Towards new small-kernel-based discrete orthogonal transforms and their algorithmic implementation

Aleksandr Cariow

West Pomeranian University of Technology in Szczecin
Faculty of Computer Science and Information Technologies, Żołnierska 52,
71-210 Szczecin, Poland
e-mail: atariov@zut.edu.pl

Abstract. This note presents a simple and practical idea for synthesizing new discrete orthogonal transform (DOT) algorithms based on the properties of the Kronecker product of a sequence of orthogonal matrices. Specifically, it exploits the fact that the Kronecker product of orthogonal matrices is itself orthogonal. The proposed approach utilizes small-sized DOT kernels to construct new algorithms. By skillfully choosing kernels, one can create new orthogonal bases and simultaneously construct fast algorithms for discrete transforms in these new bases.
Keywords: discrete orthogonal transforms, fast algorithms, digital signal processing

1. Introduction

Discrete orthogonal transforms (DOTs) are a fundamental mathematical tool used in modern digital signal processing, image analysis, scientific computing, and data compression. Their central idea is to represent discrete data in an alternative basis, where essential characteristics and parameters—such as frequency components, energy distributions, or localized features—become more explicit. This basis change, based on the theory of orthonormal vector spaces, ensures energy conservation, numerical stability, and invertibility, making these transforms both theoretically rigorous and practically useful. The roots of discrete orthogonal transforms date back to the classical work of Jean Baptiste Joseph Fourier, whose ideas on representing functions using trigonometric series laid the foundation for spectral analysis [1]. Over time, these concepts have evolved into numerous discrete transforms using different systems of basis functions, including the discrete Fourier transform (DFT) [2-5], eight types of discrete cosine and sine transforms (DCTs, DSTs) [4-10], the discrete Hartley transform [11], the discrete Haar transform [5], the Walsh-Hadamard transform [5, 11], the Slant transform [12], and transforms based on orthogonal polynomials [13]. Various "exotic" transforms with unique properties have also been developed [14-17]. Each of these has unique structural properties—global versus localized representation, different boundary conditions, or varying levels of space-frequency resolution—that make them suitable for specific types of problems. Although discrete orthogonal transforms are versatile, their direct computation often requires a large number of arithmetic operations, typically $O(N^2)$. This led to the creation of fast algorithms, with the Fast Fourier Transform (FFT) introduced by Cooley and Tukey being the most well-known example. By reducing the complexity of the DFT to $O(N\log_2 N)$, the FFT sparked a wide range of efficient computational techniques. Similar approaches have been applied to other transforms, resulting in fast cosine transforms, fast sine transforms, lifting-based wavelet decompositions, and algorithms that utilize symmetries, sparsity, or factorization of orthogonal matrices. The advent of these fast algorithms has transformed digital technology, enabling real-time audio and video processing, accelerating large-scale numerical simulations, and improving highly efficient compression schemes such as JPEG, JPEG2000, MP3, and modern streaming codecs. Today, fast orthogonal transforms remain essential, especially in high-dimensional data analysis, scientific visualization, and machine learning workflows, where both accuracy and speed are crucial. As a result, discrete orthogonal transforms and their rapid implementations are fundamental to modern digital signal processing. Their development continues to be an active research area, driven by the need

for higher-dimensional methods, optimized GPU-based implementations, and adaptive or data-driven transform designs.

Alongside classical transforms, a particularly active research direction concerns the synthesis of new orthogonal bases tailored to the statistical or geometric properties of modern data. Classical bases—Fourier, cosine, or wavelet—though powerful, are not universally suitable for all classes of signals. Standard bases are often suboptimal for signals exhibiting nonstationarity, local anisotropies, irregular sampling, or high-dimensional structure. Real-world data often exhibit structural characteristics such as nonstationarity, sparsity, irregular sampling, anisotropy, or complex geometric features that traditional transforms do not capture well. Designing new orthogonal systems enables more compact representation, improved denoising performance, or more effective feature extraction.

Custom-designed orthonormal systems can offer significantly improved sparsity, better representation of edges or transients, or enhanced robustness to noise. However, the usefulness of such bases depends critically on the availability of efficient computational methods. Thus, developing fast algorithms for transforms in these bases is an essential complement to the theoretical design. This includes exploring matrix factorizations, recursive construction schemes, and structural optimizations that reduce computational cost while preserving stability. As a result, both the construction of novel orthogonal bases and the engineering of fast transform algorithms represent a coherent, mutually reinforcing research effort. This dual process—building new orthogonal bases and engineering fast transforms on them—ensures that mathematical innovations can be translated into practical tools for signal processing, scientific computing, and machine learning. Together, they expand the capabilities of contemporary computational mathematics and digital signal processing, meeting the growing demands of real-time systems, high-dimensional data analysis, and modern machine-learning applications. Taking into account the above, this paper presents a simple and realistic idea demonstrating the possibility of synthesizing new basis function systems and corresponding fast algorithms using small-sized discrete orthogonal transforms as kernels.

2. Preliminary remarks

In this note, we use the following notations:

$\mathbf{I}_N$ is an identity matrix of order N;

$\mathbf{A}_{n_i}$, $\mathbf{A}_{N_i}$ - are $n_i \times n_i$ and $N_i \times N_i$ orthogonal matrices represented small-size kernels.

$\mathbf{W}_N^{(i)}$ – are $N \times N$ block-diagonal matrices describing computations for each iteration of the algorithms;

$\otimes$ is the Kronecker product of two matrices [18];

We will present the description of the developed algorithms in two ways: using matrix-vector computational procedures and using data flow graphs. In this article, all data flow graphs represent the flow of data from left to right. Straight lines represent data transfer operations (data transfer paths), and rectangles represent the multiplication of matrices, denoted by the letters inside the rectangles, with input vectors. Circles (see Fig. 4) indicate operations of multiplication by a normalizing constant $s_n^{(N)}$.

3. Strategy for constructing discrete orthogonal transform algorithms

Let us assume that we are dealing with an N-point sequence of input data. We also assume that N is a product of positive integers: $N(i)=N_1 \times N_2 \times \ldots \times N_i$, $i=1, \ldots, K$, and $N(0)=1$. Then the base matrix of the generalized discrete orthogonal transform can be defined as follows:

$$\mathbf{A}_N^{(N_1, N_2, \ldots, N_K)} = \mathbf{A}_{N_1} \otimes \ldots \otimes \mathbf{A}_{N_K} \tag{1}$$

where $\{\mathbf{A}_{n_i}\}$ is a set of orthogonal $n_i \times n_i$ matrices, which are the basic kernels of a specific discrete transform (for now, it doesn't matter which one). We exploit the fact that the Kronecker product of orthogonal matrices is also orthogonal.

Let K be the number of factor-matrices $\mathbf{W}_N^{(i)}$ obtained by factorizing the order $N \times N$ original transform matrix. Then $N(K)=N$.

The well-known Fundamental Tensor-Product Factorization Theorem (see, for example, [19]) is of the form $\mathbf{A}_{N_1}\otimes\ldots\otimes\mathbf{A}_{N_K}=\prod_{i=1}^{K}(\mathbf{I}_{N(i-1)}\otimes\mathbf{A}_{n_i}\otimes\mathbf{I}_{N/N(i)})$, and this factorization is true for different permutations of the factors $(\mathbf{I}_{N(i-1)}\otimes\mathbf{A}_{n_i}\otimes\mathbf{I}_{N/N(i)})$. We will use this decomposition in our further discussions.

Let $\mathbf{X}_{N\times 1}=[x_0,x_1,\ldots,x_{N-1}]^T$ be a vector, discribing input data swquence and $\mathbf{Y}_{N\times 1}^{(N_1,\,N_2,\ldots,\,N_K)}=[y_0,y_1,\ldots,y_{N-1}]^T$ be a vector, discribing coefficients of the discrete orthogonał transform.

Then, using the corollary of the Fundamental Tensor-Product Factorization Theorem, we can propose a generalized matrix-vector procedure for calculating the coefficients of some "generalized" discrete orthogonal transform that uses the "grafting" of small-sized base kernels $\mathbf{A}_{n_i}$ into a single whole:

$$\mathbf{Y}_{N\times 1}^{(N_1,\,N_2,\ldots,\,N_K)}=\prod_{i=1}^{K}(\mathbf{I}_{N(i-1)}\otimes\mathbf{A}_{N_i}\otimes\mathbf{I}_{N/N(i)})\mathbf{X}_{N\times 1} \tag{2}$$

or in more compact form:

$$\mathbf{Y}_{N\times 1}^{(N_1,\,N_2,\ldots,\,N_K)}=\mathbf{W}_N^{(K)}\cdots\mathbf{W}_N^{(1)}\mathbf{X}_{N\times 1}, \tag{3}$$

where $$\mathbf{W}_N^{(i)}=\mathbf{I}_{N(i-1)}\otimes\mathbf{A}_{N_i}\otimes\mathbf{I}_{N/N(i)}$$

Thus, expression (3) defines an algorithm for a discrete orthogonal transform synthesized via the nesting of small-size base kernels $\mathbf{A}_{n_i}$. It should be noted that in some cases it is appropriate to represent the kernels $\overline{\boldsymbol{A}}_{N_i}$ of discrete orthogonal transforms as the product of a non-orthonormal base matrix $\mathbf{A}_{N_i}$ and a diagonal matrix of normalizing constants $diag(\mathbf{S}_{N_i\times 1})$:

$$\overline{\mathbf{A}}_{N_i}=diag(\mathbf{S}_{N_i\times 1})\,\mathbf{A}_{N_i},\ i=1,\ldots,K, \tag{4}$$

where $\mathbf{A}_{N_i}$ is the non-orthonormal base matrix and $\mathbf{S}_{N_i\times 1}=\left[s_0^{(N_i)},\,s_1^{(N_i)},\,\ldots,\,s_{N_i-1}^{(N_i)}\right]^{\mathrm{T}}$ is a diagonal matrix of normalizing constants, the values of which depend on the selected base.

Substituting (4) into (1), we obtain the following expression

$$\overline{\boldsymbol{A}}_N^{(N_1,\,N_2,\,\ldots,\,N_K)}=diag(\mathbf{S}_{N_1\times 1})\,\mathbf{A}_{N_1}\otimes diag(\mathbf{S}_{N_2\times 1})\,\mathbf{A}_{N_2}\otimes\ldots\otimes diag(\mathbf{S}_{N_K\times 1})\,\mathbf{A}_{N_K}. \tag{5}$$

Then, using the properties of the Kronecker product [19-21], we can write:

$$\overline{\boldsymbol{A}}_N^{(N_1,\,N_2,\,\ldots,\,N_K)}=[diag(\mathbf{S}_{N_1\times 1})\otimes diag(\mathbf{S}_{N_2\times 1})\otimes\ldots\otimes diag(\mathbf{S}_{N_K\times 1})]\times[\mathbf{A}_{N_1}\otimes\mathbf{A}_{N_2}\otimes\ldots\otimes\mathbf{A}_{N_K}]. \tag{6}$$

This representation allows us to account for partial normalizations of small-dimensional transforms in the final stage of the algorithm implementation, which, in some cases, reduces its multiplicative complexity. Thus, we obtain a generalized computational procedure that takes these aspects into account:

$$\mathbf{Y}_{N\times 1}^{(N_1,\,N_2,\,\ldots,\,N_K)}=diag(\mathbf{S}_{N\times 1})\times\prod_{i=1}^{K}(\mathbf{I}_{N(i-1)}\otimes\mathbf{A}_{N_i}\otimes\mathbf{I}_{N/N(i)})\mathbf{X}_{N\times 1}, \tag{7}$$

where

$$diag(\mathbf{S}_{N\times 1})=[diag(\mathbf{S}_{N_1\times 1})\otimes diag(\mathbf{S}_{N_2\times 1})\otimes\ldots\otimes diag(\mathbf{S}_{N_K\times 1})]=diag(s_0^{(N)},\,s_1^{(N)},\,\ldots,\,s_{N-1}^{(N)}). \tag{8}$$

4. Some examples

Let us examine a few examples below to help you better understand the material. For clarity, let us assume that $N=12$.

1. Let $N=3\times 4$, Therefore $K=2$, $N_1=3$, $N_2=4$, and $N(1)=3$, $N(2)=12$.

Then

$$\dot{\mathbf{W}}_N^{(1)}=\mathbf{I}_{N(1-1)}\otimes\mathbf{A}_{N_1}\otimes\mathbf{I}_{N/N(1)}=\mathbf{I}_{N(0)}\otimes\mathbf{A}_3\otimes\mathbf{I}_{N/N(1)}=\mathbf{I}_1\otimes\mathbf{A}_3\otimes\mathbf{I}_{12/3}=\mathbf{A}_3\otimes\mathbf{I}_4$$

$$\dot{\mathbf{W}}_N^{(2)}=\mathbf{I}_{N(2-1)}\otimes\mathbf{A}_{N_2}\otimes\mathbf{I}_{N/N(2)}=\mathbf{I}_{N(1)}\otimes\mathbf{A}_4\otimes\mathbf{I}_{12/12}=\mathbf{I}_3\otimes\mathbf{A}_4\otimes\mathbf{I}_1=\mathbf{I}_3\otimes\mathbf{A}_4$$

Then, the computational procedure for this case is as follows:

$$\mathbf{Y}_{12\times 1}^{(3,4)}=\dot{\mathbf{W}}_{12}^{(2)}\dot{\mathbf{W}}_{12}^{(1)}\mathbf{X}_{N\times 1}, \tag{9}$$

Figure 1 illustrates a graph of the discrete orthogonal transform algorithm, as described by expression (9), utilizing 3rd- and 4th-order kernels.

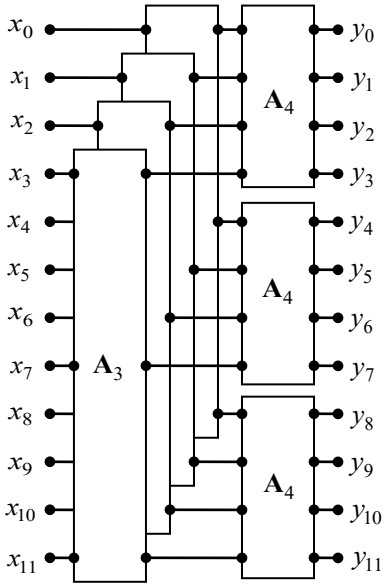

Fig. 1. Data flow graph representing the organization of calculations DOT for N=3×4.

2. Let $N = 4\times3$, Therefore $K=2$, $N_1=4$, $N_2=3$, and $N(1)=4$, $N(2)=12$.
Then

$$\ddot{\mathbf{W}}_N^{(1)}=\mathbf{I}_{N(1-1)}\otimes\mathbf{A}_{N_1}\otimes\mathbf{I}_{N/N(1)}=\mathbf{I}_{N(0)}\otimes\mathbf{A}_4\otimes\mathbf{I}_{N/N(1)}=\mathbf{I}_1\otimes\mathbf{A}_4\otimes\mathbf{I}_{12/4}=\mathbf{A}_4\otimes\mathbf{I}_3\,,$$

$$\ddot{\mathbf{W}}_N^{(2)}=\mathbf{I}_{N(2-1)}\otimes\mathbf{A}_{N_2}\otimes\mathbf{I}_{N/N(2)}=\mathbf{I}_{N(1)}\otimes\mathbf{A}_3\otimes\mathbf{I}_{12/12}=\mathbf{I}_4\otimes\mathbf{A}_3\otimes\mathbf{I}_1=\mathbf{I}_4\otimes\mathbf{A}_3\,,$$

The computational procedure for this case is as follows:

$$\mathbf{Y}_{12\times1}^{(4,3)}=\ddot{\mathbf{W}}_{12}^{(2)}\ddot{\mathbf{W}}_{12}^{(1)}\mathbf{X}_{N\times1}, \tag{10}$$

Figure 2 illustrates a graph of the discrete orthogonal transform algorithm, as described by expression (10), utilizing 3rd- and 4th-order kernels.

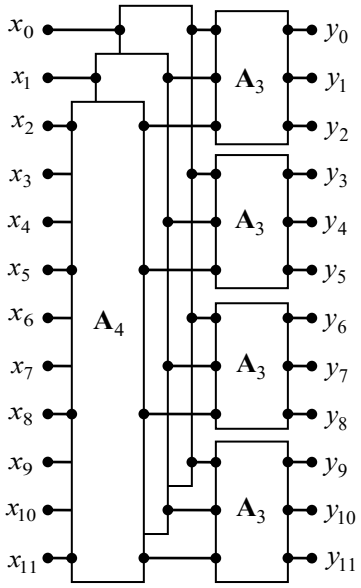

Fig. 2. Data flow graph representing the organization of calculations DOT for N=4×3.

3. Let N = 2×3×2. Therefore K=3, N_1=2, N_2=3, N_3=2 and $N(1)$=2 , $N(2)$=6, $N(3)$=12.
Then

$$\ddot{\mathbf{W}}_N^{(1)}=\mathbf{I}_{N(1-1)}\otimes\mathbf{A}_{N_1}\otimes\mathbf{I}_{N/N(1)}=\mathbf{I}_{N(0)}\otimes\mathbf{A}_2\otimes\mathbf{I}_{N/N(1)}=\mathbf{I}_1\otimes\mathbf{A}_2\otimes\mathbf{I}_{12/2}=\mathbf{A}_2\otimes\mathbf{I}_6\,,$$

$$\ddot{\mathbf{W}}_N^{(2)}=\mathbf{I}_{N(2-1)}\otimes\mathbf{A}_{N_2}\otimes\mathbf{I}_{N/N(2)}=\mathbf{I}_{N(1)}\otimes\mathbf{A}_3\otimes\mathbf{I}_{12/6}=\mathbf{I}_2\otimes\mathbf{A}_3\otimes\mathbf{I}_2\,,$$

$$\ddot{\mathbf{W}}_N^{(2)}=\mathbf{I}_{N(3-1)}\otimes\mathbf{A}_{N_3}\otimes\mathbf{I}_{N/N(3)}=\mathbf{I}_{N(2)}\otimes\mathbf{A}_2\otimes\mathbf{I}_{12/12}=\mathbf{I}_6\otimes\mathbf{A}_2\otimes\mathbf{I}_1=\mathbf{I}_6\otimes\mathbf{A}_2\,,$$

The computational procedure for this case is as follows:

$$\mathbf{Y}_{12\times1}^{(4,3)}=\ddot{\mathbf{W}}_{12}^{(3)}\,\ddot{\mathbf{W}}_{12}^{(2)}\ddot{\mathbf{W}}_{12}^{(1)}\mathbf{X}_{N\times1}, \tag{11}$$

Figure 3 shows a graph of the discrete orthogonal transform algorithm described by expression (11), based on 2rd- and 3th-order kernels.

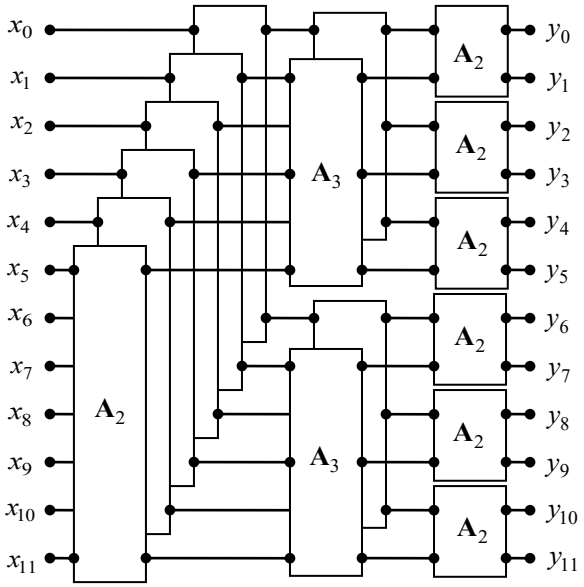

Fig. 3. Data flow graph representing the organization of calculations DOT for N=2×3×2.

4. Now let N=2×2×3. Therefore K=3, N_1=2, N_2=2, N_3=3 and $N(1)$=2, $N(2)$=4, $N(3)$=12. Besides $\bar{\mathbf{A}}_{N_1}=diag\,(\mathbf{S}_{N_1})\mathbf{A}_{N_1}$, $\bar{\mathbf{A}}_{N_2}=diag(\mathbf{S}_{N_2})\mathbf{A}_{N_2}$, $\bar{\mathbf{A}}_{N_3}=diag(\mathbf{S}_{N_3})\mathbf{A}_{N_3}$.
The computational procedure for this case is as follows:

$$\mathbf{Y}_{12\times1}^{(2,2,3)}=diag\,(\mathbf{S}_{N\times1})\bar{\mathbf{W}}_{12}^{(3)}\bar{\mathbf{W}}_{12}^{(2)}\bar{\mathbf{W}}_{12}^{(1)}\mathbf{X}_{N\times1}, \tag{12}$$

where

$$diag\,(\mathbf{S}_{N\times1})=diag\,(\mathbf{S}_{N_1})\otimes diag\,(\mathbf{S}_{N_2})\otimes diag\,(\mathbf{S}_{N_3})=\left[s_0^{(N)},\,s_1^{(N)},\,\ldots,\,s_{N-1}^{(N)}\right]^{\mathrm{T}}\text{, and}$$

$$\bar{\mathbf{W}}_{12}^{(1)}=\mathbf{I}_{N(1-1)}\otimes\mathbf{A}_{N_1}\otimes\mathbf{I}_{N/N(1)}=\mathbf{I}_{N(0)}\otimes\mathbf{A}_2\otimes\mathbf{I}_{N/N(1)}=\mathbf{I}_1\otimes\mathbf{A}_2\otimes\mathbf{I}_{12/2}=\mathbf{A}_2\otimes\mathbf{I}_6\,,$$

$$\bar{\mathbf{W}}_{12}^{(2)}=\mathbf{I}_{N(2-1)}\otimes\mathbf{A}_{N_2}\otimes\mathbf{I}_{N/N(2)}=\mathbf{I}_{N(1)}\otimes\mathbf{A}_2\otimes\mathbf{I}_{12/4}=\mathbf{I}_2\otimes\mathbf{A}_2\otimes\mathbf{I}_3\,,$$

$$\bar{\mathbf{W}}_{12}^{(3)}=\mathbf{I}_{N(3-1)}\otimes\mathbf{A}_{N_3}\otimes\mathbf{I}_{N/N(3)}=\mathbf{I}_{N(2)}\otimes\mathbf{A}_3\otimes\mathbf{I}_{12/12}=\mathbf{I}_4\otimes\mathbf{A}_3\otimes\mathbf{I}_1=\mathbf{I}_4\otimes\mathbf{A}_3\,,$$

Figure 4 shows a graph of the discrete orthogonal transform algorithm described by expression (12), based on 2rd, and 3th-order kernels.

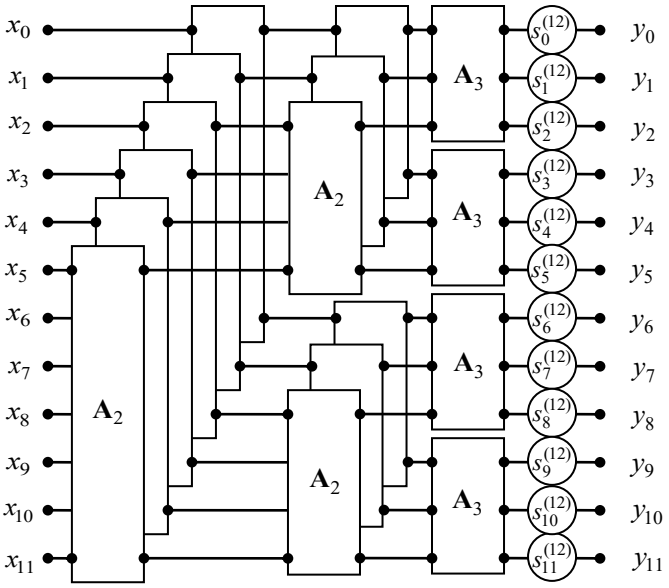

Fig. 4. Data flow graph representing the organization of calculations DOT for N=2×2×3.

5. What's next?

Now we can choose the concrete type of orthogonal transformation and "insert" the basis matrices of this transformation into the rectangles shown in the figures, which until now have represented "black boxes" of given sizes. Now these small-sized transforms will become the building blocks for the implementation of a new transformation. We will not go into the details of applying specific systems of basis functions here, but will simply outline the idea. This approach allows us to synthesize new discrete orthogonal transforms using small-size transforms (nested kernels) of traditional discrete orthogonal transforms as building blocks. Such kernels can include small-size kernels of discrete Hartley transforms; small-size kernels of trigonometric transforms of all 16 types, as well as their integer modifications; kernels of the slant transform; Haar transform; orthogonal polynomials of Legendre, Krawtchouk, Tchebichef, Racah, Hahn, etc. The list of kernel types that can serve as a basis for synthesizing new systems of basis functions and algorithms for their implementation can be extended. New transforms utilizing various kinds of DOT kernels can be tailored to the structure of specific data, and their efficient algorithms make them practically applicable. The new basis can be tuned to particular signal parameters, saving computing resources and memory and improving the quality of digital signal representation and processing.

And one more thing: a common drawback of transforms in traditional orthogonal bases is that the arithmetic complexity of the computational algorithm increases with the transform size. Another advantage of the proposed approach is that it enables the synthesis of algorithms for new large-scale discrete transforms that take into account and preserve the local properties of the embedded small-scale transforms (kernels), while simultaneously producing efficient algorithms for large-scale transforms. It is also worth noting that the algorithms implemented using the approach under consideration are iterative. If, at each iteration, we use sets of kernels representing not one but different types of discrete orthogonal transforms, we can synthesize "hybrid" discrete orthogonal transforms.

Conclusion

In general, the problem is clear: for a compact and efficient representation of signals that takes into account their structure, more and more orthogonal bases are needed, and fast algorithms make this practical and feasible in real time. The proposed approach makes this possible. It should be understood, however, that synthesizing a new orthogonal base is not enough. Its properties must be investigated and its validity demonstrated. We have already demonstrated its successful application in the development of fast algorithms for the pseudo-fractional Fourier and Hadamard transforms [22-23]. In conclusion, we would like to note that many remarkable, more or less universal methods for synthesizing new basis systems exist. However, the purpose of this article is not to provide an overview, a detailed analysis, or to identify the advantages, disadvantages, and limitations of these approaches. The purpose of this note is to clarify the idea underlying the approach described here, which allows for a relatively simple combination of the solution to the problem of synthesizing new basis systems and the construction of algorithms for fast data transformations within these systems. We plan to further develop this approach.

References

[1]. J.-B.-J. Fourier. Théorie analytique de la chaleur. Paris: Firmin-Didot, père et fils, 1822. Digital copy: doi: https://doi.org/10.3931/e-rara-19706.

[2]. M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the FFT," IEEE Acoustics, Speech, and Signal Processing Magazine, 1985, vol. 1, pp. 14-21, October 1984. also in the Archive for History of Exact Sciences.

[3]. R. E. Blahut, Fast Algorithms for Digital Signal Processing. Reading, MA: Addison-Wesley, Inc., 1984.

[4]. D. F. Elliott, K.R. Rao, Fast Transforms: Algorithms, Analyses, Applications; Academic Press, 1983.

[5]. N. Ahmed and K. R. Rao, Orthogonal Transforms for Digital Signal Processing. New York: Springer, 1975.

[6]. F. M. Wang and P. Yip, "Fast prime factor decomposition algorithms for a family of discrete trigonometric transforms," Circuits, Systems, and Signal Processing, 1989, vol. 8, no. 4, pp. 401-419.

[7]. K. R. Rao and P. Yip, Discrete Cosine Transform: Algorithms, Advantages, Applications. San Diego, CA: Academic Press, 1990.

[8]. X. Shao, S.G. Johnson, Type-II/III DCT/DST algorithms with reduced number of arithmetic operations. Signal Processing 2008, 88(6), 1553–1564.

[9]. R.K. Chivukula, Y.A. Reznik, Fast computing of discrete cosine and sine transforms of types VI and VII. In Proceedings of the SPIE 8135, Applications of Digital Image Processing XXXIV, 20–25 September 2011, San Diego, CA, USA.

[10]. W. Hamidouche, P. Philippe, M. Camar-Eddine, A. Kammoun, D. Menard, O. Deforges, Hardware-friendly DST-VII/DCT-VIII approximations for the versatile video coding standard. In Proceedings of the Picture Coding Symposium (PCS 2019), 12–15 November 2019, Ningbo, China.

[11]. H. V. Sorensen, D. L. Jones, C. S. Burrus, and M. T. Heideman, On computing the discrete Hartley transform, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 33, pp. 1231-1238, October 1985.

[12]. W. K. Pratt, Wen-Hsiung Chen, and Lloyd R. Welc, Slant transform image coding, IEEE Transactions on Communications, 1974, vol. com-22, no. 8, pp. 1075 – 1095.

[13]. B. M. Mahmmod, Abd R. bin Ramli, S. H. Abdulhussain, S. A. R. Al-Haddad, W. A. Jassim, Signal compression and enhancement using a new orthogonal-polynomial-based discrete transform, IET Signal Processing, 2018, vol. 12, no. 1, pp. 129-142.

[14]. A. Dziech, New orthogonal transforms for signal and image processing. Appl. Sci. 2021, 11, 7433.
[15]. V.M. Chernov, A. G. Dmitriyev. Image compression using discrete orthogonal transforms with the «noise-like» basis functions. Computer Optics, 1999, no.19. pp. 184-187.
[16]. V. M. Chernov, M. V. Pershina. Fibonacci-Mersenne and Fibonacci-Fermat discrete transforms. Baletin de Informatica. Мапуту, 1999, no. 9/10. pp. 25-31.
[17]. S. S. Agayan and D. Z. Gevorkyan. Synthesis of a class of orthogonal transforms, Pattern Recognition and Image Analysis, 1992, vol. 2, no. 4, pp. 394-408.
[18]. J. Granata, M. Conner, R. Tolimieri. The tensor product: a mathematical programming language for FFTs and other fast DSP operations. IEEE Signal Processing Magazine 1992, 9, 40–48, doi:10.1109/79.109206.
[19]. J. R. Johnson, R. W. Johnson, D. Rodriguez, and R. Tolimieri. A methodology for designing, modifying and implementing Fourier transform algorithms on various architectures. Circuits, Systems, and Signal Processing, 1990, vol. 9, pp. 449–500.
[20]. A. Graham, Kronecker Products and Matrix Calculus: With Applications. Ellis Horwood Ltd, 1981.
[21]. P. A. Regalia, S. K. Mitra, Kronecker Products, Unitary Matrices and Signal Processing Applications, SIAM Review, 1989, v. 31, no 4, pp. 586-613.
[22]. D. Majorkowska-Mech, and A. Cariow. 2021. Discrete pseudo-fractional Fourier transform and its fast algorithm. Electronics 10, no. 17: 2145, doi:10.3390/electronics10172145.
[23]. D. Majorkowska-Mech, A. Cariow, Discrete pseudo-fractional Hadamard transform and its fast algorithm", IEEE Signal Processing Letters, 2020, vol. 27, pp. 1195-1199, doi: 10.1109/LSP.2020.3005535.