Integer DCT Approximation With Arbitrary Size and Adjustable Precision

Lucas A. Thomaz, *Member, IEEE*, Pedro A. A. Assunção, *Senior Member, IEEE*, Luis M. N. Tavora, *Member, IEEE*, and Sérgio M. M. de Faria, *Senior Member, IEEE*

***Abstract*—This letter proposes a method to obtain integer reversible discrete cosine transforms for generic transform-based coding schemes. The novelty of the proposed method, which is based on decomposition of the DCT-II matrix into two triangular and one diagonal matrices, is twofold: (i) the new matrices can be of arbitrary size, i.e., any square $N \times N$ dimension, thus suitable for applications where non power-of-2 dimensions are required; (ii) they can be designed with adjustable precision in a trade-off with the number of representation bits. Furthermore, improvements are also proposed over the base scheme to avoid numerical issues when working with large matrices and to obtain more reliable approximations. The performance evaluation demonstrate the effectiveness of the proposed transforms to approximate the coding gain capabilities of the original DCT-II.**

***Index Terms*—Integer DCT, Lossless Coding, Separable Transform, Transform Coding, Triangular Decomposition.**

I. Introduction

DISCRETE Cosine Transforms (DCT) have been employed in many coding schemes since its original publication [1]. Several coding standards are based on such transforms, as the JPEG [2], decades ago, and the more recent HEVC [3]. The wide adoption of the DCT is partially due to the fact that they approximate very well the energy concentration attributes of the Karhunen-Love Transform [4] (the optimal one in terms of this property) but, differently from the latter, it is not signal-dependent and its implementation only requires real coefficients.

The widespread use of the DCTs, especially of the DCT Type 2 (DCT-II), for transform coding has also motivated a lot of research in related aspects, giving rise to variations of the original transform. One of the most important ones is the integer DCT, which can be employed in lossy and lossless coding tools. The integer DCT has the advantage of being reversible even with finite numerical precision (inherent of digital computing) and much more computationally efficient than floating point implementations, as it can run in dedicated fix-point units. Historically, the DCT-based coding standards evolved from using square ($N \times N$) transform blocks, where N is a power-of-2 (as JPEG and HEVC), to more flexible ($N \times M$) where both N and M are power-of-2 blocks (as in the VVC [5]), and more recently the upcoming standard JPEG Pleno [6] employs 4D blocks with arbitrary integer size [7].

Integer DCTs find useful applications in lossless coding (*e.g.* medical images), since the reversibility with finite numerical precision allows perfect reconstruction of images in DCT-based codecs. Another promising area of application is lossless light field coding, where recent advances ([8], [9], and [10]) may be seen as precursors of a future extension of the JPEG Pleno [11]. However, the use of non-integer 4D-DCTs [7] does not allow such coding mode in the current version of the codec.

This letter introduces a novel approach to obtain integer reversible $N \times N$ transforms that approximate the DCT-II, for any integer N, based on the triangular matrix scheme [12]. Our approach solves some issues related to the matrix conditioning and numerical precision during the matrices generation, guaranteeing that the integer transform can be efficiently computed. Also, the proposed approach uses a genetic algorithm to find the best integer approximation for the $N \times N$ DCT matrices, retaining the DCT-II properties in terms of coding-gain.

The remainder of the letter is organized as follows: Section II presents the related work discussing other approaches and uses of the integer DCTs. Section III introduces the proposed method explaining in detail how to obtain the integer reversible transforms. In Section III experimental results demonstrate the capabilities of the proposed transform in terms of coding-gain. Finally, in Section V the conclusions and final discussions are presented.

II. Related Work

One of the most efficient ways to apply DCT transforms to an image is to divide the image into blocks and use DCT matrices to pre-multiply (and post-multiply in case of 2D transform) each image block. Several coding standards use such approach to implement their transform-based data compression. Two of the most important image and video coding standards (H.264/AVC [13] and H.265/HEVC [14]) adopt integer approximations of the DCTs in their framework, in order to optimize the transform computations.

The integer DCT approximation employed in the H.264/AVC standard is presented in [15], and requires only the use of

Manuscript received March 6, 2020; revised May 13, 2020; accepted May 23, 2020. Date of publication May 28, 2020; date of current version June 18, 2020. This work was supported in part by the Programa Operacional Regional do Centro, under Project PlenoISLA POCI-01-0145-FEDER-028325 and in part by the FCT/MCTES through national funds and when applicable co-funded by EU funds under the Project UIDB/EEA/50008/2020. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Matteo Naccari *(Corresponding author: Lucas A. Thomaz.)*

Lucas A. Thomaz, Pedro A. A. Assunção, and Sérgio M. M. de Faria are with the Instituto de Telecomunicações, 2411-901 Leiria, Portugal, and also with the ESTG, Polytechnic of Leiria, 2411-901 Leiria, Portugal (e-mail: lucas.thomaz@co.it.pt; amado@co.it.pt; sergio.faria@co.it.pt).

Luis M. N. Tavora is with the ESTG, Polytechnic of Leiria, 2411-901 Leiria, Portugal (e-mail: luis.tavora@ipleiria.pt).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes a MATLAB function with the method's implementation and a readme file. This material is 49 KB in size.

Digital Object Identifier 10.1109/LSP.2020.2998362

additions and bit shift operations, which simplify the computation of the transform matrices. Similarly, in [16] the DCT approximation matrices employed in the H.265 (HEVC) standard are presented. The H.265 standard employs 4 fixed-size matrices to compute 4×4, 8×8, 16×16, and 32×32 transforms. These matrices are integer approximations of the DCT, however since they are not perfectly reversible they cannot be employed in lossless coding, although they present low inherent distortion.

In order to obtain proper approximation of the DCT transform, which can be used in lossless coding applications, some requirements must be fulfilled [17]: the input and output of the transform must be integer, that is integer-to-integer mapping, the transform must be reversible, that is, bijective. In addition to those requirements, it is expected that the transform matrix entries are all binary-valued, which means they can be represented as $a_{m,n} \cdot 2^{-k}$, with $k \in \mathbb{Z}^+$, $a_{m,n} \in \mathbb{Z}$, and m, n being the indexes of the transform matrix entries.A popular method to obtain integer DCT approximations (for matrices whose sizes are $2^k, k \in \mathbb{Z}^+$) is the lifting scheme [18], which produces direct and inverse transforms that can be used with fast algorithms, such as the Fast Fourier Transform (FFT).

The triangular matrix scheme presented in [12] generalizes the lifting scheme for any $N \times N$ full-rank transform matrix. This approach, decomposes the original transform matrix into three triangular matrices, which are approximated into binary-valued matrices, resulting in a binary-valued approximation of the original transform matrix. This method presents two main issues: the first one related with the conditioning of the intermediate matrices that can cause the method to diverge; and the second related to the large approximation errors induced by truncation that reduce the matrices transform coding gain.

This work presents an algorithm to generate reversible integer DCT matrices based on the triangular matrix scheme [12] for any $N \times N$ dimension, with simplifications focused on the DCT-II. Improvements are proposed to handle the truncation-induced errors and the matrix conditioning, increasing the reliability in the generation of the transform matrices. The proposed algorithm features a trade-off between transform coding gain and bit depth of the fixed-point representation.

III. Proposed Method

This section describes the method herein proposed to determine the integer reversible DCT and its inverse matrices based on triangular matrix scheme (available as a Matlab function in the supplemental material). As this work focus on providing matrices to be used in transform-based coding, the proposed method is limited to square transform matrices. If transforms of non-square image blocks ($N \times M$) are required (as in some partitions of VVC [5]), it is possible to perform a composition of two DCT approximation matrices ($N \times N$ and $M \times M$), pre-multiply the image block by the $N \times N$ and post-multiply it by the $M \times M$ matrix, resulting in an $N \times M$ block.

A. Square DCT Matrix Adaptation

The conversion is performed by starting with a $N \times N$ DCT-II matrix $\mathbf{G}$ and decomposing this matrix as $\mathbf{G} = \mathbf{D}\mathbf{T}_3\mathbf{T}_2\mathbf{T}_1$, where $\mathbf{D}$ is a diagonal $N \times N$ matrix, $\mathbf{T}_1$ and $\mathbf{T}_3$ are $N \times N$ upper triangular matrices, and $\mathbf{T}_2$ is an $N \times N$ lower triangular matrix.

Matrix $\mathbf{D}$ is similar to an $N \times N$ identity matrix, however, the first entry of $\mathbf{D}$, may be -1 depending on the size N, thus $\mathbf{D}_{1,1} = \gamma$, where $\gamma = (-1)^{N-1}$. After determining the matrix $\mathbf{D}$, the following step is to find an upper triangular matrix $\mathbf{L}_1$ such that $\mathbf{E} = \mathbf{G}\mathbf{L}_1$, with

$$\mathbf{E} = \begin{bmatrix} e_{1,1} & e_{1,2} & e_{1,3} & \dots & e_{1,N-1} & e_{1,N} \\ e_{2,1} & 1 & 0 & \dots & 0 & 0 \\ e_{3,1} & e_{3,2} & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ e_{N-1,1} & e_{N-1,2} & e_{N-1,3} & \dots & 1 & 0 \\ e_{N,1} & e_{N,2} & e_{N,3} & \dots & e_{N,N-1} & 1 \end{bmatrix} \tag{1}$$

and

$$\mathbf{L}_1 = \begin{bmatrix} 1 & \tau_{1,2} & \tau_{1,3} & \dots & \tau_{1,N-1} & \tau_{1,N} \\ 0 & 1 & \tau_{2,3} & \dots & \tau_{2,N-1} & \tau_{2,N} \\ 0 & 0 & 1 & \dots & \tau_{3,N-1} & \tau_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & \tau_{N-1,N} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}. \tag{2}$$

To find the columns $\mathbf{l}_n$ of $\mathbf{L}_1$ and thus all the entries of the matrix, we can use the identity $\mathbf{S}_n\mathbf{l}_n + \hat{\mathbf{g}}_n = \mathbf{z}_n$, where

$$\mathbf{S}_n = \begin{bmatrix} g_{2,1} & g_{2,2} & \dots & g_{2,n-1} \\ g_{3,1} & g_{3,2} & \dots & g_{3,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n-1,1} & g_{n-1,2} & \dots & g_{n-1,n-1} \\ g_{n,1} & g_{n,2} & \dots & g_{n,n-1} \end{bmatrix}, \tag{3}$$

and

$$\mathbf{l}_n = \begin{bmatrix} \tau_{1,n} \\ \tau_{2,n} \\ \vdots \\ \tau_{n-2,n} \\ \tau_{n-1,n} \end{bmatrix}, \quad \hat{\mathbf{g}}_n = \begin{bmatrix} g_{2,n} \\ g_{3,n} \\ \vdots \\ g_{n-1,n} \\ g_{n,n} \end{bmatrix}, \quad \mathbf{z}_n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{4}$$

From these definitions it is possible to determine $\mathbf{l}_n = \mathbf{S}_n^{-1}(\mathbf{z}_n - \hat{\mathbf{g}}_n)$. Then, the expression can be rewritten such that $\mathbf{G} = \mathbf{E}\mathbf{T}_1$, where $\mathbf{T}_1 = \mathbf{L}_1^{-1}$. Since $\mathbf{L}_1$ is an upper triangular matrix there are many efficient and simple ways to calculate its inverse, which will also be upper triangular.

Now, $\mathbf{E}$ is decomposed as $\mathbf{E} = \mathbf{F}\mathbf{T}_2$, where

$$\mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ e_{2,1} & 1 & 0 & \dots & 0 & 0 \\ e_{3,1} & e_{3,2} & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ e_{N-1,1} & e_{N-1,2} & e_{N-1,3} & \dots & 1 & 0 \\ e_{N,1} & e_{N,2} & e_{N,3} & \dots & e_{N,N-1} & 1 \end{bmatrix}, \tag{5}$$

and $\mathbf{F}$ is an $N \times N$ identity matrix whose first line is replaced by $[f_{1,1}\ f_{1,2}\ f_{1,3}\ \dots\ f_{1,N}]$. From that, we get

$$[f_{1,1}\, f_{1,2}\, f_{1,3}\, \dots\, f_{1,N}] = [e_{1,1}\, e_{1,2}\, e_{1,3}\, \dots\, e_{1,N}] \cdot \mathbf{T}_2^{-1}. \tag{6}$$

Again, finding the inverse of $\mathbf{T}_2$ is straightforward since $\mathbf{T}_2$ and its inverse are lower diagonal matrices.

Finally, $\mathbf{F}$ is decomposed as $\mathbf{F} = \mathbf{D}\mathbf{T}_3$. But, since $\mathbf{D}$ is already defined, we get that $\mathbf{T}_3$ is an identity matrix whose first line is replaced by $[\gamma f_{1,1}\ \gamma f_{1,2}\ \gamma f_{1,3}\ \ldots\ \gamma f_{1,N-1}\ \gamma f_{1,N}]$, which can also be obtained as $\mathbf{T}_3 = \mathbf{D}\mathbf{F}$, since $\mathbf{D}$ is an involutory matrix ($\mathbf{D} = \mathbf{D}^{-1}$).

To obtain the integer DCT approximation we need to approximate the triangular matrices $\mathbf{T}_1$, $\mathbf{T}_2$, and $\mathbf{T}_3$ into binary-valued ones ($\mathbf{J}_1$, $\mathbf{J}_2$, and $\mathbf{J}_3$). Assuming that the non-zero entries in the m, n position of the $\mathbf{T}_1$, $\mathbf{T}_2$, and $\mathbf{T}_3$ matrices are $\eta_{m,n}$, $\zeta_{m,n}$, and $\rho_{m,n}$ respectively, we approximate them as $h_{m,n} \approx \eta_{m,n}$, $c_{m,n} \approx \zeta_{m,n}$, and $r_{m,n} \approx \rho_{m,n}$ so that

$$h_{m,n} = 2^{-b_1}\mathrm{round}(2^{b_1}\eta_{m,n}), \tag{7}$$

$$c_{m,n} = 2^{-b_2}\mathrm{round}(2^{b_2}\zeta_{m,n}), \text{ and} \tag{8}$$

$$r_{m,n} = 2^{-b_3}\mathrm{round}(2^{b_3}\rho_{m,n}), \tag{9}$$

where b_1, b_2, and b_3 are the number os bits used to approximate $\mathbf{T}_1$, $\mathbf{T}_2$, and $\mathbf{T}_3$.

With (7), (8), (9), we obtain the binary approximation $\mathbf{B}$ of $\mathbf{G}$ as $\mathbf{B} = \mathbf{D}\mathbf{J}_3\mathbf{J}_2\mathbf{J}_1$. Since $\mathbf{D}$, $\mathbf{J}_3$, $\mathbf{J}_2$, and $\mathbf{J}_1$ are all binary-valued matrices, so will be $\mathbf{B}$. Also, in order to obtain the inverse approximated DCT matrix we simply do $\tilde{\mathbf{B}} = \tilde{\mathbf{J}}_1\tilde{\mathbf{J}}_2\tilde{\mathbf{J}}_3\mathbf{D}$, with $\tilde{\mathbf{J}}_1 = \mathbf{J}_1^{-1}$, $\tilde{\mathbf{J}}_2 = \mathbf{J}_2^{-1}$, and $\tilde{\mathbf{J}}_3 = \mathbf{J}_3^{-1}$, which also results in a binary-valued matrix (as $\tilde{\mathbf{J}}_1$, $\tilde{\mathbf{J}}_2$, $\tilde{\mathbf{J}}_3$, and $\mathbf{D}$ are binary-valued and triangular matrices with diagonals composed by 1 and -1, thus with unitary determinants [19]).

B. Improving the Matrices Conditioning

In order to obtain matrix $\mathbf{T}_1$, and thus matrix $\mathbf{L}_1$, it is necessary to compute several inverse matrices, as explained in Section III-A. However, the inversion of the $\mathbf{S}_n$ matrices defined in Eq. (3) can introduce errors in the algorithm if the matrices are not well conditioned, that is if its condition number (ratio of the largest to the smallest eigenvalue of the matrix) is too small. To cope with such issues, a permutation step is introduced at the beginning of the algorithm to provide a better conditioned version of the $\mathbf{G}$.

In order to find the new, better conditioned, matrix $\mathbf{G}'$ that replaces $\mathbf{G}$ we need to iteratively find the permuting $\mathbf{P}$ and $\mathbf{Q}$ so that $\mathbf{G}' = \mathbf{P}\mathbf{G}\mathbf{Q}$, and

$$\left|\det\left(\begin{bmatrix} g'_{2,1} & g'_{2,2} & \cdots & g'_{2,n-1} \\ g'_{3,1} & g'_{3,2} & \cdots & g'_{3,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g'_{n-1,1} & g'_{n-1,2} & \cdots & g'_{n-1,n-1} \\ g'_{n,1} & g'_{n,2} & \cdots & g'_{n,n-1} \end{bmatrix}\right)\right| \tag{10}$$

is large for every value of $n \in [2, N]$. To do so, we use Algorithm 1. Since $\mathbf{P}$ and $\mathbf{Q}$ are permutation matrices, it follows that $\mathbf{P}\mathbf{P}^{\mathrm{T}} = \mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \mathbf{I}$, thus we have that $\mathbf{G}' = \mathbf{P}\mathbf{G}\mathbf{Q} = \mathbf{P}\mathbf{D}\mathbf{T}_3\mathbf{T}_2\mathbf{T}_1\mathbf{Q}$, $\mathbf{G} = \mathbf{P}^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\mathbf{T}_2\mathbf{T}_1\mathbf{Q}^{\mathrm{T}}$, $\mathbf{B} = \mathbf{P}^{\mathrm{T}}\mathbf{D}\mathbf{J}_3\mathbf{J}_2\mathbf{J}_1\mathbf{Q}^{\mathrm{T}}$, and finally $\tilde{\mathbf{B}} = \mathbf{Q}\tilde{\mathbf{J}}_3\tilde{\mathbf{J}}_2\tilde{\mathbf{J}}_1\mathbf{D}\mathbf{P}$.

C. Improving the Truncation Approximation

Another improvement to the standard version of the algorithm is related to the method used to truncate the $\mathbf{T}_n$ matrices coefficients in order to generate the $\mathbf{J}_n$ matrices. In some cases, the values of the coefficients are very different from those of the truncated matrices, thus reflecting in large errors when the final binary matrix $\mathbf{B}$ is generated. To mitigate this issue, in the cases where the coefficient error is large, i.e. $2^{b_s}\mathbf{T}_{s(m,n)} - \mathrm{round}(2^{\mathrm{b_s}}\mathbf{T}_{\mathrm{s(m,n)}}) > 0.25$, the original truncation procedure that replaces the number by $2^{-b_s}\mathrm{round}(2^{\mathrm{b_s}}\mathbf{T}_{\mathrm{s(m,n)}} \pm 1)$ is modified.

Algorithm 1: Find $\mathbf{P}$ and $\mathbf{Q}$ Permutation Matrices.

```
S' = [], clist =[], rlist = [], detCost = [0,0,0];
P = Q = zeros(N,N); rowList = 1:N, columnList = 1:N;
for n = 1: N-1 do
    for psi = rowList do
        for nu = columnList do
            S"=[S', G(rlist,nu); G(psi,clist), G(psi,nu)];
            detTemp = abs(det(S"));
            if detTemp > detCost(1) then
                detCost = [detTemp, psi, nu];
            end
        end
    end
    psi = detCost(2); nu = detCost(3);
    S' = [S', G(rlist,nu); G(psi,clist), G(psi,nu)];
    rowList(rowList == psi) = [];
    columnList(columnList == nu) = [];
    clist = [clist, nu]; rlist = [rlist, psi];
    P(n+1,psi) = 1; Q(nu,n) = 1;
end
P(1,rowList) = 1; Q(columnList,N) = 1;
G' = P*G*Q;
```

Checking the effect of individual changes to the values of the coefficients in the binary approximation matrix would be impractical, as even in small transform matrices the number of possible combinations can be prohibitive. Thus, it is proposed to employ an alternative method based on a genetic algorithm that will feature the sum of absolute differences (SAD) of the two transform matrices ($\mathrm{SAD} = \sum_{\mathrm{n=1}}^{\mathrm{N}}\sum_{\mathrm{m=1}}^{\mathrm{N}}|\mathbf{G}_{\mathrm{n,m}} - \mathbf{B}_{\mathrm{n,m}}|$) as a fitness function. This algorithm finds the best truncation approximation among a limited amount of combinations, using 1-point crossover and mutation to explore the space of combinations. In this method the gene is a ternary-sequence of values $(-1, 0, 1)$ with the length equal to the number of values with approximation equal to or larger than 0.25. At every iteration the gene values corresponding to each matrix entry are added to the original coefficient, and the fitness function for the resulting matrix is measured.

In the experiments, the population was initialized with all different elements whose only one position was non-zero. The population size was twice the number of positions in a gene. The stop-condition was 100 epochs without improvement.

The use of such methods allow the approximation SAD to be lowered several orders of magnitude (depending on the size of the transform), as will be shown in the following section.

IV. Experimental Assessment

The presented method allows one to obtain a binary-valued approximation to the DCT matrices. These approximations will

TABLE I
TRANSFORM CODING GAIN OF THE PROPOSED TRANSFORM IN dB

Sizes	DCT-II	Prop. 8 bits	Prop. 12 bits	Prop. 16 bits	Prop. 20 bits
2×2	5.0550	5.0550	5.0550	5.0550	5.0550
4×4	7.5701	7.5700	7.5701	7.5701	7.5701
8×8	8.8259	8.8239	8.8259	8.8259	8.8259
16×16	9.4555	9.1513	9.4542	9.4555	9.4555
32×32	9.7736	6.5274	8.1063	9.7587	9.7736
48×48	9.8817	-83.8137	6.3915	7.952	9.1212

TABLE II
CODING GAIN OF DCT APPROXIMATIONS IN dB WITH 16 bits

Sizes	DCT-II	Prop.	H.264 [15]	binDCT [18]	Bink2.2 [22]
4×4	7.5701	7.5701	-	7.5697	-
8×8	8.8259	8.8259	8.7833	8.8251	8.8250

be as close to the original DCT matrices as the number of bits (b_1, b_2, b_3) increases, thus being arbitrarily accurate with an arbitrarily large number of bits. However, since in practice the number of bits is limited, this section shows the impact of adjusting the bit depth on the quality of the DCT approximation to different sizes of transform.

One of the most effective metrics to demonstrate the usefulness of transforms in terms of transform-based block coding is to measure the transform coding gain, as defined in [20].

Since the DCT is a good approximation to the ideal transform in terms of transform coding gain (Karhunen-Love transform) [21], the proposed transform approach has been compared to the DCT-II in terms of transform coding gain, for different sizes of the transform matrices, as well as for different number of bit depths. The achieved results are shown in Table I (using first-order Gauss-Markov process image model with autocorrelation coefficient $\rho = 0.95$). The proposed integer approximations of the DCT were obtained using both the truncation and permutation improvements.

Table I shows that even using a small number of bits to represent the transform matrices, the proposed method performs similarly to the DCT for block sizes up to 8×8. Moreover, it is clear that, for larger sizes of the transform matrix, equal or larger than 16×16, more bits are needed to obtain similar coding gain to the DCT-II, as is evidenced by the poor coding gain of the 48×48 DCT at 8-bits precision.

Particularly for DCT sizes 4×4 and 8×8 and 16-bit precision, which are common in block transform coding, other integer approximations for the DCT have their coding gain results available in the literature. Some of the most important examples are the H.264 transform proposed in [15], the Bink 2.2 transform [22] which is based on matrix rotations, and the binDCT which is a lifting based approximation from [18], where it was employed in a lossless approach of the H.263 method. Table II shows the comparison of their coding gain with the proposed transform. Other relevant transforms, such as the ones used in the HEVC [3] are not compared, since they are not perfectly reversible. The coding gains in this table show that our approach better approximates the coding gain of the DCT-II for the given bit precision.

In addition, we also measured the impact of the proposed truncation method based on genetic algorithms. To this aim the SAD that arises from different sizes of the transform was measured, with and without the proposed method, as shown in Table III. These experimental results were determined using a fixed number of 8 bits.

TABLE III
SAD OF INTEGER DCT WITH AND WITHOUT GENETIC OPTIMIZATION

Sizes	Regular	Optimized
2×2	$5.5 \cdot 10^{-4}$	$5.5 \cdot 10^{-4}$
4×4	$1.8 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$
8×8	$4.0 \cdot 10^{-1}$	$1.6 \cdot 10^{-1}$
16×16	$2.3 \cdot 10^{1}$	$1.7 \cdot 10^{1}$
32×32	$4.4 \cdot 10^{4}$	$6.6 \cdot 10^{2}$
48×48	$1.8 \cdot 10^{8}$	$5.0 \cdot 10^{4}$

The results in Table III indicate that the truncation improvement has a high impact on the quality of the obtained transform, which improves by orders of magnitude, depending on the size of the transform, namely for sizes larger than 4×4.

The produced transforms are separable, thus not limited to 1D and 2D applications, but also applicable in 3D and 4D coding. Also, since the coding gain is close to that of the DCT-II it can be employed to convert lossy transform-based codecs into lossless codecs without much change of the original framework (except the removal of quantization), as in [18].

An in-depth analysis of the computational complexity of the proposed transforms is beyond our scope. However, it is worth noting that for some of the produced transforms (with power-of-2 sizes) it would be possible to obtain fast implementations based on the Cooley-Turkey algorithm [23]. Their computational complexity should be similar to that of the lifting DCT [18]. If no fast methods are employed, the complexity would be that of standard matrix multiplication, hence the same complexity of standard DCT without changes.

V. CONCLUSIONS

This work describes a new approach to derive integer approximations of DCT transform matrices of various dimensions, which may be used in lossy and lossless transform-based codecs. Unlike other methods in the literature, where there is a restriction to the transform dimensions, namely the limitation to power-of-2 sizes, the dimension of these transform matrices may be adapted to the image characteristics making them suitable for conversion of lossy to lossless codecs.

The proposed method consists on deriving integer approximations for the DCT-II, based on the triangular matrix approach. In order to enhance the quality of the approximation, a genetic algorithm-based truncation improvement is described, and the total approximation error decreases in orders of magnitude depending on the size of the transform matrix. In terms of transform coding gain, the achieved transforms presents similar results to the original DCT-II matrices and superior to other methods in the literature for sizes between 2×2 and 8×8, disregarding the number of bits used to represent its values. For matrices of 16×16 or larger, it is possible to trade-off between coding gain and the bit depth to represent the matrix values. This flexibility makes the proposed approach very attractive to be used in lossy to lossless transform-based encoders. Not only for 2D images, but also for video, volumetric or light field data, where higher dimension transforms matrices might be efficiently used.

REFERENCES

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine trans-form," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974, doi: 10.1109/T-C.1974.223784.

[2] *Information Technology Digital Compression and Coding of Continuous-tone Still Images: Requirements and Guide-lines*, International Organization for Standardization, Geneva, Switzerland, Feb. 1994.

[3] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012, doi: 10.1109/TCSVT.2012.2221191.

[4] W. K. Pratt, *Digital Image Processing.*, 3rd ed. New York, NY, USA: Wiley, 2001.

[5] M. Siekmann *et al.*, "Set of transforms," presented at the Joint Video Experts Team 10th Meeting, San Diego, CA, USA, Contribution JVET-J0040, Apr. 2019.

[6] P. Schelkens *et al.*, "JPEG Pleno light field coding technologies," in *Proc. SPIE Opt. Eng. + App.*, San Diego, CA, USA, 2019, pp. 391–401.

[7] M. B. de Carvalho *et al.*, "A 4D DCT-based lenslet light field codec," in *25th IEEE Int. Conf. Image Process. (ICIP)*, Athens, Greece, 2018, pp. 435–439.

[8] J. M. Santos, P. A. A. Assuncao, L. A. da Silva Cruz, L. Tavora, R. Fonseca-Pinto, and S. M. M. Faria, "Lossless light-field compression using reversible colour transformations," in *7th Int. Conf. Image Process. Theory, Tools Appl. (IPTA)*, Montreal, QC, Canada, 2017, pp. 1–6.

[9] P. Helin, P. Astola, B. Rao, and I. Tabus, "Sparse modelling and predictive coding of subaperture images for lossless plenoptic image compression," in *Proc. 3DTV*, Hamburg, Germany, 2016, pp. 1–4.

[10] I. Schiopu and A. Munteanu, "Deep-learning-based macro-pixel synthesis and lossless coding of light field images" *APSIPA Trans. Signal Inf. Process.*, vol. 8, pp. 1–12, Jul. 2019, doi: 10.1017/ATSIP.2019.14.

[11] J. M. Santos, P. A. A. Assuncao, L. A. da Silva Cruz, L. M. N. Tvora, R. Fonseca-Pinto, and S. M. M. Faria, "Lossless compression of light fields using multi-reference minimum rate predictors," present at the Joint Pictures Expert Group, 82nd JPEG Meeting, Lisbon, Portugal, Contribution WG1M82036, Jan. 2019.

[12] S.-C. Pei and J.-J. Ding, "General reversible integer transform conversion using the triangular matrix scheme," 2006. [Online]. Available: https://www.semanticscholar.org/paper/General-Reversible-Integer-Transform-Conversion-Pei-Ding/bf1c3c63cfca437ac697c2ed135cf5540df5a9da

[13] *ITU-T Recommendation H.264: Advanced Video Coding for Generic Audiovisual Services*, International Telecommunications Union, Geneva, Switzerland, May 2003.

[14] *ITU-T Recommendation H.265: High Efficiency Video Coding*, International Telecommunications Union, Geneva, Switzerland, Apr. 2013.

[15] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC" *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 598–603, Dec. 2003, doi: 10.1109/TCSVT.2003.814964.

[16] M. Wien, *High Efficiency Video Coding: Coding Tools and Specification.,* 1st ed., Berlin, Germany: Springer, 2015.

[17] Z.-M. Lu and S.-Z. Guo, *Lossless Information Hiding in Images.,* 1st ed., Cambridge, MA, USA: Elsevier, 2017.

[18] J. Liang and T. D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme" *IEEE Trans. Singal Process.*, vol. 49, no. 12, pp. 3032–3044, Dec. 2001, doi: 10.1109/78.969511.

[19] G. Strang, *Linear Algebra and Its Applications*, 4th ed. Belmont, CA, USA: Thomson, Brooks/Cole, 2006, pp. 225–259.

[20] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video.,* 1st ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2017.

[21] S. Jana and P. Moulin, "Optimality of KLT for high-rate transform coding of Gaussian vector-scale mixtures: Application to reconstruction, estimation, and classification," *IEEE Trans. Inf. Theory*, vol. 52, no. 9, pp. 4049–4067, Sep. 2006, doi: 10.1109/TIT.2006.880056.

[22] F. Giesen, "Bink 2.2 integer DCT design," 2013. Accessed: Jan. 28, 2020. [Online]. Available: https://fgiesen.wordpress.com/2013/11/04/bink-2-2-integer-dct-design-part-1/

[23] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, 1965.