations.

The second contribution consists of:

• Introduce an appropriate quantisation matrix for integer DCT-based JPEG regarding energy distribution of some integer DCT coefficients.

• An enhancement of the baseline JPEG image compression standard based on DTT approximations according to a suitable proposed quantisation matrix.

Accordingly, this thesis has been organised into the following Chapters:

Chapter 1 starts with an introduction to the basics of image coding and some essential concepts for this thesis on JPEG image coding standards. The focus is quickly put on the transform stage and its crucial role in the image coding scheme.

Chapter 2 contains a detailed study of the transforming role inside image coding applications. As mentioned, the main motivation of this thesis is to improve JPEG image coding by making use of integer transforms, and a state-of-the-art of integer DCT and integer DTT transforms.

Chapter 3, two novel approaches to developing low complexity integer 8-points DCT approximations were discussed in this chapter. Those approaches allow us to achieve four integers DCT approximations without multiplication operation. A new multiplier-less JPEG quantisation matrix based on the energy distribution of existing 8×8 integer DCT approximations. This strategy was applied to DTT based JPEG standard and leads to a new quantisation matrix.

Chapter 4 Comparative study of existing transforms and those proposed in this study is introduced in this chapter. The discussion of the experimental result is also introduced in this chapter.

Part I

IMAGES COMPRESSION FUNDAMENTALS

Chapter 1

Basic concept of image compression

1.1 Preview

In contemporary society, the utilisation of digital images and videos has become an indispensable requirement. However, these image files often possess a considerable storage capacity due to their large sizes. To illustrate this fact, we may consider the example of "megapixel quality" in digital cameras. This term denotes a digital color image with a resolution of 1024 pixels × 1024 pixels × 1024 pixels × 24 bits, which, without compression, requires a storage capacity of 3 MB. By way of comparison, a film roll with thirty-six exposures would occupy 100 MB. While advances in communication channels such as the 4G network have facilitated the transmission of color images via mobile devices, limited air bandwidth still presents certain challenges. Given the mounting demand for high-quality images and the concomitant rise in traffic, efficient data compression algorithms have become indispensable for the storage and transmission of digital images. Thus, image compression techniques have become an imperative task in the field of multimedia computing and in Internet applications.

1.2 Basic concepts of image compression

Image compression is the process of reducing the byte size of a source image without significantly degrading its quality. This can help to store a greater number of images in a given amount of disk space or memory, and to facilitate faster image transmission over the Internet or downloading from web pages. On the other hand, uncompressed multimedia data, such as images, audio, and video, can require considerable storage capacity and bandwidth. Despite the rapid advancements in mass storage density, processor speed, and digital communication system performance,

the demand for data storage capacity and data transmission bandwidth continues to exceed the capabilities of available technology.

1.3 Needs of compression

The ability to perceive, recognise, and understand the visual information that surrounds us is of extreme importance to humans. In recent years, significant advancements in technology, particularly in Very Large-Scale Integrated (VLSI) circuits and computing power, have made it possible for video to be used extensively in our daily lives. Examples of this include video telephony, video conferencing, High-Definition TV (HDTV) [43], and digital video. With the increasing demand for high-quality video services, advanced compression techniques for image and video data have become an essential requirement. Compression serves as a critical enabling technology that helps to bridge the gap between the massive amount of video data required and the limited hardware capabilities available [44].

To gain a better understanding of the significance of data compression, let us examine an example. Digital cinema and HDTV currently employ a 4K system, which consists of approximately 4096 x 2160 pixels per frame. However, the newly developed Super Hi-Vision (SHV) format uses 7680 x 4320 pixels per frame, with a frame rate of 60 frames per second. Consider a three-hour color video file based on SHV technology, where each pixel has a precision of 48 bits. The size of the resulting video file would be (7680 x 4320 x 48 x 60 x 3 x 60 x 60) bits, or roughly 120,135.498 GB in size [45]. For non-professional users, storage capacity between 500 GB to 1 TB is typically sufficient for storing movies [45]. According to Seagate, an American data storage company, the average capacity of Hard Disk Drives (HDDs) worldwide was 4.1 TB in the third quarter of 2020 [45]. Therefore, data compression plays a vital role in reducing the size of images and video files, making them more manageable and allowing for efficient storage and transmission.

1.4 Image compression algorithms

Generally, image compression algorithms are divided into two categories: lossless and lossy algorithms, as shown in Fig. 1.1. If we eliminate statistical redundancy with no information loss, this is lossless compression, and it is also called a reversible process. On the other hand, if it has information loss, it will be called lossy compression or irreversible process.

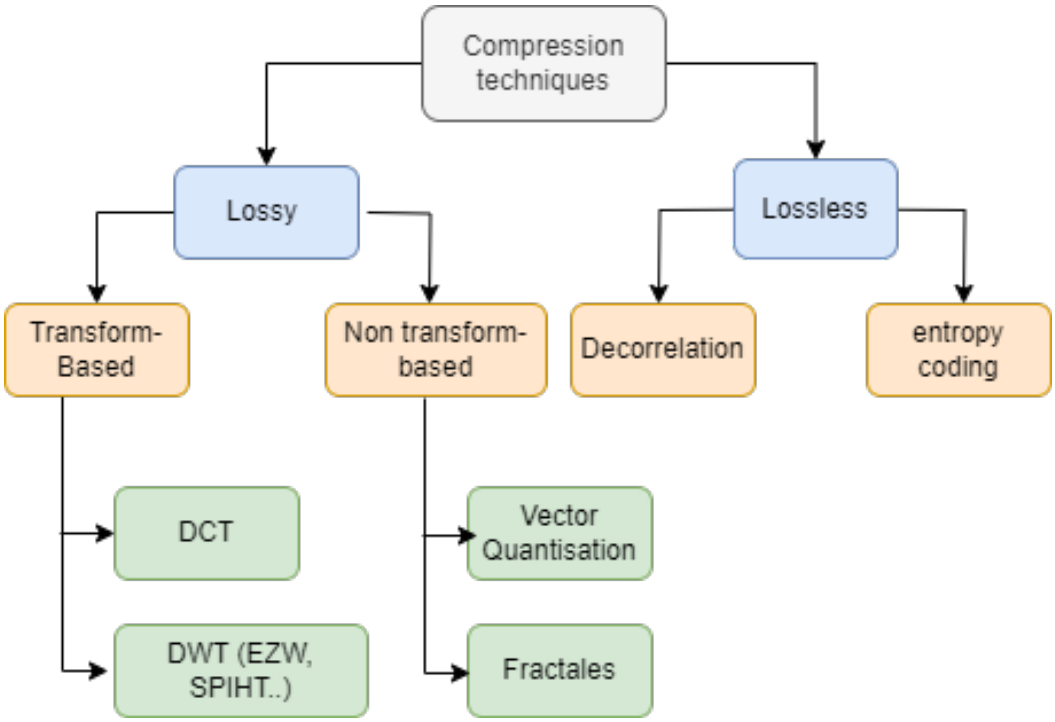


Fig. 1.1: Classification of image compression algorithms

1.4.1 Lossless image compression algorithms

The principle of lossless image compression is to minimise the number of bits used to represent the original image without any loss of information. In a lossless image compression scheme, the reconstructed image after compression is numerically identical to the original image. This is an important requirement for some application domains, e.g. medical imaging where not only high quality is in demand but unaltered archiving is a legal requirement. The lossless technique can also be used for the compression of other data types where the loss of information is not acceptable, e.g. text documents and program executables.
Lossless image compression techniques depend on two-stage procedures [46]:
a) Decorrelation.
b) Entropy coding.

1.4.1.1 Decorrelation

The correlation between samples, which is presented in almost all types of signals, represents redundant information that does not need to be transmitted if reversible decorrelation techniques are applied. These techniques are classified into three

categories: prediction-based techniques, transform-based techniques, and multi-resolution-based techniques [46].

1.4.1.2 Entropy coding

Once the data has been decorrelated, compression can be achieved by applying entropy coding. An entropy encoding or reversible compression of a data source corresponds to a lossless encoding of symbols, the purpose of which is to eliminate the redundancies of information present in the source. Entropy coding is used with some lossy image compression algorithms, i.e. JPEG Standard. It reduces the number of bits produced from quantised coefficients output for more compression. The fundamental concept of entropy coding is to assign the short code words to symbols that appear most often and the long code words to those that appear infrequently. Most entropy coders employ Run Length Coding (RLC), statistical coding (such as Huffman coding or arithmetic coding), or the Lempel-Ziv-Welch (LZW) coder.

1.4.2 Lossy image compression algorithms

The lossy term is applied to a data compression approach in which some quantity of the original data is lost during the compression process. Lossy image compression applications attempt to eliminate redundant or unnecessary information in terms of what the human eye can perceive. It provides a high compression ratio compared to lossless ones.

An image reconstructed following lossy compression is usually not the same as the original, it contains degradation relative to the original image. Lossy image compression is useful for application to the worldwide image for quicker transfer across the internet. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme discards the redundant information. The lossy image compression techniques are classified into 2 categories, as shown in Fig. 1.1.

1.4.2.1 Nontransform-based compression techniques

(a) *Vector quantisation*

As shown from Vector Quantisation (VQ) the encoding block diagram described in Fig. 1.2, the image is partitioned into blocks of pixels, and each block is represented by a vector x. This vector is compared against code words in the code book on the encoder side, which will get the index of the best codeword match. Therefore, an index stored in fewer bits will be transmitted instead of

a code word, which achieves more compression ratio. The performance of VQ depends on the appropriate codebook, so researchers proposed optimisation techniques for global codebook generation. The VQ technique performance is better than scalar quantisation methods such as Pulse Code Modulation (PCM), Differential PCM (DPCM), Adaptive DPCM. VQ [47, 48] is basically a c-means clustering method widely used for image compression [49, 50], pattern recognition [51, 52], speech recognition [53, 54], face detection [55], speech and image coding because of its excellent rate-distortion performance Traditional vector quantisation suffers several problems [56]:

• High computational cost: visual codebook generation is computationally expensive, especially with many features.
• Limited reliability: codebook construction in vector quantisation relies on the collection of image features and codebook generation methods.
• Update inefficiency: with many new features collected, the codebook/quantiser should be updated accordingly. However, the codebook updating needs lots of effort.

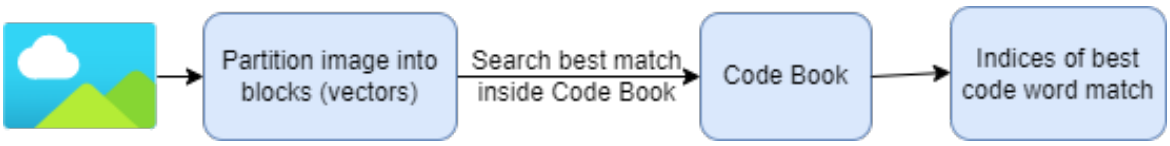


Fig. 1.2: Block diagram of vector quantisation encoder

(b) *Fractal compression*
Fractal compression is a lossy compression technique based on fractal theory [57], according to which an image can be described by a set of fractals. The idea behind this technique is that any image is a finite set of geometric transformations (rotations, enlargements, reductions) that apply to the subsets of identical patterns and varying sizes that compose it. Fractal compression then consists of replacing the input image with a series of mathematical formulas that allow it to be recomposed in its entirety. The efficiency of the compression operation is therefore proportional to the importance of the geometric properties of the image. The more these are numerous, the fewer the number of mathematical formulas will be.

Fractal image compression, as explored by [58], demonstrates notable capabilities for achieving high compression rates while maintaining image reconstruc-

tion quality. Additionally, the decoding process is characterized by its simplicity, offering a straightforward interpretation of fractal codes. However, it is important to acknowledge a significant drawback associated primarily with the coding process, namely its extensive computational requirements, and time consumption. Consequently, the application of fractal compression within the domain of sensor arrays proves unfavorable due to increased power consumption, potentially leading to a shortened operational lifespan of the array

1.4.2.2 Transform-based compression techniques

Transform coding has been widely used in image and video compression. There are several characteristics desirable for the purpose of data compression. Transforms are useful entities that encapsulate the (some/all) following characteristics [59] :

(i) *Data decorrelation*: The ideal transform completely decorrelates the data in a sequence/block; i.e., it packs the most amount of energy in the fewest number of coefficients. In this way, many coefficients can be discarded after quantization and prior to encoding. It is important to note that the transform operation itself does not achieve any compression. It aims at decorrelating the original data and compacting a large fraction of the signal energy into relatively few transform coefficients.

(ii) *Data-independent basis functions*: Owing to the large statistical variations among data, the optimum transform usually depends on the data, and finding the basis functions of such transformation is a computationally intensive task. This is particularly a problem if the data blocks are highly non-stationary, which necessitates the use of more than one set of basis functions to achieve high decorrelation. Therefore, it is desirable to trade optimum performance for a transform whose basis functions are data-independent.

(iii) *Fast implementation*: The number of operations required for an N-point transform is generally of the order $O(n^2)$. Some transforms have fast implementations, which reduce the number of operations to $O(n.log_2 n)$. For a separable $n \times n$ 2D transform, performing the row and column 1D transforms successively reduces the number of operations from $O(n^4)$ to $O(2n^2.log_2 n)$.

Transform-based compression algorithms are generally required in three basic steps, see Fig. 1.3.

(i) The first step is the transformation of the image data to eliminate the spatial redundancy or correlation between neighboring pixels.

(ii) The second step is quantisation or thresholding, which reduces the space in which the data is represented. It is in this step that the loss of information occurs because quantisation involves dividing and rounding the result into an integer.

(iii) The last step is the encoding of the quantised data. It helps organise the binary train by compressing the quantised values without altering the information for the purpose of transmission or archiving.

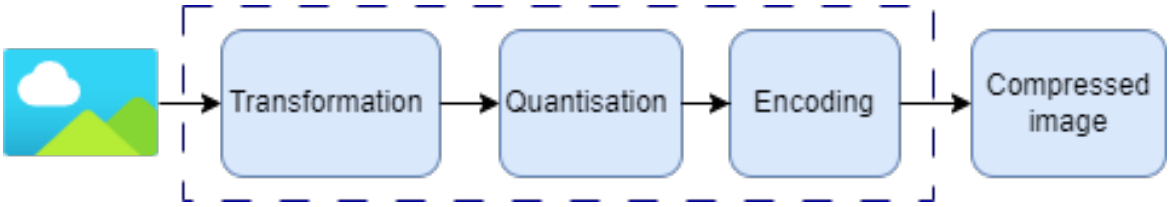


Fig. 1.3: The encoding process based on transform encoder

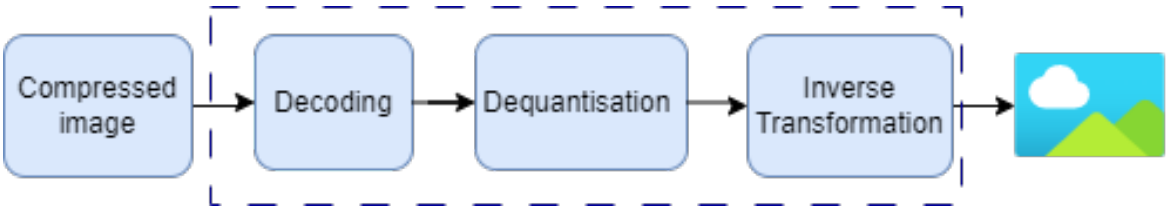


Fig. 1.4: The decoding process based on transform encoder

In the phase of decompression or reconstruction of the image, three steps are necessary:
•The transmitted or archived data is decoded.
•Inverse quantisation is applied to the decoding result.
•Then the inverse transformation is applied to the dequantised data as presented in Fig. 1.4.

(i) *Transformation*: The objective of the transformation is the decorrelation of information. It eliminates the redundancy between neighboring pixels and concentrates the relevant information of the image on a few significant coefficients, which makes it possible to optimise the subsequent quantisation and coding steps. This processing step is reversible and lossless. The choice of the transformation method must allow a reduction of the correlation of the input data. Transform selection is an important factor in data compression. The

best-known transforms in the field of compression are:
• Discrete Fourier Transform (DFT) [60],
• Discrete Cosine Transform (DCT) [61],
• Discrete Hadamard Transform (DHT),
• Karhunen Loève Transform (KLT) [62],
• Generalized Walsh Transform (WGT),
• Discrete Wavelet Transform (DWT) [63].
DCT and DWT are the most used in data compression, and in particular in image compression.

(ii) *Quantisation*: In the compression scheme, the quantisation step is primarily responsible for image degradation. During this step, the image will be really compressed. Quantisation is used in practice, and it is based on the principle that low-frequency coefficients make the greatest contribution to the quality of the reconstructed block. Only the coefficients, whose magnitudes exceed a threshold, are important, and all those that remain can be excluded from the reconstruction of the image.
By quantisation and thresholding, a degradation of quality and a reduction of precision are introduced. This degradation is based on the choice of the quantisation factor and the threshold value [46, 50]. If the threshold value is high, the degradation will be considerable but if the threshold value is low, then the degradation value is negligible or low with such a low compression ratio.

(iii) *Encoding*: After quantisation or thresholding, the matrix obtained is available for coding. For transmission or archiving, it is very advantageous to minimise the number of signal bits. The methods used are the reversible compression methods. Examples include Huffman coding, EZW coding, etc. Among the most famous entropy coders, we can cite Huffman coding, arithmetic coding, or even Golomb coding.

1.5 Lossy image compression standards

1.5.1 Standard JPEG

JPEG, proposed as a compression standard by the International Organization for Standardization (ISO) and the Consultative Committee for International Telegraphy and Telephony (CCITT), serves as a widely adopted method for compressing digital images color, monochrome, multiple greyscale, and continuous-tone formats.

Among the several modes available within the JPEG framework, the sequential mode, based on the DCT, emerges as the most commonly employed approach. The JPEG compression algorithm can be conceptually divided into distinct steps, as detailed by Yuan in [64], to facilitate its operation procedure and comprehension. These steps encompass the following:

1.5.1.1 Color change

JPEG employs the YC_bC_r color space, requiring a color space transformation from RGB. The RGB information from the original bitmap is converted into the luminance component (Y), representing brightness, and the chrominance components (C_b and C_r), representing chroma. This transformation enables subsequent processing steps to be performed efficiently.

1.5.1.2 Minimum Coded Unit (MCU), Data Unit (DU) and image sampling

The luminance component (Y) carries significant data, whereas the chrominance components (C_b and C_r) hold relatively less important information. To enhance compression efficiency, it is possible to selectively consider only a portion of the chrominance data (C_bC_r). Modern software supporting the JPEG format commonly offers two sampling methods, namely × and YC_bC_r422, as described by [65], which correspond to the data sampling ratios of the three components within the YC_bC_r color space. Considering image quality factors, the JPEG standard specifies the minimum coding unit, known as the Minimum Coding Unit (MCU).
In the encoding and decoding processes of JPEG images, the smallest data block processed is an $8 \times s$ data block, referred to as a DU.

1.5.1.3 Discrete Cosine Transform

The JPEG (Joint Photographic Experts Group) algorithm is a widely adopted two-dimensional discrete cosine transform (DCT) technique that operates on 8×8 sub-blocks. In the initial step, the algorithm sequentially partitions the original image into multiple 8×8 sub-blocks. Within an 8×8 image block, the pixel values exhibit a relatively smooth transition, resulting in lower spatial frequencies within the image. Subsequently, the DCT is applied to each image block, enabling the concentration of energy in a few coefficients located in the upper left corner, with these coefficients typically exhibiting significantly reduced absolute values. This characteristic facilitates the subsequent compression process by facilitating the removal of high-frequency components and preserving only the essential image information.

Considering the primary focus of this thesis on transforms for image coding, Chapter 2 will provide a comprehensive explanation of the transform stage.

1.5.1.4 Quantisation

After the DCT transform, the low-frequency components are concentrated in the upper left corner, and the high-frequency components in the lower right corner. Quantification is to discard the information that has little effect on the visual effect under the premise of maintaining a certain quality. The linear uniform quantizer is used in the JPEG standard. The quantification process is to divide 64 DCT coefficients by quantification step size and rounding. The frequency component is kept, and the high-frequency component is suppressed by quantification processing. That is to say, the compression ratio can be further improved by using fine quantification for Y and coarse quantification for C_bC_r. In decoding, inverse quantification is used, that is, the value to be processed is multiplied by the corresponding position value of the corresponding quantification table.

1.5.1.5 Zigzag scan

In order to ensure that low-frequency components appear first, high-frequency components appear afterward to increase the number of continuous "0" in the run length, and the Alternating Current (AC) coefficient of the other 63 elements of 8×8 except the DC coefficient $F(0,0)$, the "Zigzag" arrangement method is used, and then run-length encoding is performed.

1.5.1.6 Run-length coding

The principle of run-length coding: The neighboring pixels with the same color value in a row are replaced with a count value and the color value. When the data is quantized, many generated "0" can describe their length with only one data.

1.5.1.7 Differential coding in the intermediate format

Since the DC coefficients of the two adjacent 8x8 blocks are very small, differential coding Differential Predictive Coding Modulation (DCPM) is used to increase the compression ratio.

1.5.1.8 Huffman coding

After getting the middle format, the number of parentheses in the example above is encoded by Huffman.

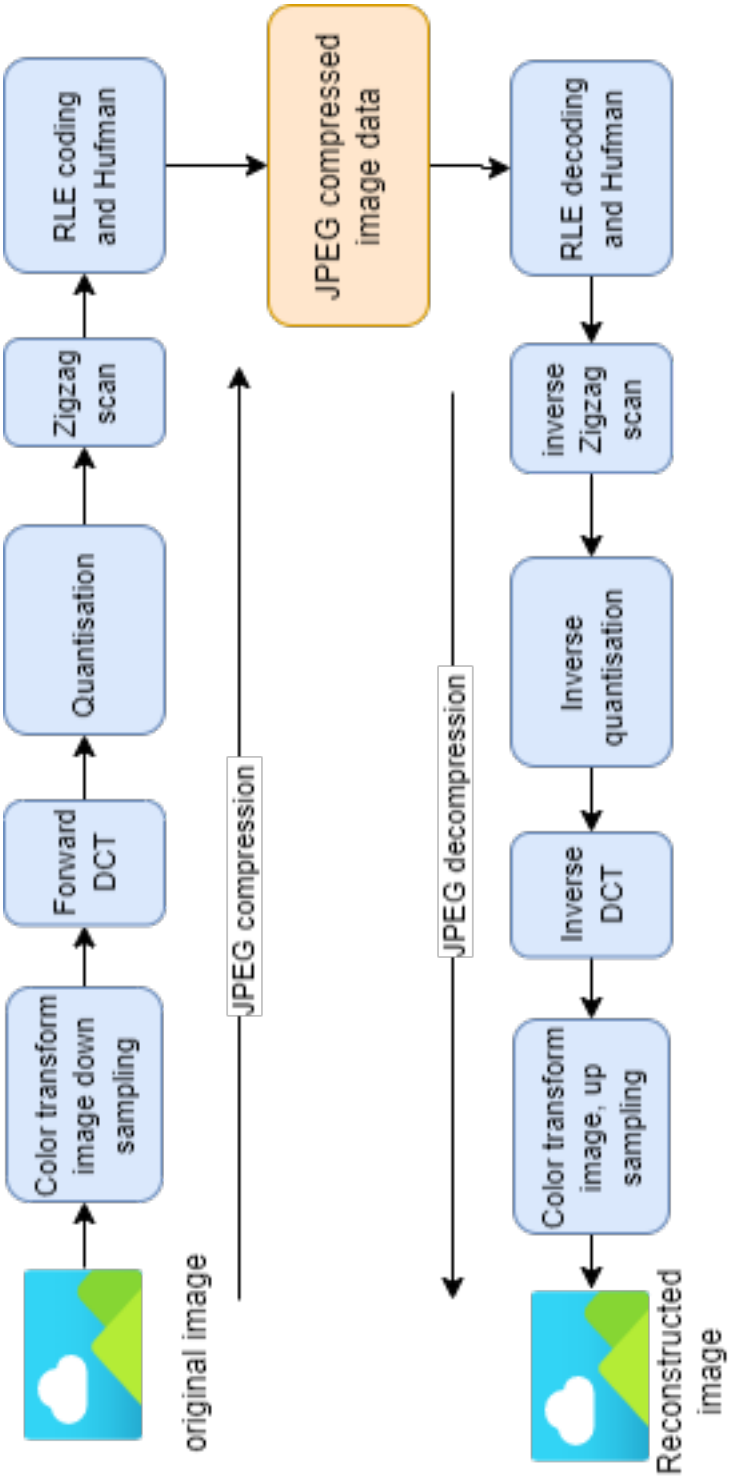


Fig. 1.5: JPEG compression/decompression algorithm

1.5.2 JPEG 2000

In the JPEG2000 standard, the block-based DCT of JPEG has been replaced by the full-frame Discrete Wavelet Transform (DWT). The DWT inherently provides a multiresolution image representation, and it also improves compression efficiency because of good energy compaction and the ability to decorrelate the image across a larger scale [66].

Fig. 1.6 illustrates the core elements of a JPEG 2000 encoder, which include preprocessing, discrete wavelet transform, quantisation, encoding. The subsequent sections of the JPEG 2000 book provide a thorough examination of each of these components, delving into their intricate details and functionalities [67].

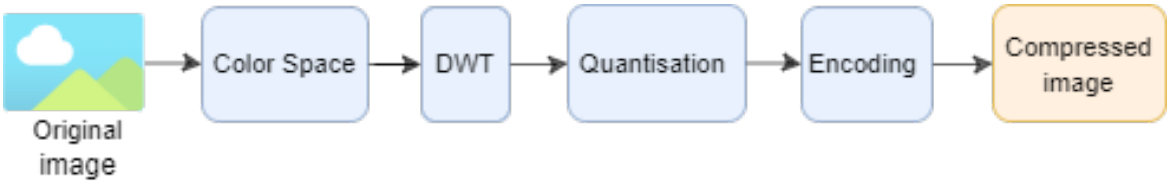


Fig. 1.6: JPEG2000 encoder algorithm

1.5.2.1 The preprocessing stage

The preprocessing stage in JPEG 2000 involves:

(i) Dividing the input image into non-overlapping tiles of equal size, which allows for independent compression of each tile using specific compression parameters.

(ii) Next, unsigned sample values in each component are level-shifted (DC offset) by subtracting a fixed value of 2^{B-1} from each sample to make its value symmetric around zero.

(iii) Finally, the level-shifted values can be subjected to a forward transformation to decorrelate the color data. One restriction on applying the intercomponent transformation is that the components must have identical bit-depths and dimensions. One transform is the Irreversible Color Transform (ICT), which is identical to the traditional RGB to YC_bC_r color transformation and can only be used for lossy coding. The other transform is the reversible color transform reversible color transform (RCT), which is a reversible integer-to-integer transform that approximates the ICT for color decorrelation and can be used for both lossless and lossy coding.

During the decoding process, the decompressed image is subjected to the inverse color transform, if required, to restore the original color representation. Subsequently, the DC level shift, applied during encoding, is removed from the image, ensuring its fidelity to the original data.

1.5.2.2 The Discrete Wavelet Transform

In JPEG 2000, the full-frame discrete wavelet transform (DWT) replaces the block discrete cosine transform (DCT) transformation used in baseline JPEG.

The discrete wavelet transform (DWT) possesses several characteristics that align with the requirements established by the JPEG 2000 committee. One key attribute is its inherent ability to provide a multiresolution representation of an image. Additionally, the full-frame nature of the DWT allows for decorrelation across a broader scale, effectively eliminating blocking artefacts, particularly at high compression ratios. Another advantage lies in the use of integer DWT filters, enabling the integration of both lossless and lossy compression within a single compressed bitstream.

1.5.2.3 Quantisation

In the JPEG 2000 standard, the quantization step is performed on each subband individually. The user selects a quantizer step size, for each subband, and this step size is applied to quantize all the coefficients within that subband. The determination of the quantizer step size can be guided by the perceptual importance of each subband, taking into account information derived from the Human Visual System (HVS).

1.5.2.4 Entropy Coding

In the JPEG 2000 standard, the quantizer indices that correspond to the quantized wavelet coefficients in each subband undergo entropy encoding to generate the compressed bit-stream. The selection of the entropy encoder is influenced by multiple factors, one of which is the need to create an embedded bit-stream. This is achieved through the bit-plane encoding of the quantiser indices. The technique of bit-plane encoding, which involves encoding the wavelet coefficients in successive bit-planes, has been employed by established embedded wavelet encoders like EZW.

Despite its superior coding performance and a range of novel features, JPEG 2000 has long been regarded as less suitable for consumer market applications

due to its significantly higher algorithmic complexity compared to its predecessor, JPEG.

1.6 The performance assessment

The quality evaluation measures the amount of distortion brought by an image compression algorithm. An image compression standard can be considered better than another one if it gives reconstructed images of better quality for the same compression rate, or conversely a lower rate for the same quality of image reconstruction. The most obvious way of evaluating the quality of image reconstruction is to use subjective experiments, as defined in [68]. In these approaches, a panel of human observers assesses the quality of reconstructed images. Then, the results are averaged as a mean opinion score. This way, an estimation of the perceived distortion is obtained. These methods remain the most reliable, yet are generally impractical as a significant number of viewers are needed over a long time. In practice, objective metrics enable to evaluate quickly an image compression algorithm with many different parameters on large sets of images. Below are presented three objective metrics widely used in image compression: PSNR, Structural SIMilarity (SSIM), and Percentage Energy Error Norm (PEEN)

1.6.1 Peak Signal-to-Noise Ratio

The PSNR is a mathematical measure of image quality based on the pixel difference between two images. PSNR is defined as:

$$PSNR = 10.log_{10}.\frac{MAX^2}{MSE} \tag{1.1}$$

For 8-bit depth images, which are the main format considered in this thesis, the maximum pixel values write $MAX = 2^8 - 1 = 255$.

Mean Square Error (MSE) is computed by averaging the squared intensity of the original image $X(m,n)$ and the resultant image pixels $Y(m,n)$.The measure between two windows of size $N \times N$ is:

$$MSE = \frac{1}{N.N}\Sigma_{n=1}^{n}\Sigma_{m=1}^{n}(X(m,n) - Y(m,n))^2 \tag{1.2}$$

1.6.2 Structural Similarity Index Measure

The Structural Similarity Index SSIM is a full reference metric, in other words, the measuring of image quality based on an initial uncompressed or distortion-free image as a reference:

$$SSIM(X,Y) = \frac{(2\mu_X.\mu_Y + c_1)(2cov_{XY} + c_2)}{(\mu_X^2 + \mu_Y^2 + c_1)(\sigma_X^2 + \sigma_Y^2 + c_2)} \tag{1.3}$$

where μ_X is the average of X; μ_Y is the average of Y; σ_X the variance of X; σ_Y the variance of Y ; cov_{XY} the covariance between X and Y; $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilise the division with weak denominator; L the dynamic range of the pixel values.; $k_1 = 0.01$ and $k_2 = 0.03$, these values are used in [69], and the performance of the SSIM index algorithm is fairly insensitive to variations of these values [69]. The SSIM is in the range between 0 and $+1$. The closer the result is to $+1$, the higher the similarity.

1.6.3 Percentage Energy Error Norm

Percentage Energy Error Norm,PEEN, is a measure of the distortion (or relative distance) between the original image and the reconstructed image after the decompression process [25].

$$PEEN = \sqrt{\frac{\Sigma_{m=1}^{N}\Sigma_{n=1}^{N}(X(m,n) - Y(m,n))^2}{\Sigma_{m=1}^{N}\Sigma_{n=1}^{N}(X(m,n)^2)}} \tag{1.4}$$

1.6.4 Absolute Percentage Error

An extra measure is considered for a more idea of the resulting quality, i.e., the Absolute percentage error (APE) relative to the DCT. This last one represents relative errors in percentage for each metric cited above. For example, the APE for PSNR is calculated according to the following formula:

$$APE(PSNR) = \frac{(PSNR_e - PSNR_T)}{PSNR_e} \tag{1.5}$$

$PSNR_e$ and $PSNR_T$ are the values of PSNR considering the exact DCT and a given approximation T, respectively. The values of APE (SSIM) and APE (PEEN) are calculated similarly.

1.6.5 Transform efficiency

The efficiency for an approximate transform C is given by the following formula [70] :

$$\eta(C) = \frac{\sigma_{i=1}^{8}|r_{i,j}|}{\sigma_{i=1}^{8}\sigma_{j=1}^{8}|r_{i,j}|}.100 \tag{1.6}$$

where $r_{i,j}$is the (i,j) the entry of R_y . $R_y = C.R_x.C^t$, R_x is the covariance if x whose elements are given by $\rho^{|i-j|}$, $i,j = 1,2,...,8$.

1.6.6 Total error energy

The spectral similarity between the conventional DCT and one of its approximations is measured by Total Error Energy ϵ_{total}. For a considered DCT approximation matrix C_a , it is given by [71]:

$$\epsilon_{total} = \pi||C_e - C_a||_F^2 \tag{1.7}$$

Note that $||.||_F$ is the Frobenius norm, and C_e is the exact DCT matrix.

1.6.7 Calculation time

The time constraint is an essential factor in evaluating the performance of any compression method, it comes down to calculating the time taken by the compression and decompression of images. This constraint is imposed depending on the application targeted by the compression (transmission or archiving). Indeed, it would be unfortunate, in a transmission application, if the time saved by reducing the size of the data to be transmitted were less than the time spent on compression/ decompression, however, this quality (time constraint) will be less crucial in applications aimed at archiving data.

1.7 Conclusion

In this chapter, we have explained the basic concept of image compression, discussing the two categories of image compression algorithms: lossless and lossy. we have also explained the process of transformation, quantization, and encoding in transform-based compression algorithms.The popular transforms used in image compression, such as Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), and Discrete Wavelet Transform (DWT) have been also introduced. Our work is devoted to JPEG encoders and to their enhancement based on the integer approximations of DCT (detailed in Chapter 3).

Chapter 2

Integer transforms and fast algorithms

2.1 Introduction

In the previous Chapter, transforms were identified as an important part of current coding standards. This chapter studies the design and properties that make transforms useful in signal coding. A transform is a mathematical transfer function of a signal from one representation domain to another. The high energy compaction offered by the transformation process has led this technique to be part of all the international signal coding standards [72]. Transforms allow reducing existing signal correlations in the spatial domain, leading to a more decorrelated signal in the transform domain and ensuring a more compact representation. This is of great importance for the upcoming stages of scanning and entropy coding. Transforms can be very abstract since they tend to work in N-dimensional spaces, where N represents the number of residual samples processed by the transform. Typical values vary from $N = 4 \times 4$ to $N = 32 \times 32$ using powers of two in modern video codecs, such as HEVC.

Among the discrete transforms: the Karhunen–Loève transform (KLT) is considered as the optimum transformation for data decorrelation [73]. However, the KLT transformation kernel depends on the input data statistical properties, which implies that a fast algorithm cannot be easily attainable [74], Discrete Fourier Transform (DFT) [60], DCT emerged as an alternative to the KLT computation [73], Discrete Sine Transform (DST) and the DWT [75].

2.2 Block transforms

Block-based coding is widely adopted in image/video systems, such as JPEG [76] H.264/MPEG-4 AVC [77, 78] and HEVC [79]. In these systems, the image to be transformed is split into non-overlapping blocks, and each one is treated and transformed independently [80]. This provides the advantage of being less expensive in terms of computing than other kinds of transforms, such as wavelet transforms used in JPEG-2000 [81]. Since JPEG [76] uses DCT block transform only, they will be used in this thesis to build up new systems.

2.3 Orthogonal transforms

Transforms used in image processing and video coding systems are orthogonal. Orthogonal matrices are square matrices whose rows and columns are orthogonal unit vectors, also known as orthonormal vectors, with [72]:

$$A^t A = A A^t = I \tag{2.1}$$

As a consequence, the inverse matrix of an orthogonal matrix is its transposed version:

$$A^t = A^{-1} \tag{2.2}$$

where A is a square matrix, A^t is the transpose of matrix A and A^{-1} is its inverse. This property offers some benefits:

(i) Fast computation of inverse transform with no need to store it separately.

(ii) Re-use of fast algorithms for both direct and inverse transform applications.

(iii) Energy preservation.

2.4 Separability

Image and video coding deal with image blocks, which are two-dimensional signals, and, consequently, use transforms able to process those signals. The straightforward approach to work with those signals is to use non-separable transforms. These transforms take the residual samples from a block previously reshaped into a single-dimensional signal. For instance, a block 4×4 becomes a 16×1 vector. Afterward, the transform is applied normally [72]:

$$X = A.x \tag{2.3}$$

where X is a $N \times N$ block, reshaped into a $N^2 \times 1$ vector and A is a $N^2 \times N^2$ matrix. The main disadvantage of this approach is the number of calculations required to obtain the transformed signal: for a $N \times N$ block, the number of operations required to transform it in a non-separable way is N^4 multiplications and $N^2(N^21)$ additions.

Due to the high number of operations needed to transform a block using non-separable transforms, separable transforms are widely used in image and video coding. A block is transformed separately using horizontal and vertical transforms A_h and A_v for its rows and columns, respectively, as:

$$Y = A_v.(A_h.X^t)^t = A_v.X.A_h^t \tag{2.4}$$

The operation inside the parenthesis transforms the rows of x, and the outer part, the columns of the result. By performing the horizontal and vertical transforms separately, the number of operations required is reduced to $2N^3$ multiplications and $2N^2(N-1)$ additions.

In the context of signal processing, the discrete cosine transform (DCT) serves as a method to convert a signal into its constituent frequency components. This technique finds extensive application in the field of image compression. In the next section, we present a set of straightforward functions designed to calculate the DCT and facilitate image compression.

2.5 Discrete Cosine Transform

The Discrete Cosine Transform was first introduced in 1974 by Ahmed and al. [82]. It is an orthogonal transformation, which has a fixed set of basis functions, an efficient algorithm for its computation, and good energy compaction and correlation reduction properties. Like other transformations, the DCT attempts to de-correlate data from a signal. There are several variants of DCT with slightly modified definitions and properties, such as DCT type I, II, III, IV, V-VIII [59]. Among the different versions of DCT, type II and type III (inverse DCT) have received a lot of attention in digital signal processing. In the following sections, we review the DCT transform for both cases: one-dimensional and two-dimensional.

2.5.1 One-dimentional DCT

The formal definition of the one-dimensional (1-D) N-point DCT of N input samples: $x(n) = x(0), \ldots, x(N-1)$ is given by:

$$y(u) = \sqrt{\frac{2}{N}}\alpha(u)\sum_{n=0}^{N} x(n) cos \frac{\pi(2n+1)u}{2N} \tag{2.5}$$

with $u = 0, 1, \ldots, N-1$, and $\alpha(u)$ is defined as follows:

$$\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if} \quad u = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.6}$$

Thus, the first transform coefficient is the average value of the sample sequence. In literature, this value is referred to as the DC coefficient. All other transform coefficients are called the AC coefficient.

$$y(u = 0) = \frac{1}{N}\sum_{n=0}^{N} x(n) \tag{2.7}$$

The inverse one-dimensional DCT (IDCT 1-D) is defined as given in the following equation:

$$x(n) = \sqrt{\frac{2}{N}}\sum_{n=0}^{N} \alpha(u) y(n) cos \frac{\pi(2n+1)u}{2N} \tag{2.8}$$

The one-dimensional N-point DCT presented in equation 2.5 can be expressed in matrix form as:

$$\mathrm{y} = \mathrm{C.x} \tag{2.9}$$

where $\mathrm{x} = [x_0, x_1, \ldots, x_{N-1}]^t$ is an input vector of size $N \times 1$, $\mathrm{y} = [y_0, y_1, \ldots, y_{N-1}]^t$ is the transformed (output) vector of size $N \times 1$, and C is a $N \times N$ matrix called the kernel matrix of the transformation, whose basis vectors are cosines sampled, given by:

$$C_{i,j} = \sqrt{\frac{2}{N}}\alpha(i).cos \frac{\pi(2j+1)i}{2N} \tag{2.10}$$

$$\mathrm{C} = \begin{bmatrix} C_{0,0} & C_{0,1} & \ldots & C_{0,N-1} \\ C_{1,0} & C_{1,1} & \ldots & C_{1,N-1} \\ \ldots & \ldots & \ldots & \ldots \\ C_{N-1,0} & C_{N-1,1} & \ldots & C_{N-1,N-1} \end{bmatrix} \tag{2.11}$$

The inverse one-dimensional N-point DCT presented in equation 2.8, can also be formulated as a matrix multiplication according to:

$$\mathrm{x} = \mathrm{C}^{\mathrm{t}}.\mathrm{y} \tag{2.12}$$

By referring to the equation 2.8, it is possible to deduce therefrom a $N \times N$ matrix of the transform which, multiplied by a signal, provides the coefficients of this same signal in the domain of the transform. If this matrix is applied to the two dimensions of an image, i.e. once in the direction of the rows and once in the direction of the columns, the two-dimensional version of this transform is obtained. Equation 2.13 illustrates the DCT-II matrix for $N = 8$.

$$\mathrm{C} = \frac{1}{2}\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ cos\frac{\pi}{16} & cos\frac{3\pi}{16} & sin\frac{3\pi}{16} & sin\frac{\pi}{16} & -sin\frac{\pi}{16} & -sin\frac{3\pi}{16} & -cos\frac{3\pi}{16} & -cos\frac{\pi}{16} \\ cos\frac{\pi}{8} & sin\frac{\pi}{8} & -sin\frac{\pi}{8} & -cos\frac{\pi}{8} & -cos\frac{\pi}{8} & -sin\frac{\pi}{8} & sin\frac{\pi}{8} & cos\frac{\pi}{8} \\ cos\frac{3\pi}{16} & -sin\frac{\pi}{16} & -cos\frac{\pi}{16} & -sin\frac{3\pi}{16} & sin\frac{3\pi}{16} & cos\frac{\pi}{16} & sin\frac{\pi}{16} & -cos\frac{3\pi}{16} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ sin\frac{3\pi}{16} & -cos\frac{\pi}{16} & sin\frac{\pi}{16} & cos\frac{3\pi}{16} & -cos\frac{3\pi}{16} & -sin\frac{\pi}{16} & cos\frac{\pi}{16} & -sin\frac{3\pi}{16} \\ sin\frac{\pi}{8} & -cos\frac{\pi}{8} & cos\frac{\pi}{8} & -sin\frac{\pi}{8} & -sin\frac{\pi}{8} & cos\frac{\pi}{8} & -cos\frac{\pi}{8} & sin\frac{\pi}{8} \\ sin\frac{\pi}{16} & -sin\frac{3\pi}{16} & cos\frac{3\pi}{16} & -cos\frac{\pi}{16} & cos\frac{\pi}{16} & -cos\frac{3\pi}{16} & sin\frac{3\pi}{16} & -sin\frac{\pi}{16} \end{bmatrix} \tag{2.13}$$

Each row of C can be seen as a basis function $C_u(n), n = (0, 1, ..., 7)$, and the figures of these functions are shown in Fig. 2.1.

As we can see, the top-left waveform $(u = 0)$ is simply a constant, whereas other waveforms $(n = 1, 2, ..., 7)$ show the behavior at progressively higher frequencies [83]. These waveforms are called cosine basis functions, which are orthogonal and independent [84]. In accordance with our previous description, the DC coefficient $C(0)$ is the average value of $x(n)$.

2.5.2 Bidimentional DCT

The DCT can be extended to the transformation of 2-D signals or images. This can be achieved in two steps: by computing the 1-D DCT of each of the individual rows of the two-dimensional image, and then computing the 1-D DCT of each column of the image. Let $X(n, m]$ an input block of size $N \times N$. The 2-D DCT of $X(n, m]$ defined as the output matrix of transformation domain of size $N \times N$ given by the following

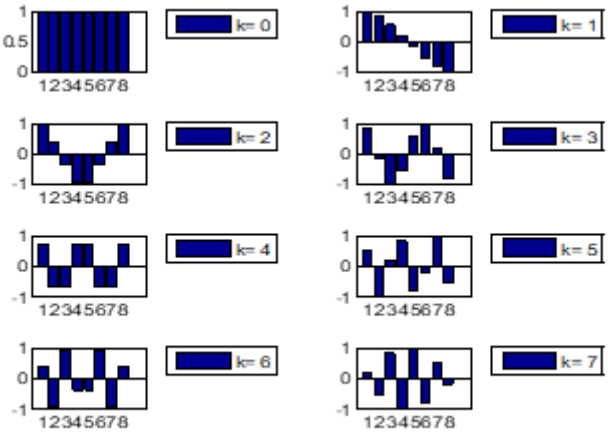


Fig. 2.1: Basis functions of 1-D DCT($N = 8$)

equation:

$$Y(u,v) = \frac{2}{N}\alpha(u)\alpha(v)\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}X(n,m)cos\frac{(2n+1)u\pi}{2N}cos\frac{(2m+1)v\pi}{2N} \tag{2.14}$$

where the scale factors $\alpha(u)$ and $\alpha(v)$ equal to:

$$\alpha(u),\alpha(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if} \quad u,v = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.15}$$

Similarly, the two-dimensional inverse DCT (2-D IDCT) of $Y(u,v)$ is defined as :

$$X(n,m) = \frac{2}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\alpha(u)\alpha(v)Y(n,m)cos\frac{(2n+1)u\pi}{2N}cos\frac{(2m+1)v\pi}{2N} \tag{2.16}$$

for $n,m = 0,1,\ldots,N-1$.
Applying the DCT to a 8×8 block gives 64 coefficients. The first DCT coefficient, $Y(0,0)$, is the DC coefficient. This coefficient corresponds to the average value of the input sequence and represents the roughest details of the block (lowest spatial frequency). The remaining coefficients are called AC coefficients. The AC coefficients represent the finer details of the block (higher spatial frequencies). We can represent the frequency distribution of the DCT of a 8×8 block as shown in Fig. 2.2.

All the information in the image is in the low frequencies. On the other hand, the details of the image are localised in high frequencies.

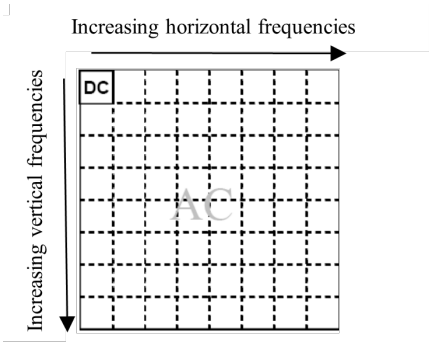


Fig. 2.2: Frequency distribution of 1-D DCT($N = 8$)

In equation 2.16:

$$C(u,v) = \alpha(u)\alpha(v)cos(\frac{(2n+1)u\pi}{2N}cos\frac{(2m+1)v\pi}{2N} \tag{2.17}$$

$C(u,v)$ are the basis functions of 2-D DCT. These basis functions can be generated by multiplying the horizontally oriented set of cosine basis functions with the vertically oriented set of cosine basis functions:

$$C(u,v) = C_u C_v^t \tag{2.18}$$

$$Y(u,v) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} X(n,m))C(u,v) \tag{2.19}$$

In the case of $N = 8$, these functions are shown in Fig. 2.3. The functions located at the top left represent the low frequencies of the transform, the spatial frequencies increase as one moves towards the lower right corner of the block.

The DCT matrix in equation 2.13 cannot be applied optimally in its native form. In addition to the precision required to perform the calculations, a DCT-II matrix of dimension N requires N^2 multiplications and $N(N-1)$ additions for a total of $2N^2$ elementary arithmetic operations. It is therefore necessary to perform simplifications, rounding and optimisations to consider a more efficient hardware implementation of the DCT.
Section 2.5.3 presents some alternatives with a view to simplifying the calculation of the DCT.

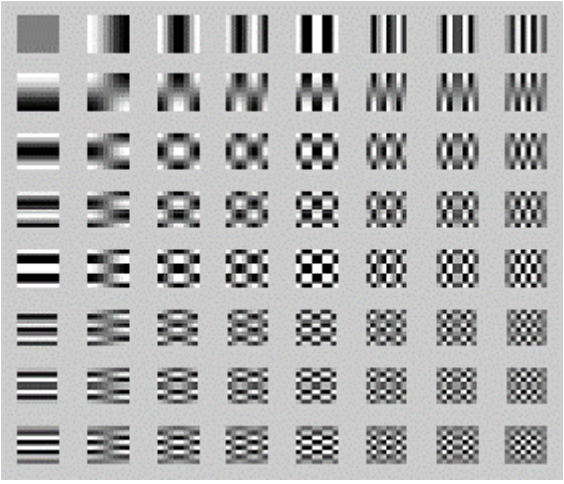


Fig. 2.3: Basis functions of 2-D DCT ($N = 8$)

2.5.3 Fast algorithms of DCT

The process of computing the DCT can be intricate and requires significant time and resources within image and video compression standards. Nevertheless, this step holds immense importance as it serves to distinguish between the low and high frequencies present in the image. Thus, despite its complexity, the DCT plays a crucial role in effectively compressing images and videos by separating their frequency components.

As a result, the calculation of the traditional 8-point DCT required 64 multiplications and 56 additions, thus these numbers are very great. The development of fast DCT algorithms is therefore an essential task for a successful hardware implementation.

The literature is very rich in fast DCT algorithms, a review of these algorithms is presented with citations. Chen et al. [23] is the first published with 16 multiplications and 26 additions. Later, several algorithms were published [22, 85, 86, 24], most of which resulted in 12 multiplications and 29 additions. Wahid et al. [24] minimize this number into 5 multiplications and 29 additions. All the fast algorithms still require floating-point multiplications, which are slow in both hardware and software implementations.

2.6 Discrete Tchebichef Transform

Despite being adopted for image compression, DCT is not the only option for transform encoding method [87]. JPEG-like compression schemes based on the Discrete Tchebichef Transform (DTT) [88] have been demonstrated to be realistic alternatives

[89, 90]. Discrete Tchebichef Transform (DTT) is an integer orthogonal transform that has been derived from the Discrete Tchebichef Polynomials (DTP) [91].

2.6.1 One-dimentional DTT

Discrete Tchebichef Transform is increasingly used in the literature because of its high coding gain, transform efficiency, and low computational complexity. The DTT of an input signal $x(n) = [x_0, x_1, \ldots, x_{N-1}]^t$ is the output transform domain signal $y(n) = [y_0, y_1, \ldots, y_{N-1}]^t$ given by:

$$y = C_T.x \tag{2.20}$$

where C_T is the Tchebichef transformation matrix, whose entries are given by [89].

$$C_T = \tilde{t}_k(n) \tag{2.21}$$

$n, k = 0, 1, 2, \ldots, N-1$ and $\tilde{t}_k(n)$ is the normalized k^{th} order discrete Tchebichef polynomial. Because such polynomial class represents a set of orthogonal polynomials [**?**], we have that $C_T^{-}1 = C_T^t$ The 8-point DTT matrix is given as:

$$C_T = D_T. \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -7 & -5 & -3 & -1 & 1 & 3 & 5 & 7 \\ 7 & 1 & -3 & -5 & -5 & -3 & 1 & 7 \\ -7 & 5 & 7 & 3 & -3 & -7 & -5 & 7 \\ 7 & -13 & -3 & 9 & 9 & -3 & -13 & 7 \\ -7 & 23 & -17 & -15 & 15 & 17 & -23 & 7 \\ 1 & -5 & 9 & -5 & -5 & 9 & -5 & 1 \\ -1 & 7 & -21 & 35 & -35 & 21 & -7 & 1 \end{bmatrix} \tag{2.22}$$

$$C_T = D_T.T_T \tag{2.23}$$

where,

$$D_T = 1/2.diag(1/\sqrt{2}, 1/\sqrt{42}, 1/\sqrt{42}, 1/\sqrt{66}, 1/\sqrt{142}, 1/\sqrt{546}, 1/\sqrt{66}, 1/\sqrt{858}) \tag{2.24}$$

Then, the inverse transformation is given by:

$$x = C_T.y \tag{2.25}$$

thus, the matrix D_T does not add any computational overhead. Hence, the only source of computational complexity is the matrix T_T.

2.6.2 Two-dimentional DTT

The two-dimensional DTT (2-D DTT) of an input matrix X is the output matrix, Y, given by:

$$Y = C_T.X.C_T \tag{2.26}$$

The inverse 2-D DTT is furnished by :

$$X = C_T.Y.C_T \tag{2.27}$$

Many researchers have developed DTT-based image compression and enhanced DTT-based image compression techniques. In image compression, the DTT is applied to the forward and inverse transforms. As a result, most studies focus on improving DTT's performance in image compression by modifying the size of image blocks processed by DTT or simplifying DTT operation.

2.6.3 Fast algorithms of DTT

However, the exact DTT has a high arithmetic complexity because of the number of additions and float-point multiplications that it involves. It is well known that such multiplications need more computational power than additions or fixed-point multiplications, both in terms of software and hardware. As a result, the DTT's high computation complexity prevents it from being used in systems with low power consumption [92, 93], or in real-time processing, such streaming media [94, 95]. Thus, the DTT's computational efficiency might be increased by using quick algorithms.

Although these rapid methods have reduced arithmetic difficulties as compared to direct DTT calculations, they nevertheless have a high arithmetic complexity, necessitating a substantial number of additions and bit-shifting operations.

Nakagaki et al.[96] suggested a 4×4 forward DTT method that is fast, required only 32 multiplications and 66 additions compared to previous methods took full the two advantages of DTT's properties: separability and symmetry, requiring 64 multiplication operation and 96 additions.

The procedure described in [97] requires just 32 multiplications and 66 additions. It significantly decreases the algorithm's complexity and, as a result, the transformation time for images of varied resolution. For processing the 44-picture block, Senapati et al. [36] suggested a novel image compression technique. To simplify

processing, the multiplication terms in its algorithm are replaced with shift and addition operations. As a result, the non-multiplicative operation is realized, and the forward and inverse DTT transformation speeds are increased. However, in his paper [36], he mentioned and used a novel rearrangement approach, zigzag prune, to replace the standard block prune. In terms of reconstructed image quality, the results show that the zigzag prune outperforms the classical block prune.

Ishawar described DTT's non-multiplicative operation in even more detail in [98]. A fast algorithm for the exact DTT matrix was derived in [88], requiring 44 additions and 29 bit-shifting operations. Such arithmetic complexity is considered excessive, when compared to state-of-the-art discrete transform approximations, which generally require less than 24 additions [3, 38].

2.7 Integer DCT approximations

Image, video, and audio digital signals are represented by integer values. The floating-point DCT converts these integer values to real coefficients. Although the fast algorithms of the DCT such as [22, 24, 23, 99, 100], significantly reduce the number of their arithmetical operations and required floating-point operations. They still make hardware and software implementations very slow, require a lot of memory space and consume too much electrical energy. To remedy these problems, the DCT coefficients are approximated by integers, so the floating-point multiplication is replaced by integer multiplication [38, 101, 102, 29, 28, 31, 35].

A scheme of approximation of an integer transform should have the following features [103]:

(i) It should have low computational complexity.

(ii) It should have low error energy to provide compression performance close to the exact transform, and preferably should be orthogonal.

(iii) It should work for higher lengths of transform to support modern video coding standards, and other applications like tracking, surveillance, and simultaneous compression and encryption.

The main objective of DCT approximations is to eliminate the floating-point multiplication operations, which consumed many times for its computation and a lot of materials for their implementations. To analyse the image compression based on DCT approximations, an overview of DCT approximations methods is mainly presented in this section.

2.7.1 Signed DCT

The approach in [25] is simple, it involves the application of the "$sign$" function operator to the forward DCT matrix. The resulting matrix transform, termed as SDCT, has good energy compaction and decorrelation properties, 24 additions required for its computation, with the inputs $\{0, 1, -1\}$. Applying the Signum function to the DCT matrix gives this simple matrix:

$$T_N^{SDCT}(i,j) = \frac{1}{\sqrt{N}} sign\{T_N(i,j)\} \tag{2.28}$$

where "$sign$" means the $Signum$ function, defined by:

$$sign\{x\} = \begin{cases} +1 & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x = 0 \\ -1 & \text{if} \quad x < 0 \end{cases} \tag{2.29}$$

The application of the $signum$ function to the DCT matrix gives the simple matrix, T_1, in 2.30. It can be verified that the transform matrix T_1 is not orthogonal, i.e. the number of non-zero off-diagonal elements of $T_1 \times T_1^t$ is 12%. So different fast algorithms are used to calculate the forward and inverse SDCT.

$$T_1 = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \end{bmatrix} \tag{2.30}$$

2.7.2 Bouguezel et al. DCT Approximation series

2.7.2.1 BAS-2008-a

Bouguezel et al. introduce in [7] a low complexity transform for image compression, by properly replacing some elements of SDCT transform proposed in [25] by zeros. It is the most recent version of 8-point DCT-II. It is orthogonal,

i.e.$T_2^{-1} = T_2^t$, required 18 additions and 2 bit-shifts. It is given by :

$$T_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & 1/2 & -1/2 & -1 & -1 & -1/2 & 1/2 & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1/2 & -1 & 1 & -1/2 & -1/2 & 1 & -1 & 1/2 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.31}$$

2.7.2.2 BAS-2008-b

With the same approach introduced in [7], Bouguezel et al. proposed a new version in [104], the resulting transform is not-orthogonal, thus the same fast algorithm is not used for its computational. The matrix of this transform required 21 additions and 3-bit shifts. It is given by:

$$T_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{2.32}$$

2.7.2.3 BAS-2009

Bouguezel et al. implement the same idea as their previous work cited above. A transform with 18 additions is carried out [26]. This transform is less signifi-

cant at low compression ratio. It is defined as :

$$T_4 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.33}$$

2.7.2.4 BAS-2010

A new multiplication-free transform is proposed by the authors in [27], by an extension of the 4-order integer DCT. The transform matrix is orthogonal, requiring 24 additions and 4-bit shifts.

$$T_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 2 & 1 & -1 & -2 & 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 1 & -2 & 2 & -1 & 1 & -2 & 2 & -1 \end{bmatrix} \tag{2.34}$$

2.7.3 Parametric DCT transform

Bouguezel et al. proposed a new approximate DCT transform in [105] who they called parametric transform. It consists of introducing an arbitrary parameter a, in the transform reported in [104], and performing some row permutation. Its orthogonality is verified in [105].
A fast algorithm is developed in [105] to reduce the number of arithmetical operations from 36 additions and 8 multiplications to a few additions and bit shift operations. For example:
• $a = 0$, T_6 required 16 additions.
• $a = 1$, T_6 required 18 additions.

• $a = 2$, T_6 required 18 additions and 2 bit-shift.

$$T_6 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & a & -a & -1 & -1 & -a & a & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 \\ a & -1 & 1 & -a & -a & 1 & -1 & a \end{bmatrix} \tag{2.35}$$

2.7.4 Binary Discrete Cosine Transform

The authors of [28] developed a binary version of DCT called Binary Discrete Cosine Transform (BDCT) using Walsh Hadamard Transform (WHT). This transform differs from that of the SDCT by only four entries. It required 24 additions. The BDCT matrix is given by:

$$T_7 = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \end{bmatrix} \tag{2.36}$$

2.7.5 DCT approximations based on integer functions

The authors in the paper [29] propose a collection of DCT approximations (twelve transforms). Their idea involves applying a series of integer functions to the forward floating points DCT matrix. These functions include floor, ceiling, truncation, and rounding-off. The resulting matrices are orthogonal or quasi-orthogonal, with a computational complexity ranging from 18 to 24 additions. The two important orthogonal DCT approximations of this collection are given:

$$T_{4\_cintra} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & -1 & 1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & -1 & 1 & -1 & 1 & -1 & 1 & 0 \end{bmatrix} \tag{2.37}$$

$$T_{6\_cintra} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 0 & 0 & -1 & -1 & -2 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 1 & 0 & -2 & -1 & 1 & 2 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -2 & 0 & 1 & -1 & 0 & 2 & -1 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 0 & -1 & 1 & -2 & 2 & -1 & 1 & 0 \end{bmatrix} \tag{2.38}$$

2.7.6 DCT approximations based on angle similarity

In their study, Oliveira et al. [35] put forth a novel approach involving two transformations aimed at reducing the angle between the rows of Discrete Cosine Transforms (DCTs) and the corresponding rows of the matrix approximations.

$$T_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 0 & 0 & -1 & -2 & -2 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 1 & 0 & -2 & -2 & 2 & 2 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 2 & -2 & 0 & 1 & -1 & 0 & 2 & -2 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 0 & -1 & 2 & -2 & 2 & -2 & 1 & 0 \end{bmatrix} \tag{2.39}$$

$$T_9 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 2 & 0 & 0 & -2 & -1 & -2 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 2 & 0 & -2 & -1 & 1 & 2 & 0 & -2 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -2 & 0 & 2 & -2 & 0 & 2 & -1 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 0 & -2 & 1 & -2 & 2 & -1 & 2 & 0 \end{bmatrix} \tag{2.40}$$

2.7.7 DCT approximations based on 16-point DCT

In their research, Ezhilarasi et al. ([38]) introduce a pair of transforms derived from the 16-point Discrete Cosine Transform (DCT) matrix, utilizing fundamental elements consisting of $0, \pm 1, \pm 2$. The two proposed transforms are as follows:

$$T_{10} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 & -1 & -2 & -2 & -2 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 2 & 1 & -2 & -2 & 2 & 2 & -1 & -2 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 2 & -2 & -1 & 2 & -2 & 1 & 2 & -2 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 1 & -2 & 2 & -2 & 2 & -2 & 2 & -1 \end{bmatrix} \tag{2.41}$$

$$T_{11} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{2.42}$$

2.7.8 PADCT transform

The transform presented in [6] is based on an approach introduced in [25]. It consists of introduce the Signum function in a matrix transform. In the case of PADCT [6] the matrix is that reported in [36]. Then the resulting transform, PADCT, is quasi-

orthogonal as its original transform. The complexity of this transform is 17 additions only, but its performances in image compression application remains limited. It is given by:

$$T_{12} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.43}$$

2.7.9 Transform with 14 addition only

Bayer and Cintra in [32] proposed DCT approximation with 14 additions only, hereafter referred to as T_{13}. This method consists of replacing judiciously elements of transform matrix reported in [2] with zero.
They obtained the following matrix:

$$T_{13} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.44}$$

2.7.10 A Multiparametric Class of Low-complexity Transforms

Based on a series of research published by Bouguezel, Ahmed, and Swamy, the authors of [106] offer a new class of low-complexity 8-point DCT approximations. A multiparametric fast technique that includes both known and new transformations is also developed. After solving a multicriteria optimization task, they choose the best-performing DCT approximations and subject them to a scaling approach to get

larger size transforms.

$$T_{14} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 0 & -1 & 0 & 1 & -1 & 0 & 1 & -1 \\ 1 & -1 & 1 & 0 & 0 & 1 & -1 & 0 \\ 1 & -1 & 0 & -1 & 1 & 0 & 1 & -1 \end{bmatrix} \tag{2.45}$$

The low-complexity matrices T_i, cited above, can be modified to provide orthogonal transformations C_i by the procedure of orthogonalisation based on the polar decomposition as described in a good deal of research paper [28, 104, 7, 105]. Thus, the orthogonal DCT approximations associated to a low-complexity matrix T_i, are given by:

$$C_i = D_i.T_i \tag{2.46}$$

$$D_i = (\sqrt{T_i.T_i^t})^{-1}, \quad i = 1,2,3,\ldots,12. \tag{2.47}$$

2.8 Integer DTT approximations

The use of approximate transforms presents an alternative to exact transform computations. This approach has been effectively employed in approximating the discrete cosine transform (DCT), resulting in the development of several approximation techniques [27, 107].

Recently, the discrete Tchebichef transform (DTT) has attracted significant attention because of its efficient signal decorrelation properties, and at the same time straightforward hardware implementation.Prominent examples of approximate DTT transforms include: [3], [4] and [108] respectively.

2.8.1 Near-orthogonal DTT approximation

Oliveira et al. [3] suggested a low-complexity near-orthogonal 8-point DTT approximation suitable for image and video coding. A quick technique for the suggested DTT approximation that involves just 24 additions and 6 bit-shifting operations was also provided. The suggested approximation has an additive arithmetic cost that is 45.5% and 2.05% less than the conventional DTT fast technique and DTT approximation in [109], respectively. In terms of video coding, the suggested technique

produced results that were almost identical to the approximation in [109]. The suggested design might excel in ensuring superior operation frequency and very low power consumption, as well as low computational complexity while keeping good coding performance, therefore it could be employed in an expanding wide range of low power hand-held devices.

$$T_{T1} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ -2 & -1 & -1 & 0 & 0 & 1 & 1 & 2 \\ 2 & 0 & -1 & -1 & -1 & -1 & 0 & 2 \\ -2 & 1 & 2 & 1 & -1 & -2 & -1 & 2 \\ 1 & -2 & 0 & 1 & 1 & 0 & -2 & 1 \\ -1 & 2 & -1 & -1 & 1 & 1 & -2 & 1 \\ 0 & -1 & 2 & -1 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -2 & 1 & 0 & 0 \end{bmatrix} \tag{2.48}$$

$$C_{T1} = D_{T1}.T_{T1} \tag{2.49}$$

$$centering D_{T1} = diag(1/\sqrt{32}, 1/\sqrt{12}, 1/\sqrt{12}, 1/\sqrt{20}, 1/\sqrt{12}, 1/\sqrt{14}, 1/\sqrt{12}, 1/\sqrt{10}) \tag{2.50}$$

2.8.2 Orthogonal DTT Approximation

In [4], Farsiani et al. introduce a novel compression strategy based on the discrete Tchebichef transform. When compared to the original transform, a new integer transformation matrix reduces hardware complexity by up to 74. The transform in [4]:

$$T_{T2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -2 & -2 & -1 & 0 & 0 & 1 & 2 & 2 \\ 3 & 0 & -1 & -2 & -2 & -1 & 0 & 3 \\ -2 & 1 & 2 & 0 & 0 & -2 & -1 & 2 \\ 1 & -2 & -1 & 2 & 2 & -1 & -2 & 1 \\ -1 & 2 & -2 & -1 & 1 & 2 & -2 & 1 \\ 0 & -1 & 2 & -1 & -1 & 2 & -1 & 0 \\ 0 & 1 & -2 & 0 & 0 & 2 & -1 & 0 \end{bmatrix} \tag{2.51}$$

$$C_{T2} = D_{T2}.T_{T2} \tag{2.52}$$

$$D_{T2} = diag(1/\sqrt{8}, 1/\sqrt{12}, 1/\sqrt{28}, 1/\sqrt{18}, 1/\sqrt{20}, 1/\sqrt{20}, 1/\sqrt{12}, 1/\sqrt{10}) \tag{2.53}$$

2.9 Conclusion

This chapter is dedicated to the exploration of block transforms, specifically focusing on the DCT (Discrete Cosine Transform) and DTT (Discrete Tchebichef Transform) as the two investigated transform design methodologies. We shed light on the key characteristics that render these transforms suitable for various image and video compression standards. Additionally, we delve into the presentation of integer approximations, which further enhance the applicability of these transforms.

Part II

CONTRIBUTIONS

Chapter 3

Proposed enhanced JPEG encoding standard

3.1 Introduction

The primary focus of this chapter revolves around the development of novel integer DCTs (Discrete Cosine Transforms) aimed at enhancing the compression performance of the integer DCT-based JPEG standard. The objective is to strike a balance between computation complexity and performance. To achieve this, two distinct approaches are employed, leading to the creation of four integer DCT approximations. Additionally, an appropriate quantisation matrix is proposed for these approximations.

The approximations presented in this chapter frequently utilise transform matrices with entries defined within the set $\{0, \pm 1, \pm 2\}$. Consequently, these transforms possess zero multiplicative complexity, as the necessary arithmetic operations can be exclusively implemented through additions and bit-shifting operations.

The resulting transforms have been subjected to a comprehensive objective assessment, wherein they were meticulously compared to the existing integer DCT approximations.

To ensure efficient hardware implementation of each proposed transform, a fast algorithm has been proposed. Those algorithms are specifically designed to optimise execution speed and resource utilisation in hardware architectures for each transform.

Through a detailed analysis of the performance of our transforms, it has been demonstrated that they strike a desirable balance between performance and arithmetic complexity. This means that they achieve competitive compression perfor-

mance while minimising the computational burden imposed by complex arithmetic operations. Such a compromise is essential in real-world applications where both speed and efficiency are critical factors.

It is known that a good approximation of the DCT transform must satisfy the following conditions:

(i) To have low computational complexity, the most important requirement;

(ii) The error energy of the approximation must be small in order to provide a compression performance close to the exact DCT;

(iii) It is preferred that the approximation be orthogonal;

(iv) Approximation must be able to be expanded to larger sizes to support modern video encoding standards.

The proposed transform provides a compromise between the first three requirements above, the latter can be obtained by using the scalable recursive algorithm proposed in [103]. This algorithm has resulted in more dimensional versions of the proposed matrix that are suitable for video experiments. Note that this algorithm is already used in a more recent work published in [27].

3.2 Proposed transforms based on 16-point DCT transform

In this section, a novel approach for developing integer approximation of the two-dimension 8 points conventional DCT-II is introduced. The proposed method is based on 16 points DCT-II and rounding off operations. As a result, we obtained a multiplication-free 8-point approximation of conventional DCT-II, noted T_{pa}, with the parameter $a = \{1, 2\}$, and its basic elements are $S = \{0, 1, 2\}$; no multiplication operations are required. These two transforms are the more efficient DCT approximations with a reduction in the term of complexity computation, requiring 24 additions and 2-bit shifts for $a = 1$ or 24 additions and 6-bit shifts for $a = 2$; no multiplication is required.

3.2.1 Proposed algorithm

Our proposed approach, based on 16-floating point DCT-II, consists of the following steps:

1. Generate the forward two-dimension 16-floating points DCT matrix.

2. Construct the 8×8 DCT matrix, while the elements of the base vectors with even symmetry constitute the even part of the 16-floating points DCT transform matrix.

3. Multiply the resulting matrix by the scaling factor 4.

4. For the odd rows of the resulting matrix: replace the matrix element magnitude superior to 1 by the factor a, the element's magnitude between 1 and 0.5 by the value '1', and by '0' the rest of the matrix elements.

5. For the even rows, if the element of the matrix is superior to 0.75 replace it with '2', otherwise with '1'. We found the following parametric 8-point DCT approximation.

$$T_{pa} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a & a & 1 & 0 & 0 & -1 & -a & -a \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ a & 0 & -a & -1 & 1 & a & 0 & -a \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -a & 0 & a & -a & 0 & a & -1 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 0 & -1 & a & -a & a & -a & 1 & 0 \end{bmatrix} \tag{3.1}$$

6. For $a = 1$, we get the orthogonal transform matrix T_{p1}:

$$T_{p1} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & -1 & -1 & -1 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 1 & 0 & -1 & -1 & 1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 0 & -1 & 1 & -1 & 1 & -1 & 1 & 0 \end{bmatrix} \tag{3.2}$$

7. For $a = 2$, we get the following non-orthogonal transform matrix, noted $\hat{T}_{p2}$.

$$\hat{T}_{p2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 0 & 0 & -1 & -2 & -2 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 2 & 0 & -2 & -1 & 1 & 2 & 0 & -2 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -2 & 0 & 2 & -2 & 0 & 2 & -1 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 0 & -1 & 2 & -2 & 2 & -2 & 1 & 0 \end{bmatrix} \tag{3.3}$$

3.2.2 Proposed fast algorithms

The direct computation of the proposed approximate DCT required 48 additions and 8-bit shift for the first proposed transform T_{p1} or 48 additions and 28 bit-shift operations for the second one, $\hat{T}_{p2}$. However, these numbers are very large, and therefore a fast algorithm is required. Consequently, we factorised the proposed matrices into a product of sparse matrices of very low complexity. The proposed transforms can be obtained by multiplying 3 simple matrices as follows:

$$T_{p1} = P_1 \times A_3 \times A_2 \times A_1 \tag{3.4}$$

$$\hat{T}_{p2} = P_2 \times A_4 \times A_2 \times A_1 \tag{3.5}$$

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 \end{bmatrix}, A_4 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{-1} & \mathbf{-2} & \mathbf{-2} \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{2} & \mathbf{0} & \mathbf{-2} \\ 0 & 0 & 0 & 0 & \mathbf{-2} & \mathbf{0} & \mathbf{2} & \mathbf{-1} \\ 0 & 0 & 0 & 0 & \mathbf{2} & \mathbf{-2} & \mathbf{1} & \mathbf{0} \end{bmatrix}$$

The bold block can be decomposed as described in [38]:

$$\begin{bmatrix} 0 & -1 & -2 & -2 \\ 1 & 2 & 0 & -2 \\ -2 & 0 & 2 & -1 \\ 2 & -2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} 2^1 + \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} 2^0 \tag{3.6}$$

- The matrices P_1 and P_2 do not include any arithmetic operation;
- The matrix A_1 presents eight rows with two '1's including 8 additions;
- The matrix A_2 presents 4 rows with two '1's, including only 4 additions;
- The matrix A_3 presents two rows with two '1's, including 2 additions; two rows with '1' and '2', including 2 additions and two-bit shift and 4 rows with three '1', including 8 additions.

- The matrix A_4 presents two rows with two '1's, including 2 additions; two rows with '1' and '2', including 2 additions 2-bit shift operations, and the four last rows including 4 additions and 4 bit-shift operations

The Signal Flow Graphs (SFG) appropriate to the above factorisations are shown in Fig. 3.2 for T_{p1} and $\hat{T}_{p2}$ respectively using the basic structure shown in Fig. 3.1.

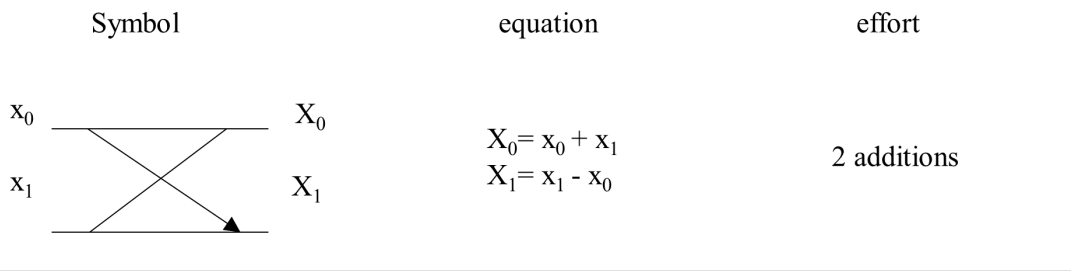
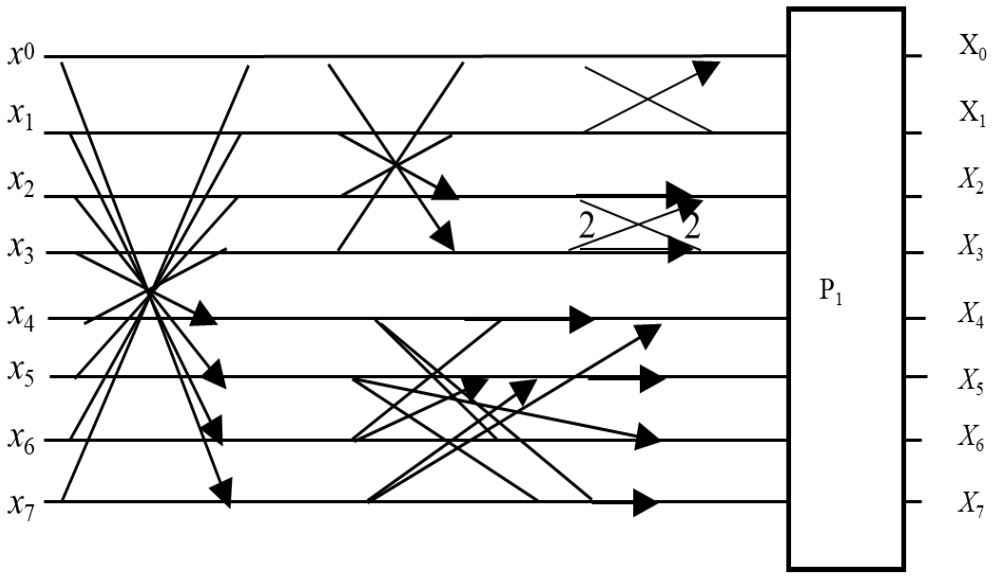


Fig. 3.1: The basic symbol used in the graph

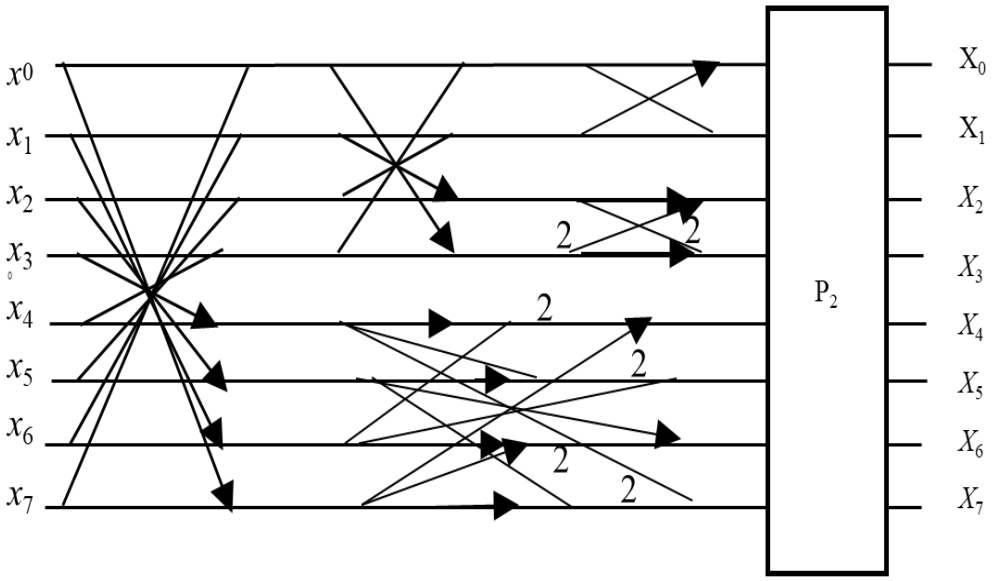
The first and second stages are the same for the two proposed transforms. The difference emerges in the third stage. The permutation matrices P_1 and P_2 do not contain any operation. The computational cost is only 24 additions and two bits shifting for T_{p1}, 24 additions and 6 bits shifting for the proposed second one. Therefore, the proposed fast algorithm requires 50% fewer additions and 75% less bit-shifting operations

3.2.3 Computational complexity

Table 3.1 provides a comparison of the number of additions, multiplications, and shifts required between the existing and proposed transforms. All elements of approximate transform matrices are $\{0, \pm 1, \pm 2\}$, so no multiplication operations are required. The computational cost of different transforms varies between 16 additions and 24 additions with 6 bit-shifting. We focused our research on increasing the PSNR while maintaining the number of operations required in recent research [38] and [35].



(a)



(b)

Fig. 3.2: The signal flow graphs (SFG): (a) forward T_{p1}, (b) forward $\hat{T}_{p2}$

Table 3.1: Number of arithmetic operations required by our proposed transformsT_{p1}, T_{p2} and other DCT approximations

Transforms	Multiplications	Additions	Bit-shifts	Total
Scaled DCT [24] (2007)	05	29	00	34
Proposed T_{p1}	**00**	**24**	**02**	**26**
Proposed $\hat{T}_{p2}$	**00**	**24**	**06**	**30**
T_1 [25] (2001)	00	24	00	24
T_2 [7] (2008)	00	18	02	20
T_3 [104] (2008)	00	21	03	24
T_4 [26] (2009)	00	18	00	18
T_5 [27] (2010)	00	24	04	28
T_6 [5] (2011)				
$a=0$	00	16	00	16
$a=1$	00	18	00	18
$a=2$	00	18	02	20
T_7 [28](2012)	00	24	00	24
T_8 [35] (2019)	00	24	06	30
T_9 [35] (2019)	00	24	06	30
T_{10} [38] (2018)	00	28	06	34
T_{11} [38](2018)	00	24	02	26
T_{12} [6](2016)	00	17	00	17
$T_{4-cintra}$ [29] (2014)	00	24	00	24
$T_{6-cintra}$ [29] (2014)	00	24	06	30

3.3 Proposed transform based on introducing of element nul

3.3.1 First obtained transform

Indeed, the bit-shift operations cause computational problems on the hardware implementation [6]. To remedy these problems, a new approximation has been proposed in [6] resulting from the application of the signum function to the transform given in [36]. It only requires additions; however, it is quasi-orthogonal and its image compression performance is quite low. The transform in [7] is a variant of SDCT [25], which consists of introducing some 0 and $1/2$ elements. The resulting transform requires 18 additions, and 2 bits shift operations. To avoid such problems, we propose a different variant of [7] completely free of bit shift and multiplication operations by replacing the elements equal to $1/2$ of the matrix transform in [7] with 0.

The associated proposed transform, referred to as T_{p3}, with its diagonal, D_{p3}, are given by:

$$T_{p3} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.7}$$

$$centering D_{p3} = diag(1/\sqrt{8}, 1/2, 1/2, 1/\sqrt{2}, 1/\sqrt{8}, 1/2, 1/2, 1/\sqrt{2}) \tag{3.8}$$

$$C_{p3} = D_{p3} \times T_{p3} \tag{3.9}$$

All elements of our matrix are 0 or ± 1, so multiplication and bit-shift operations are entirely absent. It is easy also to verify that our new transform is orthogonal, i.e. $C_{p3}^{-1} = C_{P3}^{t} = T_{p3}^{t} \times D_{p3}$ where t denotes the matrix transpose operation, unlike the one proposed in [25]. Hence, the same number of operations is required to compute both the forward and inverse transforms.

3.3.1.1 Proposed fast algorithm

It is clear from equation 3.7, the proposed transform T_{p3} requires 27 additions. To make our transform more obvious, we propose a fast algorithm for its calculation. This algorithm, based on sparse matrix factorisation, consists of factorising the matrix T_{p3} into three simple matrix products. Accordingly, the number of required operations is reduced to 16 additions only.
The three matrices are given as follows:

$$T_{p3} = A_5 \times A_6 \times A_7 \tag{3.10}$$

$$A_5 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, A_6 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_7 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The structure corresponding to our fast algorithm is illustrated in Fig. 3.3. As Fig. 3.3 clearly illustrates, our transform requires only 16 additions. The structure can be divided into 3 independent stages. from left to right, the first stage requires 8 additions, the second one 4 additions, and finally the third 4 additions.

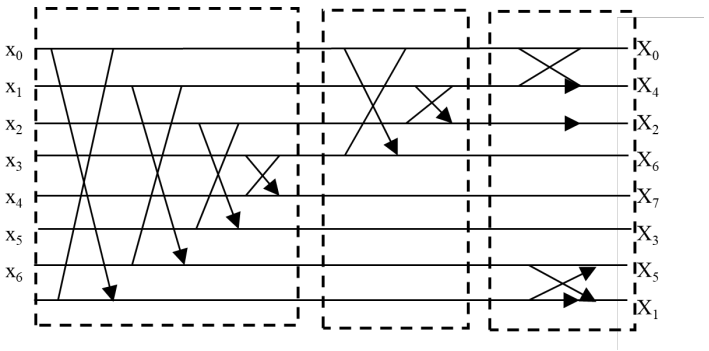


Fig. 3.3: Signal flow graph for forward T_{p3}

3.3.1.2 Computational complexity evaluation

To highlight our transform, we compared it with other existing transforms in the literature in terms of complexity, and the results are summurised in Table 3.2. The

Scaled DCT is the fastest algorithm for calculating the DCT transform, and it is used only as a reference for performance assessment.

Table 3.2: Number of arithmetic operations required by our proposed transform T_{p3} and other transforms

Transforms	Multiplications	Additions	Bit-shifts	Total
Scaled DCT [24] (2007)	05	29	00	34
Proposed transform T_{p3}	00	16	00	16
T_2 [7] (2008)	00	18	02	20
T_3 [104] (2008)	00	21	03	24
T_{13} [32] (2012)	00	14	00	14
T_6 [5] (2011)				
$a=0$	00	16	00	16
$a=1$	00	18	00	18
$a=2$	00	18	02	20
T_{12} [6](2016)	00	17	00	17

From Table 3.2 we can show a reduction of 4 arithmetic operations of our proposed transform compared to its original transform in [7]. Indeed, the transform we propose, with only 16 additions, also allows a significant reduction in the number of operations required compared to other transforms; it has 33.34 %, 20%, 11.11%, and 5.9% lower arithmetic costs than the SDCT [25], $T_6(a=2), T_6(a=1)$ [5], and T_{12} [6], respectively. The transform T_{13} with 14 additions in [32] remains the best in terms of complexity; however, it shows a significant degradation in terms of performance.

3.3.2 Second obtained transform

The T_9 transform has not been implemented in the paper [35], its performance in image compression is limited despite its high complexity cost. To obtain a performance improvement of the transform T_9 reported in [35] with less complexity in terms of computation costs, the coefficients of the even rows have been kept while some coefficients of the odd rows have been set to zero, taking into consideration the following conditions:

- The proposed matrix T_{p4} must possess its elements from $\{0, \pm 1, \pm 2\}$;
- Only addition and bit-shift operations are needed to implement the forward and inverse transform;
- $T_{p4} \times T_{p4}^t$ must be a diagonal matrix, therefore the inverse transform T_{p4}^{-1} guaranteed to have low computational complexity.

The resulting transform requires only 18 additions and 6 bit-shift operations. A reduction of 25% in the computational cost complexity, compared to its original transform, is successfully achieved. The associated resulting transform T_{p4} is given by:

$$T_{p4} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 & -1 & -2 \\ 2 & 1 & -1 & -2 & -2 & -1 & 1 & 2 \\ 0 & 0 & -2 & 0 & 0 & 2 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -2 & 0 & 0 & 0 & 0 & 2 & -1 \\ 1 & -2 & 2 & -1 & -1 & 2 & -2 & 1 \\ 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 \end{bmatrix} \tag{3.11}$$

T_{p4} can be modified to provide an orthogonal transform C_{p4}. where $(.)^t$ denotes the matrix transpose operation and considering: $C_{p4} = D_{p4} \times T_{p4}$ and $C_{p4} \times C_{p4}^t = I$.

$$D_{p4} = \sqrt{(T_{p4} \times T_{p4}^t)^{-1}} \tag{3.12}$$

$$D_{p4} = diag(1/\sqrt{8}, 1/\sqrt{10}, 1/\sqrt{20}, 1/\sqrt{8}, 1/\sqrt{10}, 1/\sqrt{10}, 1/\sqrt{20}, 1/\sqrt{8}) \tag{3.13}$$

All elements of our matrix are $\{0, \pm 1, \pm 2\}$. Hence, no multiplication is required because the required arithmetic operations can be implemented exclusively through addition and bit-shifting operations. This means that without using multiplication, a reduction in computational complexity and an increase in speed are achieved.

It should be noted that the proposed matrix C_{p4} is orthogonal. Accordingly, the inverse transform is as follows:

$$C_{p4}^t = C_{p4}^{-1} = T_{p4} \times D_{p4} \tag{3.14}$$

Therefore, the forward and inverse transforms require the same number of operations for their computation. The multiplication with the diagonal matrix D_{p4} can be eliminated from the transform step by integrating its entries into the quantisation matrix.

3.3.2.1 Proposed fast algorithm

The computational complexity of direct implementation of the proposed DCT transform consists of 36 additions and 16 bit-shifting. However, to make our transform more evident, such cost can be significantly reduced through sparse matrix factori-

sation [110]. For this, we propose a fast algorithm for its computation. This algorithm is based on factorising the matrix T_{p4} into three simple matrices product. Accordingly, the number of operations required is reduced to 18 additions only and 6 bits shifts. The three resultant matrices are given as follows:

$$T_{p4} = A_8 \times A_9 \times A_{10} \tag{3.15}$$

$$A_8 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -2 \\ 0 & 0 & -1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}, A_9 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_{10} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The proposed fast algorithm for the new multiplier-less transform, T_{p4}, summarised in the following steps, can be used to calculate the transformed vector $y = T_{p4}.x$, of an input vector x of size 8.

Step 1 :

$$\begin{aligned} u_0 &= x_0 + x_7 & u_4 &= -x_3 + x_4 \\ u_1 &= x_1 + x_6 & u_5 &= -x_2 + x_5 \\ u_2 &= x_2 + x_5 & u_6 &= -x_1 + x_6 \\ u_3 &= x_3 + x_4 & u_7 &= -x_0 + x_7 \end{aligned}$$

Step2 :

$$\begin{aligned} k_0 &= u_0 + u_3 & k_4 &= u_4 \\ k_1 &= u_1 + u_2 & k_5 &= u_5 \\ k_2 &= -u_1 + u_2 & k_6 &= u_6 \\ k_3 &= -u_0 + u_3 & k_7 &= u_7 \end{aligned}$$

Step3 :

$$\begin{aligned} y_0 &= -k_0 + k_1 & y_4 &= k_0 - k_1 \\ y_1 &= -k_6 - 2k_7 & y_5 &= 2k_6 - k_7 \\ y_2 &= -k_2 - 2k_3 & y_6 &= 2k_2 - k_3 \\ y_3 &= 2k_5 & y_7 &= 2k_4 \end{aligned}$$

As can be shown from the algorithm above:

- step 1 required 8 additions;
- step 2 required 4 additions;
- step 3 required 6 addition and 6-bit shift operations.

The total is 24 additions and 6-bit shift operations. Using the separability property of the DCT transform, the 2-D of the proposed transform can be calculated using the 1-D implementation proposed, row-wise first and then column-wise.

3.3.2.2 Computational complexity evaluation

Table 3.3 presents a summary of the comparative analysis of several DCT approximations considered in our study in terms of the degree of complexity. We also include the exact DCT for comparison purposes. The number of addition, shift, and multiplication operations needed for each transform, on one and two dimensions, has been compared.

Table 3.3 clearly demonstrates that the proposed transform exhibits a notable reduction in the number of operations when compared to its original transform counterpart. Specifically, it necessitates 20% fewer operations than the T_9 [35] transform.

3.4 Proposed transforms in image compression

All elements of our proposed transform matrices are $\{0, \pm 1, \pm 2\}$, so multiplication operations are completely absent. The proposed low-complexity matrices $T_{pi} = \{i =$

Table 3.3: Number of arithmetic operations required our proposed transform T_{p4} and other DCT approximations

Transforms	Multiplications	Additions	Bit-shifts	Total
Scaled DCT [24] (2007)	05	29	00	34
Proposed transform T_{p4}	00	18	06	24
T_1 [25] (2001)	00	24	00	24
T_3 [104] (2008)	00	21	03	24
T_7 [28] (2012)	00	24	00	24
T_{12} [6](2016)	00	17	00	17
$T_{4-cintra}$ [29] (2014)	00	24	00	24
T_{10} [38](2018)	00	24	02	26
T_{14} [106](2020)	00	18	01	19
T_9 [35](2019)	00	24	06	30

$1, 2, 3, 4\}$ can approximate the DCT matrix by means of orthogonalisation or quasi-orthogonalisation. The first value of a leads to an orthogonal transform T_{p1}. Therefore, the same fast algorithm is used to calculate both the forward and inverse transform and the case of the other proposed transform for $i = 3$ and $i = 4$:

$$T_{pi}.T_{pi}^{-1} = I \tag{3.16}$$

where I is the identity matrix. As with all, the proposed approximate DCT transform, T_{pi}, can be modified to be C_{pi} to ensure the orthonormality conditions, this procedure is approved in different papers of integer DCT approximations.

$$C_{pi} = D_{pi}.T_{pi} \tag{3.17}$$

$$C_{pi}.C'_{pi} = I \tag{3.18}$$

Then $C_{pi}^{-1} = C'_{pi}$ and D_{pi} is a diagonal matrix ensuring the orthogonality of the DCT approximations, which is obtained by the following equation:

$$D_{pi} = diag\sqrt{(T_{pi}.T'_{pi})^{-1}} \tag{3.19}$$

where $\sqrt{(.)}$ signifies the matrix square root operation and $diag(.)$ rends a diagonal matrix with the diagonal elements of its matrix argument. Since the quantisation/dequantisation is applied on the block matrix in the transform domain, the diagonal matrix D_{pi} can be merged into the quantisation/dequantisation matrix.

On the other side, the second value of $'a'$ leads to a non-orthogonal approximation for the 8 floating points DCT as many transforms known in literature [25, 104, 6, 71, 3]. Then, $\hat{T}_{p2}$ does not satisfy the orthogonality condition. However, $\hat{T}_{p2} \times \hat{T}'_{p2}$ is a nearly diagonal matrix. Indeed, the orthogonality property is not a strictly necessary requirement for image compression applications, a good energy compaction property can be achieved by a near orthogonality. The evaluation of the proximity of a given non-orthogonal transform matrix to another orthogonal matrix is performed by computing the deviation from the diagonal measurement, given by [111]:

$$\delta(A) = \frac{1 - ||diag(A)||_F^2}{||A||_F^2} \tag{3.20}$$

A is a square matrix and $||.||_F$ is the Frobenius norm of matrices [111]. The deviation from diagonality of the matrix diagonalisation of SDCT T_1 matrix has been adopted as the maximum value that can be accepted for saying that a non-orthogonal transform is a quasi-orthogonal [71]. This value is calculated from:

$$\delta(T_1.T_1^t) \approx 0.20 \tag{3.21}$$

For a transform matrix T:

$$\delta(T.T^t) \leq 0.20 \tag{3.22}$$

$$\delta(\hat{T}_{p2}.\hat{T}_{p2}^t) = 0.0544 \leq 0.20 \tag{3.23}$$

All the following transforms: SDCT [25], BAS transform proposed in [104], transforms given in [71] and PADCT introduced in [6] are non-orthogonal, and they are considered as good approximations of the DCT, just like the approximate DTT T_T proposed in [3]. Table 3.4 summarises the deviation from the orthogonality of these transforms. The deviation from orthogonality value of SDCT is the greatest one, and it is considered as the maximum value that can be taken by a transform. Since our transform $\hat{T}_{p2}$ has the smallest value of deviation from orthogonality than the well-known existing non-orthogonal transforms, it is more "nearly-orthogonal" than these transforms. Therefore, the non-orthogonal approximation $\hat{T}_{p2}$ referred to be $\hat{C}_{p2}$ can be obtained

Table 3.4: Deviation from diagonal measurement for non-orthogonal transforms

Transform	T_1 [25]	T_3[104]	$\hat{T}$ [71]	$\tilde{T}$ [71]	T_{12} [6]	T_T [3]	$\hat{T}_{p2}$
δ	0.200	0.177	0.071	0.058	0.133	0.133	**0.054**

by:

$$\hat{C}_{p2} = \hat{D}_{p2}.\hat{T}_{p2} \tag{3.24}$$

$$\hat{D}_{p2} = diag\sqrt{(\hat{T}_{p2}.\hat{T}'_{p2})^{-1}} \tag{3.25}$$

$$\hat{D}_{p2} = diag(1/\sqrt{8}, 1/\sqrt{18}, 1/\sqrt{20}, 1/\sqrt{18}, 1/\sqrt{8}, 1/\sqrt{18}, 1/\sqrt{20}, 1/\sqrt{18}) \tag{3.26}$$

The second investigation for JPEG standard quality enhancement is to propose the approach to generate quantisation matrix adapted to JPEG standard based on integer DCT or DTT approximations.

3.5 Integer DCT-based JPEG quantisation matrix

This section proposes an enhancement of the baseline JPEG compression standard based on integer DCT approximations.

- Regarding energy distribution of integer DCT coefficients, an appropriate quantisation matrix has been proposed.

- This matrix is approximated to a power of two elements to reduce the complexity cost in image compression standard JPEG.

3.5.1 Default JPEG luminance quantisation matrix

The JPEG is the well-known standard for still grayscale and color image compression. It is based on DCT (or fast DCT/FDCT) whereas quantisation is an important stage in this standard. Indeed, it is part of the process that allows reducing the size of the images to be stored. However, it is also during this stage, that information losses are generated.
In the JPEG standard, after a preprocessing stage, the $M \times N$ input image is subdivided into blocks of 8×8 pixels $X_{i,j}$ whose DCT is applied. The resulting coefficients, $Y_{u,v}$, are quantified afterward, $Y_{u,v}^{quant}$.

$$Y_{u,v}^{quant} = round(Y_{u,v} \oslash Q_{JPEG}) \tag{3.27}$$

where $round(.)$ is the rounding function, and $\oslash$ denotes the elements-by-element division. JPEG proposes a luminance quantisation matrix where each of the 64 DCT

coefficients is quantised by the uniform quantiser, Q_{JPEG} [76].

$$Q_{JPEG} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 77 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{3.28}$$

The dequantisation stage in decoding process is given by :

$$\hat{Y}_{u,v}^{dequant} = Y_{u,v}^{quant} \odot Q_{JPEG} \tag{3.29}$$

where $\hat{Y}_{u,v}^{dequant}$ is the dequantised coefficients, and $\odot$ denotes the element-wise multiplication. In the field of image and video compression, an effective alternative to the traditional discrete cosine transform (DCT) called Integer Discrete Cosine Transform (InDCT) has gained popularity over the past decade [112]. Numerous integer DCT approximations have been proposed in the literature, offering of using a minimal number of integer arithmetic operations [1, 113, 37, 29, 114, 112, 115]. To enhance the compression ratio (CR) achieved by JPEG utilising DCT approximations, it becomes essential to adjust the quantisation step in accordance with these approximations. By adopting the quantisation process to align with the characteristics of the specific DCT approximations, improved compression performance can be achieved.

3.5.2 Related work

In 2017, Oliveira et al.input forward, in [42], a novel quantisation scheme that is compatible with the JPEG standard and offers a significantly reduced complexity compared to existing methods. The new scheme eliminates the need for multiplication and addition operations, exclusively utilises bit-shifting operations. By combining this low-complexity quantisation table with discrete cosine transform (DCT) approximations, the overall complexity of the transform-quantisation pair in the JPEG encoding/decoding process can be decreased.

The technique involves applying the nearest power-of-two function called $np2$ to the default luminance JPEG quantization matrix.

$$np2(x) = 2^{round(log_2 x)}, x \in R \tag{3.30}$$

The resulting approximate quantisation matrix, denoted as $Q_{Oliveira}$, is provided in [42]:

$$Q_{Oliveira} = \begin{bmatrix} 6 & 8 & 8 & 16 & 32 & 32 & 64 & 64 \\ 16 & 16 & 16 & 16 & 32 & 64 & 64 & 64 \\ 16 & 16 & 16 & 24 & 32 & 64 & 64 & 64 \\ 16 & 16 & 16 & 32 & 64 & 64 & 64 & 64 \\ 16 & 16 & 32 & 64 & 64 & 128 & 128 & 64 \\ 32 & 32 & 64 & 64 & 64 & 128 & 128 & 64 \\ 64 & 64 & 64 & 64 & 128 & 128 & 128 & 128 \\ 64 & 64 & 64 & 128 & 128 & 128 & 128 & 128 \end{bmatrix} \tag{3.31}$$

By expressing the orthonormal integer approximation DCT matrix, denoted as C, as the product of the integer DCT matrix T and a diagonal matrix D, it is possible to integrate the diagonal matrix D into the quantisation matrix. This integration eliminates the need for separate multiplication operations associated with D, leading to improved computational efficiency.

$$C = D.T \tag{3.32}$$

Then, the modified forward and inverse quantisation matrices, $\hat{Q}_f$ and $\hat{Q}_i$, respectively, can be written as:

$$\hat{Q}_{Oliveira-f} = np2(Q_{JPEG} \oslash (d \times d^t)) \tag{3.33}$$

$$\hat{Q}_{Oliveira-i} = np2((d \times d^t) \odot Q_{JPEG}) \tag{3.34}$$

where d presents the column vector with the element corresponding to the main diagonal of D, and d^t is its transpose.

3.5.3 Proposed approach to generate quantisation matrix

The DCT coefficients are quantised by dividing them all by a factor. The JPEG norm approach has the goal of rejecting low-value coefficients that only provide limited

information in the image and reducing the dynamics of others. The resulting degradation is invisible to the eye. However, It is crucial to remember, that not all the coefficients contain the same amount of information. Therefore, this quantity must be considered by the quantisation step as well as the fact that DCT concentrates the image energy into a few altered coefficients associated with low frequencies, they are subsequently richer on the information.

The quantisation factor should, therefore, be different depending on the position of the coefficient in the block: it will be all the stronger as the coefficients will be poor in information. It will be larger at high frequency than at low frequency. Indeed, reducing the dynamic, view zeroing, a coefficient corresponding to high frequencies (often related to noise) will not cause as much degradation as a coefficient corresponding to low frequencies since the human eyes cannot sense the high-frequency components in the image. Then, the energy of DCT coefficients plays a very important role to define the best suitable quantisation step for each coefficient. In this regard, using this strategy, we present a new approach to generate a quantisation matrix intended for integer DCT-based JPEG encoder. Our approach is composed of two stages:

3.5.3.1 Energy distribution of DCT approximations

It consists of generating a quantisation matrix based on the energy distribution of integer DCT coefficients. Thus, two integer DCT approximations introduced in [1, 2] have been considered which they present the more efficient transforms that approximate the floating points DCT, and they provide the best trade-off between coding performances and computation cost.

Fig. 3.5 shows an example of the energy distribution of the 64 coefficients of the integer DCT approximations reported in [1] for test image Lena [116].

Concerning the transform in [2], we can show from Fig. 3.6 that the energy of the coefficients differs from one transform to the other. The quantisation factor should depend on the position of the coefficient in the block as well as on its energy. A coefficient with low energy has poor information, so the corresponding quantisation step should be high and vice versa. The implementation of this strategy leads us to construct the novel quantisation matrix for the integer DCT-based JPEG standard.

1 72.1db	**2** 20.6db	**4** -1.7db	**6** -11.0db	**8** -18.4db	**12** -25.5db	**25** -29.8db	**21** -28.6db
3 -1.1db	**5** -8.7db	**7** -12.0db	**9** -18.8db	**18** -28.1db	**13** -26.9db	**28** -30.1db	**22** -29.0db
11 -24.1db	**10** -24.0db	**16** -27.8db	**20** -28.4db	**17** -27.8db	**23** -29.3db	**24** -29.4db	**33** -30.5db
14 -26.9db	**19** -28.3db	**27** -29.9db	**34** -30.5db	**26** -29.9db	**44** -32.6db	**30** -30.4db	**45** -32.6db
15 -27.2db	**31** -30.4db	**32** -30.5db	**39** -31.4db	**40** -31.9db	**51** -33.2db	**59** -34.2db	**54** -33.5db
35 -31.0db	**29** -30.3db	**36** -31.1db	**47** -32.9db	**57** -33.8db	**55** -33.7db	**52** -33.3db	**53** -33.4db
37 -31.2db	**38** -31.3db	**46** -32.6db	**60** -34.5db	**42** -32.4db	**50** -33.1db	**48** -33.0db	**64** -35.7db
43 -32.5db	**49** -33.0db	**58** -34.0db	**41** -32.2db	**61** -34.8db	**62** -35.4db	**56** -33.8db	**63** -35.6db

Fig. 3.4: Energy distribution of conventional DCT for test image Lena

1 72.1db	**2** 25.1db	**4** 3.2db	**3** 16.0db	**13** -18.5db	**6** -5.5db	**9** -12.1db	**5** -0.1db
7 -8.5db	**8** -11.2db	**11** -15.7db	**10** -15.3db	**53** -37.3db	**30** -29.0db	**17** -23.8db	**12** -18.0db
38 -31.0db	**20** -25.3db	**31** -29.3db	**16** -21.3db	**50** -35.9db	**34** -30.3db	**37** -30.7db	**22** -26.1db
32 -29.9db	**26** -28.4db	**39** -31.5db	**28** -28.5db	**63** -47.6db	**55** -39.1db	**52** -37.1db	**36** -30.6db
23 -27.7db	**18** -25.2db	**19** -25.3db	**14** -19.4db	**44** -32.9db	**25** -28.1db	**24** -28.1db	**15** -20.8db
49 -35.2db	**35** -30.6db	**41** -31.8db	**21** -25.3db	**61** -43.0db	**51** -36.2db	**46** -33.4db	**27** -28.4db
56 -39.4db	**40** -31.8db	**42** -32.8db	**33** -29.9db	**59** -40.4db	**47** -33.7db	**43** -32.9db	**29** -28.9db
62 -43.2db	**54** -38.3db	**58** -40.4db	**48** -34.4db	**64** -51.3db	**60** -42.9db	**57** -40.2db	**45** -33.3db

Fig. 3.5: Energy distribution of DCT approximation in [1] for test image Lena

1 72.1db	2 22.4db	3 3.2db	4 1.9db	12 -18.4db	11 -12.9db	10 -12.0db	6 -9.6db
5 -4.4db	7 -10.2db	8 -10.7db	9 -10.9db	44 -30.6db	14 -21.8db	13 -19.2db	16 -23.8db
43 -30.2db	24 -27.3db	34 -28.7db	21 -26.1db	56 -34.4db	50 -32.2db	40 -30.0db	53 -32.8db
20 -25.8db	37 -29.2db	38 -29.5db	22 -26.6db	62 -37.4db	55 -34.3db	39 -29.9db	54 -34.2db
25 -27.3db	28 -27.7db	19 -25.0db	15 -22.2db	49 -32.0db	32 -28.6db	26 -27.5db	36 -29.0db
29 -28.2db	31 -28.5db	23 -26.7db	18 -24.7db	60 -36.8db	45 -31.4db	35 -28.9db	51 -32.4db
63 -37.4db	59 -35.8db	46 -31.8db	42 -30.2db	64 -38.0db	57 -34.9db	48 -31.9db	58 -35.7db
27 -27.6db	41 -30.2db	33 -28.6db	17 -23.9db	61 -37.0db	47 -31.8db	30 -28.3db	52 -32.7db

Fig. 3.6: Energy distribution of DCT approximation in [2] for test image Lena

The quantisation matrix we suggest is as follows:

$$Q_{P-DCT} = \begin{bmatrix} 20 & 17 & 18 & 19 & 22 & 36 & 36 & 31 \\ 19 & 17 & 20 & 22 & 24 & 40 & 23 & 40 \\ 20 & 22 & 24 & 28 & 37 & 53 & 50 & 54 \\ 22 & 20 & 25 & 35 & 45 & 73 & 73 & 58 \\ 22 & 21 & 37 & 74 & 70 & 92 & 101 & 103 \\ 24 & 43 & 50 & 64 & 100 & 104 & 120 & 92 \\ 45 & 100 & 62 & 79 & 100 & 70 & 70 & 101 \\ 41 & 41 & 74 & 59 & 70 & 90 & 100 & 99 \end{bmatrix} \tag{3.35}$$

3.6 Quantisation matrix resulting from merging the diagonal matrix

As explained in the previous sections, it is possible to integrate the diagonal matrix, denoted as D, derived from an integer approximation matrix T, into the quantisation step.

3.6.1 Equivalent quantisation matrix

If transform matrix of 2-dimentional data X is T, so:

$$Y = C.X.C^t = (D.T).X.(D.T)^t = D.T.X.T^t.D \tag{3.36}$$

where X is the entry block, Y is the transformed block by the transform T,T^t denotes the transpose of the transform matrix T and $C = D.T$.
Let

$$Y^* = T.X.T^t \tag{3.37}$$

$$Y = D.Y^*D \tag{3.38}$$

Let d be the element of the diagonal matrix, D. The entry of Y at position (i,j) is:

$$y(i,j) = d_i.d_j.y^*(i,j) \tag{3.39}$$

Assuming that the elements of the original quantisation matrix is $Q = [q_(i,j)]$, the quantised coefficient $y^Q(i,j)$ is:

$$\begin{aligned} y^Q(i,j) &= y(i,j)/q(i,j) \\ &= d_i.d_j.y^*(i,j)/q(i,j) \\ &= (y^*(i,j))/(q(i,j)/(d_i.d_j) \end{aligned} \tag{3.40}$$

Hence, the quantisation matrix of integer approximate transform T can be developed from the conventional quantisation matrix element $q(i,j)$ dividing it by $d_i.d_j$ that is $q(i,j)/(d_i.d_j)$. Accordingly, the equivalent quantisation matrix is $D^{-1}.Q.D^{-1}$.

3.6.2 Equivalent to inverse quantisation matrix

In the decoding process, the inverse quantisation matrix and the inverse transform are combined as follows:

$$\begin{aligned} C^T.(Y^Q.Q).C =& (D.T)^T(Y^Q \otimes Q)(D.T) \\ &= T^T.D.[Y^Q \otimes Q].D.T \\ &= T^T.(D.Y^Q.D) \otimes Q.T \\ &= T^T.[Y^Q \otimes D.Q.D)].T \end{aligned} \tag{3.41}$$

The symbol $\otimes$ represents the matrix dot where $y(i,j)^Q$ denotes the quantisation coefficient of $y(i,j)$. This implies that $d_i d_j y(i,j)^Q$ must be calculated before performing the inverse transform. This step can be integrated with the inverse quantisation process to make the inverse quantisation step as $q(i,j)d_i d_j$.

Therefore, when we use the integer approximate DCT transform matrix T to perform the forward and inverse transform, the quantisation and inverse quantisation of the matrices are different. the resulting proposed quantisation matrix is:

$$Q_{P-DCT-f} = Q_{P-DCT} \oslash (d \times d^t) \tag{3.42}$$

$Q_{P-DCT-f}$ is the forward quantisation matrix specifically designed for the JPEG coder, taking into account the integer DCT approximations used in the compression process. By applying the same method during the dequantisation stage, we can obtain the inverse quantisation matrix:

$$Q_{P-DCT-i} = (d \times d^t) \odot Q_{P-DCT} \tag{3.43}$$

3.6.3 Quantisation matrix design based on integer DCT

Despite the use of integer DCT approximations, the quantisation and dequantisation stages in JPEG still involve floating-point multiplications and rounding operations. However, we propose an alternative approach by introducing an integer power-of-two approximation for our quantisation matrix, denoted as $Q_{P-DCT-f}$. This approach eliminates the need for multiplication operations in the matrices used during JPEG quantisation and dequantization stages.

The new proposed forward and inverse approximate quantisation matrices are expressed as:

$$\tilde{Q}_{P-DCT-f} = 2^{ceil.log_2(Q_{P-DCT-f})} \tag{3.44}$$

$$\tilde{Q}_{P-DCT-i} = 2^{ceil.log_2(Q_{P-DCT-i})} \tag{3.45}$$

where $ceil(x)$ rounds the elements of $x, x \in R$, to the nearest integers. The resulting matrices required bit-shift operations only, and they can be integrated directly into image compression standard JPEG based on integer DCT.

3.6.4 Arithmetic cost evaluation

The arithmetic cost evaluation of forward and inverse transform /quantisation pair stages for a single 8×8 image block is tabulated in Table 3.5.

If no approximations are applied to the standard JPEG standard and only the fast-scaled DCT (5 multiplications and 29 additions for an 8-sample vector) is used, the standard JPEG requires the greatest number of operations (1180). This number is reduced by Oliveira et al. in [42], an arithmetic operations savings of 24.1% in additions, and no multiplication is needed in their implementation. An additional bit-shift operation is required; it can be implemented visually with no computation cost.

In all our four proposed scenarios mentioned in Table 3.5, the number of arithmetic operations is decreased compared with JPEG standard, it reaches to 640 operations only and no multiplication is required using the transform in [1] with 16 additions and the proposed quantisation matrix, $\tilde{Q}_{P-DCT-f}$, with bit-shift operations only. The reduction achieved to 512 additions means a saving of 44.8% compared with the JPEG baseline and 27.3% compared with the quantisation matrix in [42].

Table 3.5: Arithmetic complexity comparison required by conventional JPEG and other scenarios

Senario	Addition	Bits-Shift	Multiplication	Round(.)	Total	Operation saving
Standard JPEG	928	0	288	64	1180	-
Transf. in [1] and Q_{JPEG}	512	0	128	64	704	40.34%
Transf.in [1] and Q_{P-DCT}	512	0	128	64	704	40.34%
Transf. in [1] and $\tilde{Q}_{P-DCT}$	512	128	0	0	640	45.8%
Transf. in [2]and $\hat{Q}_{Oliveira}$	704	128	0	0	832	29.5%

3.7 Integer DTT-based JPEG quantisation matrix

The framework for image compression and decompression in the DTT-based JPEG standard follows a similar structure to the DCT-based JPEG baseline system, consisting of three fundamental steps: DTT transform, quantisation, and entropy coding. However, the key distinction lies in the use of the forward DTT in encoding, replacing the original floating-point DCT, while the inverse DTT is employed alongside the inverse floating-point DCT in decoding. This disparity stems from the different transform basis matrices employed in DTT compared to DCT.

Recently, various researchers in [3, 4] proposed integer DTTs with applications for image and video compression. According to experimental results, these approx-

imate DTTs provide comparable performance to the DCT for lossy image compression. However, they did not investigate how to design the modified quantisation matrix based on DTT. In this section, we propose to improve the baseline JPEG compression standard based on integer DTT approximations.

3.7.1 Related work

DTT-based JPEG standard consists of replacing the DCT transform with the DTT. It is mentioned that this quantisation matrix can be used as an exact luminance quantisation matrix of the DTT [36]. In [117], an optimal quantisation matrix for the DTT-based JPEG baseline is proposed. In reference, an optimal quantisation matrix, Q_{opt}, for a DTT-based JPEG baseline is proposed such as the following:

$$Q_{opt} = \begin{bmatrix} 6 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 17 & 19 \\ 16 & 16 & 16 & 16 & 16 & 17 & 19 & 26 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{bmatrix} \tag{3.46}$$

3.7.2 Energy distribution of integer DTT approximations

The proposed approach discussed in section 3.5.3.1 to generate a quantisation matrix by exploiting the energy distribution of integer DCT approximations suitable for integer DCT-based JPEG standard has been exploited for DTT approximations.
Due to their efficiency in approximating the DTT and providing the best compromise between coding performance and computational cost, the two integer approximations in [3, 4] were considered. Fig. 3.7, Fig. 3.8 and Fig. 3.9 show the energy distribution of the Lena test image coefficients, using the exact DTT and the DTT approximations reported in [3, 4], respectively.

Those figures illustrate that the energy distribution of coefficients varies across different transforms. Fig. 3.8 presents the energy distribution of the DTT approximation reported in [3] for the test image Lena.
Considering this discrepancy, it becomes necessary for the quantisation factor to depend not only on the coefficient's position within the block but also on its energy level; Coefficients with higher energy contain valuable information, thus requiring

lower quantisation factors, whereas coefficients with lower energy can be assigned higher quantisation factors to achieve efficient compression.

1 72.1db	**9** -5.9db	**32** -28.3db	**50** -41.1db	**59** -59.2db	**63** -93.4db	**60** -61.3db	**64** -Infdb
2 25.4db	**14** -8.7db	**17** -12.6db	**28** -25.3db	**47** -37.8db	**52** -46.9db	**43** -34.1db	**58** -57.8db
7 2.2db	**27** -24.8db	**31** -27.6db	**37** -31.6db	**49** -39.9db	**56** -53.5db	**40** -33.3db	**62** -61.6db
6 2.7db	**24** -24.1db	**26** -24.8db	**34** -29.9db	**46** -37.7db	**54** -51.8db	**36** -30.7db	**57** -57.4db
5 9.5db	**21** -18.8db	**22** -19.8db	**25** -24.6db	**38** -32.4db	**51** -44.7db	**29** -25.8db	**53** -48.8db
4 17.6db	**11** -7.9db	**13** -8.3db	**19** -15.1db	**23** -22.6db	**41** -33.4db	**18** -14.6db	**45** -37.1db
8 0.7db	**30** -26.9db	**33** -29.1db	**42** -33.9db	**48** -39.7db	**55** -53.0db	**39** -32.5db	**61** -61.3db
3 20.8db	**10** -6.6db	**12** -8.2db	**15** -10.8db	**20** -18.6db	**35** -30.2db	**16** -12.3db	**44** -35.4db

Fig. 3.7: Energy distribution of exact DTT for test image Lena

3.7.3 Quantisation matrix design based on integer DTT

The image compression based on DTT is similar to that of JPEG adopting a linear uniform quantisation process to quantise the transformed coefficients, by dividing the low-frequency coefficients by a small quantisation step and dividing the high-frequency coefficients by a large quantisation step.

As a result, the quantisation factor should be different depending on the coefficient's position in the block: it should be greater when the coefficients have little information content. In this context, we have proposed a quantisation matrix adapted

1 72.1db	**2** 29.8db	**3** 7.3db	**5** -3.8db	**7** -6.2db	**6** -4.8db	**11** -13.1db	**4** 0.8db
9 -10.0db	**8** -8.3db	**10** -12.5db	**15** -24.4db	**12** -22.6db	**18** -25.2db	**29** -29.5db	**14** -24.3db
49 -33.1db	**17** -24.8db	**22** -26.9db	**44** -32.0db	**23** -28.0db	**34** -30.3db	**27** -29.0db	**30** -29.6db
40 -31.7db	**13** -23.4db	**16** -24.6db	**32** -29.9db	**19** -25.2db	**26** -28.8db	**20** -25.9db	**21** -26.2db
59 -37.8db	**31** -29.8db	**35** -30.8db	**57** -36.7db	**41** -31.8db	**51** -33.7db	**42** -31.9db	**37** -31.3db
63 -38.6db	**24** -28.3db	**25** -28.8db	**56** -36.0db	**46** -32.4db	**48** -32.9db	**38** -31.5db	**28** -29.4db
62 -38.1db	**33** -30.3db	**43** -32.0db	**60** -37.9db	**36** -31.1db	**50** -33.4db	**39** -31.6db	**45** -32.4db
64 -40.8db	**47** -32.8db	**54** -35.1db	**61** -38.1db	**52** -34.2db	**58** -36.7db	**55** -35.9db	**53** -34.5db

Fig. 3.8: Energy distribution of the DTT approximation in [3] for test image Lena.

1 72.1db	**2** 13.3db	**20** -13.0db	**15** -10.8db	**36** -24.7db	**6** -0.1db	**50** -29.6db	**4** 3.0db
3 5.7db	**13** -9.2db	**24** -17.1db	**14** -10.5db	**38** -25.0db	**8** -3.7db	**35** -23.5db	**5** 2.2db
21 -13.4db	**31** -20.8db	**44** -27.7db	**23** -15.2db	**37** -25.0db	**12** -8.5db	**34** -21.8db	**7** -0.8db
29 -20.2db	**49** -28.8db	**57** -34.0db	**30** -20.3db	**51** -30.9db	**22** -13.7db	**41** -26.8db	**11** -5.9db
27 -19.2db	**47** -27.9db	**58** -34.0db	**32** -20.8db	**52** -31.6db	**18** -12.7db	**48** -28.3db	**9** -4.8db
28 -19.7db	**43** -27.3db	**54** -32.2db	**33** -21.4db	**55** -32.9db	**19** -12.7db	**45** -27.8db	**10** -5.3db
39 -26.1db	**59** -34.4db	**64** -40.2db	**42** -27.1db	**61** -35.8db	**26** -19.0db	**53** -31.9db	**17** -11.4db
46 -27.9db	**60** -34.5db	**63** -39.4db	**40** -26.3db	**62** -37.6db	**25** -18.5db	**56** -33.4db	**16** -11.3db

Fig. 3.9: Energy distribution of the DTT approximation in [4] for test image Lena

to the integer DTT transforms.

$$Q_{P-DTT} = \begin{bmatrix} 16 & 16 & 18 & 19 & 24 & 36 & 36 & 31 \\ 19 & 16 & 20 & 22 & 24 & 40 & 23 & 40 \\ 20 & 22 & 24 & 28 & 37 & 53 & 50 & 54 \\ 24 & 20 & 25 & 35 & 45 & 73 & 73 & 58 \\ 22 & 21 & 37 & 74 & 70 & 92 & 101 & 103 \\ 24 & 43 & 50 & 64 & 100 & 104 & 120 & 92 \\ 45 & 100 & 62 & 79 & 100 & 70 & 70 & 101 \\ 41 & 41 & 74 & 59 & 70 & 90 & 100 & 99 \end{bmatrix} \tag{3.47}$$

3.8 Conclusion

Two different approaches to developing low-complexity integer 8-point DCT approximations were proposed.

1. Based on a 16-floating-points DCT matrix and rounding off operations, this approach allows us to achieve two integer-8-point DCT transforms process low computational complexity and without multiplication operations.

2. As a result of a successful introduction of some zeros in the transform in specified existing transforms, two novel others and efficient 8×8 orthogonal approximate DCT-II transforms are obtained.

A new multiplier-less integer DCT-based JPEG quantisation matrix based on the energy distribution of existing 8×8 integer DCT approximations are introduced. On one hand, to assure efficient JPEG compression standard and on the other hand, to eliminate any rounding operations due to quantisation stage. The same strategy is applied to approximate the DTT-based JPEG algorithm and leads to a new and efficient quantisation matrix.

Chapter 4

Experiments, evaluations, and discussions

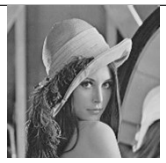
4.1 Introduction

Image compression is one of the important applications of the DCT. In order to check the effectiveness of the proposed transforms T_{pi} and compared to other existing DCT approximations, the following platform (Intel Core I5-8250U Processor ×64 at 1.6 GHz, 8 GB memory) using MATLAB R2017b programming language considered the compression of six well-known test images such as Lena, Cameraman [116], Barbara, Boat, Baboon, and Airplane [118], of size 512×512 (8 bpp). These images are different in the type of content and the amount of detail they contain, which is indicated by their patial Frequency Measure (SFM) [119] and Spectral Activity Measure (SAM) [120] parameters.

The SFM indicates the overall activity level in an image and the SAM indicates the predictability of an image; the values of SAM vary between $[1, \infty[$ and the higher values of the SAM imply that the image is more predictable. The different characteristic of images considered is tabulated in Table 4.1.
We carry out an experiment according to the following steps:

- Select one of the six test images as the input image;
- Divide each image into non-overlapping blocks of 64 (8×8) pixels;
- Apply the forward approximate DCT transform, considered in this study, to each block for each image;

Table 4.1: The six greyscale test images used in this study

Name	Image	Size ($pixel \times pixel$)	Pixel length	SAM	SFM
Lena		512×512	8 bits	910.17	14.02
Barbara		512×512	8 bits	445.82	29.45
Boat		512×512	8 bits	1144.4	17.85
Baboon		512×512	8 bits	99.70	36.51
Cameraman		512×512	8 bits	8132.4	14.36
Airplane		512×512	8 bits	2064.1	16.79

- Apply the zigzag scan;

- Select the coefficient retained of the 64 coefficients from each block and reconstruct the image by applying the inverse transform.

In order to evaluate the quality of the reconstructed images, three quantitative metrics are considered: Peak Signal to Noise Ratio, Structural Similarity Index Measure, and Percentage Energy Error Norm. They are generally used in recent works of image compression to evaluate the performance of several transformations.

4.2 Proposed transforms based on 16-point DCT performance evaluation

The efficiency curves of the proposed transform C_{p1} and the existing transforms, with the same arithmetic complexity, are shown in Fig. 4.1.

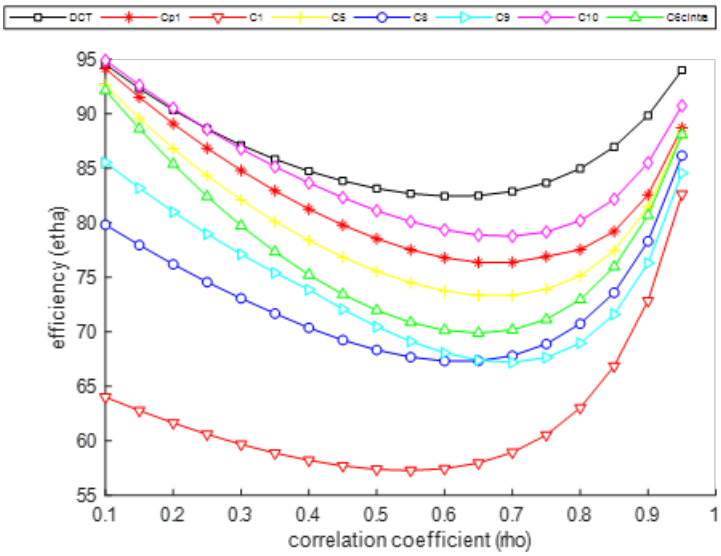


Fig. 4.1: Efficiency of proposed transform C_{p1} and other integer DCT approximations

Fig.4.1 shows that the proposed transform C_{p1} offers improved efficiency compared to the other transforms for all correlation coefficient values. It is the most efficient transform after the integer DCT approximation C_{10} proposed in [38], noted that C_{10} required 28 additions and 6-bit shift operations. Thus, an improvement of about 27.8% in the number of arithmetic operations compared to C_{p1}.

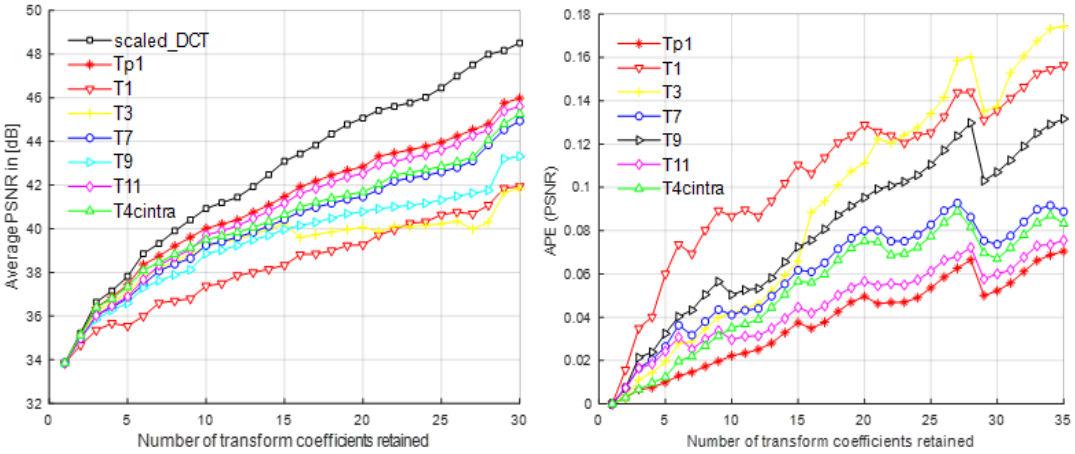
Table 4.2 presents the Total Error Energy, ϵ_{total}, of the proposed transform $\hat{C}_{p2}$ and considered approximated transforms.

Table 4.2: Total Error Energy, ϵ_{total}, for the proposed transform $\hat{C}_{p2}$ and other transforms

Transform	C_1 [25]	C_3[104]	$\hat{C}$ [71]	$\tilde{C}$ [71]	C_{12} [6]	C_T [3]	$\hat{C}_{p2}$
ϵ_{total}	3.32	4.19	1.79	3.64	10.93	0.78	**0.04**

Among all considered non-orthogonal transforms, the proposed transform $\hat{C}_{p2}$ provides the lowest value of the total error energy (grey highlighted in Table 4.2). Note that our transform requires 24 additions and 6-bit shift operations, similar to that of the approximate Tchebichef transformation reported in [3]. The transform in [71] requires 26 additions. However, it provides a high total error energy. While the transform of [6] performs the smallest number of arithmetic operations (17 additions) of the overall considered transforms, it displays the highest total error energy.

To illustrate the superiority of proposed transforms in image compression compared to existing DCT approximations, the variation of the three assessment criteria: PSNR, SSIM, PEEN, and their APE are shown in Fig.4.2, Fig.4.3, and Fig. 4.4, respectively.

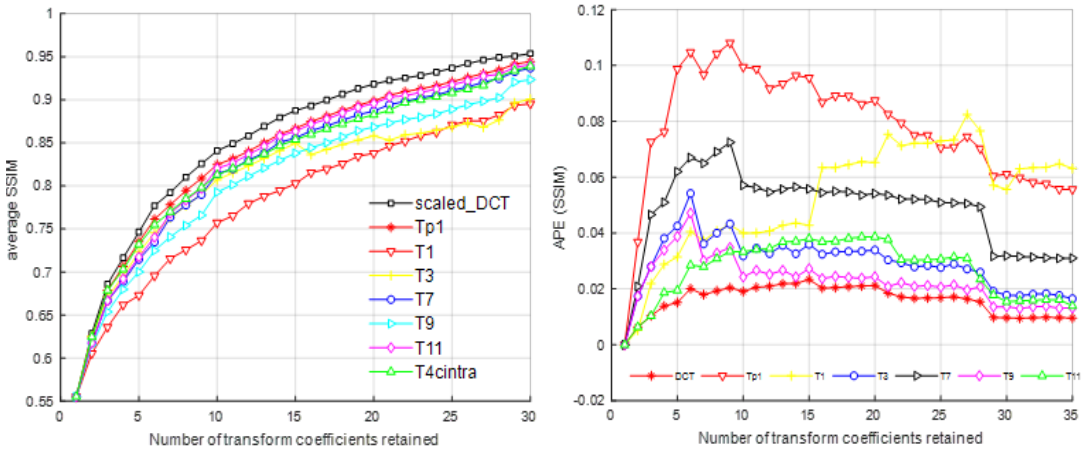


a b

Fig. 4.2: PSNR performance criteria: (a) Curves of average PSNR values for different DCT approximations. (b) Corresponding APE

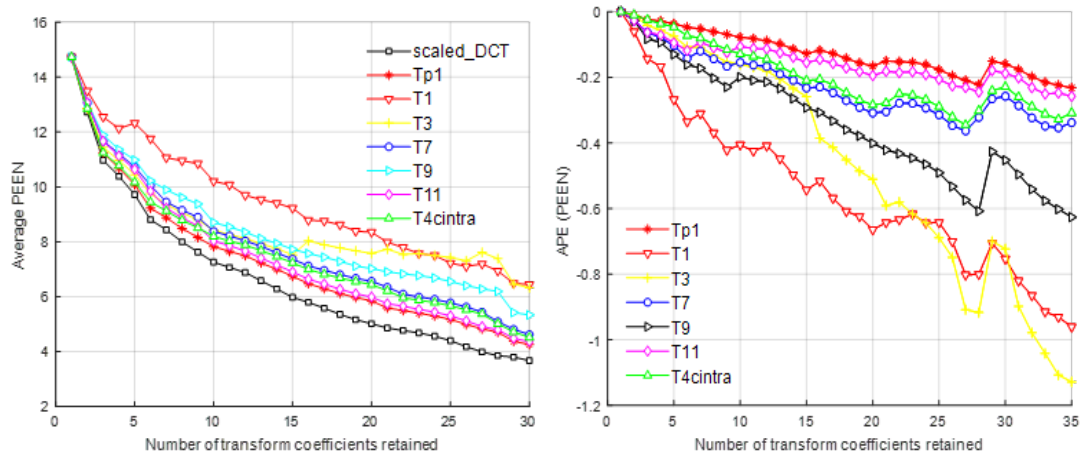
From Fig.4.2, the proposed transform T_{p1} outperforms all recent integer transforms considered in this study for all compression ratio values. It is slightly higher, about 0.085, than the transform T_{11} reported in [38] requiring the same number of operations. This superiority can be seen clearly observing the APE of PSNR's curve depicted in Fig. 4.2a. Our proposed transform demands 13.3% less operations than the transform T_9 reported in [35], on the other hand, it significantly outperformed T_9 in terms PSNR in middle and high compression ratio. The other transforms $T_{4-cintra}$ [29], T_7 [28] and T_3 [104] required the same number of operations than the proposed transform T_{p1} but their performances remain less significant.
These results are confirmed by the SSIM and PEEN assessment criteria and their

corresponding APE curves presented in Fig.4.3 and Fig.4.4 respectively. From these figures, the proposed transform T_{p1} has the lowest value of APE compared to the other transforms.



a b

Fig. 4.3: SSIM performance criteria: (a) Curves of average SSIM values for different DCT approximations. (b) Corresponding APE



a b

Fig. 4.4: PEEN performance criteria: (a) Curves of average PEEN values for different DCT approximations. (b) Corresponding APE

In the same way, the curves obtained of the second proposed transform $\hat{T}_{p2}$ by using different performance measures are shown in Fig.4.5. In view of the fact that the DCT matrices: T_8, T_9 in [35] and $T_{6-cintra}$ [29], with 24 additions and 6-bit shift

operations, have the higher performance than the T_1, we have considered them for the performance comparison of the proposed transform $\hat{T}_{p2}$

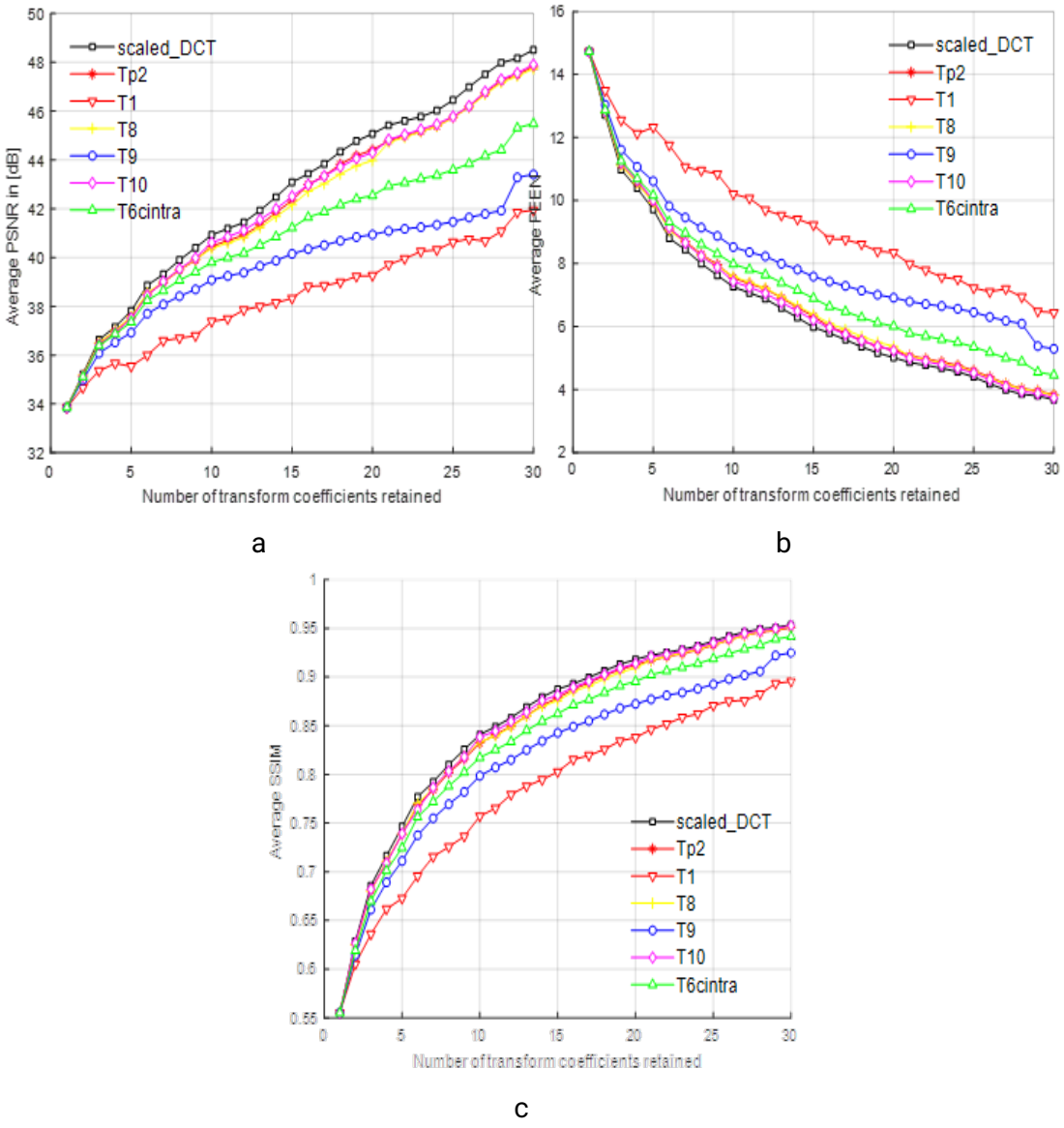


Fig. 4.5: Quality assessment in terms of: (a) PSNR,(b) PEEN, (c) SSIM of proposed transform $\hat{T}_{p2}$ and different DCT approximations

From the PSNR, SSIM, and PEEN variation plots depicted in Fig.4.5, it appears clearly that the compression performance of the proposed transform $\hat{T}_{p2}$ is very close to that Scaled DCT [75] requiring 11.8 % fewer operations. Compared to the transforms T_8, T_9 and $T_{6-cintra}$, the proposed transform $\hat{T}_{p2}$ offers a considerable

superiority in performance in terms of the comparison criteria: PSNR, PEEN, and SSIM, while it kept the same number of arithmetic operations.

In comparison with the T_{10} [38], the proposed transform performed very closely in terms of PSNR, PEEN, and SSIM, while achieving a reduction of 12% in the number of arithmetic operations. Although the proposed transform $\hat{T}_{p2}$ is non-orthogonal it provides the lowest value of deviation from the orthogonality measurement σ and the total error energy, ϵ_{total}, compared to all other existing non-orthogonal transforms as listed in Table 3.4 and Table 4.2, respectively and discussed previously, where it offers better value than that of the maximum value of deviation from the orthogonality considered in [119].

With regard to the visual effects of reconstructed images and by maintaining a compression ratio (CR) up to the CR value recommended by [29, 6], the reconstructed Lena images using the proposed transforms and the other approximate DCT transforms considered in this study are illustrated in Fig.4.7 and Fig. 4.8.

Both proposed approximate transforms produce comparable results to that of the scaled DCT, with no difference in visual perception.

From Fig. 4.6 we can show that our proposed transform $\hat{T}_{p2}$ outperforms the integer approximate Tchebichef transform reported in [3] in terms of total error energy Table 4.2 and PSNR despite they have the same number of arithmetic operations.

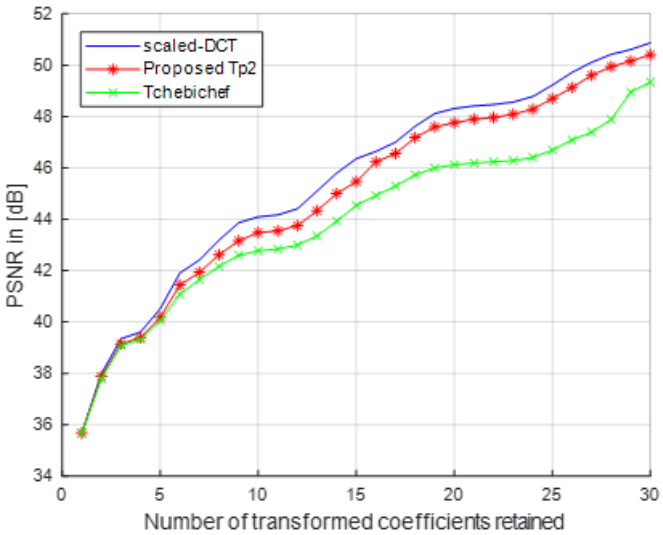


Fig. 4.6: PSNR variation of T_{p2} vs Tchebichef transform for test image Lena

a Original Lena 512×512 image

b Scaled DCT: $PSNR = 44.08dB$

c $\hat{T}_{p2}$: $PSNR = 43.47dB$

d $T_1 : PSNR = 39.905dB$

e $T_8 : PSNR = 43.419dB$

f $T_9 : PSNR = 41.462dB$

g T_{10}: $PSNR = 43.829dB$

h $T_{6-cintra}$: $PSNR = 42.723dB$

Fig. 4.7: Original and reconstructed Lena image using : $\hat{T}_{p2}$ and different DCT approximations

4.3 Proposed transform based on introducing of element nul

4.3.1 First proposed transform

Fig. 4.9 shows the plot of average PSNR, average PEEN, and average SSIM values obtained for the test images using different approximation methods mentioned above.

a T_{p1} : $PSNR = 42.95dB$ b T_1 : $PSNR = 39.91dB$ c T_3 : $PSNR = 42.33dB$

d T_5 : $PSNR = 43.13dB$ e T_7 : $PSNR = 42.21dB$ f T_9 : $PSNR = 41.46dB$

g T_{11} : $PSNR = 42.70dB$ h $T_{4-cintra}$: $PSNR = 42.42dB$

Fig. 4.8: Reconstructed Lena images using T_{p1} and different DCT approximations

Figure 4.9a indicates that the proposed transform and Bas-2011 $\{a = 1\}$ [5] achieve similar results, although our transform retains 11.11% of the number of the required operations. Compared with the same transform with $\{a = 0\}$, our transform shows a slight improvement at middle bit rates (with the number of the coefficients retained ranging from 20 to 27). However, we can notice a significant improvement in performance at high bit rates. Compared to the remaining transforms, the one we propose provides better values of PSNR at all compression ratios, even though it illustrates

the lowest complexity. Note that the transform T_{13} required 14 additions only. Nevertheless, we can notice a decline in performance.
In Fig. 4.9b, we have shown the plot of average PEEN values obtained for different approximation transforms. As can be seen from Fig. 4.9b, the proposed algorithm delivers the best performance in comparison with the most existing methods at all compression ratios. Fig. 4.9c confirms the results discussed above.

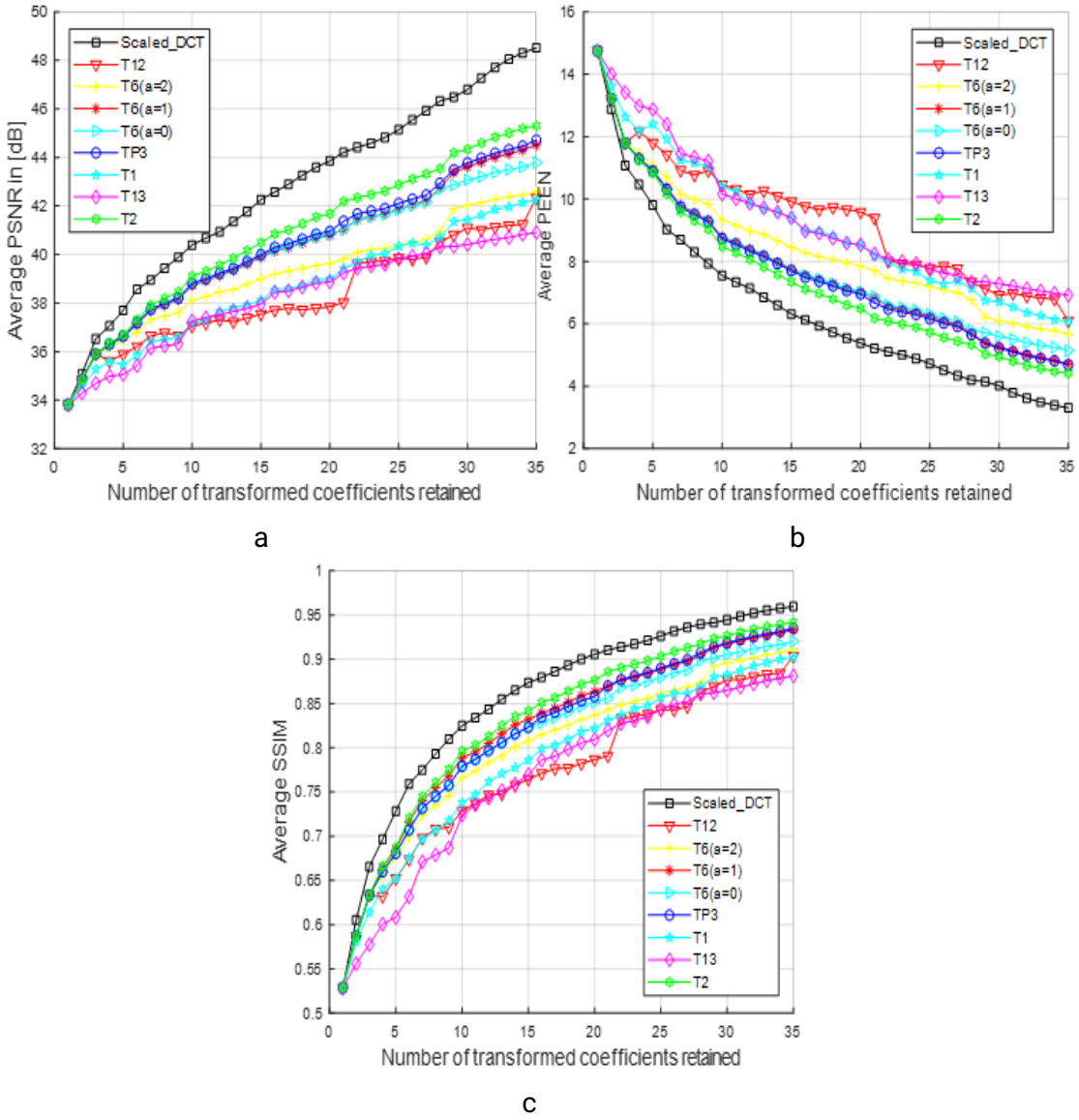


Fig. 4.9: Quality assessment of T_{p3} and other DCT approximations : (a) Average PSNR, (b) Average PEEN, and (c) Average SSIM

Fig. 4.10a illustrates the difference in PSNR between the proposed transform $\hat{T}_{p2}$ and the parametric transform mentioned in [5] with specific parameters $a = \{2, 1, 0\}$

Fig. 4.10b shows the difference between the PSNR obtained by our proposed transform $\hat{T}_{p2}$ and that obtained by the transform in [6]. It is clear from this figure that along with the number of additions reduced by one addition, another still more important improvement in PSNR is achieved by our transform.

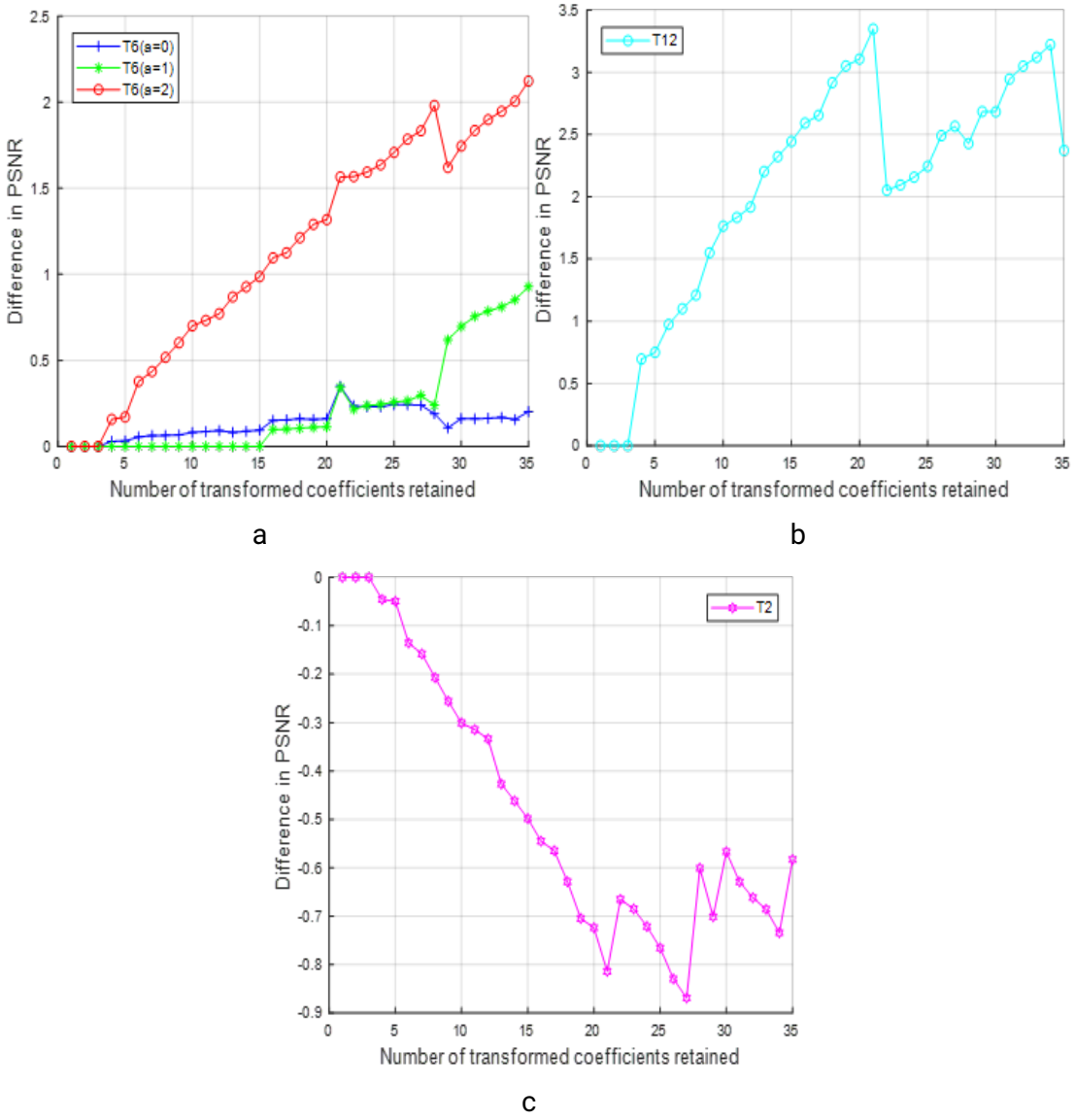


Fig. 4.10: The gain in PSNR of T_{p3} over : (a) T_6 transform in [5] with $a = \{0, 1, 2\}$, (b) T_{12} Transform in [6], (c) T_2 Transform in [7]

As indicated by the above curves, the overall performance of our transform shows an improvement as compared to the previous one. In comparison with the transform in [5] for $a = 2$, requiring 18 additions and 2 bit-shifts, the proposed transform,

with just 16 additions, provides significant improvements in performance in terms of both PSNR and computational complexity, as is evident from Fig. 4.10a and Table 4.3, respectively. With respect to the transform in [5] with $a = 1$ and 18 additions, our transform produces a slight improvement, with only 16 additions. For $a = 0$, without increasing complexity, similar results are displayed at low bit rates (the number of coefficients retained is less than 15). On the other hand, at high bit rates (the number of transformed coefficients retained is greater than 30), our transformation seems to be better.

The difference between our transform and its original transform [7] is shown in Fig. 4.10c. A slight degradation can be observed at the high compression ratio; this degradation is increased when the compression ratio is decreased. And that's due to the fact that our transform presents a reduction of 4 arithmetical operations compared to its original transform.
Table 4.3 summarises some numerical values of the average PSNR resulting from the application of each transform on the test images for different numbers of coefficients retained for the image reconstruction.

Table 4.3: Average PSNR (in dB) of different transforms

Methods	Number of arithmetic operations	Number of coefficients retained			
		05	10	20	30
T_{p3}	**16 additions**	**37.03**	**29.20**	**41.36**	**44.17**
T_2[7]	18 additions and 2 bit-shifts	37.08	39.50	42.09	44.73
T_{12}[6]	17 additions	36.29	37.45	38.26	41.49
T_6 $(a = 2)$ [5]	18 additions and 2 bit-shifts	36.86	38.50	40.03	42.44
T_6 $(a = 1)$ [5]	18 additions	37.00	39.12	41.18	44.02
T_6 $(a = 0)$ [5]	16 additions	**37.03**	**39.20**	41.23	43.47
T_{13} [32]	14 additions	35.85	37.22	39.29	40.42
T_1 [25]	24 additions	35.34	37.56	39.40	41.83
Scaled DCT	29 additions and 5 multiplications	38.08	40.78	44.26	47.19

The PSNR values of the best transform that provides a good compromise between the number of operations and the compression performance are highlighted, as shown in Table 4.3 and Table 4.4. These results provide clear evidence of an improvement in performance by using our transform, in addition to performing a few operations. As compared to the transform in [7], requiring 18 additions and 2-bit

shifts operations, the one we proposed shows a slight decrease in PSNR. In contrast, it generates cost savings in computation, without bit shift operations and with only 16 additions.

As shown in Table 4.3, the transform in [7] outperforms all other transforms, as it has the largest number of arithmetic operations. Compared to our transform, we notice a slight difference that varies from 0.2 dB (for Barbara) to 0.6 dB (for Airplane and Lena). Note that the cost savings in computation associated with our transform is 4 arithmetic operations without bit shifts.

We now compare the performance of the proposed transform with other transforms considered in this study. As Table 4.4 illustrates, the results show a significant improvement over the transform T_12 [6] up to 2 dB, Moreover, even if our transform has the smallest number of operations, it outperforms the parametric transformation of [7].

Table 4.4: Performance assessment of different transforms in terms of PSNR and compression ratio

Methods	Number of arithmetic operations		Test images					
			Lena	Baboon	Barbara	Boat	Airplane	Sailboat
T_{p3}	**16 additions**	CR	**12.06**	**7.45**	**9.75**	**10.58**	**12.22**	**9.46**
		PSNR	**30.96**	**25.00**	**25.19**	**29.90**	**31.35**	**29.66**
T_2[7]	18 additions and 2 bit-shift	CR	12.00	7.37	9.70	10.62	12.04	9.47
		PSNR	31.62	25.34	25.41	30.56	32.01	30.14
T_{12}[6]	17 additions	CR	11.88	7.48	9.71	10.52	12.22	9.51
		PSNR	29.10	23.18	23.68	26.51	27.40	26.21
$T_6\ (a=2)$ [5]	18 Additions and 2 bit-shift	CR	11.71	7.32	9.75	10.44	11.78	9.16
		PSNR	30.06	24.48	24.98	29.94	28.56	29.26
$T_6\ (a=1)$ [5]	18 additions	CR	12.06	7.35	9.67	10.48	12.00	9.40
		PSNR	30.92	25.06	25.31	29.94	28.56	29.26
$T_6\ (a=0)$ [5]	**16 additions**	CR	**12.06**	7.42	9.69	10.54	12.06	9.42
		PSNR	**30.96**	25.03	25.23	29.28	31.14	29.58
T_{13} [32]	14 additions	CR	12.56	7.41	9.55	10.62	12.11	9.44
		PSNR	30.96	25.03	25.23	29.28	31.14	29.28
T_1 [25]	24 additions	CR	11.96	7.49	9.44	10.68	12.14	9.58
		PSNR	28.76	23.70	23.67	27.76	28.94	27.38
Scaled DCT	29 additions and 5 multiplications	CR	11.96	7.54	9.50	10.54	12.15	9.44
		PSNR	34.98	26.18	26.37	33.26	35.32	32.56

Finally, a subjective evaluation is shown in Fig. 4.11 for the Lena test image. The image has been subjected to a JPEG-like compression experiment. Only 5 of the 64-transformed coefficients in each block were used to reconstruct the image, all other coefficients were set to zero. This value is already considered in a previous work [2]. As may be noted, the reconstructed image using the proposed algorithm produces fewer blocking artifacts than the methods in [2]. The quality of the image reconstructed using the transform in [2] diminished, even though the number of operations has been reduced by two compared to the proposed one.
As a result, the proposed transform offers a good compromise between performance and complexity.

a

b T_{p3} : $PSNR = 27.05dB$

c T_{13} : $PSNR = 24.77dB$

d T_{12} : $PSNR = 26.69dB$

e T_6 $\{a = 2\}$: $PSNR = 26.69dB$

f T_{p1} : $PSNR = 25.71dB$

Fig. 4.11: The original and reconstructed Lena image using proposed T_{p3} transform and different DCT approximation

4.3.1.1 Complexity analysis in terms of comparing the running time of the forward transformation

The proposed fast algorithm and the algorithm presented in [7] were compared in terms of running time. The comparison was conducted using MATLAB programming language on a platform consisting of an Intel Core I5-8250U Processor ×64 at 1.6 GHz and 8 GB memory.
The forward transform was performed on three Lena greyscale images of sizes 256×256, 512×512, and 1024×1024, and the average values obtained were presented in Table 4.5 The results in the table confirm our previous findings regarding arithmetic operations, as the proposed algorithm outperforms the other algorithm in terms of running time. This is because the proposed transform requires fewer computations.

Table 4.5: Comparison of the Running Time between the Proposed Transform T_{p3} and its Original Version

Transforms	Lena image size		
	256× 256	512 × 512	1024 × 1024
T_{p3}	**0.0111**	**0.0446**	**0.317**
T_2	0.0120	0.0462	0.0331
T_{12}	0.0116	0.0448	0.0320
Scaled DCT	0.0908	0.433	1.869

4.3.1.2 Comparison of Performance with Hadamard Transform

In this particular section, we are comparing the compression performance of the proposed transform T_{p3} with the 8-point Hadamard transform [121]. A similar experiment to the one described in section 4.1 is performed using the Hadamard transform [121], our proposed transform, and the conventional DCT transform. The performance comparison is presented in Fig. 4.12, based on the metrics of PSNR, PEEN, and SSIM. The results indicate a clear improvement in the performance of the proposed transform as compared to the Hadamard transform, with only 16 additions performed instead of the 24 required by the Hadamard transform.

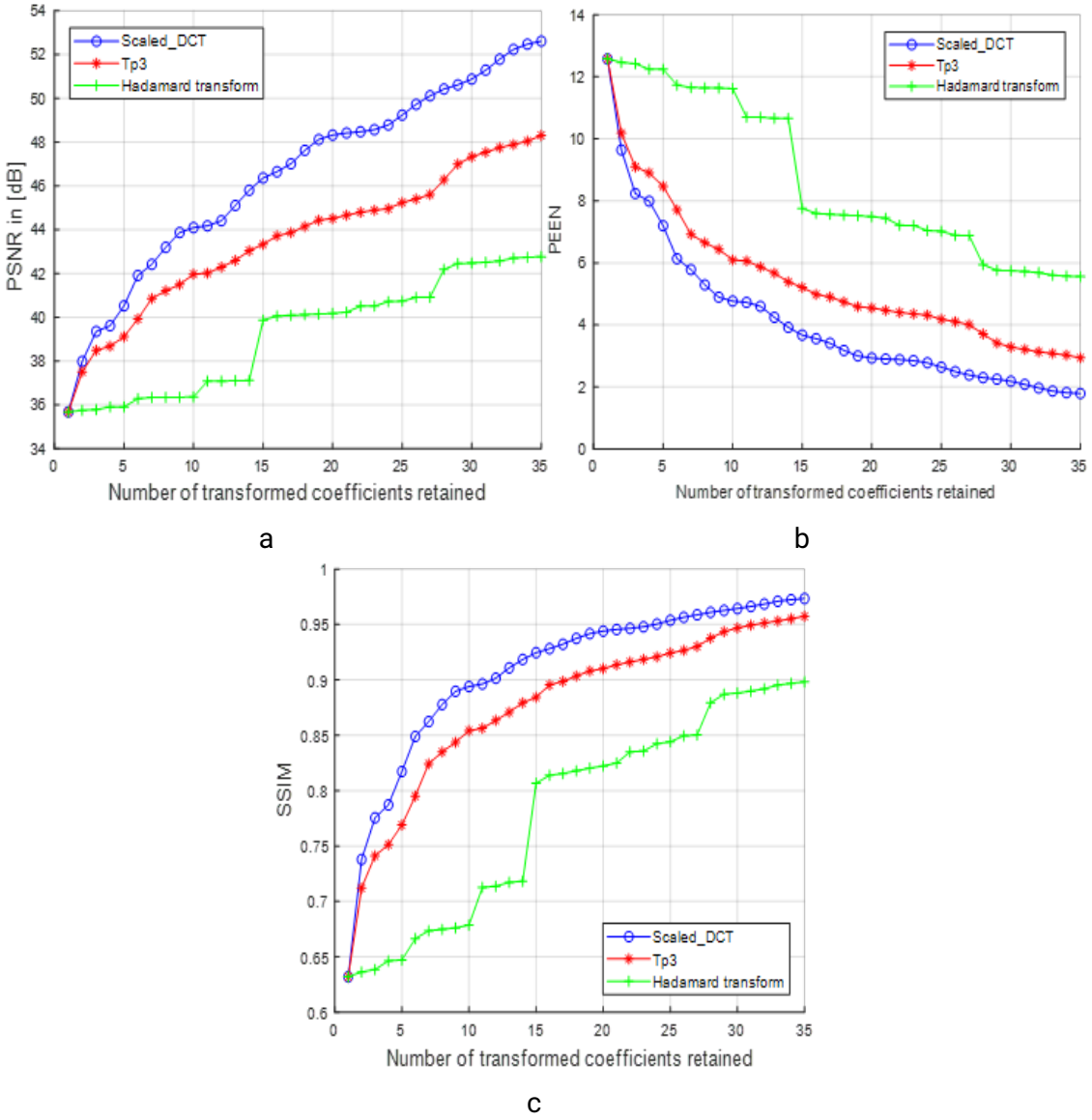


Fig. 4.12: Evaluation of T_{p3} Vs Hadamard transform for image Lena in terms : (a) PSNR (b) PEEN (c) SSIM

4.3.2 Second proposed transform

In Fig. 4.13, the variations in performance criteria PSNR, PEEN, and SSIM for the "Lena" test image are plotted. At low bit rates where the number of spectral coefficients retained is less than 15, the proposed transform, T_{p4}, is similar in performance to transforms T_1, T_3, T_7, T_{12}, and T_9, as seen in Fig. 4.13a. However, there is a slight decrease in performance compared to transform T_{10} and due to a reduction in additional operations. At middle and high bit rates, the proposed transform outperforms

all the other transforms except for the T_{10} transform, with a difference in PSNR performance of less than 1 dB, even though our transform has 6 fewer additions.

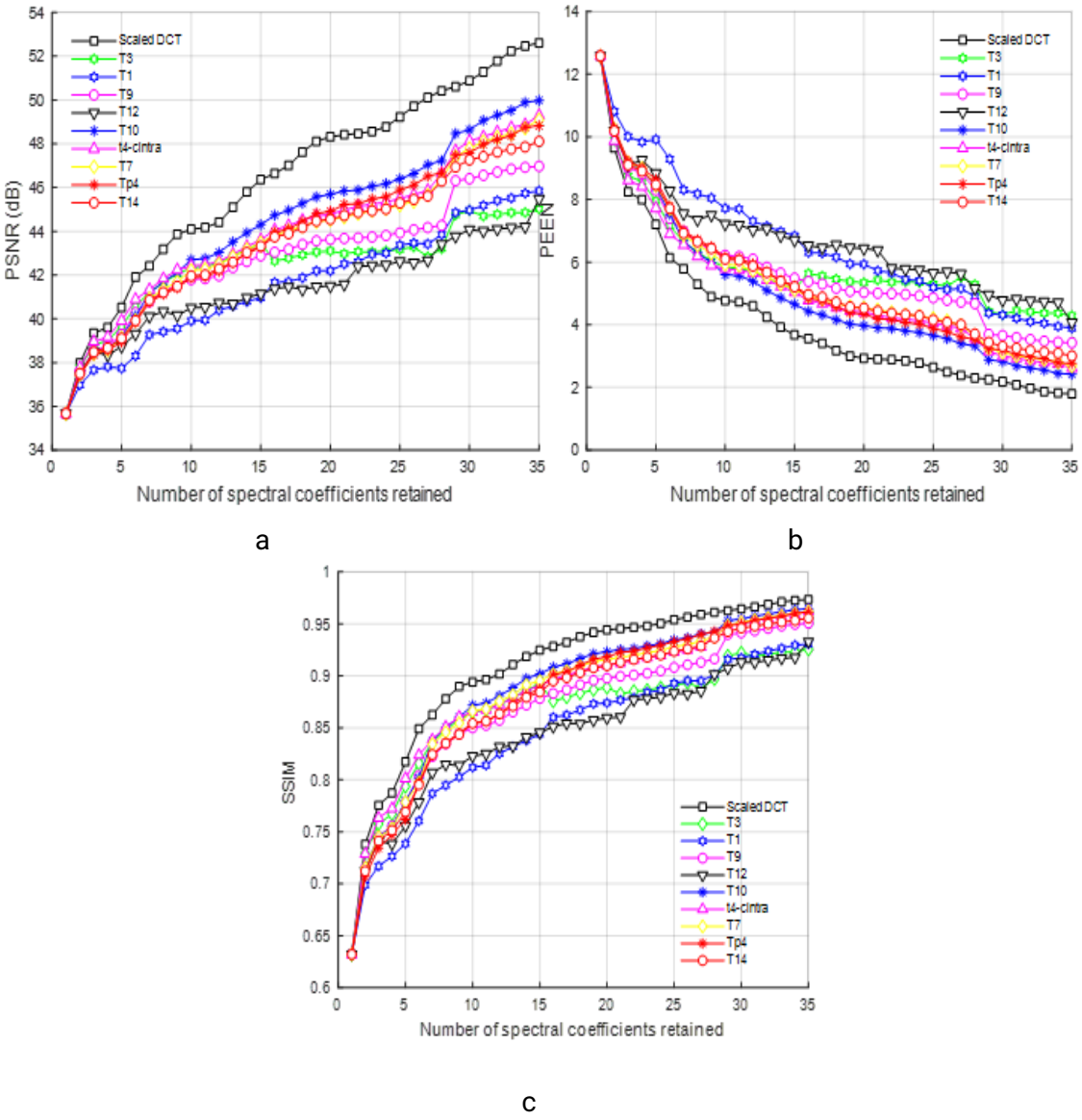


Fig. 4.13: Objective quality assessment of T_{p4} for image Lena in terms of: (a) PSNR (b) PEEN (c) SSIM

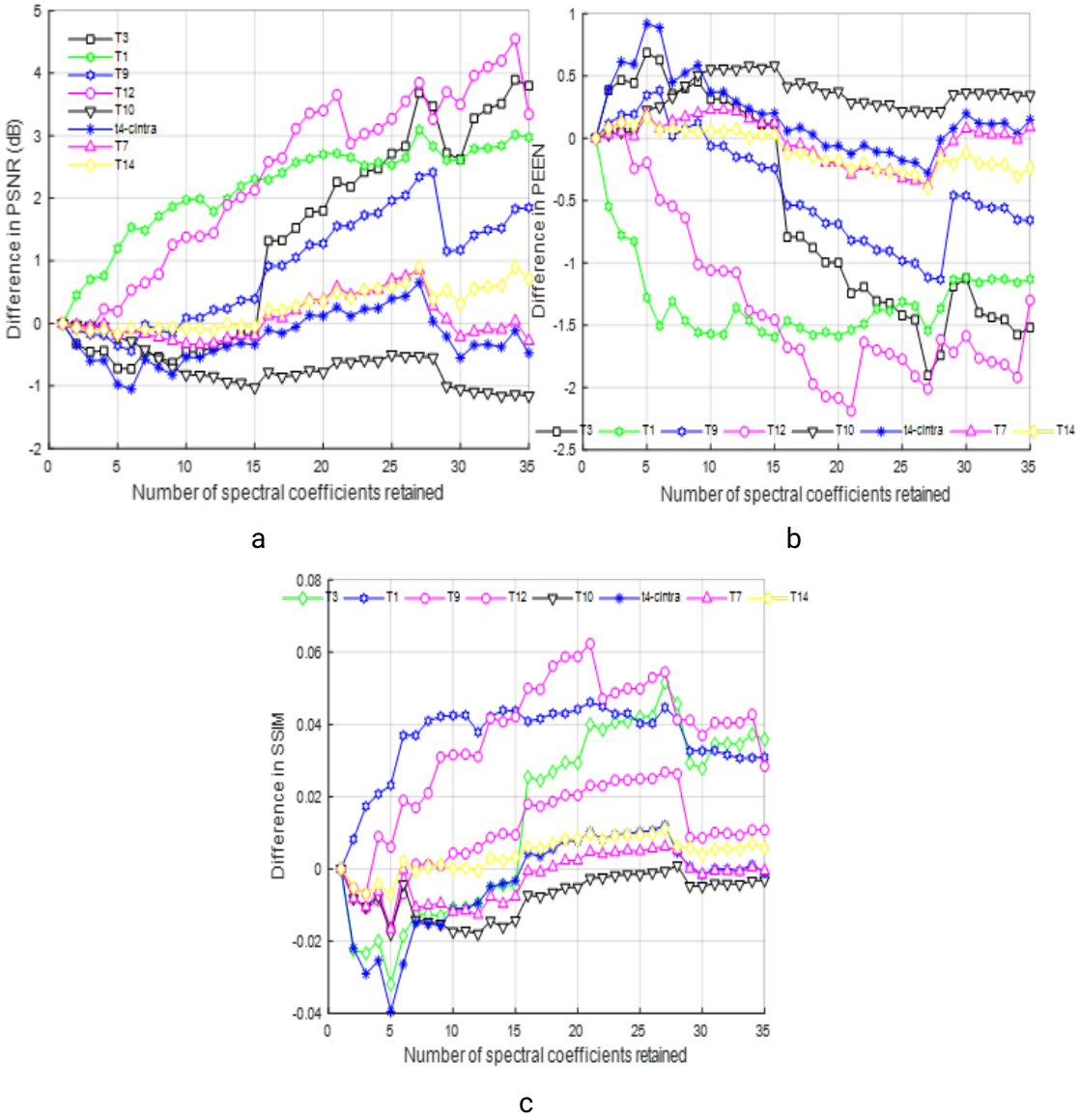


Fig. 4.14: Difference between the proposed and other DCT approximations for test image 'Lena' in terms of: (a) PSNR (b) PEEN (c) SSIM

Based on Fig. 4.13b, when the number of coefficients retained is below 15 (high bit rate), there is minimal variation in PEEN between T_{p4} and T_{14} However, at different bit rates, our proposed transform outperforms the T_{14} transform. It's worth noting that both transforms involve the same number of additions, except that our transform employs more bit shift operations. In practical implementation, most adders and shifters function concurrently, resulting in a decrease in delay.
By examining Fig. 4.13c and Fig. 4.14c, it becomes apparent that the proposed transform T_{p4} yields superior SSIM values compared to the other transforms, consistent

with the earlier findings.
To provide an objective assessment of the results, we focus on the disparity between the proposed transform T_{p4} and other approximate transforms. The results are presented in Fig. 4.14, where it can be seen from Fig. 4.14a that the T_{p4} transform surpasses most of the rival approaches in PSNR performance. T_{p4} outperforms T_1, T_3, T_9, and T_{12} transforms across all compression ratios, with significant gains of approximately 3 dB, 4 dB, 2.5 dB, and 5 dB, respectively.

Based on the discussion, it was concluded that T_{12} [6], T_{10} [38], T_{14} [106], and T_9 [35] are the approximations that strike a better balance between performance and complexity. The non-orthogonal transforms in [25] and [104] were excluded from consideration due to their limited performance in image compression. It is worth noting that while the fast algorithm for the inverse transform in [104] requires only 21 additions and 3 bit-shift operations, no algorithm was proposed in [25] for efficiently computing the inverse SDCT matrix.
Table 4.6 provides a summary of the numerical results obtained from implementing the proposed transform, T_{p4}, and the recent approximations discussed earlier under the JPEG-like standard. These results are based on PSNR and compression ratio (CR) values. To ensure a fair comparison, we calculated the PSNR value for each transform using an equal ratio.

Table 4.6: Comparison of Compression Performance among Several Approximate DCT Transforms

Image		Lena	Boat	Baboon	Airplane	Bridge	Cameraman
Scaled DCT	PSNR	33.91	32.24	24.05	33.33	27.02	36.33
	CR	12.89	11.57	8.89	12.33	08.16	13.88
T_{12}[6]	PSNR	28.77	26.32	21.69	26.30	23.23	27.78
	CR	12.55	11.56	08.31	12.22	08.19	13.91
T_{10} [38]	PSNR	31.59	29.68	23.55	29.27	26.10	31.93
	CR	12.55	11.81	08.35	12.22	08.21	13.80
T_{14} [106]	PSNR	30.73	29.14	23.15	29.28	25.43	30.80
	CR	12.32	11.53	8.26	12.45	08.29	13.70
T_9 [35]	PSNR	30.57	29.10	23.44	29.30	25.66	30.62
	CR	12.82	11.61	8.38	12.46	08.23	13.64
T_{p4}	PSNR	30.71	29.16	23.33	29.75	25.77	30.73
	CR	12.72	11.57	8.28	12.21	08.14	13.75

Table 4.6 displays the comparison of compression performance among several approximate DCT transforms, indicating that the proposed transform offers comparable compression quality to [6] for all test images. Conversely, the non-orthogonal

SDCT transform demonstrates the lowest image quality for all images considered. Furthermore, the proposed transform generally shows improved performance, achieving a cost complexity reduction of 6 additions without compromising image quality, when compared to the transform in [35]. When a single addition decrement is made, the DCT transform's performance evaluation shows a decline compared to the proposed T_{p4} transform for all test images. Moreover, the proposed transform outperforms the T_{14} transform in terms of PSNR for most of the test images, with the same number of addition operations.
The subjective evaluation in image compression applications considers different numbers of retained coefficients. For instance, in [122], the authors found that only 16 coefficients are necessary for accurate image reproduction.
The study conducted a subjective evaluation of image reconstruction using different transforms, which is presented in Fig. 4.15. The reconstruction was performed using only 16 out of 64 spectral coefficients from each block for the test image, Lena. The results show that the proposed transform produces fewer blocking artefacts compared to the transforms used in previous studies [35] and [104]. Based on this finding, the study concludes that the proposed transform provides a favorable balance between performance and arithmetic complexity.

An enlarged view of a section of the reconstructed test image 'Lena' is presented in Fig. 4.16 for the three transforms: T_9, T_{12}, T_{14} and T_{p4}. It is evident from this figure that the image reconstructed by our transform has fewer blocking artefacts compared to the other two transforms. This suggests that our transform outperforms the other two transforms in terms of blocking artefact reduction.
**** Other test images
Additional test images have been introduced in this section to validate the results, as only the reference image Lena was utilised in previous sections to evaluate the proposed transform.

The figures: Fig. 4.17, Fig. 4.18, and Fig.4.19 depict the curves illustrating the variations in objective quality assessment for the test images Boat, Bridge, and Cameraman. The assessment criteria include PSNR, PEEN, and SSIM. When considering the Bridge image, which exhibits a lower predictability level (SAM=134.48), the proposed transform demonstrates a remarkable similarity to other transforms for high and medium compression ratios, while surpassing them in other cases.

a

b PSNR=31.28 dB, CR =11.71

c PSNR=31.88 dB, CR =12.31

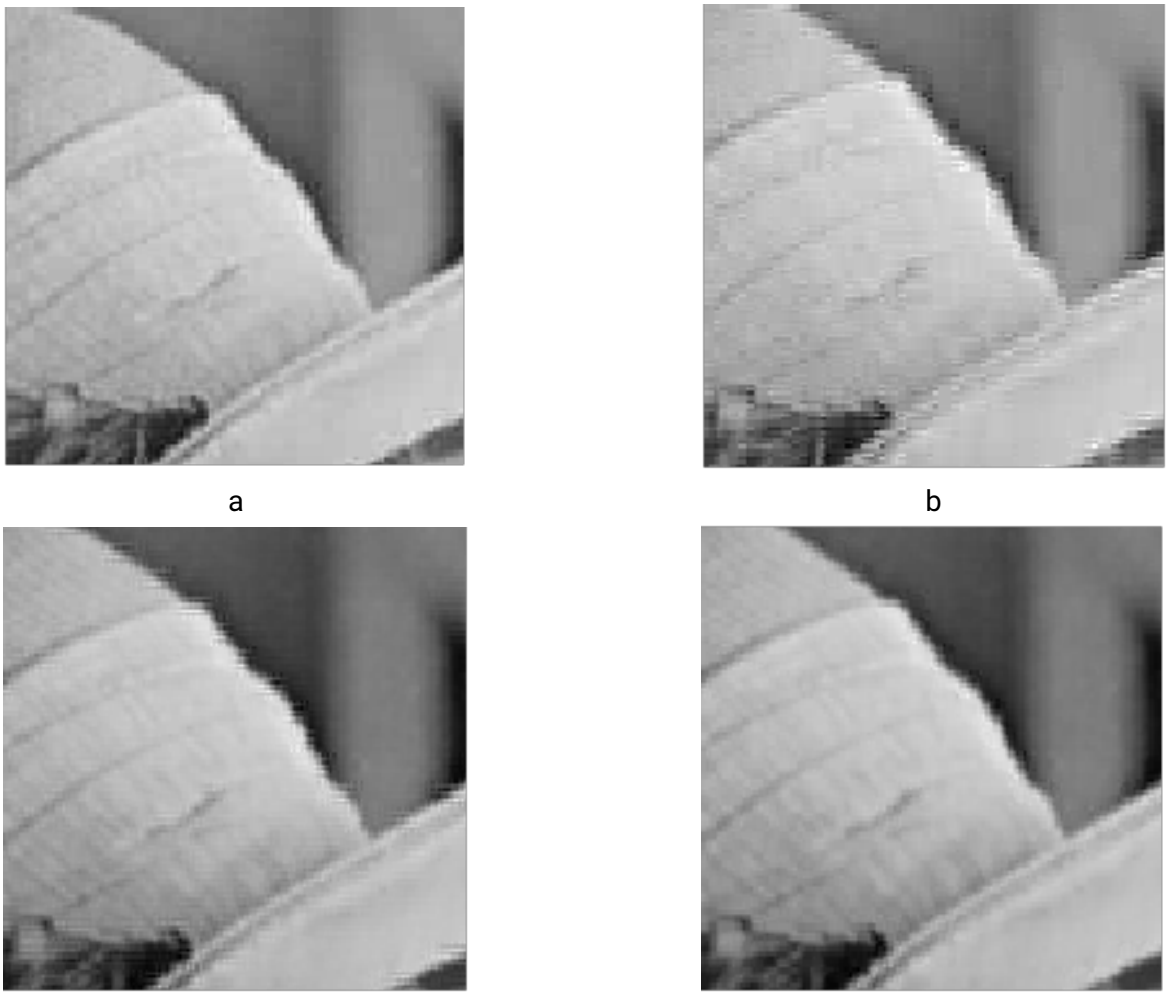
d PSNR=29.11 dB, CR =11.90

e PSNR=31.28 dB,
CR =11.71

f PSNR=30.64 dB,
CR =12.78

g PSNR=31.46 dB,
CR =12.26

Fig. 4.15: The reconstructed Lena image using various transforms: (a) Original image, (b) Scaled DCT, (c) T_{10}, (d) T_{12}, (e) T_{14}, (f) T_{9}, (g) T_{p4}



a b c d

Fig. 4.16: An enlarged view of an area of the Lena test image using: (a) T_{14}, (b) T_{12}, (c) T_9 , (d) T_{p4}

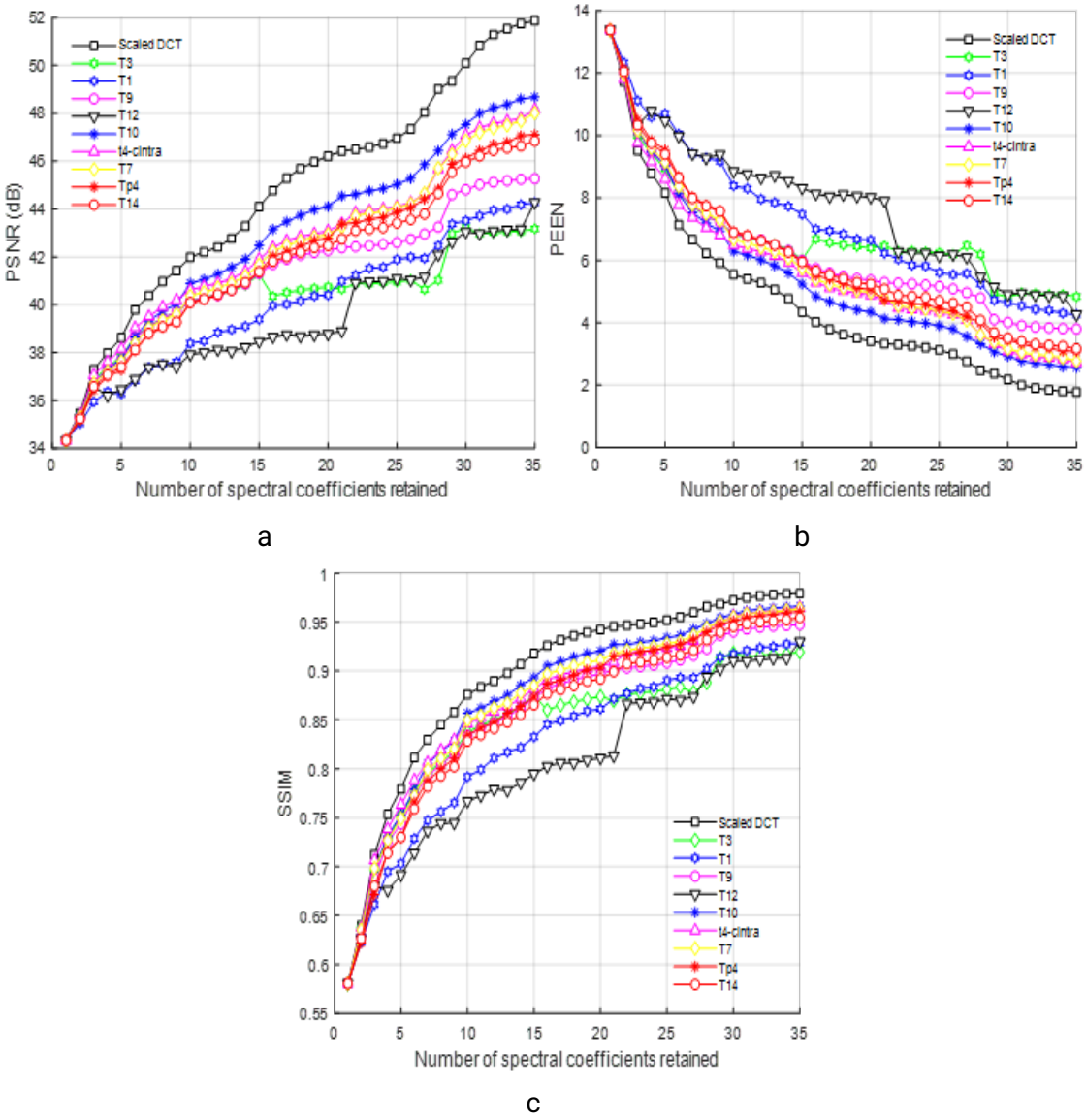


Fig. 4.17: Objective quality assessment in terms of (a) PSNR, (b) PEEN, and (c) SSIM of proposed transform T_{p4} for the test image Boat

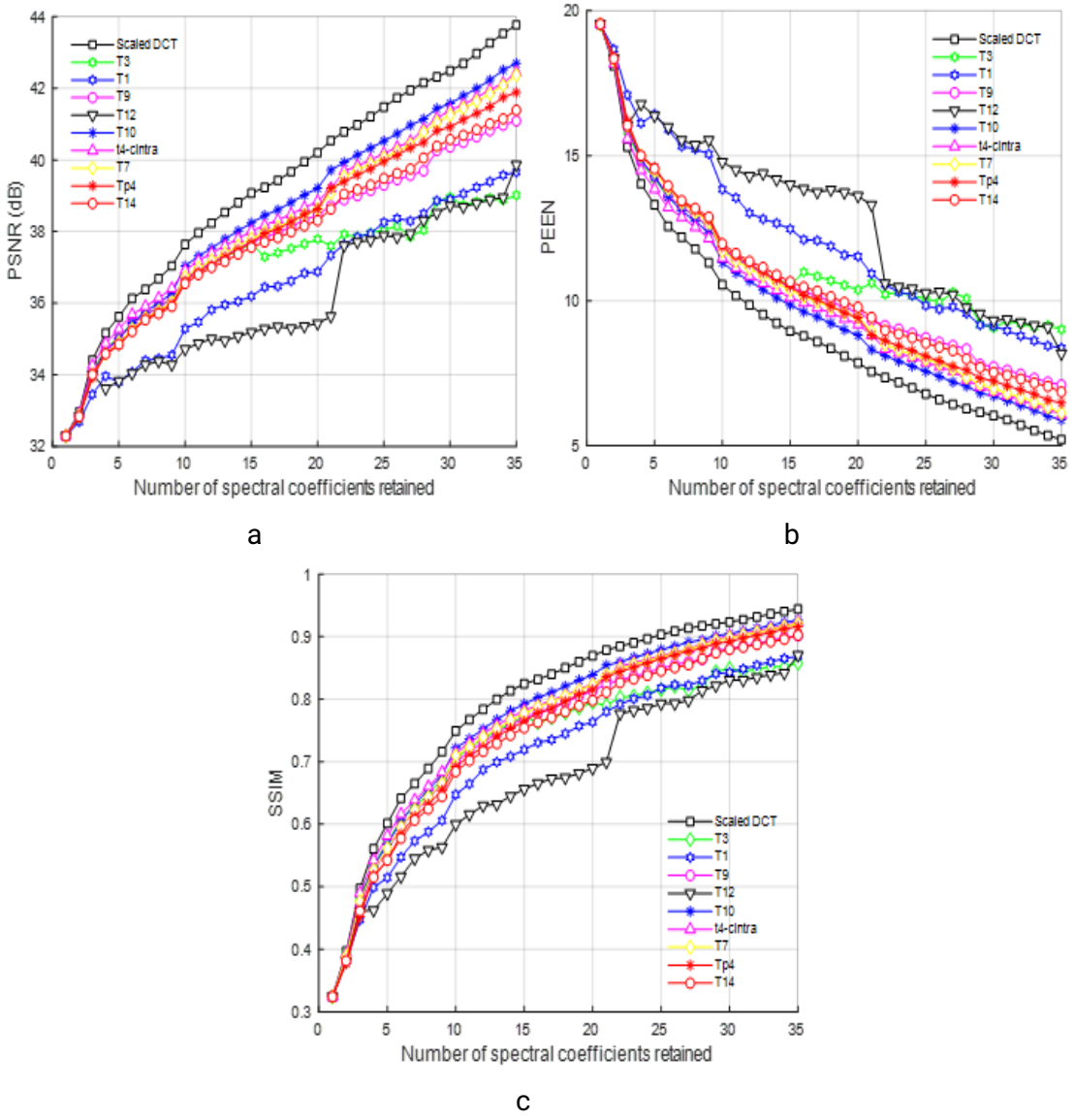


Fig. 4.18: Objective quality assessment in terms of : (a) PSNR, (b) PEEN, and (c) SSIM of proposed transform T_{p4} for the test image Bridge

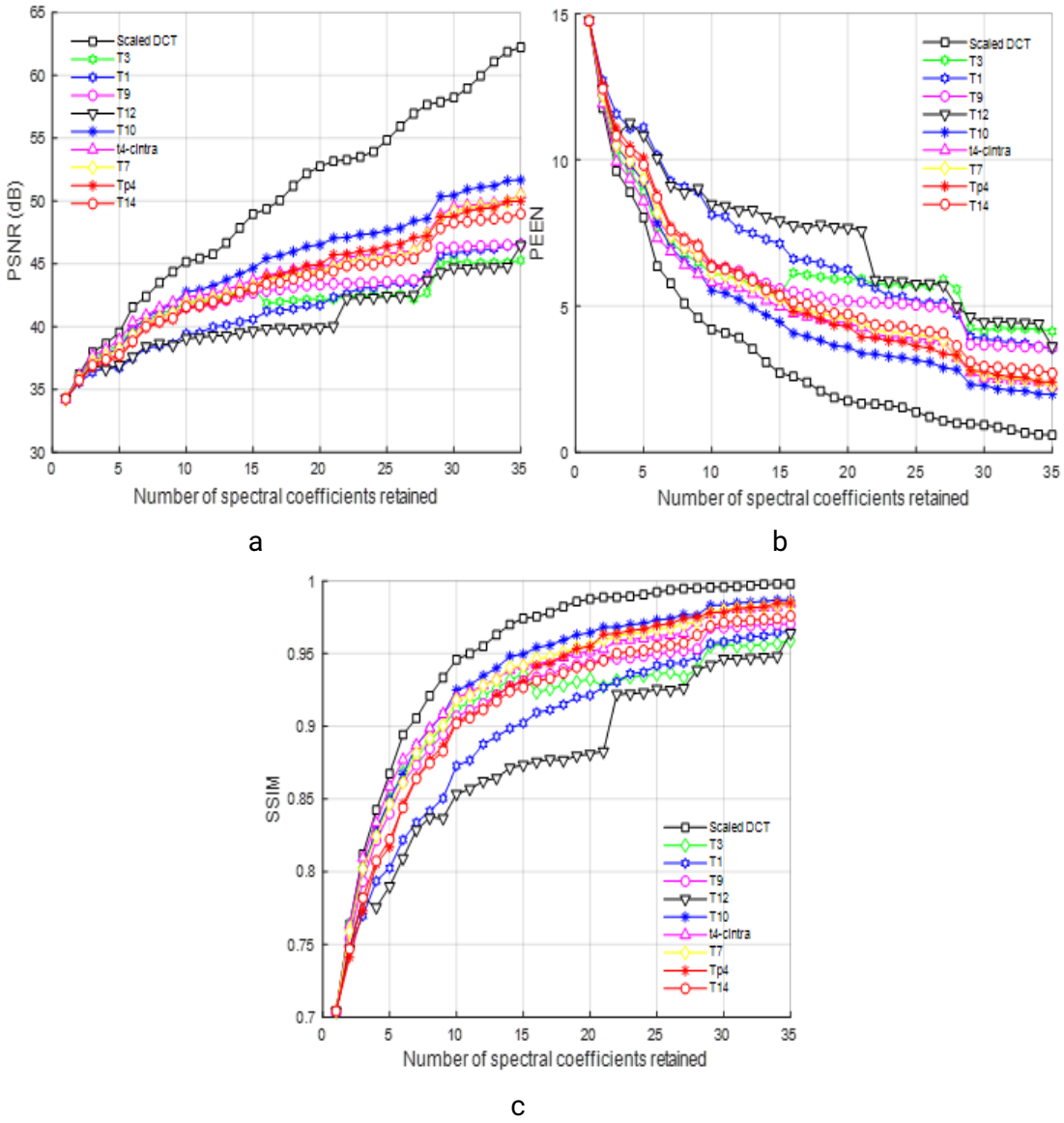


Fig. 4.19: Objective quality assessment in terms of (a) PSNR, (b) PEEN, and (c) SSIM of proposed transform T_{p4} for the test image Cameraman

4.4 Performance comparison of proposed transform versus Hadamard and Tchebichef transforms

Two other transforms which have recently been used in the field of compression are the Walsh – Hadamard transform (WHT) and the Tchebichef transform. These two linear image transforms are chosen because of their flexibility, energy compaction, and decorrelation properties [3, 123, 124]. In this section, we compare the compression performance of the proposed transform to the 8-points Hadamard transform in [121] and Tchebichef transforms T_{Tch1} and T_{Tch2} in [3, 109], respectively.

The arithmetic complexity of these transforms is summarised in Table 4.7. Note that the two Tchebichef transforms matrices are not orthogonal. However, the multiplication of the forward and inverse matrices has a deviation from the diagonality.

Table 4.7: A cost comparison of arithmetic operations with the Hadamard and Tchebichef transforms

Transform	One dimension			
	add.	Shift	Mult.	Total
Hadamard	24	0	0	24
T_{Tch1}	20	0	0	20
$(T_{Tch1})^{-1}$	29	8	0	37
T_{Tch2}	24	6	0	30
T_{p4}	18	6	0	24

We perform a similar experiment to that described in section 4.1, using these two transforms, with our proposed transform and the conventional DCT transform. The performance comparison is provided in terms of PSNR, PEEN, and SSIM, for Lena test image. The results obtained are illustrated in Fig. 4.20. In comparison with the Hadamard transform, requiring 24 additions, the results of our study show significant improvements in the visual performance of the proposed transform. According to the results presented in Fig.4.20, the proposed integer DCT significantly improves the overall compression performance of the approximate Tchebichef transform proposed in [3] while it takes advantage of the significantly lower operating costs compared to the approximation non-orthogonal of Tchebichef in [109]. In addition, a degradation of about 1 dB in terms of PSNR is obtained by our transformation compared to the Tchebichef approximation introduced in [3], which can be interpreted by the significant reduction in the arithmetic computation of our proposed transform. At low bit rates, we can observe from the SSIM and PEEN plots, that the results of our transform are close to those in [3].

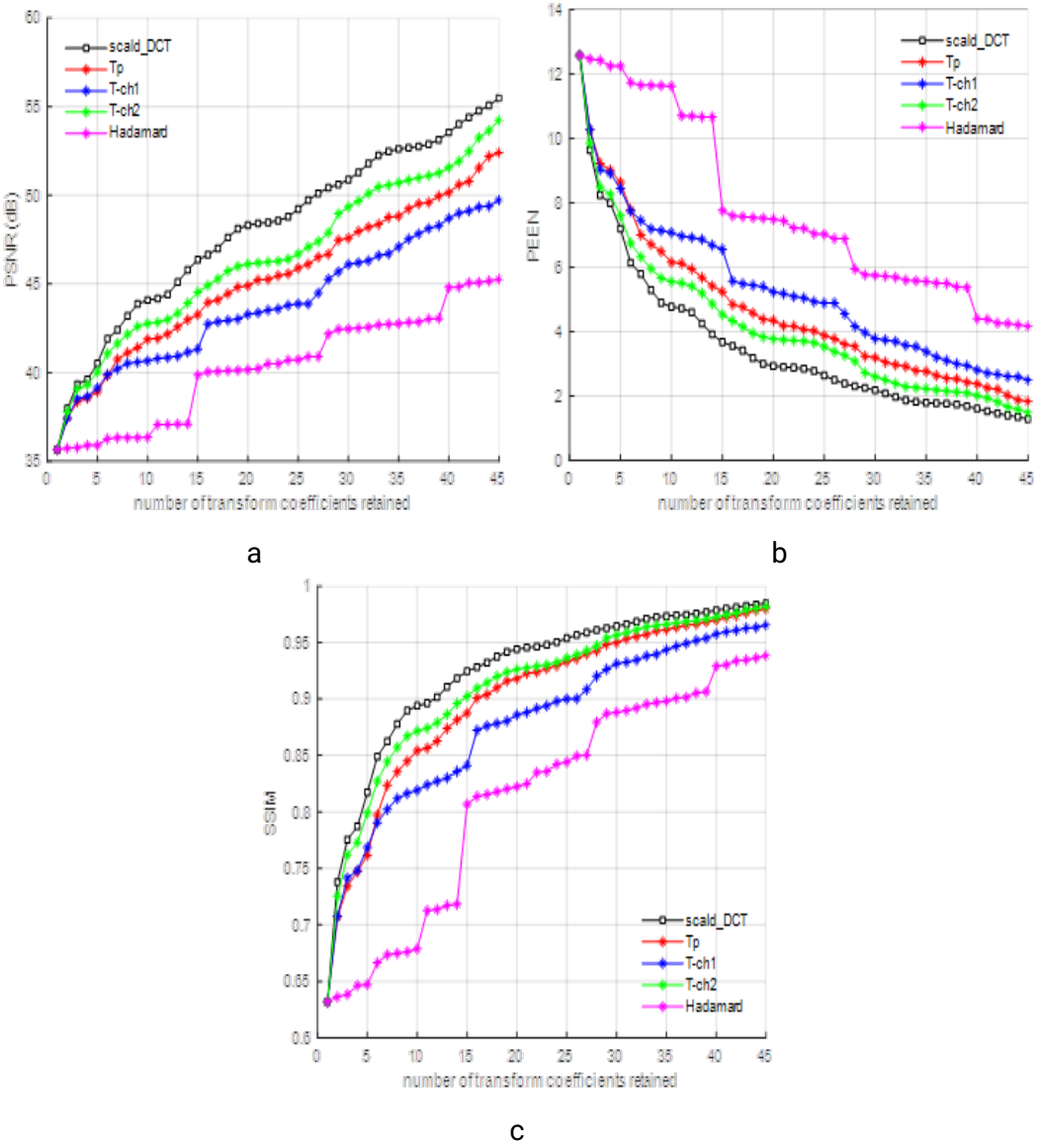


Fig. 4.20: Objective quality assessment in terms of (a) PSNR, (b) PEEN, and (c) SSIM of proposed transform T_{p4} for the Lena test image

4.5 Integer DCT-based JPEG quantisation matrix

For an appropriate assessment of our proposed quantisation matrix in an image compression application, the simulation of the JPEG-like standard is considered. Four different scenarios summarized in Table 4.8 have been applied to the three gray-scale test images Lena, Boat, and Bridge. These images are chosen for their differences in the type of content and the amount of detail they contain, which is

indicated by their Spatial Frequency Measure (SFM) and Spectral Activity Measure (SAM) parameters.

Kipping the same values of Compression Ratio (CR) for all different considered scenarios, the resultant images quality criteria PSNR is given in Table 4.8. From Table 4.8, it can be shown that the high PSNR values are delivered when we used the proposed bit-shift approximate quantisation matrix,$\tilde{Q}_{P-DCT}$, combined with the integer DCT transform suggested in [1] for all considered test images. It can offer an average gain in PSNR until about 2 dB. A slight improvement can be shown using the approach suggested in [42], about 0.3 dB compared to the conventional JPEG approach.

Table 4.8: Image quality comparison in terms of PSNR for different test images

Test Image	CR	Quantisation Matrix	PSNR (dB)	PSNR Improvement (dB)
Lena	14.70	Q_{JPEG}	29.81	-
	14.56	Q_{P-DCT}	31.08	1.27
	14.65	$\tilde{Q}_{P-DCT}$	32.03	2.22
	14.86	$\hat{Q}_{Oliveira}$	29.84	0.03
Boat	12.16	Q_{JPEG}	28.76	-
	11.92	Q_{p-DCT}	29.83	1.07
	12.10	$\tilde{Q}_{P-DCT}$	31.14	2.38
	12.09	$\hat{Q}_{Oliveira}$	29.14	0.38
Bridge	7.72	Q_{JPEG}	25.70	-
	7.67	Q_{P-DCT}	26.31	0.61
	7.66	$\tilde{Q}_{P-DCT}$	27.29	1.59
	7.81	$\hat{Q}_{Oliveira}$	25.77	0.07

4.6 Integer DTT-based JPEG quantisation matrix

For a fair assessment of our proposed quantisation matrix of greyscale image compression, the JPEG-like standard is considered. The image compression /decompression system follows the framework of the DTT-based JPEG standard, with three important steps: transform (exact DTT or integer approximate DTT transforms), quantisation and entropy coding.

For each transform, at the same compression ratio, we compute the PSNR using the standard quantisation matrix JPEG, Q_{JPEG}, the optimal quantisation matrix, Q_{opt} in [117] for exact DTT-based JPEG standard and the proposed quantisation matrix, Q_{P-DTT}. The three well-known greyscale test images of size 512×512: Lena, Boat, and Bridge were used for performance comparison purposes. These images

Table 4.9: Performance evaluation of the proposed quantisation matrix

Transforms	Quantisation matrices	Lena		Boat		Bridge	
		CR	PSNR	CR	PSNR	CR	PSNR
Exact DTT	Q_{JPEG}	13.01	29.80	11.71	28.31	08.29	24.49
	Q_{opt}	13.06	29.77	11.75	28.41	08.29	24.51
	Q_{P-DTT}	13.01	29.69	11.62	28.32	08.21	24.45
DTT apprx. in [3]	Q_{JPEG}	13.04	31.09	11.74	29.59	08.32	25.70
	Q_{opt}	13.23	31.10	11.92	29.61	08.37	26.02
	Q_{P-DTT}	13.33	31.77	11.86	30.01	08.37	26.02
DTT approx. in [4]	Q_{JPEG}	13.11	31.94	11.63	30.55	08.22	26.22
	Q_{opt}	12.84	32.31	11.78	30.59	08.30	26.23
	Q_{P-DTT}	12.86	**33.40**	11.70	**31.19**	08.28	**26.59**

are chosen for their different content and the amount of detail they contain, which is indicated by their parameters: spatial frequency measure (SFM) and spectral activity measure (SAM). Table 4.9 shows the PSNR values obtained using similar compression ratio values for the various cases considered. When using the exact DTT-based JPEG baseline, the PSNR values of our proposed quantisation matrix are slightly lower than those of other existing quantisation matrices. For all test images, the PSNR values again for the approximate DTT-based JPEG, using the proposed quantisation matrix, are higher than those of the other quantisation matrices. For instance, an improvement of about 0.6 dB is obtained when we compare our proposed quantisation matrix with the optimal quantisation matrix Q_{opt} proposed in [117].
Fig. 4.21 shows the reconstructed image using the exact DTT and approximate DTT for different quantisation matrices. As can be seen from Fig. 4.21, the quality of the reconstructed images using the quantisation matrices Q_{opt} or Q_{P-DTT} is slightly better than that of Q_{JPEG} when employing the exact DTT transform in the JPEG standard. Moreover, by using the approximate DTT [4] in JPEG standard, it can be shown that the visual quality of the images reconstructed with the proposed quantisation matrix is superior to that of the two alternative quantisation matrices. Note that the transform in [4] requires a smaller number of arithmetic operations compared to the exact DTT transform.

a Exact DTT, Q_{JPEG}, CR=17.84, PSNR=29.19 dB

b Exact DTT, Q_{opt}, CR=17,35, PSNR=29.42 dB

c Exact DTT, Q_{P-DTT}, CR=17.85, PSNR=29.42 dB

d DTT in [4], Q_{JPEG}, CR=17.87, PSNR=29.64 dB

e DTT in [4] , Q_{JPEG}, CR=17.39, PSNR=30.56 dB

f DTT in [4], Q_{P-DTT}, CR=17.92, PSNR=30.65 dB

Fig. 4.21: The reconstructed Lena image by different quantisation matrices

4.7 Conclusion

The results obtained in our research were presented in this chapter. The implementation of our strategy in the JPEG still image compression standard shows the superiority of our proposed integer transforms over other transforms in the literature. The combination of our transforms with the proposed appropriate quantization matrix ensures an improvement of the JPEG compression standard in terms of the number of operations as well as the quality of the reconstructed images.

Conclusion and Perspectives

We proposed two new approaches for developing low-complexity approximations of the integer 8-point DCT. These approaches aim to simplify the computation of the DCT while maintaining its accuracy, which can be useful in applications such as image and video compression.

Based on 16-floating-points DCT matrix and rounding-off operations: this approach uses a 16-floating-points DCT matrix and rounding-off operations to achieve two integer 8-point DCT transforms with low computational complexity and without multiplication operations. The proposed transforms outperform popular alternatives in terms of compression performance criteria and computational complexity. For example, in our tests on image compression, the proposed transforms achieved a higher compression ratio than the popular DCT transforms while requiring less computational resources. These results demonstrate the potential of the proposed transforms for practical applications in image and video compression.

Successful introduction of some zeros in existing transforms: the second approach involves introducing some zeros into existing transforms, resulting in a new transform that performs better than existing approximate transforms with the same number of operations, and even some transforms with more operations. This new transform requires only 16 additions, and an efficient algorithm has been developed for its computation. The other approach results in a new transform that provides improved image compression performance compared to other transforms with similar arithmetic operations. While it performs similar operations to its counterparts, this transform produces higher-quality output images. Our proposed methods save computational cost compared to the original transform, with a reduction of 4 additions, and an efficient and fast algorithm has been developed, making the new transform suitable for hardware implementation.

We also propose a new multiplier-less JPEG quantisation matrix based on the energy distribution of existing 8×8 integer DCT approximations. This matrix aims to improve integer DCT-based JPEG standard efficiency and eliminate rounding op-

erations that occur during the quantisation stage. Our proposed quantisation matrix improves the capability of image compression by providing high PSNR improvement compared to existing quantisation matrices.

Specifically, we developed the quantisation matrix by analysing the energy distribution of existing 8x8 integer DCT approximations. Our matrix is designed to allocate quantisation step sizes according to the energy distribution of the DCT coefficients, which can result in more efficient compression. We tested our matrix on a variety of images and found that it consistently outperformed existing quantisation matrices in terms of PSNR improvement. In summary, our proposed multiplier-less JPEG quantisation matrix offers a promising approach for improving image compression efficiency and maintaining high image quality.

To design an efficient approximate DTT-based quantisation matrix, we investigated the energy distribution of coefficients in existing integer DTT approximations. Our experiments show that combining integer DTT approximations with the proposed quantisation matrix outperforms the DTT based on the optimal quantisation matrix by an average of about 0.6 dB in terms of PSNR. Additionally, compared to the standard JPEG quantisation matrix, our approach achieves an improvement of more than 0.43 dB.

Future work encompasses incorporating the suggested transformations into real-time applications and the domain of video compression. Additionally, another interesting perspective involves adapting these transforms for data cryptography purposes. Such an adaptation would enable the exploration of the potential applications of these transformations in data security and confidentiality protection. This research aims to investigate the performance of the transforms in the context of cryptography by evaluating their resistance to attacks and their ability to ensure the integrity and confidentiality of sensitive data. This exploration thus presents opportunities for enhancing data security while capitalising on the advantages offered by the proposed transforms.

library

[1] Nabila Brahimi, Toufik Bouden, Tahar Brahimi, and Larbi Boubchir. A novel and efficient 8-point dct approximation for image compression. *Multimedia Tools and Applications*, 79(11-12):7615–7631, 2020.

[2] Renato J Cintra and Fábio M Bayer. A dct approximation for image compression. *IEEE Signal Processing Letters*, 18(10):579–582, 2011.

[3] Paulo A M Oliveira, Renato J Cintra, Fábio M Bayer, Sunera Kulasekera, and Arjuna Madanayake. Low-complexity image and video coding based on an approximate discrete tchebichef transform. *IEEE Transactions on Circuits and Systems for Video Technology*, 27:1066–1076, 2016.

[4] Sirous Farsiani and Amir M Sodagar. Hardware and power-efficient compression technique based on discrete tchebichef transform for neural recording microsystems. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 3489–3492. IEEE, 2020.

[5] Saad Bouguezel, M Omair Ahmad, and MNS Swamy. A low-complexity parametric transform for image compression. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2145–2148. IEEE, 2011.

[6] Reem T Haweel, Wail S El-Kilani, and Hassan H Ramadan. Fast approximate dct with gpu implementation for image compression. *Journal of Visual Communication and Image Representation*, 40:357–365, 2016.

[7] Saad Bouguezel, M Omair Ahmad, and MNS Swamy. Low-complexity 8$\times$ 8 transform for image compression. *Electronics Letters*, 44(21):1249–1250, 2008.

[8] Tahar Brahimi, Ali Melit, and Fouad Khelifi. An improved spiht algorithm for lossless image coding. *Digital Signal Processing*, 19(2):220–228, 2009.

[9] Tahar Brahimi, Farid Laouir, Larbi Boubchir, and Arab Ali-Chérif. An improved wavelet-based image coder for embedded greyscale and colour image compression. *AEU-International Journal of Electronics and Communications*, 73:183–192, 2017.

[10] Tahar Brahimi, Larbi Boubchir, Régis Fournier, and Amine Naït-Ali. An improved multimodal signal-image compression scheme with application to natural images and biomedical data. *Multimedia Tools and Applications*, 76(15):16783–16805, 2017.

[11] Khalid Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017.

[12] William B Pennebaker and Joan L Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[13] Thomas Richter and Richard Clark. Why jpeg is not jpeg—testing a 25 years old standard. In *2018 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2018.

[14] Zhigang Fan and Ricardo L De Queiroz. Identification of bitmap compression history: Jpeg detection and quantizer estimation. *IEEE Transactions on Image Processing*, 12:230–235, 2003.

[15] Tiziano Bianchi and Alessandro Piva. Image forgery localization via block-grained analysis of jpeg artifacts. *IEEE Transactions on Information Forensics and Security*, 7:1003–1017, 2012.

[16] Qing Wang and Rong Zhang. Double jpeg compression forensics based on a convolutional neural network. *EURASIP Journal on Information Security*, 2016(1):1–12, 2016.

[17] Weixiang Li, Wenbo Zhou, Weiming Zhang, Chuan Qin, Huanhuan Hu, and Nenghai Yu. Shortening the cover for fast jpeg steganography. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1745–1757, 2019.

[18] Jinyuan Tao, Sheng Li, Xinpeng Zhang, and Zichi Wang. Towards robust image steganography. *IEEE Transactions on Circuits and Systems for Video Technology*, 29:594–600, 2 2019.

[19] Shuming Jiao, Zhi Jin, Chenliang Chang, Changyuan Zhou, Wenbin Zou, and Xia Li. Compression of phase-only holograms with jpeg standard and deep learning. *Applied Sciences*, 8(8):1258, 2018.

[20] Michael Parker. *Digital Signal Processing 101: Everything you need to know to get started*. Elsevier, 2 nd edition, 2017.

[21] Lulin Cai, Yiduan Qian, Yajuan He, and Wen Feng. Design of approximate multiplierless dct with csd encoding for image processing. In *2021 IEEE International symposium on circuits and systems (ISCAS)*, pages 1–4. IEEE, 2021.

[22] Yukihiro Arai, Takeshi Agui, and Masayuki Nakajima. A fast dct-sq scheme for images. *IEICE TRANSACTIONS (1976-1990)*, 71:1095–1097, 1988.

[23] Wen-Hsiung Chen, C H Smith, and S C Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on communications*, 25:1004–1009, 1977.

[24] Khan A Wahid, Vassil S Dimitrov, and Graham A Jullien. On the error-free realization of a scaled dct algorithm and its vlsi implementation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 54:700–704, 2007.

[25] Tarek I Haweel. A new square wave transform based on the dct. *Signal processing*, 81:2309–2319, 2001.

[26] Saad Bouguezel, M Omair Ahmad, and MNS Swamy. A fast 8$\times$ 8 transform for image compression. In *2009 International Conference on Microelectronics-ICM*, pages 74–77. IEEE, 2009.

[27] Saad Bouguezel, M Omair Ahmad, and MNS Swamy. A novel transform for image compression. In *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, pages 509–512. IEEE, 2010.

[28] Saad Bouguezel, M Omair Ahmad, and MNS Swamy. Binary discrete cosine and hartley transforms. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(4):989–1002, 2012.

[29] R J Cintra, F M Bayer, and C J Tablada. Low-complexity 8-point dct approximations based on integer functions. *Signal Processing*, 99:201–214, 2014.

[30] Uma Sadhvi Potluri, Arjuna Madanayake, Renato J Cintra, Fábio M Bayer, Sunera Kulasekera, and Amila Edirisuriya. Improved 8-point approximate dct for image and video compression requiring only 14 additions. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(6):1727–1740, 2014.

[31] Parvin Tamboli and Aparna Shinde. A low complexity 8x8 dct transform for image compression. *IJAREEIE*, 4(7):6185–6190, 2015.

[32] Fábio M Bayer and Renato J Cintra. Dct-like transform for image compression requires 14 additions only. *Electronics letters*, 48:919–921, 2012.

[33] Vítor A Coutinho, Renato J Cintra, Fábio M Bayer, Sunera Kulasekera, and Arjuna Madanayake. A multiplierless pruned dct-like transformation for image and video compression that requires ten additions only. *Journal of Real-Time Image Processing*, 12:247–255, 2016.

[34] Chaouki Araar, Salim Ghanemi, Mohamed Benmohammed, and Hamza Atoui. Pruned improved eight-point approximate dct for image encoding in visual sensor networks requiring only ten additions. *Journal of Real-Time Image Processing*, pages 1–12, 2019.

[35] Raíza S Oliveira, Renato J Cintra, Fábio M Bayer, Thiago LT da Silveira, Arjuna Madanayake, and André Leite. Low-complexity 8-point dct approximation based on angle similarity for image and video coding. *Multidimensional Systems and Signal Processing*, 30:1363–1394, 2019.

[36] Ranjan K Senapati, Umesh C Pati, and Kamala K Mahapatra. A low complexity orthogonal 8$\times$ 8 transform matrix for fast image compression. In *2010 Annual IEEE India Conference (INDICON)*, pages 1–4. IEEE, 2010.

[37] Nabila Brahimi and Saad Bouguezel. An efficient fast integer dct transform for images compression with 16 additions only. In *International Workshop on Systems, Signal Processing and their Applications, WOSSPA*, pages 71–74. IEEE, 2011.

[38] R Ezhilarasi, K Venkatalakshmi, and B Pradeep Khanth. Enhanced approximate discrete cosine transforms for image compression and multimedia applications. *Multimedia Tools and Applications*, pages 1–14, 2018.

[39] Y-G Wu. Ga-based dct quantisation table design procedure for medical images. *IEE Proceedings-Vision, Image and Signal Processing*, 151:353–359, 2004.

[40] Beatrice Lazzerini, Francesco Marcelloni, and Massimo Vecchio. A multi-objective evolutionary approach to image quality/compression trade-off in jpeg baseline algorithm. *Applied Soft Computing*, 10:548–561, 2010.

[41] Milan Tuba and Nebojsa Bacanin. Jpeg quantization tables selection by the firefly algorithm. In *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, pages 153–158. IEEE, 2014.

[42] P A M Oliveira, R S Oliveira, R J Cintra, F M Bayer, and A Madanayake. Jpeg quantisation requires bit-shifts only. *Electronics Letters*, 53:588–590, 2017.

[43] Albert Moran. The technology of television. In *Framing Technology*, pages 29–44. Routledge, 2023.

[44] Yun qing Shi and Huifang Sun. *Image and Video Compression for Multimedia Engineering*. Taylor Francis Group, third edit edition, 2019.

[45] Md Rahman, Mohamed Hamada, and Jungpil Shin. The impact of state-of-the-art techniques for lossless still image compression. *Electronics*, 10:360, 2021.

[46] Hanaa ZainEldin, Mostafa A Elhosseini, and Hesham A Ali. Image compression algorithms in wireless multimedia sensor networks: A survey. *Ain Shams engineering journal*, 6:481–490, 2015.

[47] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[48] Abul Hasnat, Dibyendu Barman, and Bandana Barman. Luminance approximated vector quantization algorithm to retain better image quality of the decompressed image. *Multimedia Tools and Applications*, 80:11985–12007, 2021.

[49] Xi Zhang and Xiaolin Wu. Lvqac: Lattice vector quantization coupled with spatially adaptive companding for efficient learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10239–10248, 2023.

[50] Jonas Löhdefink, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. Scalar and vector quantization for learned image compression: A study on the effects of mse and gan loss in various spaces. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.

[51] Zakarya Oubrahim, Yassine Amirat, Mohamed Benbouzid, and Mohammed Ouassaid. Power quality disturbances characterization using signal processing and pattern recognition techniques: A comprehensive review. *Energies*, 16(6):2685, 2023.

[52] M Khalil Gibran, Erna Budhiarti Nababan, and Poltak Sihombing. Analysis of face recognition with fuzzy c-means clustering image segmentation and learning vector quantization. In *2020 3rd international conference on mechanical, electronics, computer, and industrial technology (MECnIT)*, pages 188–193. IEEE, 2020.

[53] Akanksha Kalia, Shikar Sharma, Saurabh Kumar Pandey, Vinay Kumar Jadoun, and Madhulika Das. Comparative analysis of speaker recognition system based on voice activity detection technique, mfcc and plp features. In *Intelligent Computing Techniques for Smart Energy Systems: Proceedings of ICTSES 2018*, pages 781–787. Springer, 2020.

[54] Joshua Roy Palathinkal and Vaddi Chandra Sekhar. Impact of pre-processing on the performance of text-independent speaker recognition system. In *2020 IEEE REGION 10 CONFERENCE (TENCON)*, pages 1227–1232. IEEE, 2020.

[55] Rudy Chandra, Shahira An-Nissa, and Elviawaty M Zamzami. Comparative analysis of eigenface and learning vector quantization (lvq) to face recognition. In *Journal of Physics: Conference Series*, volume 1566, page 012012. IOP Publishing, 2020.

[56] Wengang Zhou, Yijuan Lu, Houqiang Li, and Qi Tian. Scalar quantization for large scale image search. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 169–178, 2012.

[57] Yuval Fisher. *Fractal image compression: theory and application*. Springer Science & Business Media, 2012.

[58] Olga Svynchuk, Oleg Barabash, Joanna Nikodem, Roman Kochan, and Oleksandr Laptiev. Image compression using fractal functions. *Fractal and Fractional*, 5(2):31, 2021.

[59] Humberto Ochoa-Dominguez and Kamisetty Ramamohan Rao. *Discrete Cosine Transform*. CRC Press, 2019.

[60] Duraisamy Sundararajan. *The discrete Fourier transform: theory, algorithms and applications*. World Scientific, 2001.

[61] Syed Ali Khayam. The discrete cosine transform (dct): theory and application. *Michigan State University*, 114:1–31, 2003.

[62] R Dony et al. Karhunen-loeve transform. *The transform and data compression handbook*, 1(1-34):29, 2001.

[63] D Sundararajan. *Discrete wavelet transform: a signal processing approach*. John Wiley Sons, 2016.

[64] Shuyun Yuan and Jianbo Hu. Research on image compression technology based on huffman coding. *Journal of Visual Communication and Image Representation*, 59:33–38, 2019.

[65] Qinghua Zhang. Regressor selection and wavelet network construction. pages 3688–3693. IEEE, 1993.

[66] Peter Schelkens, Athanassios Skodras, and Touradj Ebrahimi. *The JPEG 2000 suite*. John Wiley & Sons, 2009.

[67] Peter Schelkens, Athanassios Skodras, and Touradj Ebrahimi. *The JPEG 2000 suite*. John Wiley & Sons, 2009.

[68] ITU Recommendation ITU-T P. 910, subjective video quality assessment methods for multimedia applications, itu telecom. *Standardization Sector of ITU, Geneva, Switzerland*, 1999.

[69] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13:600–612, 2004.

[70] David S Watkins. *Fundamentals of matrix computations*, volume 64. John Wiley Sons, 2004.

[71] C J Tablada, Thiago Lopes Trugillo da Silveira, Renato J Cintra, and Fábio M Bayer. Dct approximations based on chen's factorization. *Signal Processing: Image Communication*, 58:14–23, 2017.

[72] Adria Arrufat Batalla. *Multiple transforms for video coding*. PhD thesis, INSA de Rennes, 2015.

[73] Vladimir Britanak, Patrick C Yip, and Kamisetty Ramamohan Rao. *Discrete cosine and sine transforms: general properties, fast algorithms and integer approximations*. Elsevier, 2010.

[74] K Ramamohan Rao and Ping Yip. *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014.

[75] V. Britanak, P. C.Yip, and K. R. Rao. *Discrete cosine and sine transforms: general properties, fast algorithms and integer approximations*. Academic, 2007.

[76] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38:xviii–xxxiv, 1992.

[77] Detlev Marpe, Thomas Wiegand, and Gary J Sullivan. The h. 264/mpeg4 advanced video coding standard and its applications. *IEEE communications magazine*, 44:134–143, 2006.

[78] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[79] Mahsa T Pourazad, Colin Doutre, Maryam Azimi, and Panos Nasiopoulos. Hevc: The new gold standard for video compression: How does hevc compare with h. 264/avc? *IEEE consumer electronics magazine*, 1:36–46, 2012.

[80] Jizheng Xu, Feng Wu, and Wenjun Zhang. Intra-predictive transforms for block-based image coding. *IEEE transactions on signal processing*, 57:3030–3040, 2009.

[81] Joint Photographic Experts Group et al. Jpeg 2000 image coding system, iso/iec 15444-1: 2000, 2000.

[82] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23:90–93, 1974.

[83] Otávio A B Penatti, Eduardo Valle, and Ricardo da S Torres. Comparative study of global color and texture descriptors for web image retrieval. *Journal of visual communication and image representation*, 23:359–380, 2012.

[84] Cong Bai. *Analyse d'images pour une recherche d'images basée contenu dans le domaine transformé.* PhD thesis, Rennes, INSA, 2013.

[85] Muhammad Usman Karim Khan, Muhammad Shafique, Lars Bauer, and Jörg Henkel. Multicast fullhd h. 264 intra video encoder architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34:2049–2053, 2015.

[86] Wenjia Yuan, Pengwei Hao, and Chao Xu. Matrix factorization for fast dct algorithms. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 3, pages III–III. IEEE, 2006.

[87] Jae S Lim. Two-dimensional signal and image processing. *Englewood Cliffs*, 1990.

[88] Soni Prattipati, Sujata Ishwar, MNS Swamy, and Pramod Kumar Meher. A fast $8\times$ 8 integer tchebichef transform and comparison with integer cosine transform for image compression. In *2013 IEEE 56th international midwest symposium on circuits and systems (MWSCAS)*, pages 1294–1297. IEEE, 2013.

[89] Vítor A Coutinho, Renato J Cintra, Fábio M Bayer, Paulo A M Oliveira, Raíza S Oliveira, and Arjuna Madanayake. Pruned discrete tchebichef transform approximation for image compression. *Circuits, Systems, and Signal Processing*, 37:4363–4383, 2018.

[90] Nasreddine Kouadria, Khaoula Mechouek, Saliha Harize, and Noureddine Doghmane. Region-of-interest based image compression using the discrete tchebichef transform in wireless visual sensor networks. *Computers Electrical Engineering*, 73:194–208, 2019.

[91] Ramakrishnan Mukundan, S H Ong, and Poh Aun Lee. Image analysis by tchebichef moments. *IEEE Transactions on image Processing*, 10:1357–1364, 2001.

[92] Ferda Ernawan, Edi Noersasongko, and Nur Azman Abu. An efficient 2$\times$ 2 tchebichef moments for mobile image compression. In *2011 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*, pages 1–5. IEEE, 2011.

[93] Li Wern Chew, Li-Minn Ang, and Kah Phooi Seng. Survey of image compression algorithms in wireless sensor networks. In *2008 International Symposium on Information Technology*, volume 4, pages 1–9. IEEE, 2008.

[94] Meng Guo, Mostafa H Ammar, and Ellen W Zegura. V3: A vehicle-to-vehicle live video streaming architecture. *Pervasive and Mobile Computing*, 1:404–424, 2005.

[95] David H Friedman. Streaming implementation of video algorithms on a low-power parallel architecture. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 650–653. IEEE, 2013.

[96] Kiyoyuki Nakagaki and Ramakrishnan Mukundan. A fast 4$\times$4 forward discrete tchebichef transform algorithm. *IEEE Signal Processing Letters*, 14:684–687, 2007.

[97] Guobao Wang and Shigang Wang. Recursive computation of tchebichef moment and its inverse transform. *Pattern Recognition*, 39:47–56, 2006.

[98] Sujata Ishwar, Pramod Kumar Meher, and MNS Swamy. Discrete tchebichef transform-a fast 4x4 algorithm and its application in image/video compression. In *2008 IEEE International Symposium on Circuits and Systems*, pages 260–263. IEEE, 2008.

[99] Woonsung Park, Bumshik Lee, and Munchurl Kim. Fast computation of integer dct-v, dct-viii, and dst-vii for video coding. *IEEE Transactions on Image Processing*, 28:5839–5851, 2019.

[100] Christoph Loeffler, Adriaan Ligtenberg, and George S Moschytz. Practical fast 1-d dct algorithms with 11 multiplications. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 988–991. IEEE, 1989.

[101] W-K Cham. Development of integer cosine transforms by the principle of dyadic symmetry. *IEE Proceedings I (Communications, Speech and Vision)*, 136:276–282, 1989.

[102] Renato J Cintra, Fábio M Bayer, Vítor A Coutinho, Sunera Kulasekera, Arjuna Madanayake, and André Leite. Energy-efficient 8-point dct approximations: Theory and hardware architectures. *Circuits, Systems, and Signal Processing*, 35:4009–4029, 2016.

[103] Maher Jridi, Ayman Alfalou, and Pramod Kumar Meher. A generalized algorithm and reconfigurable architecture for efficient and scalable orthogonal approximation of dct. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62:449–457, 2014.

[104] Saad Bouguezel, M Omair Ahmad, and MNS Swamy. A multiplication-free transform for image compression. In *2008 2nd International Conference on Signals, Circuits and Systems*, pages 1–4. IEEE, 2008.

[105] Saad Bouguezel, M Omair Ahmad, and MNS Swamy. A low-complexity parametric transform for image compression. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2145–2148. IEEE, 2011.

[106] Diego Ramos Canterle, Thiago L T da Silveira, Fábio M Bayer, and Renato J Cintra. A multiparametric class of low-complexity transforms for image and video coding. *Signal Processing*, page 107685, 2020.

[107] Fábio M Bayer and Renato J Cintra. Dct-like transform for image compression requires 14 additions only. *arXiv preprint arXiv:1702.00817*, 2017.

[108] Abdelkader Mefoued, Saliha Harize, and Nasreddine Kouadria. Efficient, low complexity 8-point discrete tchebichef transform approximation for signal processing applications. *Journal of the Franklin Institute*, 360(7):4807–4829, 2023.

[109] Paulo A M Oliveira, Renato J Cintra, Fábio M Bayer, Sunera Kulasekera, and Arjuna Madanayake. A discrete tchebichef transform approximation for image and video coding. *IEEE Signal Processing Letters*, 22:1137–1141, 2015.

[110] Richard E Blahut. *Fast algorithms for signal processing*. Cambridge University Press, 2010.

[111] Jinyuan Tao, Sheng Li, Xinpeng Zhang, and Zichi Wang. Towards robust image steganography. *IEEE Transactions on Circuits and Systems for Video Technology*, 29:594–600, 2018.

[112] Maher Jridi and Pramod Kumar Meher. Scalable approximate dct architectures for efficient hevc-compliant video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 27:1815–1825, 2016.

[113] Nabila Brahimi, Toufik Bouden, Tahar Brahimi, and Larbi Boubchir. Lossy image compression based on efficient multiplier-less 8-points dct. *Multimedia Systems*, pages 1–12, 2021.

[114] Vítor de A Coutinho, Renato J Cintra, and Fábio M Bayer. Low-complexity multidimensional dct approximations for high-order tensor data decorrelation. *IEEE Transactions on Image Processing*, 26:2296–2310, 2017.

[115] Ismail Gassoumi, Lamjed Touil, Bouraoui Ouni, and Abdellatif Mtibaa. An efficient design of dct approximation based on quantum dot cellular automata (qca) technology. *Journal of Electrical and Computer Engineering*, 2019, 2019.

[116] Image databases. [online]. available. http://www.imageprocessingplace.com/root_files_V3/image_databases.htm. Accessed: 19-Mar-2018.

[117] Bin Xiao, Wenming Shi, Gang Lu, and Weisheng Li. An optimized quantization technique for image compression using discrete tchebichef transform. *Pattern Recognition and Image Analysis*, 28:371–378, 2018.

[118] Cvg - ugr - image database. http://decsai.ugr.es/cvg/dbimagenes/g512.php. Accessed: 19-Mar-2018.

[119] Ahmet M. Eskicioglu and Paul S. Fisher. Image quality measures and their performance. *IEEE Transactions on Communications*, 43:2959–2965, 1995.

[120] Sonja Grgic, Marta Mrak, and Mislav Grgic. Comparison of jpeg image coders. *University of Zagreb, Faculty of Electrical Engineering and Computing Unska*, 3, 2001.

[121] M Lee and Mostafa Kaveh. Fast hadamard transform based on a simple matrix factorization. *IEEE transactions on acoustics, speech, and signal processing*, 34:1666–1667, 1986.

[122] Jonathan Robinson and Vojislav Kecman. Combining support vector machine learning with the discrete cosine transform in image compression. *IEEE Transactions on Neural Networks*, 14:950–958, 2003.

[123] A Diana Andrushia and R Thangarjan. Saliency-based image compression using walsh–hadamard transform (wht). *Biologically rationalized computing techniques for image processing applications*, pages 21–42, 2018.

[124] Nasreddine Kouadria, Khaoula Mechouek, Saliha Harize, and Noureddine Doghmane. Region-of-interest based image compression using the discrete tchebichef transform in wireless visual sensor networks. *Computers & Electrical Engineering*, 73:194–208, 2019.