x-DTT: A package for calculating real and Integer Discrete Tchebichef Transform kernels based on orthogonal polynomials

Ahcen Aliouat, Nasreddine Kouadria, Doru Florin Chiper

SoftwareX 23 (2023) 101441

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

x-DTT: A package for calculating real and Integer Discrete Tchebichef Transform kernels based on orthogonal polynomials

Ahcen Aliouat [a,*], Nasreddine Kouadria [a], Doru Florin Chiper [b]

[a] LASA Laboratory, Electronics Department, Faculty of Technology, Badji Mokhtar - Annaba University, Algeria
[b] Dept. of Applied Electronics, Technical University "Gh. Asachi" Iasi, Romania

ARTICLE INFO

Article history:
Received 1 March 2023
Received in revised form 6 June 2023
Accepted 12 June 2023

Dataset link: https://github.com/ahcen23/x-DTT/blob/main/data.rar, https://github.com/ahcen23/x-DTT/blob/main/data.rar

Keywords:
Discrete Tchebichef Transform
Integer Discrete Tchebichef Transform
Image compression
Features extraction

ABSTRACT

This paper presents a MATLAB implementation of the exact real Discrete Tchebichef Transform (DTT) and Integer Discrete Tchebichef Transform (IDTT) with their inverse functions. It is a simple and scalable package that is the first of its kind for MATLAB. This software provides an efficient way to advance research related to DTT, IDTT, and their inverses. It is applicable to all domains where Chebyshev moments are employed, including feature extraction, image and video compression, and other related image processing. The package supports matrix generation of any size, from 2×2 to any desired size, which makes it scalable. The presented software is easy to use and generates the exact DTT transformation kernel, integer IDTT kernel, and diagonal matrix. The performance of DTT and IDTT-based JPEG compression compared to DCT-based JPEG compression was evaluated using standard metrics. Overall, this package is a valuable tool for researchers and practitioners working in the field of image and video processing.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00146
Permanent link to Reproducible Capsule	https://github.com/ahcen23/x-DTT
Legal Code License	The 3-Clause BSD License
Code versioning system used	/Github
Software code languages, tools, and services used	MATLAB
Compilation requirements, operating environments & dependencies	MATLAB 2009a or newer
Link to developer documentation	https://github.com/ahcen23/x-DTT/blob/main/README.md
Support email for questions	ahcen.aliouat@univ-annaba.org

1. Motivation and significance

The Discrete Tchebichef Transform (DTT) is a transformation that has lately been utilized as an alternative to the Discrete Cosine Transform (DCT) [1] for image and video compression [2–6], feature extraction [7–9] and image watermarking [10]. Generating DTT is based on Discrete Chebyshev Polynomials (DCP) which are orthogonal polynomials widely used for data fitting and approximation. The popularity of the DTT is due to its high transform efficiency and energy compaction. Other advantages are the high performance in data compression and relatively low complexity [11,12]. Consequently, the DTT is a potential competitor to the DCT in low delay video coding standards [13–15].

Various attempts have been made, like the DCT [16,17], to minimize the spatial and temporal complexity of the DTT to enable its implementation in embedded systems [18,19]. One such attempt is the Integer DTT (IDTT), which expresses the DTT transform matrix using only integer coefficients. This approach reduces the floating-point multiplications and division operations required during the transform computation process. However, calculating the integer DTT matrix involves multiple steps, including the calculation of DCP values using a recursive-based approach. This can be a computationally challenging problem for any programming language, especially when considering the scalability of the algorithm.

* Corresponding author.
E-mail address: ahcen.aliouat@univ-annaba.org (Ahcen Aliouat).

https://doi.org/10.1016/j.softx.2023.101441

We believe that a crucial aspect in advancing research that utilizes DTT and IDTT is the ability to quickly generate scalable and variable-sized kernel with minimal effort. To address this need, we have developed the x-DTT package, which enables the generation of DTT, IDTT, and their inverses using a single MATLAB function for matrices of various sizes.

The paper is structured as follows: The next section provides a brief overview of the real and integer discrete Tchebichef transform, which is necessary to understand the paper and examples presented. Section 3 describes the software, while Section 4 outlines the software functionalities. Implementation examples are presented in Section 1. In Section 4, we compare DTT, IDTT, and DCT to demonstrate the accuracy of xDTT results in image compression applications. Finally, Section 6 provides concluding remarks.

2. Real DTT and IDTT overview

This section offers a concise summary of the mathematical basis for the DTT and IDTT computations. Additionally, we present the depiction of the DTT based on the essential elements of the DCP. The DCP can be expressed as Δ, where Δ is the forward difference operator.

$$t_k(n) = k!\Delta^k\left[\binom{n}{k}\binom{n-N}{k}\right] \text{ for } n = 0, 1 \ldots N-1 \tag{1}$$

Some properties of the Chebyshev polynomials are given below.

2.1. Properties of DCP

(1) **Orthogonality**: The DCPs are orthogonal with respect to unit weight

$$\sum_{n=0}^{N-1} t_k(n)t_m(n) = \delta_{km}d_k, \tag{2}$$

where d_k is the squared norm of the DCP defined as

$$d(k,N) = \sum_{n=0}^{N-1}[t_k(n)]^2 = \frac{N\left(N^2-1\right)\left(N^2-2^2\right)\cdots\left(N^2-k^2\right)}{2k+1} \tag{3}$$

(2) **Symmetry**: The DCP satisfies the symmetry property

$$t_k(N-1-n) = (-1)^k t_k(n) \tag{4}$$

(3) **Recurrence Relation**: Based on Eq. (1), the DCP satisfies a three-term recurrence relation that can be expressed as follows:

$$t_0(n) = 1, t_1(n) = 2n - N + 1 \tag{5}$$

$$t_k(n) = \left[\frac{2k-1}{k}t_k(1)\right]t_{k-1}(n) - \left[\left(\frac{k-1}{k}\right)\left(N^2-(k-1)^2\right)\right]t_{k-2}(n) \tag{6}$$

2.2. Derivation of real DTT from normalized DCP

DCPs, while orthogonal, can lead to numerical instability and error magnification. To address this issue, we perform normalization using Eq. (2) to obtain $\tau_k(n) = \frac{t_k(n)}{\sqrt{d(n,N)}}$. Here, t_k represents the classical DCP, while the orthonormal DCP is denoted by τ_k. The set $\{\tau_k(n)\}$ is orthonormal, as $\forall n, m : \langle\tau_n, \tau_m\rangle = \delta_{n,m}, \{\tau_k(n)\}$. We can obtain the orthonormal DCP iteratively using τ_k in Eq. (7), which can be expressed concisely as:

$$\tau_k(n) = (a_1 n + a_2)\,\tau_{k-1}(n) + a_3\tau_{k-2}(n) \tag{7}$$

with

$$\tau_0(n) = \frac{1}{\sqrt{N}} \tag{8}$$

and

$$\tau_1(n) = (2n+1-N)\sqrt{\frac{3}{N\left(N^2-1\right)}} \tag{9}$$

where

$$a_1 = \frac{2}{k}\sqrt{\frac{4k^2-1}{N^2-k^2}}, \quad a_2 = \frac{1-N}{k}\sqrt{\frac{4k^2-1}{N^2-k^2}} \tag{10}$$

and

$$a_3 = \frac{1-k}{k}\cdot\frac{2k+1}{2k-3}\sqrt{\frac{N^2-(k-1)^2}{N^2-k^2}} \tag{11}$$

2.3. Definition of the DTT

The N-point 1-D DTT, $Y(k)$ of an input sequence $x(n)$, for $n, k = 0, 1\cdots N-1$ and the inverse 1-D DTT can be defined as

$$Y(k) = \sum_{n=0}^{N-1}\tau(k,n)x(n) \text{ and } x(n) = \sum_{k=0}^{N-1}\tau(k,n)Y(k), \tag{12}$$

where the kernel $\tau(k,n)$ represents $\tau_k(n)$, the orthonormal basis of DCP, which is given by (7).

2.4. Formulation of IDTT

The 2-D DTT can be represented in matrix form [20] as shown in

$$Y = \tau X\tau' = \left(\tau(\tau X)'\right)', \tag{13}$$

Here, X is the 2-D input matrix, τ is the Chebyshev basis, and Y is the 2-D matrix of transform coefficients.

As seen in (13), the 2-D DTT of input data X of size $N \times N$, given by Eq. (5), can also be expressed in the form shown above.

$$Y = \tau X\tau', \tag{14}$$

where τ is the ' $N \times N$ ' DTT kernel and can be factorized as shown below to separate the integer and float numbers. Consider a variable

$$q(k) = \frac{p(k)}{\sqrt{d(k,N)}}, \tag{15}$$

where

$$p(k) = \begin{cases} k! & \text{for } k = 1 \\ k!(k+1) & \text{for } k = 0, 2, 4, 6, \ldots, N \\ k!(N-k) & \text{for } k = 3, 5, 7\cdots N-1 \end{cases} \tag{16}$$

and $d(k,N)$ is the squared norm of the DTP given by

$$d(k,N) = \frac{N\left(N^2-1\right)\left(N^2-2^2\right)\cdots\left(N^2-k^2\right)}{2k+1} \tag{17}$$

Let $\boldsymbol{Q}$ be a diagonal matrix with $q(k)$ as the diagonal element of the k^{th} row. The transformation matrix τ can be expressed as

$$\begin{aligned} \tau &= \boldsymbol{Q}\hat{\tau} \\ \hat{\tau} &= \boldsymbol{Q}^{-1}\tau \end{aligned} \tag{18}$$

It is worth noting that the matrix $\boldsymbol{Q}$ can be shifted in various scenarios during the process without introducing any computational overhead. The linear representation of the DTP polynomials of the order 16×16 and 32×32 are shown in Figs. 1 and 2.

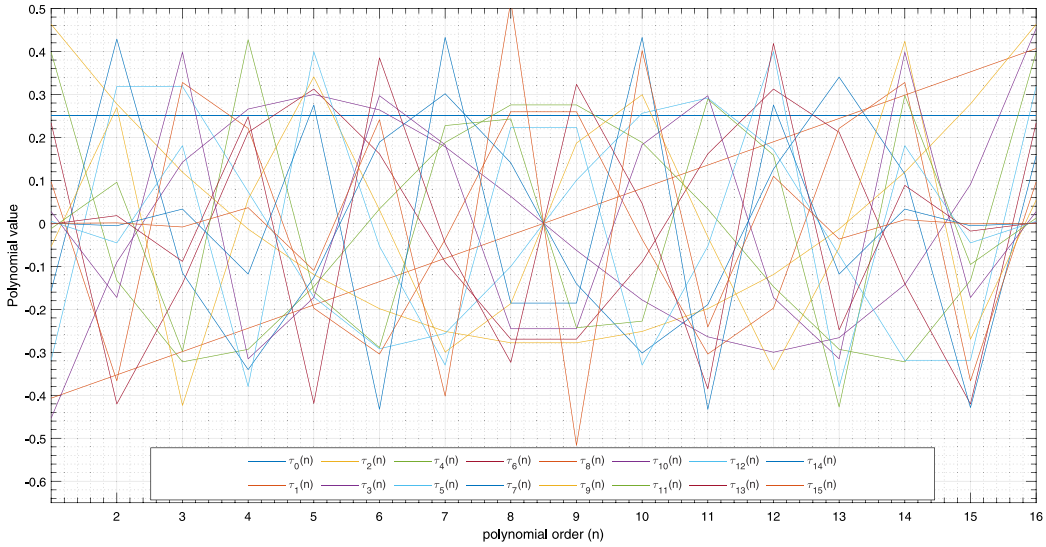


Fig. 1. An illustration of the first 16 orthonormal DCPs ($\tau_0(n)$ to $\tau_{15}(n)$).

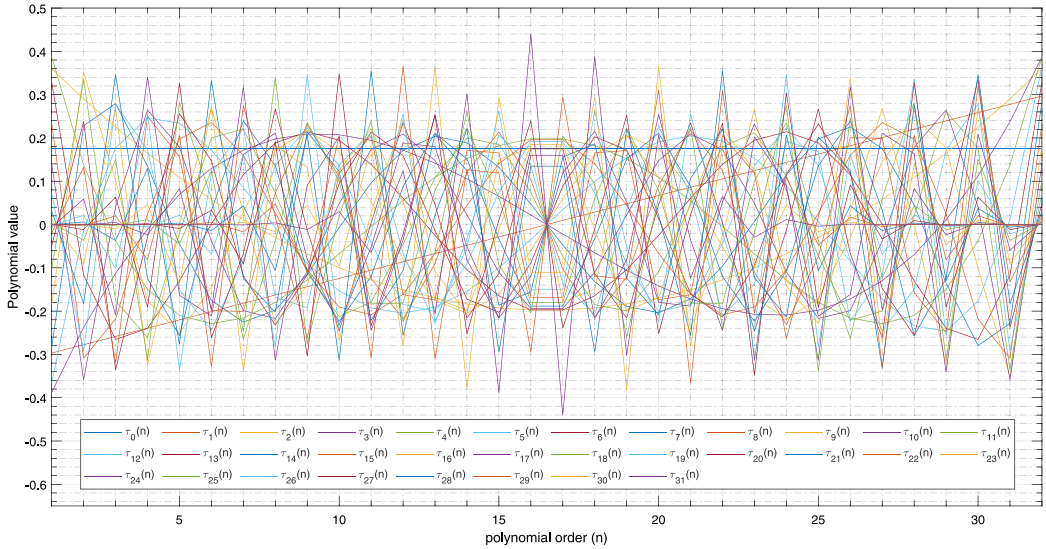


Fig. 2. An illustration of the first 32 orthonormal DCPs ($\tau_0(n)$ to $\tau_{31}(n)$).

3. Software description

The proposed software package includes a function named DTT, which takes an input N, an integer value representing the size of the generated matrix. The DTT function calculates the recursive polynomials based on the equations described in Section 2 and outputs three matrices:

- **Matrix** τ: This output is a real matrix that represents the exact DTT transformation kernel of size $N \times N$.
- **Matrix** $\hat{\tau}$: This output represents an integer matrix that is the integer kernel of the IDTT.
- **Matrix** *diag*: This output represents the diagonal matrix as shown previously in Eq. (18).

If we let *diag* represent the Q matrix, the function can be called using the following line of MATLAB code: [τ, $\hat{\tau}$, *diag*] = DTT(N). Executing this call generates the three matrices based on the theory shown in previous sections. The DTT function is easy to use, just like the integrated DCT function in MATLAB

The function can generate kernels of any size, ranging from 2×2 to a maximum tested size of 256×256, which is considerably high. However, it should be noted that generating a large kernel may take a long time due to the exponential complexity of the DCP generation process.

4. Comparison with DCT in a JPEG image compression

To confirm the correctness of the generated kernel matrices, we compared the image compression performance of a standard JPEG technique using DCT, quantization, zigzag, and Huffman encoding with a modified JPEG chain, where DCT transformation is replaced by DTT matrices generated using the proposed software.

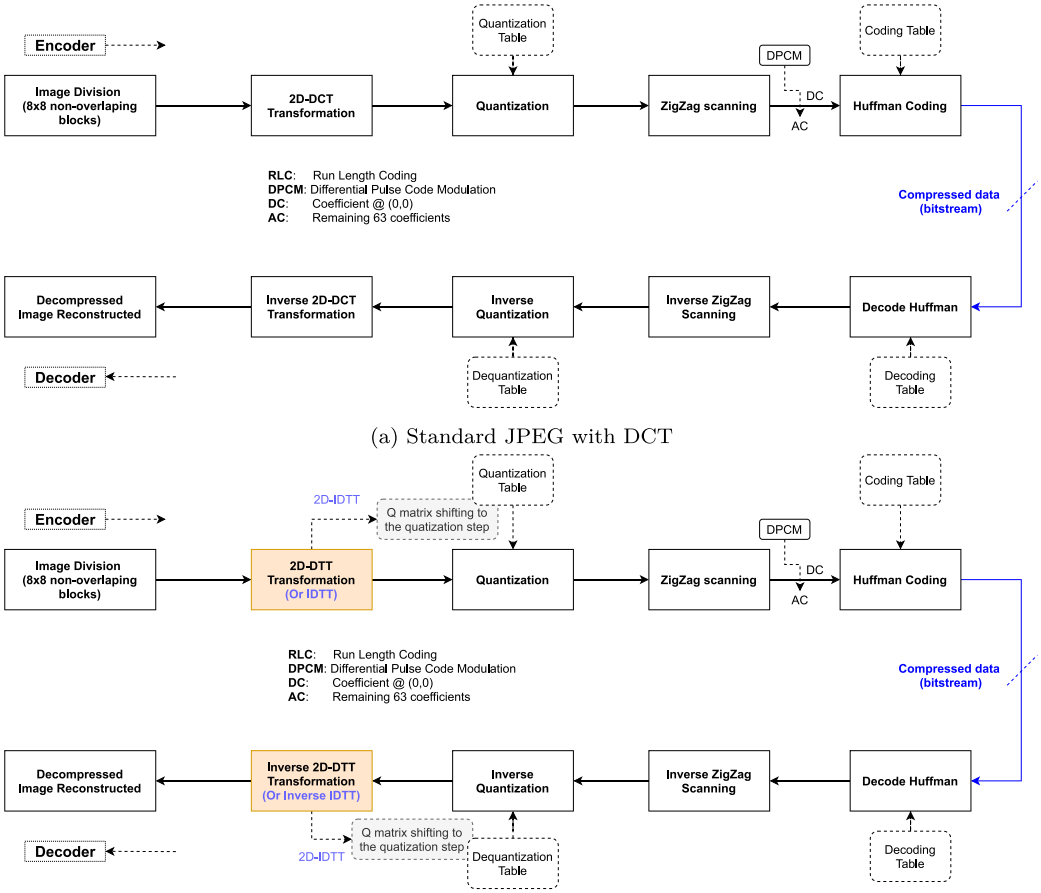


Fig. 3. JPEG chain using DCT and DTT.

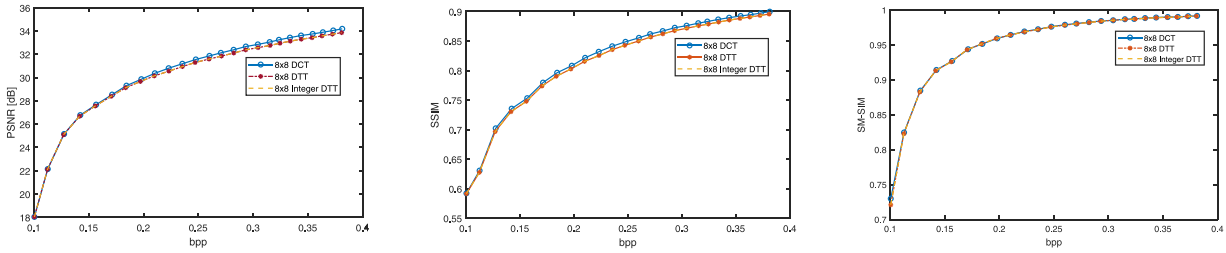
Fig. 4. Used images dataset (part from USC-SIPI Dataset [21]) for experimentation: *lena* 512 × 512, *cameraman* 256 × 256 and *boat* 512 × 512.

Fig. 3(a) shows the scheme and steps of the JPEG transform and the replacement of DCT with DTT in Fig. 3(b) (see Fig. 3).

The used standard images are shown in Fig. 4.

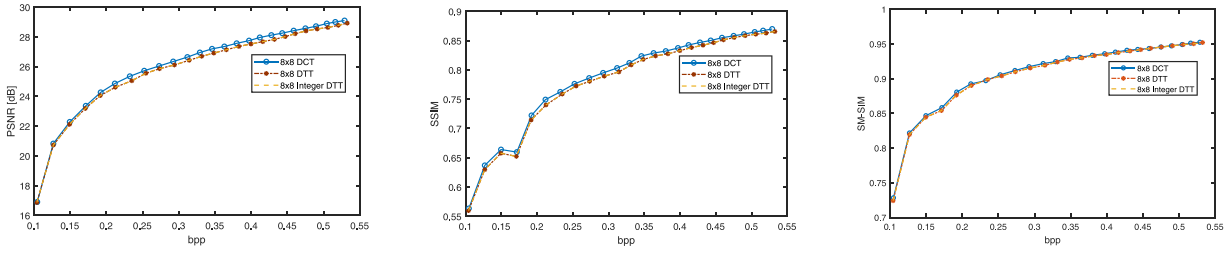
We use peak signal-to-noise ratio (PSNR) and Structural similarity (SSIM) with multiple bit-per-pixel (bpp) values for quality evaluation. Where bpp represented the ratio of the compression size to the number of original bits formulated as $bpp = S_{comp}/N_{pixels}$, where S_{comp} is the size of the compressed data and N_{pixels} is the number of pixels to compress. The formula of PSNR has been presented in [22], while the SSIM is described in [23]. The experiments are performed using MATLAB 2021a software running on a quad-core i7 2.5 GHz laptop.

Figs. 5, 6, and 7 show the compression results for the dataset used in the experiment when using DTT and IDTT-based JPEG



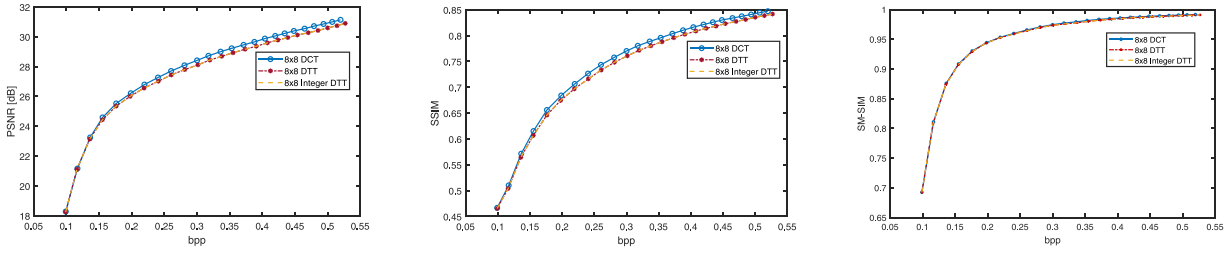
(a) PSNR (b) SSIM (c) SR-SIM

Fig. 5. Compression results comparison for **Lena** image.



(a) PSNR (b) SSIM (c) SR-SIM

Fig. 6. Compression results comparison for **Cameraman** image.



(a) PSNR (b) SSIM (c) SR-SIM

Fig. 7. Compression results comparison for **Boat** image.

compression. It is observed that the qualitative results for DTT and IDTT-based compression are almost similar to those of DCT-based compression.

However, for relatively high bpp values, DCT slightly outperforms DTT and IDTT for all the used sequences. These results are consistent with the theoretical findings reported in the literature regarding the difference between DTT and DCT for natural images [11].

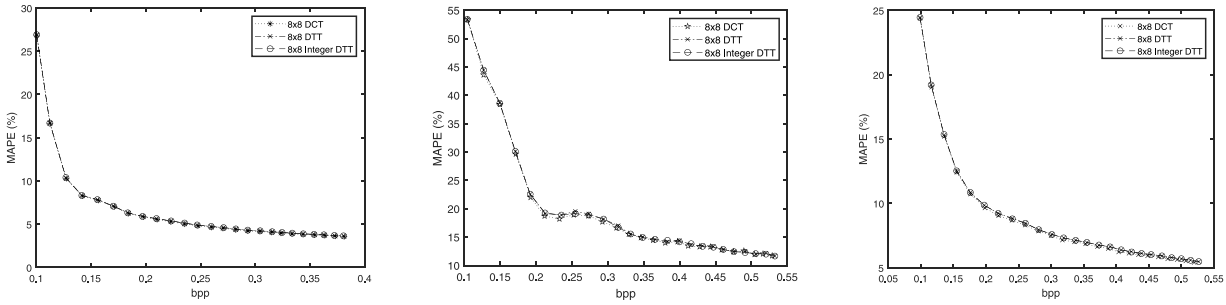
Fig. 8 shows the percentage of the error between the original image and the compression image using different transformation kernels and for different compression rates.

5. Impact

The x-DTT package offers a user-friendly solution for generating DCPs utilized in DTT, IDTT, and their corresponding inverses. This enables researchers to effortlessly engage in and advance the fields of image compression [2], Feature Extraction [8,9], and fast transform algorithms [3]. The versatility of x-DTT lies in its capability to generate orthonormal Chebyshev polynomials of various sizes. The impact of this remarkable feature lies in its ability to simplifies the development of theories behind the algorithms based on DCPs. Notably, the application of DTT and IDTT as compression techniques continues to be a prominent and rapidly evolving research area in the literature. It has been identified as a promising candidate for replacing DCT-based compression techniques. We firmly believe that the x-DTT package has the potential to significantly contribute to the advancement of this field.

6. Conclusion

The x-DTT software presented in this paper is a user-friendly and powerful package for generating the kernels of exact and integer DTT transformations in MATLAB. This package enables researchers to easily develop, utilize, and study the features and



(a) Lena (b) Cameraman (c) Boat

Fig. 8. Average percentage error for different compression rate for each transform.

MATLAB Command Window Page 1
27 décembre 2022 03:57:20

```
>> [a,b,c]=DTT(2)

a =

   0.7071    0.7071
  -0.7071    0.7071


b =

    1    1
   -1    1


c =

   1.4142         0
        0    1.4142

>>
```

Fig. A.9. 2×2 DTT results.

concepts related to the DTT transform. The strength of the proposed package lies in its simplicity and scalability for almost all transform sizes. In future work, we plan to add other algorithms for fast computing of the DTT in the same package.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The raw data required to reproduce the above findings are available to download from https://github.com/ahcen23/x-DTT/blob/main/data.rar. The processed data required to reproduce the above findings are also available to download from https://github.com/ahcen23/x-DTT/blob/main/data.rar.

Appendix A. Examples: output matrices of the DTT function

In this section, we demonstrate the generation of kernels of different sizes using our function. In the examples below, matrix a represents τ, matrix b represents $\hat{\tau}$, and matrix c represents the diagonal matrix $diag$.

Figs. A.9–A.11 show examples for generating 2×2, 4×4 and 8×8 DTT kernels. Figs. A.12–A.14 show the exact DTT matrix τ, the IDTT $\hat{\tau}$, and the diagonal matrix $diag$ for 16×16 size.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2023.101441.

```
MATLAB Command Window                                          Page 1

>> [a,b,c]=DTT(4)

a =

    0.5000    0.5000    0.5000    0.5000
   -0.6708   -0.2236    0.2236    0.6708
    0.5000   -0.5000   -0.5000    0.5000
   -0.2236    0.6708   -0.6708    0.2236


b =

     1     1     1     1
    -3    -1     1     3
     1    -1    -1     1
    -1     3    -3     1


c =

    2.0000         0         0         0
         0    4.4721         0         0
         0         0    2.0000         0
         0         0         0    4.4721

>>
```

Fig. A.10. 4 × 4 DTT results.

```
MATLAB Command Window                                          Page 1

>> [a,b,c]=DTT(8)

a =

    0.3536    0.3536    0.3536    0.3536    0.3536    0.3536    0.3536    0.3536
   -0.5401   -0.3858   -0.2315   -0.0772    0.0772    0.2315    0.3858    0.5401
    0.5401    0.0772   -0.2315   -0.3858   -0.3858   -0.2315    0.0772    0.5401
   -0.4308    0.3077    0.4308    0.1846   -0.1846   -0.4308   -0.3077    0.4308
    0.2820   -0.5238   -0.1209    0.3626    0.3626   -0.1209   -0.5238    0.2820
   -0.1498    0.4922   -0.3638   -0.3210    0.3210    0.3638   -0.4922    0.1498
    0.0615   -0.3077    0.5539   -0.3077   -0.3077    0.5539   -0.3077    0.0615
   -0.0171    0.1195   -0.3585    0.5974   -0.5974    0.3585   -0.1195    0.0171


b =

     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


c =

    2.8284         0         0         0         0         0         0         0
         0   12.9615         0         0         0         0         0         0
         0         0   12.9615         0         0         0         0         0
         0         0         0   16.2481         0         0         0         0
         0         0         0         0   24.8193         0         0         0
         0         0         0         0         0   46.7333         0         0
         0         0         0         0         0         0   16.2481         0
         0         0         0         0         0         0         0   58.5833

>>
```

Fig. A.11. 8 × 8 DTT results.

```
MATLAB Command Window

>> [a,~,~]=DTT(16)
a =
   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500   0.2500
  -0.4067  -0.3525  -0.2983  -0.2440  -0.1898  -0.1356  -0.0813  -0.0271   0.0271   0.0813   0.1356   0.1898   0.2440   0.2983   0.3525   0.4067
   0.4631   0.2779   0.1191  -0.0132  -0.1191  -0.1985  -0.2514  -0.2779  -0.2779  -0.2514  -0.1985  -0.1191  -0.0132   0.1191   0.2779   0.4631
  -0.4532  -0.0906   0.1424   0.2660   0.2998   0.2640   0.1783   0.0628  -0.0628  -0.1783  -0.2640  -0.2998  -0.2660  -0.1424   0.0906   0.4532
   0.3981  -0.1327  -0.3223  -0.2931  -0.1473   0.0335   0.1881   0.2756   0.2756   0.1881   0.0335  -0.1473  -0.2931  -0.3223  -0.1327   0.3981
  -0.3185   0.3185   0.3185   0.0735  -0.1715  -0.2918  -0.2562  -0.1002   0.1002   0.2562   0.2918   0.1715  -0.0735  -0.3185  -0.3185   0.3185
   0.2335  -0.4202  -0.1401   0.2119   0.3125   0.1616  -0.0898  -0.2694  -0.2694  -0.0898   0.1616   0.3125   0.2119  -0.1401  -0.4202   0.2335
  -0.1569   0.4288  -0.1150  -0.3403  -0.1263   0.1890   0.3017   0.1408  -0.1408  -0.3017  -0.1890   0.1263   0.3403   0.1150  -0.4288   0.1569
   0.0964  -0.3664   0.3278   0.2210  -0.1973  -0.3041  -0.0371   0.2596   0.2596  -0.0371  -0.3041  -0.1973   0.2210   0.3278  -0.3664   0.0964
  -0.0539   0.2697  -0.4238   0.0563   0.3408   0.0314  -0.2993  -0.1867   0.1867   0.2993  -0.0314  -0.3408  -0.0563   0.4238  -0.2697   0.0539
   0.0272  -0.1725   0.3982  -0.3152  -0.1725   0.2970   0.1829  -0.2452  -0.2452   0.1829   0.2970  -0.1725  -0.3152   0.3982  -0.1725   0.0272
  -0.0123   0.0957  -0.2976   0.4273  -0.1588  -0.2892   0.2275   0.2429  -0.2429  -0.2275   0.2892   0.1588  -0.4273   0.2976  -0.0957   0.0123
   0.0048  -0.0454   0.1808  -0.3800   0.3993  -0.0532  -0.3297   0.2233   0.2233  -0.3297  -0.0532   0.3993  -0.3800   0.1808  -0.0454   0.0048
  -0.0016   0.0180  -0.0884   0.2479  -0.4187   0.3851  -0.0462  -0.3235   0.3235   0.0462  -0.3851   0.4187  -0.2479   0.0884  -0.0180   0.0016
   0.0004  -0.0056   0.0333  -0.1180   0.2754  -0.4328   0.4328  -0.1855  -0.1855   0.4328  -0.4328   0.2754  -0.1180   0.0333  -0.0056   0.0004
  -0.0001   0.0012  -0.0084   0.0365  -0.1096   0.2411  -0.4019   0.5167  -0.5167   0.4019  -0.2411   0.1096  -0.0365   0.0084  -0.0012   0.0001
>>
```

Fig. A.12. The "a" matrix results for 16 × 16 DTT.

```
MATLAB Command Window

>> [~,~,~]=DTT(16),num2str(b,'%10d')
ans =
  16×154 char array
    '  1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1'
    ' -15     -13     -11      -9      -7      -5      -3      -1       1       3       5       7       9      11      13      15'
    '  35      21       9      -1      -9     -15     -19     -21     -21     -19     -15      -9      -1       9      21      35'
    ' -35      -7      11      21      23      20      14       5      -5     -14     -20     -23     -21     -11       7      35'
    ' 273     -91    -221    -201    -101      23     129     189     189     129      23    -101    -201    -221     -91     273'
    '-273     273     273      63    -147    -250    -220     -86      86     220     250     147     -63    -273    -273     273'
    ' 715   -1287    -429     649     957     495    -275    -825    -825    -275     495     957     649    -429   -1287     715'
    '-715    1954    -524   -1551    -576     862    1375     642    -642   -1375    -862     576    1551     524   -1954     715'
    ' 715   -2717    2431    1639   -1463   -2255    -275    1925    1925    -275   -2255   -1463    1639    2431   -2717     715'
    '-715    3575   -5618     746    4518     416   -3968   -2475    2475    3968    -416   -4518    -746    5618   -3575     715'
    ' 273   -1729    3991   -3159   -1729    2977    1833   -2457   -2457    1833    2977   -1729   -3159    3991   -1729     273'
    '-273    2129   -6622    9508   -3533   -6435    5062    5405   -5405   -5062    6435    3533   -9508    6622   -2129     273'
    '  35    -329    1309   -2751    2891    -385   -2387    1617    1617   -2387    -385    2891   -2751    1309    -329      35'
    ' -35     390   -1916    5369   -9070    8342   -1001   -7007    7007    1001   -8342    9070   -5369    1916    -390      35'
    '   1     -13      77    -273     637   -1001    1001    -429    -429    1001   -1001     637    -273      77     -13       1'
    '  -1      15    -105     455   -1365    3003   -5005    6435   -6435    5005   -3003    1365    -455     105     -15       1'
>>
```

Fig. A.13. The "b" matrix results for 16 × 16 DTT.

```
MATLAB Command Window

>> [~,~,~]=DTT(16),num2str(c,'%10d')
ans =
  16×181 char array
    '4          0          0          0          0          0          0          0          0          0          0          0          0          0          0          0'
    '03.687818e+01          0          0          0          0          0          0          0          0          0          0          0          0          0          0'
    '0          07.557777e+01          0          0          0          0          0          0          0          0          0          0          0          0          0'
    '0          0          07.722096e+01          0          0          0          0          0          0          0          0          0          0          0          0'
    '0          0          0          06.857755e+02          0          0          0          0          0          0          0          0          0          0          0'
    '0          0          0          0          08.570776e+02          0          0          0          0          0          0          0          0          0          0'
    '0          0          0          0          0          03.062666e+03          0          0          0          0          0          0          0          0          0'
    '0          0          0          0          0          0          04.557937e+03          0          0          0          0          0          0          0          0'
    '0          0          0          0          0          0          0          07.415665e+03          0          0          0          0          0          0          0'
    '0          0          0          0          0          0          0          0          01.325619e+04          0          0          0          0          0          0'
    '0          0          0          0          0          0          0          0          0          01.002198e+04          0          0          0          0          0'
    '0          0          0          0          0          0          0          0          0          0          02.225339e+04          0          0          0          0'
    '0          0          0          0          0          0          0          0          0          0          0          07.240098e+03          0          0          0'
    '0          0          0          0          0          0          0          0          0          0          0          0          02.166062e+04          0          0'
    '0          0          0          0          0          0          0          0          0          0          0          0          0          02.312765e+03          0'
    '0          0          0          0          0          0          0          0          0          0          0          0          0          0          01.245462e+04'
>>
```

Fig. A.14. The "c" matrix results for 16 × 16 DTT.

References

[1] Ahmed N, Natarajan T, Rao KR. Discrete cosine transform. IEEE Trans Comput 1974;100(1):90–3.

[2] Chen J, Liu S, Deng G, Rahardja S. Hardware efficient integer discrete cosine transform for efficient image/video compression. IEEE Access 2019;7:152635–45.

[3] Kouadria N, Mechouek K, Harize S, Doghmane N. Region-of-interest based image compression using the discrete Tchebichef transform in wireless visual sensor networks. Comput Electr Eng 2019;73:194–208.

[4] Aliouat A, Kouadria N, Harize S, Maimour M. Multi-threshold-based frame segmentation for content-aware video coding in WMSN. In: Advances in computing systems and applications: proceedings of the 5th conference on computing systems and applications. Springer; 2022, p. 337–47.

[5] Aliouat A, Kouadria N, Maimour M, Harize S. Region-of-interest based video coding strategy for low bitrate surveillance systems. In: 2022 19th international multi-conference on systems, signals & devices. SSD, IEEE; 2022, p. 1357–62.

[6] Aliouat A, Kouadria N, Maimour M, Harize S, Doghmane N. Region-of-interest based video coding strategy for rate/energy-constrained smart surveillance systems using WMSNs. Ad Hoc Netw 2022;103076.

[7] Jassim WA, Raveendran P. Face recognition using discrete Tchebichef-Krawtchouk transform. In: 2012 IEEE international symposium on multimedia. IEEE; 2012, p. 120–7.

[8] Marcos JV, Cristóbal G. Texture classification using discrete Tchebichef moments. J Opt Soc Amer A 2013;30(8):1580–91.

[9] Sutojo T, Rachmawanto EH, Sari CA, et al. Fast and efficient image watermarking algorithm using discrete tchebichef transform. In: 2017 5th international conference on cyber and IT service management. CITSM, IEEE; 2017, p. 1–5.

[10] Setiadi DRIM, Sutojo T, Rachmawanto EH, Sari CA. Fast and efficient image watermarking algorithm using discrete tchebichef transform. In: 2017 5th international conference on cyber and IT service management. CITSM, 2017, p. 1–5. http://dx.doi.org/10.1109/CITSM.2017.8089229.

[11] Liang W-D, Liu X-D. Comparison of approximate DCT and approximate DTT for image compression. In: 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering. ICBAIE, IEEE; 2021, p. 337–41.

[12] Paim G, Rocha LMG, Santana GM, Soares LB, da Costa EAC, Bampi S. Power-, area-, and compression-efficient eight-point approximate 2-D discrete tchebichef transform hardware design combining truncation pruning and efficient transposition buffers. IEEE Trans Circuits Syst I Regul Pap 2018;66(2):680–93.

[13] Aliouat A, Kouadria N, Harize S, Maimour M. An efficient Low Complexity Region-of-interest detection for video coding in wireless visual surveillance. IEEE Access 2023;1. http://dx.doi.org/10.1109/ACCESS.2023.3248067.

[14] Žádník J, Mäkitalo M, Vanne J, Jääskeläinen P. Image and video coding techniques for ultra-low latency. ACM Comput Surv 2022;54(11s):1–35.

[15] Mansri I, Doghmane N, Kouadria N, Harize S, Bekhouch A. Comparative evaluation of VVC, HEVC, H. 264, AV1, and VP9 encoders for low-delay video applications. In: 2020 fourth international conference on multimedia computing, networking and applications. MCNA, IEEE; 2020, p. 38–43.

[16] Chiper D-F, Swamy M, Ahmad MO. An efficient unified framework for implementation of a prime-length DCT/IDCT with high throughput. IEEE Trans Signal Process 2007;55(6):2925–36.

[17] Cintra RJ, Bayer FM. A DCT approximation for image compression. IEEE Signal Process Lett 2011;18(10):579–82.

[18] Oliveira PA, Cintra RJ, Bayer FM, Kulasekera S, Madanayake A. Low-complexity image and video coding based on an approximate discrete Tchebichef transform. IEEE Trans Circuits Syst Video Technol 2016;27(5):1066–76.

[19] Kouadria N, Mechouek K, Messadeg D, Doghmane N. Pruned discrete Tchebichef transform for image coding in wireless multimedia sensor networks. AEU-Int J Electron Commun 2017;74:123–7.

[20] Jain AK. Fundamentals of digital image processing. Prentice-Hall, Inc.; 1989.

[21] Weber AG. The USC-SIPI image database: Version 5. 2006, http://sipi.usc.edu/database/.

[22] Huynh-Thu Q, Ghanbari M. Scope of validity of PSNR in image/video quality assessment. Electron Lett 2008;44(13):800–1.

[23] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 2004;13(4):600–12.