PLOS ONE

RESEARCH ARTICLE

Performance enhancement of high order Hahn polynomials using multithreading

Basheera M. Mahmmod[1☯], Wameedh Nazar Flayyih[1☯], Zainab Hassan Fakhri[1☯], Sadiq H. Abdulhussain[1☯], Wasiq Khan[2‡*], Abir Hussain[2,3‡]

1 Department of Computer Engineering, University of Baghdad, Baghdad, Iraq, **2** School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool, United Kingdom, **3** Department of Electrical Engineering, University of Sharjah, Sharjah, United Arab Emirates

☯ These authors contributed equally to this work.
‡ WK and AH also contributed equally to this work.
* W.Khan@ljmu.ac.uk

Citation: Mahmmod BM, Flayyih WN, Fakhri ZH, Abdulhussain SH, Khan W, Hussain A (2023) Performance enhancement of high order Hahn polynomials using multithreading. PLoS ONE 18(10): e0286878. https://doi.org/10.1371/journal.pone.0286878

Editor: Viacheslav Kovtun, Vinnytsia National Technical University, UKRAINE

Received: August 20, 2022

Accepted: May 25, 2023

Published: October 25, 2023

Data Availability Statement: All relevant data are within the paper.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Abbreviations: The following abbreviations are used in this manuscript:CPU, Central Processing Unit; DHaM, Discrete Hahn Moment; DHaPs, Discrete Hahn Polynomials; DOMs, Discrete

Abstract

Orthogonal polynomials and their moments have significant role in image processing and computer vision field. One of the polynomials is discrete Hahn polynomials (DHaPs), which are used for compression, and feature extraction. However, when the moment order becomes high, they suffer from numerical instability. This paper proposes a fast approach for computing the high orders DHaPs. This work takes advantage of the multithread for the calculation of Hahn polynomials coefficients. To take advantage of the available processing capabilities, independent calculations are divided among threads. The research provides a distribution method to achieve a more balanced processing burden among the threads. The proposed methods are tested for various values of DHaPs parameters, sizes, and different values of threads. In comparison to the unthreaded situation, the results demonstrate an improvement in the processing time which increases as the polynomial size increases, reaching its maximum of 5.8 in the case of polynomial size and order of 8000 × 8000 (matrix size). Furthermore, the trend of continuously raising the number of threads to enhance performance is inconsistent and becomes invalid at some point when the performance improvement falls below the maximum. The number of threads that achieve the highest improvement differs according to the size, being in the range of 8 to 16 threads in 1000 × 1000 matrix size, whereas at 8000 × 8000 case it ranges from 32 to 160 threads.

Introduction

Moment theory is a powerful tool in the areas of image processing, pattern recognition, and computer vision applications [1]. Signals are described using scalar values called moments (one, two, or more dimensions). A set of polynomial basis functions is utilized to compute moments. These basis functions are used to convert signals, voice or images, to the transform domain [2, 3]. To deal with the problem of pattern identification, Hu [4] introduced geometric moments and moments invariants. The proposed momemnts are not orthogonal, which result in numerical difficulties [5].

Orthogonal Moments; DOPs, Discrete Orthogonal Polynomials; RRGSOP, Recurrence Relation Based on Gram-Schmidt orthonormalization process; RRnd, Recurrence Relation in the *n*-direction; RRxd, Recurrence Relation in the *x*-direction.

Continuous moments could be determined using continuous orthogonal polynomials like Zernike [6] or by using Tchebichef with altered radius [7]. Continuous moment functions can be incorrect due to two common types of errors: image coordinate transformation and continuous integral approximation. [8]. Due to the utilization of discretization and approximation throughout the process of image reconstruction, the obtained image will be imperfect [9].

In order to avoid the aforementioned constraints, Discrete Orthogonal Polynomials (DOPs) have been concentrated on by the researchers. This is due to their remarkable image reconstruction features [8–10]. In addition, 1D and 2D signals can be represented by discrete orthogonal moments (DOMs) without redundancy. Also, they have great energy compaction, and spectrum resolution characteristics [11–14]. Signal representation and feature extraction have recently been used to discrete Tchebichef polynomials [15, 16], discrete Hahn moments [17], and discrete Krawtchouk moments [18, 19]. It is note worthy that the DOPs are utilized for solving linear functional differential equations [20].

DOPs' are solid owing to their significant features, which involve localization, energy compression, watermarking, signal extraction features, numerical stability, efficient data processing, and resilient data analysis [3, 21–25]. At the same time, the vital characteristics of majority of DOMs are not applied to large-sized images which is due to limitation in the computation of polynomials [26].

The DOPs limitations such as overflow, the instability of the polynomial values, and the high computational complexity have resulted in this constraints. Therefore, an improved recurrence technique for generating higher orders are being improved, for example Tchebichef [16] and Krawtchouk [18] polynomials. Recently, Researchers have considered other DOPS, for example Charlier polynomials [27] and Hahn polynomials [26].

The computation of DOP coefficients and the propagation of errors have been simplified by using the recursive algorithms [28, 29]. Regarding degree n either a single or double recursive formula can be employed. It also considers the time or spatial coordinate. To overcome the instability in the numerical values, the DOP coefficients should be calculated in the direction of the variable n. However, when the size of one or two dimensional signals turns large these calculations become inefficient. Since small values are assumed for the squared norm of the scaled Tchebichef polynomials, the coefficients of Tchebichef polynomials, for example, suffer from instabilities in the numerical values. To solve the aforementioned problem, the recurrence method in the x-direction was introduced by Mukundan [9]. After this study, this issue has received significant attention in many studies, for example, [28].

Generally, there has been a lot of focus on computation cost [30, 31]. It is considered as a key element which assist in ill-conditioning. For this reason, large number have considered it [32, 33]. This drawback is addressed in [34] by using a rapid and efficient computation method for Meixner moment coefficients. Another research introduced a fast and stable approach of Tchebichef moments for higher polynomial order, this is performed by combining the recurrence algorithms in the n and x directions [16]. Daoui et al. [26] utilized Gram-Schmidt orthogonalization procedure to reduce numerical error propagation. However, this technique is relatively slow.

This paper proposes a novel technique inspired by discrete orthogonal Hahn moments. Based on the literature, the three-term recurrence algorithms have been utilized in several existing works to tackle the problem of computational cost and propagation error due to gamma and binomial functions [35]. In [28], the n-direction recurrence algorithm was employed with an initial value starting at n, x = 0. The drawback of the algorithm presented in [28] (the recurrence algorithm in the n-direction) comes from starting the sets of initial values which are based on the initial value at n, $x = 0$. The initial sets are computable for a very restricted DHaP polynomials size and parameters. This results in a maximum

computable polynomial size of 135. In other words, the limitation is due to the employed formula. In addition to the issue of high computational cost, algorithm used has numerical instability. To resolve the issue of the recurrence algorithm in the n-direction, the x-direction recurrence relation is adopted with a symmetry relation for equal values of the polynomials parameters [28]. Using the symmetry relation allows a reduction of 50% in the computed coefficients, which reduces the computation cost. Yet the recurrence relation in the x-directions has two limitations. The first limitation is that, according to the nature of the formula being utilized, the initial set becomes 0 when samples size or parameter values tends to be large. The second limitation is that when the degree of the polynomial increases, the coefficient values underflow because the initial value are less than 10^{-324}, which equals zero for various values of the polynomials' parameters. The highest possible order that can be calculated occurs at $n = 1423$. To overcome the limitation of previous recurrence algorithms, Daoui et al. [26] presented a technique based on the n-direction recurrence relation and the Gram-Schmidt orthonormalization process (GSOP). The utilization of the GSOP minimizes the numerical errors due to the use of the n-direction recurrence algorithm. However, the GSOP-based recurrence resolves the orthogonality of the DHaPs, but it has several limitations. First, the algorithm is unable to accurately calculate the coefficients of the DHaP when the parameters are not equal. Second, due to the technique used to calculate the initial values, the algorithm is still unable to generate DHaP for a wide range of DHaP parameters. Third, the nested loops of the GSOP algorithm result in a high computational cost, which in turn raises the number of processes required to compute the coefficients of the DHaP. Recently, a new mathematical model has been presented by [36], which can compute the DHaP's initial value for a wide range of DHaP parameters values. In order to stabilize the computation of the DHaP coefficients, the algorithm also consists of two recurrence algorithms with adaptive thresholds. Although the algorithm in [36] can compute the coefficients of the polynomials more accurately than other algorithms, it still suffers from computation overhead.

In this paper, a fast approach for computing the DHaPs is proposed and applied to high orders. This work takes advantage of the multithread for the computation of Hahn polynomials coefficients. To take advantage of the available processing capabilities, independent calculations are divided among threads. The research provides a distribution method to achieve a more balanced processing burden among the threads.

This paper is organised as follows: in Section "Mathematical definition of DHaP and its moments" Preliminaries and current three-term recurrence algorithms are discussed. The proposed recurrence algorithm is presented in Section "Proposed Recurrence Algorithm". In Section "Experimental Results", the proposed recurrence method is evaluated by an experimental investigation. Finally, the paper is concluded in Section "Conclusion".

Mathematical definition of DHaP and its moments

This section presents the mathematical principles of the DHaP and their moments.

The definition of DHaPs

The nth order of the DHaP is defined as [28]:

$$H_n^{\alpha,\beta}(x;N) = \frac{(-1)^n(\beta+1)_n(N-n)_n}{n!}\,{}_3F_2\left(\begin{matrix}-n,-x,n+1+\alpha+\beta\\ \beta+1,1-N\end{matrix}\middle|\,1\right), \tag{1}$$

where ${}_3F_2(\cdot)$ is the generalised hypergeometric series denoted by:

$$ {}_3F_2\left(\begin{matrix} a_1, a_2, a_3 \\ b_1, b_2 \end{matrix}\middle| c\right) = \sum_{k=0}^{\infty} \frac{(a_1)_k (a_2)_k (a_3)_k}{(b_1)_k (b_2)_k k!} (c)^k \tag{2} $$

and $(\cdot)_k$ is the Pochhammer symbol [37].

The orthogonality of the DHaPs is satisfied as follows:

$$ \sum_{x=0}^{N-1} H_n^{\alpha,\beta}(x;N) H_m^{\alpha,\beta}(x;N) \omega(x) = \rho(n)\delta_{nm} \ , \tag{3} $$

where ρ demotes the norm function of DHaP, ω represents the weight function of the DHaP, and δ_{nm} is the Kronecker delta. The norm and weight functions of the DHaP are defined as follows:

$$ \omega(x) = \frac{\Gamma(\beta + x + 1)\Gamma(N - x + \alpha)}{\Gamma(x+1)\Gamma(N-x)} \tag{4} $$

$$ \rho(x) = \frac{(\alpha + \beta + n + 1)_N \Gamma(\alpha + n + 1)\Gamma(\beta + n + 1)}{(2n + \alpha + \beta + 1)\Gamma(N-n)\Gamma(n+1)}. \tag{5} $$

The nth degree of the weighted and normalized DHaP is given by

$$ \hat{H}_n^{\alpha,\beta}(x;N) = H_n^{\alpha,\beta}(x;N)\sqrt{\frac{\omega}{\rho}}. \tag{6} $$

The definition of DHaM

DHaMs are the projection of the signals on the basis of the DHaP. Suppose a 2D signal $f(x, y)$ of size $N_1 \times N_2$. Then, the DHaMs, Ψ_{nm}, can be calculated:

$$ \Psi_{nm} = \sum_{x=0}^{N_1-1} \sum_{y=0}^{N_2-1} \hat{H}_n^{\alpha,\beta}(x;N_1) \hat{H}_m^{\alpha,\beta}(y;N_2) f(x,y) \tag{7} $$

$$ \begin{aligned} &n = 0, 1, \ldots, N_1 - 1, \quad \text{and} \\ &m = 0, 1, \ldots, N_2 - 1, \end{aligned} \tag{8} $$

To reconstruct back the 2D signal, image, to the spatial domain, the reconstructed signal $\hat{f}(x, y)$ can be computed as follows:

$$ \begin{aligned} &\hat{f}(x,y) = \sum_{n=0}^{N_1-1} \sum_{m=0}^{N_2-1} \hat{H}_n^{\alpha,\beta}(x;N_1) \hat{H}_n^{\alpha,\beta}(y;N_2) \Psi_{nm} \\ &x = 0, 1, \ldots, N_1 - 1; \quad \text{and} \quad y = 0, 1, \ldots, N_2 - 1. \end{aligned} \tag{9} $$

Related work

It is well known that hypergeometric series defined in Eq (1) is computationally cost and shows imprecise precision of the polynomials coefficients; thus, the three term recurrence relations are used. The available recurrence relations with their analysis are discussed in this section.

The recurrence relation in the *n*-direction (RRnd). The nth degree of the DHaP at the xth index is defined as follows [28]

$$\hat{H}_n^{\alpha,\beta}(x;N) = \frac{AB}{E}\,\hat{H}_{n-1}^{\alpha,\beta}(x;N) + \frac{CD}{E}\,\hat{H}_{n-2}^{\alpha,\beta}(x;N) \tag{10}$$

$$\begin{aligned} n &= 2,3,\ldots,N-1, \quad \text{and} \\ x &= 0,1,\ldots,N-1, \end{aligned} \tag{11}$$

the recurrence relation parameters are defined by:

$$\begin{aligned} A &= x - \frac{2N+\gamma_2-2}{4} - \frac{(-\alpha^2+\beta^2)(2N+\gamma_1)}{4(2n+\gamma_1-2)(2n+\gamma_1)} \\ B &= \sqrt{\frac{n(n+\gamma_1)(2n+\gamma_1+1)}{(N-n)(n+\alpha)(n+\beta)(2n+\gamma_1-1)(N+n+\gamma_1)}} \\ C &= -\frac{(n+\alpha-1)(n+\beta-1)(N+n+\gamma_1-1)(N-n+1)}{(2n+\gamma_1-2)(2n+\gamma_1-1)} \\ D &= \sqrt{\frac{n(n-1)(n+\gamma_1)(n+\gamma_1-1)(2n+\gamma_1+1)}{(\alpha+n)(\alpha+n-1)(\beta+n)(\beta+n-1)(N-n+1)(N-n)}}\times \\ &\quad \sqrt{\frac{1}{(\gamma_1+2n-3)(\gamma_1+N+n)(\gamma_1+N+n-1)}} \\ E &= \frac{n(\gamma_1+n)}{(\gamma_1+2n-1)(\gamma_1+2n)} \\ \gamma_1 &= \alpha+\beta \\ \gamma_2 &= \alpha-\beta \end{aligned} \tag{12}$$

with initial values

$$\hat{H}_0^{\alpha,\beta}(x;N) = \sqrt{\frac{\omega(x)}{\rho(0)}} \tag{13}$$

$$\hat{H}_1^{\alpha,\beta}(x;N) = [-(\beta+1)(N-1) + x(\alpha+\beta+2)]\sqrt{\frac{\omega(x)}{\rho(1)}}. \tag{14}$$

The problem of the recurrence relation in the n-direction recurrence algorithm is due to the utilized initial values $\hat{H}_0^{\alpha,\beta}(x;N)$ and $\hat{H}_1^{\alpha,\beta}(x;N)$. These initial values bound the polynomial to low values of polynomial size N, where the largest size can be obtained is 135 samples which occurs at limited range of DHaP parameters, $\alpha = 20$ and $\beta = 20$. This limitation occurs because of the utilized formulas. To resolve this problem, the complexity of the utilized formulas can be reduced; however, the recurrence relation in the n-direction still shows numerical propagation error [36].

The recurrence relation in the *x*-direction (RRxd). To compute the DHaPs at the *x*th index with *n*th degree, the following recurrence is used [28]:

$$\hat{H}_n^{\alpha,\beta}(x;N) = \eta_1\left[\eta_2\hat{H}_n^{\alpha,\beta}(x-1;N) + \eta_3\hat{H}_n^{\alpha,\beta}(x-2;N)\right] \tag{15}$$

$$\begin{aligned} &x = 2,3,\ldots,N-1, \quad \text{and} \\ &n = 0,1,\ldots,N-1, \end{aligned} \tag{16}$$

The recurrence relation coefficients η_1, η_2, and η_3 are computed as follows:

$$\begin{aligned} \eta_1 &= \frac{\sqrt{\omega(x)}}{\tau(x-1)+\sigma(x-1)} & \sigma(x) &= x(\alpha+N-x) \\ \eta_2 &= \frac{\tau(x-1)+2\sigma(x-1)-\lambda(n)}{\sqrt{\omega(x-1)}} & \tau(x) &= (\beta+1)(N-1)-x(\alpha+\beta+2) \\ \eta_3 &= -\frac{\sigma(x-1)}{\sqrt{\omega(x-2)}} & \lambda(n) &= n(n+\alpha+\beta+1) \end{aligned} \tag{17}$$

with initials

$$\hat{H}_n^{\alpha,\beta}(0;N) = (1-N)_n\binom{n+\beta}{n}\sqrt{\frac{\omega(0)}{\rho(n)}} \tag{18}$$

$$\begin{aligned} \hat{H}_n^{\alpha,\beta}(1;N) \quad &= \frac{(n+\beta+1)(N-n-1)-n(N+\alpha-1)}{(\beta+1)(N-1)}\times \\ &\times\sqrt{\frac{\omega(1)}{\omega(0)}}\hat{H}_n^{\alpha,\beta}(0;N). \end{aligned} \tag{19}$$

It is noteworthy that for the recurrence relation in the *x*-direction, symmetry relation [26] is employed to recude the computation cost:

$$\hat{H}_n^{\alpha,\beta}(x;N) = (-1)^n\hat{H}_n^{\alpha,\beta}(N-1-x;N) \text{ for } \alpha=\beta. \tag{20}$$

The utilization of the symmetry relation Eq (20) will reduce the computed coefficients to 50%. However, the recurrence relation in the *x*-direction still has two limitations. These limitations are:

1. The values of the $\hat{H}_n^{\alpha,\beta}(0;N)$ tend to zero as the number of samples (N) increases and the values of the DHaP parameters becomes big. This is due to the formula used in Eq (18), and
2. The coefficient values of the DHaPs become underflowed as the polynomial degree (n) becomes large. This is due to the values of the initial becomes less than 10^{-324}, which in turn goes to zero in different environments [36].

It is noteworthy that the DHaPCs become zero as the polynomial degree increases. For instance, the maximum non-zero coefficients occurred at $n = 1423$ when $N = 1600$ and $\alpha = \beta = 10$ [36].

Recurrence relation based on Gram-Schmidt orthonormalization process (RRGSOP). Daoui et al. [26] introduced their algorithm which is based on Gram-Schmidt orthonormalization process (GSOP), as well as n-direction recurrence relation to determine DHaP. In this case, GSOP is introduced to solve the issue instability in the DHaPCs. The proposed computes

the initial sets $\hat{H}_0^{\alpha,\beta}(x;N)$ and $\hat{H}_1^{\alpha,\beta}(x;N)$. Recurrence relation in the n-direction is utilised to find the coefficients for $n > 1$ is then applied in order to reduce computational errors produced through the n-direction recurrence algorithm. Despite the fact that GSOP-based recurrence algorithm fulfil orthogonality condition, it has the following problems:

1. When $\alpha \neq \beta$, the algorithm fails to correctly identify the DHaP coefficients [26].
2. This algorithm is incapable to provide DHaP for a large range of values for α and β.
3. GSOP-based recurrence algorithm suffers from computational complexity as a result of the nested loops utilized to reduce the error for each polynomial degree. This can increase the number of operations required to calculate DHaP coefficients.

Hybrid recurrence algorithm. The authors in [36] provided a new algorithm for the purpose of solving the issues and limitations from previous algorithm. In this case the authors looked at two recurrence algorithms (which are the n- and x-recurrence relations) as well as adaptive threshold in order to be able to stabilize the generation of the DHaP coefficients. It should be noted that computational cost is considered important factor which is increased in this algorithm.

Proposed recurrence algorithm

In this section, the proposed multi-thread recurrence algorithm for DHaPs is presented in details.

The matrix of the DHaPs is divided into four parts (Part H1, Part H2, Part H3, and Part H4) similar to the partitions made in [36] (see Fig 1). First, computation of the first two columns at $x = 0$ and 1 are performed before the computation of the coefficients at part H1 (see Fig 2). Then, the proposed algorithm distributes the rows among a set of threads rather than the sequential processing of the H1 rows, which permits parallel processing of the coefficients. As every row is computed independently of the other rows. Part H2 is similarly divided among the same number of threads as part H1. Following the calculation of the last two columns, the threads start calculating the coefficient.

Furthermore, parts 3 and 4 (H3 and H4), begin respectively after parts H1 and H2 have been completed. The coefficients of the last two rows in part H1 are required to calculate part H3, however each column is independent of the others. Consequently, to perform parallel computation of the coefficients, the columns in part H3 are distributed among a set of threads. In the same way, after calculating part H2 coefficient, part H4 columns are processed using a set of threads.

For the case where $\alpha = \beta$, there is a mirroring property that can be used to reduce the processing load. In the proposed algorithm, parts H2 and H4 coefficients are calculated inline with the part H1 and H3 calculations respectively. Using the available parallel resources, which are represented by multicores. The number of threads assigned to each scenario guarantees that parts H1 and H2 have the same processing burden. In parts H1 and H2 for $2N/5$ rows, the number of rows for each thread will be $bunch1 = 2N/5/Th$; Th is the number of threads. The number of columns assigned for each thread in parts H3 and H4 will be $bunch2 = (N/2)/Th$.

Experimental results

In order to evaluate the proposed threaded algorithm different matrix sizes are considered, namely 1000 × 1000, 2000 × 2000, 4000 × 4000, and 8000 × 8000. The unthreaded case in [36] is compared to some works in literature TRx [28], TRn [28], and TRGSOP [26] for the cases

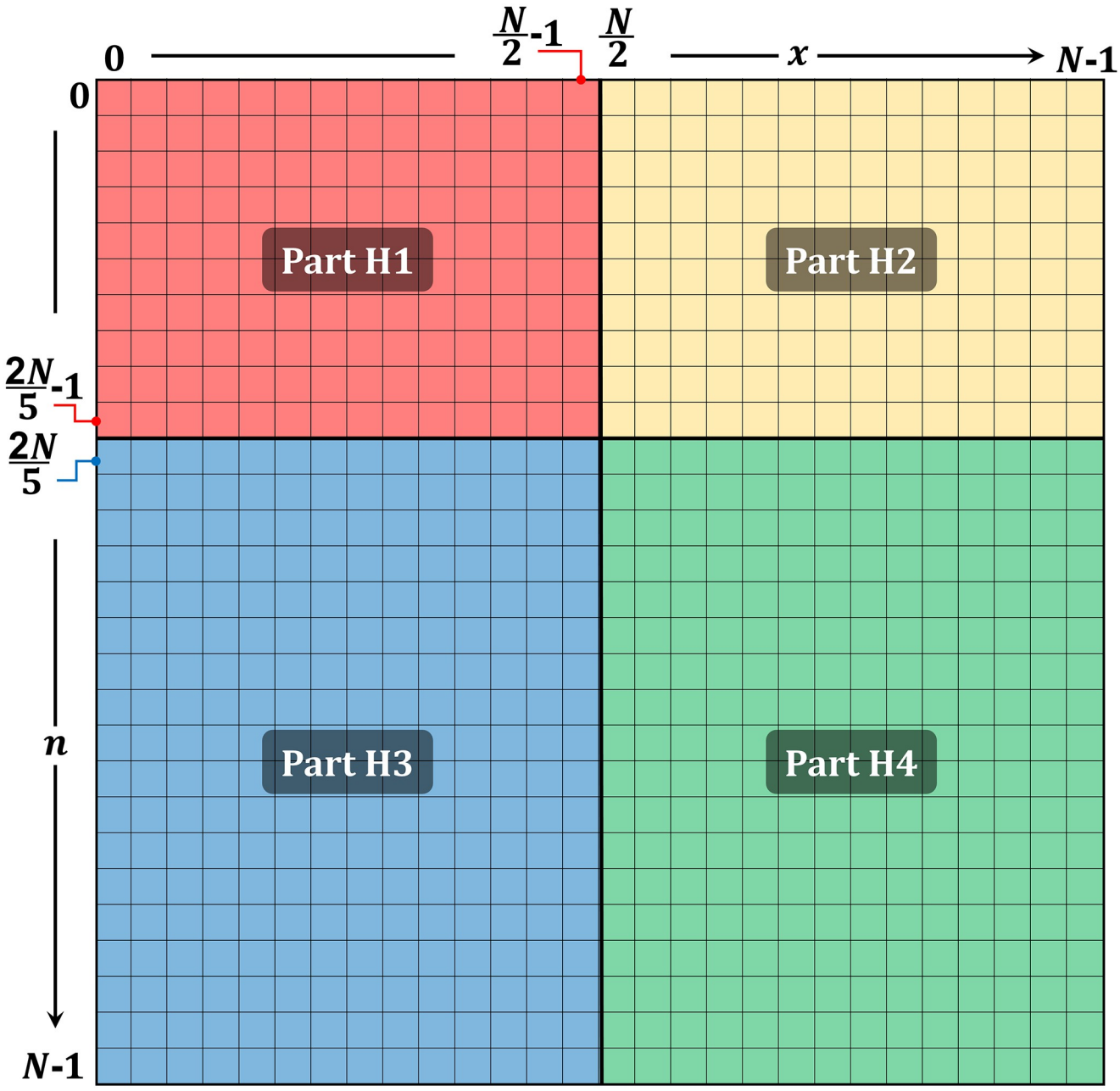

Fig 1. Partitions of the DHaPs.

https://doi.org/10.1371/journal.pone.0286878.g001

where $\alpha = \beta$ and $a \neq b$ to evaluate their performance. Tables 1 and 2 show the normalized performance of the different algorithms as compared to the unthreaded case [36]. For the case of $\alpha = \beta$, the unthreaded case outperforms the other algorithms in all cases, except the TRn at 1000 matrix size. On the other hand, TRx outperforms the unthreaded case when $\alpha \neq \beta$ for all matrix sizes, whereas TRn falls behind the unthreaded case at Large matrix sizes (4000 and 8000). TRGSOP show the worst performance at both $\alpha = \beta$ and $\alpha \neq \beta$ in all matrix sizes. It should be noted that TRn, TRx, and TRGSOP are unstable in all the tested cases while the unthreaded algorithm is stable.

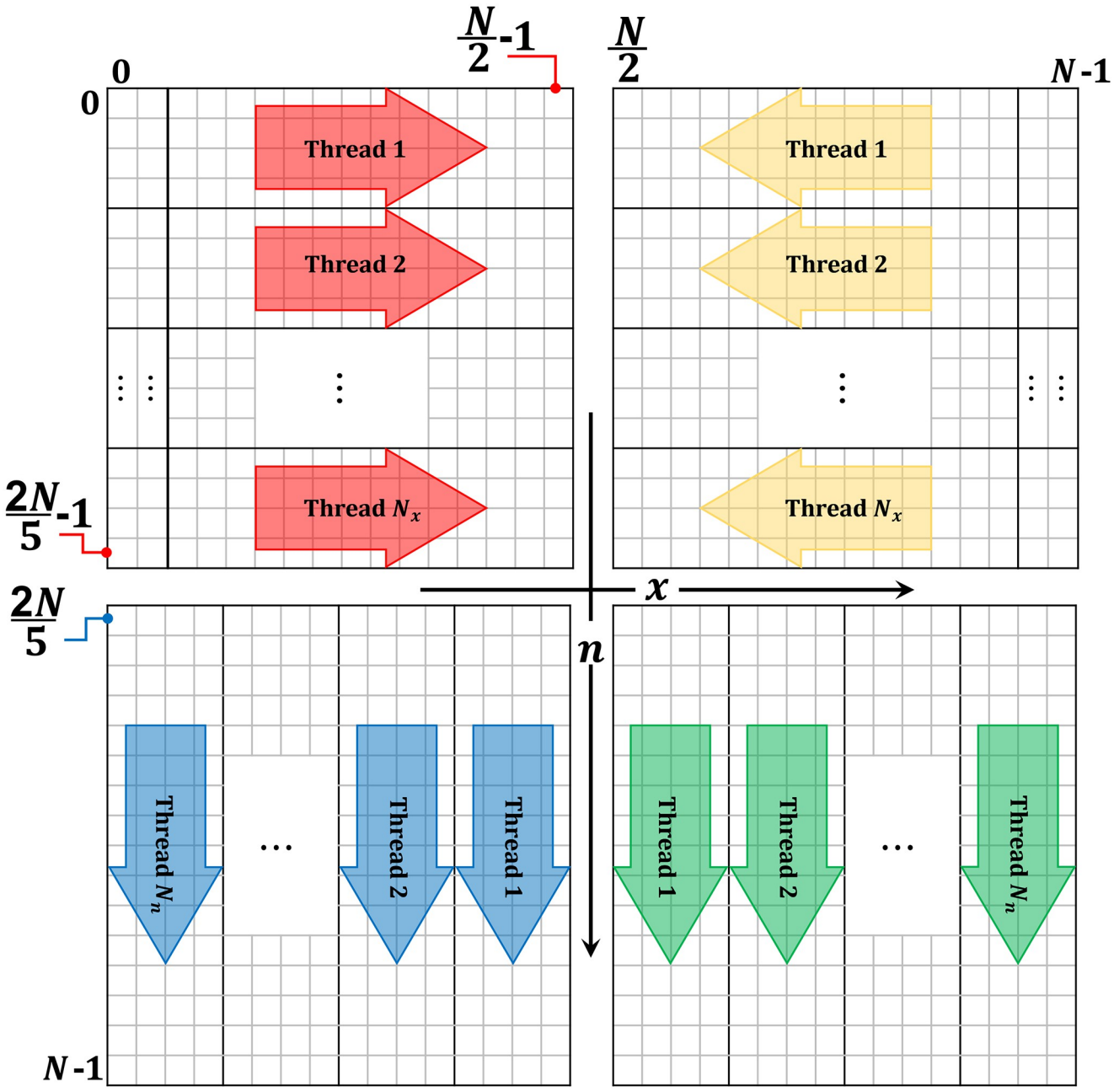

Fig 2. Steps of the proposed algorithm with threads.

https://doi.org/10.1371/journal.pone.0286878.g002

Table 1. Normalized performance of the different algorithms for $\alpha = \beta$.

Size	TRn	TRx	TRGSOP	Unthreaded Case
1000	1.152	0.824	0.001540	1
2000	0.977	0.845	0.000444	1
4000	0.726	0.830	0.000281	1
8000	0.539	0.982	0.000159	1

https://doi.org/10.1371/journal.pone.0286878.t001

Table 2. Normalized performance of the different algorithms for $\alpha \neq \beta$.

Size	TRn	TRx	TRGSOP	Unthreaded Case
1000	1.187	1.146	0.002090	1
2000	1.202	1.217	0.000570	1
4000	0.997	1.122	0.000360	1
8000	0.586	1.074	0.000170	1

https://doi.org/10.1371/journal.pone.0286878.t002

The proposed threaded algorithm is evaluated and compared with the unthreaded case at the same cases of matrix sizes considered previously with different combinations of parameters α and β. In addition, different combinations of parameters α and β are considered as well. The proposed algorithm is based on the distribution of the computation load among independent threads to provide parallelism, thus enhancing performance. The effect of number of threads on the performance is evaluated by considering 2 threads up to 250 threads.

Starting with the cases where $\alpha = \beta = 10$, 50, 100, 150, 200, and 250, Fig 3 shows the performance improvement achieved by the proposed algorithm with respect to the unthreaded case. It can be noticed that the improvement has been slightly affected by the changes in the parameter values $\alpha = \beta$, this is applicable at different number of threads and different matrix sizes. The highest difference can be observed at size = 1000 for $\alpha = \beta = 10$ where it achieves up to 23% lower improvement as compared to the other parameter cases.

As can be noticed in Fig 3, all cases achieve about 100% improvement when the process is divided among two threads. This improvement linearly increases with the increase in the number of threads up to 8 threads. The threading technique has higher improvement as compared to the unthreaded case at higher matrix size, as it achieves up to 4.7, 5.5, 5.6, and 5.8 improvement at 1000, 2000, 4000, and 8000 matrix sizes, respectively. At the size of 1000, the improvement reaches its maximum value at number of threads ranging from 8 to 16. As the number of threads increases over the aforementioned range the improvement gradually declines and the performance worsens as compared to the unthreaded case when the number of threads exceeds 128. This can be linked to the exceeding overhead imposed by the high number of threads as compared to the useful processing carried out by each thread. This drop in improvement can also be noticed in the case of size = 2000, but at a slower slope and at worst case the improvement does not fall below 1.4. For the size of 4000 and 8000, as the number of threads is increased above 8 the improvement is maintained above 4.5 and 5 respectively.

Higher CPU utilization may imply an improved performance as more processing power is allocated to the algorithm which leads to reduced execution time. To some extent this relation is valid as can be seen in Fig 4. In the unthreaded case, the CPU utilization is limited to 13% which results in high delay. This CPU utilization increases as the number of threads increases and results in lower delays. This relation does not persist for matrix sizes 1000 and 2000, where the delay increases again when the number of threads exceed 16 and 32, respectively. This results in increased processing power with no performance gain which should be avoided.

For the case where $\alpha \neq \beta$, the following values were considered (α/β) 100/50, 200/100, 400/200, 400/300, 500/250 and 500/400. Fig 5 shows the performance improvement achieved by the proposed algorithm with respect to the unthreaded case.

The threading technique achieves improvement with respect to the unthreaded case, and this improvement increases as the matrix size increases. It achieves up to 3.1 times improvement as compared to the unthreaded case at the size of 8000. At the size of 1000, the improvement reaches its maximum value at number of threads ranging from 8 to 16 which is similar to the cases where $\alpha = \beta$. As the number of threads increases over the aforementioned range the

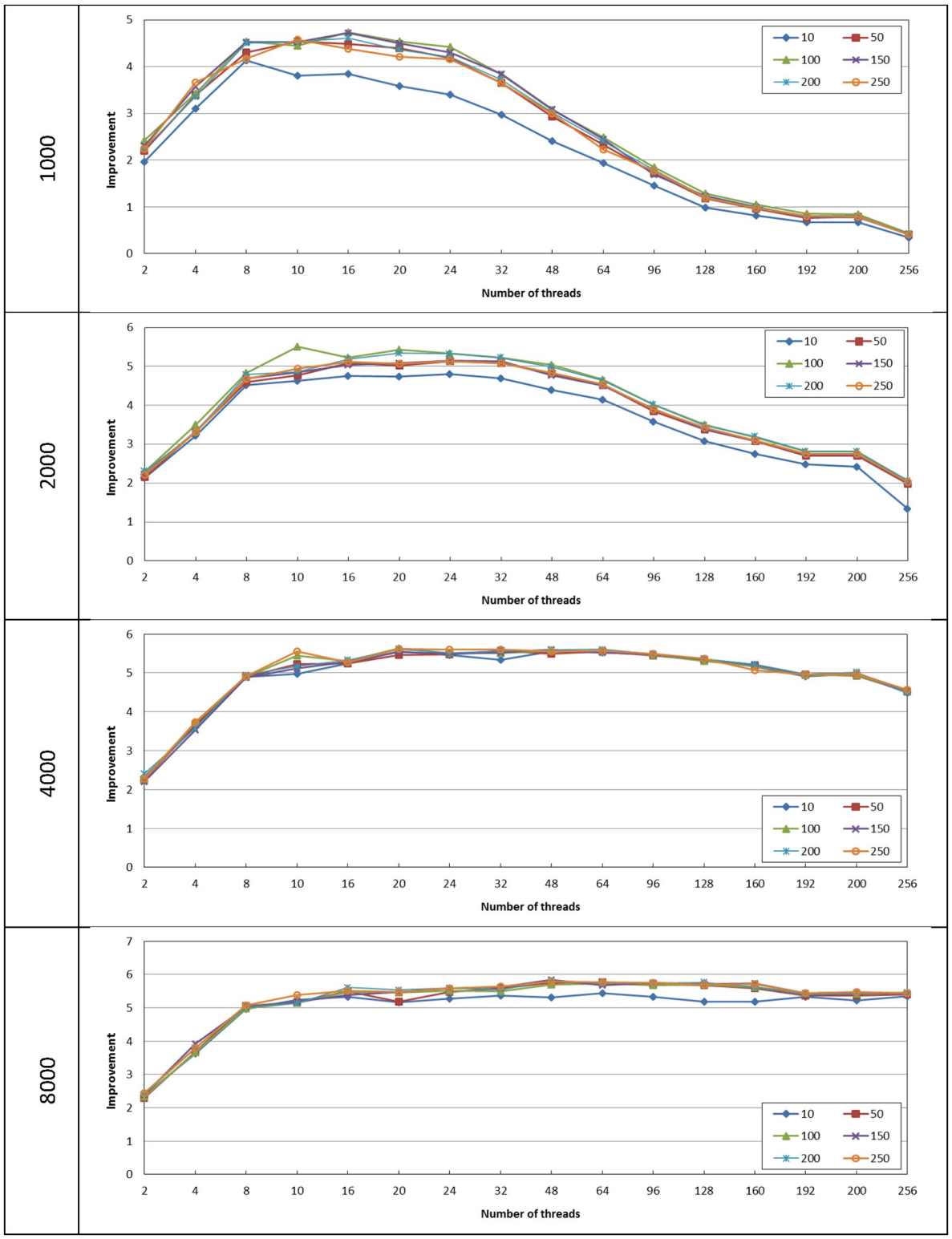

Fig 3. The improvement achieved by the proposed algorithm with respect to the unthreaded case for different size values with $\alpha = \beta$.

https://doi.org/10.1371/journal.pone.0286878.g003

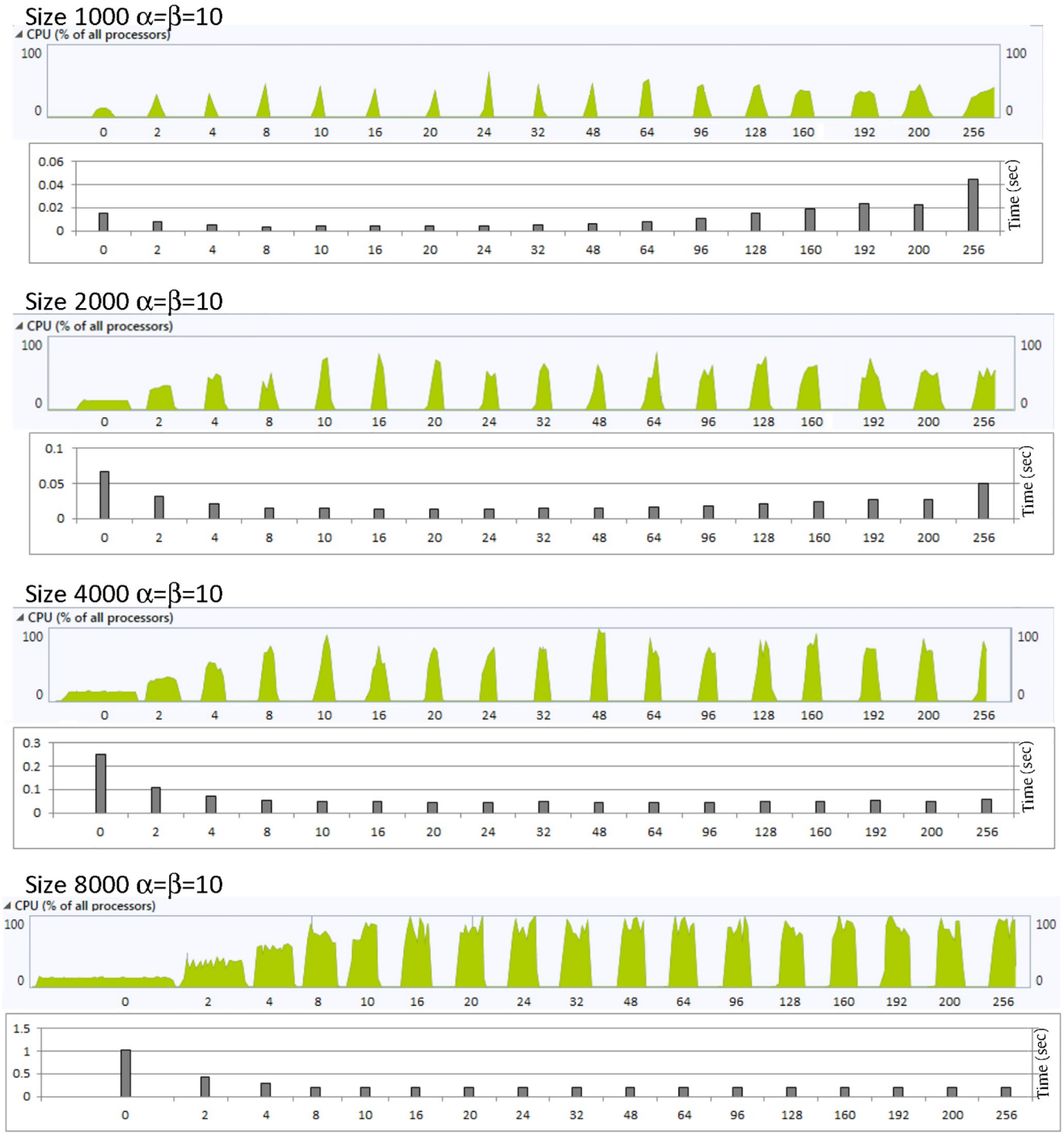

Fig 4. CPU utilization and the execution time for different values of polynomial size with $\alpha = \beta$. Note that 0 in the x axis means unthreaded case.

https://doi.org/10.1371/journal.pone.0286878.g004

improvement gradually declines and the performance worsens as compared to the unthreaded case when the number of threads exceeds 96. It reaches up to 5 times higher delay as compared to the unthreaded case, while the CPU usage is higher as shown in Fig 6. Any processing overhead with no performance gain should be avoided as it results in wasted power consumption.

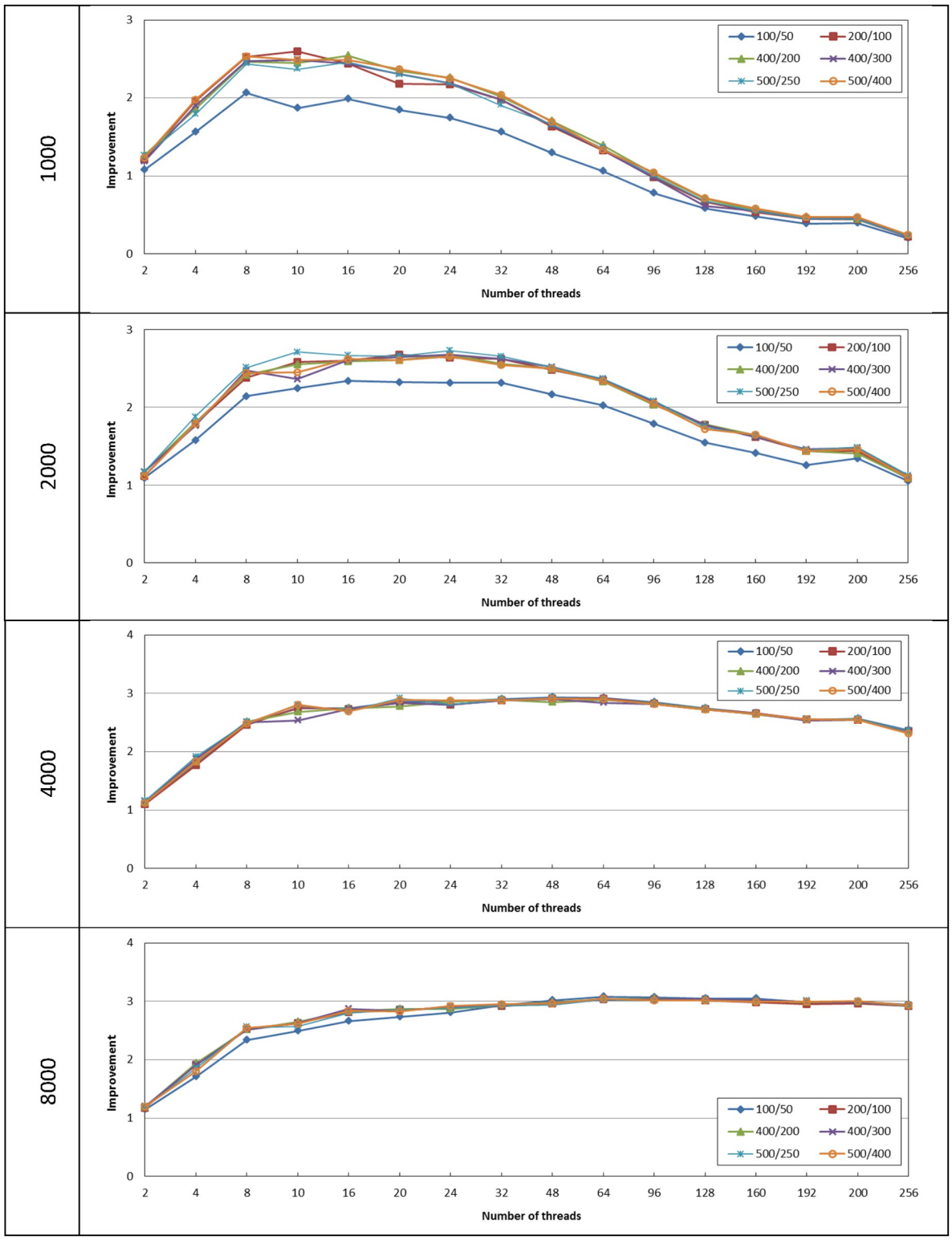

Fig 5. The improvement achieved by the proposed algorithm with respect to the unthreaded case for different size values with $\alpha \neq \beta$.

https://doi.org/10.1371/journal.pone.0286878.g005

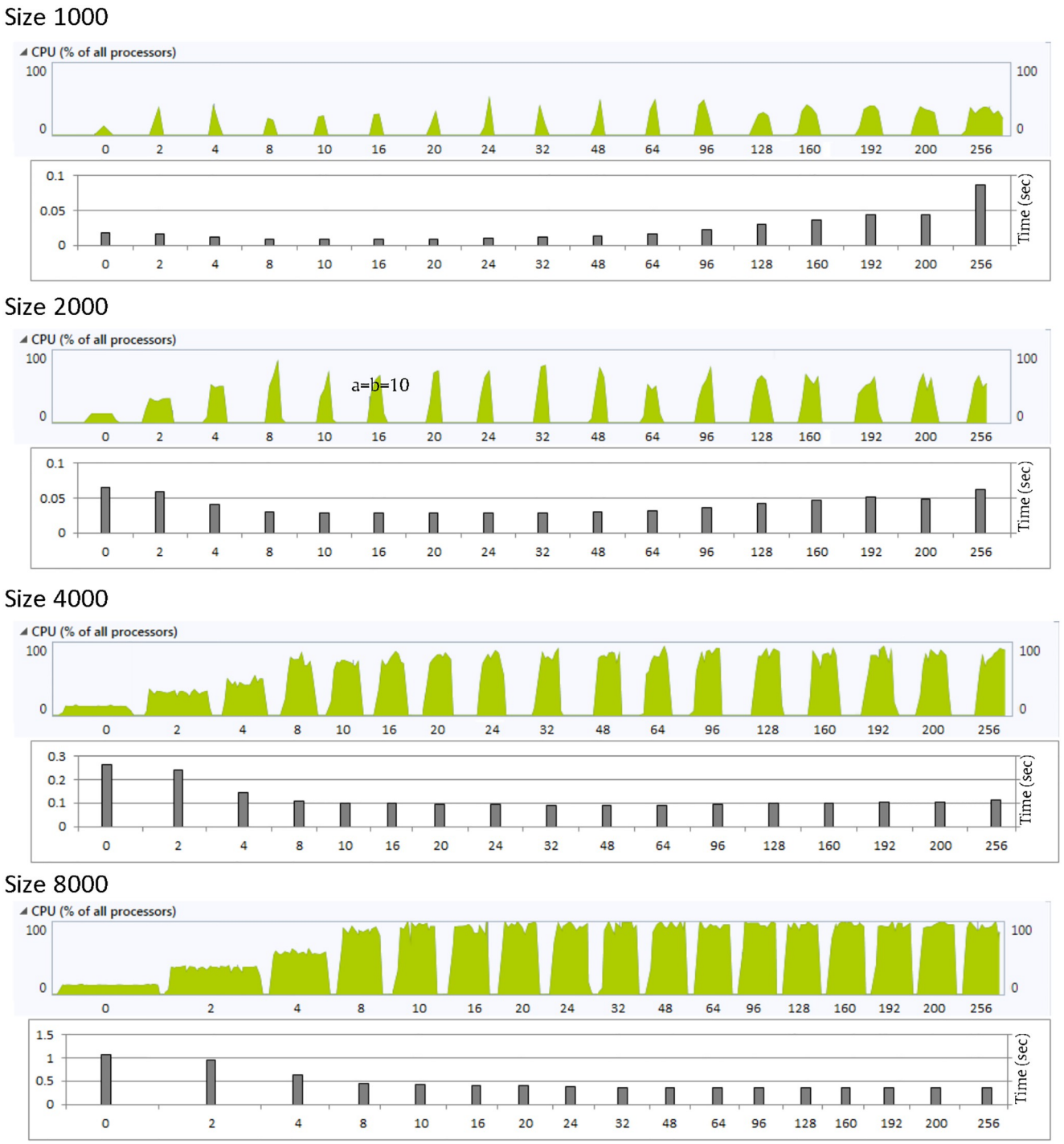

Fig 6. CPU utilization and the execution time for different values of polynomial size with $\alpha \neq \beta$. Note that 0 in the x axis means unthreaded case.

https://doi.org/10.1371/journal.pone.0286878.g006

It can be noticed that threading has higher improvement for the cases where $\alpha = \beta$ as compared to those cases where $\alpha \neq \beta$. In the former case, the coefficients in P2 and P4 are a mirror of those in P1 and P3 respectively. Thus, as soon as a coefficient in P1 and P3 is calculated, its mirror is immediately written and to avoid accessing the data again, mitigating memory reads penalties.

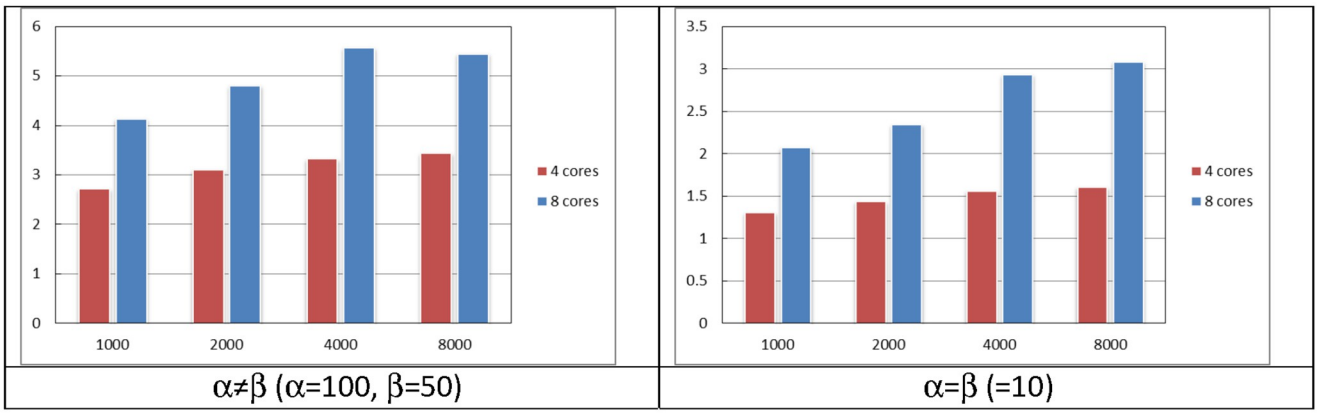

Fig 7. The effect of number of cores on the improvement for $\alpha = \beta$ and $\alpha \neq \beta$.

https://doi.org/10.1371/journal.pone.0286878.g007

The number of cores affects the performance improvement of the threading technique. To evaluate this, the instead of enabling all of the eight logical cores as in the previous tests, only four logical cores were enabled. The tests were repeated with the new configuration and Fig 7 shows the effect of number of cores on the maximum improvement. The improvement in the four cores case is lower than that of the eight cores case by up to 48% when $\alpha \neq \beta$ ($\alpha = 100$ and $\beta = 50$). For the case where $\alpha = \beta = 10$, the four cores show up to 40% lower performance improvement as compared to the eight cores. It should be noted that at the unthreaded case both, the four cores and eight cores, perform similarly. The unthreaded case uses one core only which represents 25% and 12.5% of CPU usage in the 4 cores and 8 cores, respectively.

Conclusion

The paper proposed to exploit the parallelism in coefficient calculations allowing independent threads to process coefficients in parallel to enhance performance. The considered algorithm was analyzed at different matrix sizes, parameters' values, and number of threads. The maximum performance improvement as compared to the unthreaded case was found at the case of $\alpha = \beta$ ranging from 4.7 at 1000 × 1000 matrix size to 5.8 at 8000 × 8000 matrix size. On the other hand, when $\alpha \neq \beta$, the maximum improvement was at the matrix size of 8000, achieving a maximum improvement of 3.1. The results show that the number of threads should be carefully selected to achieve the optimal improvement, since increasing this number over certain limits may lead to performance degradation due to the threading overheads. Small matrix size (1000 × 1000) reaches its maximum improvement when the number of threads ranges from 8 to 16 whereas the optimum range was from 32 to 160 at large matrix size (8000 × 8000). Reducing the number of cores from 8 to 4 reduces the performance by 48% and 40% for $\alpha = \beta$ and $\alpha \neq \beta$, respectively Other candidate algorithms exist that it is encouraged to be analyzed when the threading technique is introduced into them which requires identifying the independence within each algorithm in order to be parallelized.

Author Contributions

Conceptualization: Sadiq H. Abdulhussain.

Formal analysis: Basheera M. Mahmmod.

Investigation: Wameedh Nazar Flayyih.

Methodology: Basheera M. Mahmmod, Sadiq H. Abdulhussain.

Software: Wameedh Nazar Flayyih, Wasiq Khan.

Supervision: Sadiq H. Abdulhussain, Abir Hussain.

Validation: Zainab Hassan Fakhri.

Visualization: Zainab Hassan Fakhri.

Writing – original draft: Basheera M. Mahmmod, Wameedh Nazar Flayyih.

Writing – review & editing: Wasiq Khan, Abir Hussain.

References

1. Abdulhussain SH, Al-Haddad SAR, Saripan MI, Mahmmod BM, Hussien A. Fast Temporal Video Segmentation Based on Krawtchouk-Tchebichef Moments. IEEE Access. 2020; 8:72347–72359. https://doi.org/10.1109/ACCESS.2020.2987870
2. Hameed IM, Abdulhussain SH, Mahmmod BM. Content-based image retrieval: A review of recent trends. Cogent Engineering. 2021; 8(1):1927469. https://doi.org/10.1080/23311916.2021.1927469
3. Mahmmod BM, bin Ramli AR, Abdulhussain SH, Al-Haddad SAR, Jassim WA. Signal Compression and Enhancement Using a New Orthogonal-polynomial-based Discrete Transform. IET Signal Processing. 2018; 12(1):129–142. https://doi.org/10.1049/iet-spr.2016.0449
4. Ming-Kuei Hu. Visual pattern recognition by moment invariants. IEEE Transactions on Information Theory. 1962; 8(2):179–187. https://doi.org/10.1109/TIT.1962.1057692
5. Zhu H, Shu H, Zhou J, Luo L, Coatrieux JL. Image analysis by discrete orthogonal dual Hahn moments. Pattern Recognition Letters. 2007; 28(13):1688–1704. https://doi.org/10.1016/j.patrec.2007.04.013
6. Deng AW, Wei CH, Gwo CY. Stable, Fast Computation of High-order Zernike Moments Using a Recursive Method. Pattern Recognition. 2016; 56:16–25. https://doi.org/10.1016/j.patcog.2016.02.014
7. Hosny KM, Darwish MM. New set of multi-channel orthogonal moments for color image representation and recognition. Pattern Recognition. 2019; 88:153–173. https://doi.org/10.1016/j.patcog.2018.11.014
8. Mukundan R, Ong SH, Lee PA. Discrete vs. Continuous Orthogonal Moments for Image Analysis. In: International Conference on Imaging Science, Systems, and Technology (CISST'01); 2001. p. 23–29.
9. Mukundan R. Some Computational Aspects of Discrete Orthonormal Moments. IEEE Transactions on Image Processing. 2004; 13(8):1055–1059. https://doi.org/10.1109/TIP.2004.828430 PMID: 15326847
10. Pew-Thian Yap, Paramesran R, Seng-Huat Ong. Image Analysis by Krawtchouk Moments. IEEE Transactions on Image Processing. 2003; 12(11):1367–1377. https://doi.org/10.1109/TIP.2003.818019
11. Zhou J, Shu H, Zhu H, Toumoulin C, Luo L. Image Analysis by Discrete Orthogonal Hahn Moments. In: Image Analysis and Recognition. Springer; 2005. p. 524–531.
12. Mahmmod BM, Ramli AR, Baker T, Al-Obeidat F, Abdulhussain SH, Jassim WA. Speech Enhancement Algorithm Based on Super-Gaussian Modeling and Orthogonal Polynomials. IEEE Access. 2019; 7:103485–103504. https://doi.org/10.1109/ACCESS.2019.2929864
13. den Brinker AC. Stable Calculation of Krawtchouk Functions from Triplet Relations. Mathematics. 2021; 9(16):1972. https://doi.org/10.3390/math9161972
14. Mahmmod BM, Abdulhussain SH, Naser MA, Alsabah M, Mustafina J. Speech Enhancement Algorithm Based on a Hybrid Estimator. In: IOP Conference Series: Materials Science and Engineering. vol. 1090. Samawah, Iraq: IOPscience; 2021. p. 012102. Available from: https://iopscience.iop.org/article/10.1088/1757-899X/1090/1/012102.
15. Camacho-Bello C, Rivera-Lopez JS. Some Computational Aspects of Tchebichef Moments for Higher Orders. Pattern Recognition Letters. 2018; 112:332–339. https://doi.org/10.1016/j.patrec.2018.08.020
16. Abdulhussain SH, Ramli AR, Al-Haddad SAR, Mahmmod BM, Jassim WA. On Computational Aspects of Tchebichef Polynomials for Higher Polynomial Order. IEEE Access. 2017; 5(1):2470–2478. https://doi.org/10.1109/ACCESS.2017.2669218
17. Yap PT, Paramesran R, Ong SH. Image Analysis Using Hahn Moments. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2007; 29(11):2057–2062. https://doi.org/10.1109/TPAMI.2007.70709 PMID: 17848784
18. Mahmmod BM, Abdul-Hadi AM, Abdulhussain SH, Hussien A. On Computational Aspects of Krawtchouk Polynomials for High Orders. Journal of Imaging. 2020; 6(8):81. https://doi.org/10.3390/jimaging6080081 PMID: 34460696

19. AL-Utaibi KA, Abdulhussain SH, Mahmmod BM, Naser MA, Alsabah M, Sait SM. Reliable Recurrence Algorithm for High-Order Krawtchouk Polynomials. Entropy. 2021; 23(9):1162. https://doi.org/10.3390/e23091162 PMID: 34573787

20. Mizel AKE. Orthogonal functions solving linear functional differential equationsusing chebyshev polynomial. Baghdad Science Journal. 2008; 5(1):143–148. https://doi.org/10.21123/bsj.5.1.143-148

21. Abdulhussain SH, Mahmmod BM, Naser MA, Alsabah MQ, Ali R, Al-Haddad SAR. A Robust Handwritten Numeral Recognition Using Hybrid Orthogonal Polynomials and Moments. Sensors. 2021; 21(6). https://doi.org/10.3390/s21061999 PMID: 33808986

22. Salih AM, Mahmood SH. Digital Color Image Watermarking Using Encoded Frequent Mark. Journal of Engineering. 2019; 25(3):81–88. https://doi.org/10.31026/j.eng.2019.03.07

23. Abdulqader DA, Hathal MS, Mahmmod BM, Abdulhussain SH, Al-Jumeily D. Plain, Edge, and Texture Detection Based on Orthogonal Moment. IEEE Access. 2022; 10:114455–114468. https://doi.org/10.1109/ACCESS.2022.3217225

24. Naser MA, Alsabah M, Mahmmod BM, Noordin NK, Abdulhussain SH, Baker T. Downlink Training Design for FDD Massive MIMO Systems in the Presence of Colored Noise. Electronics. 2020; 9 (12):2155. https://doi.org/10.3390/electronics9122155

25. Abood ZI. Composite Techniques Based Color Image Compression. Journal of Engineering. 2017; 23 (3):80–93. https://doi.org/10.31026/j.eng.2017.03.06

26. Daoui A, Yamni M, Ogri OE, Karmouni H, Sayyouri M, Qjidaa H. New Algorithm for Large-Sized 2D and 3D Image Reconstruction using Higher-Order Hahn Moments. Circuits, Systems, and Signal Processing. 2020;. https://doi.org/10.1007/s00034-020-01384-z

27. Abdul-Hadi AM, Abdulhussain SH, Mahmmod BM. On the computational aspects of Charlier polynomials. Cogent Engineering. 2020; 7(1). https://doi.org/10.1080/23311916.2020.1763553

28. Zhu H, Liu M, Shu H, Zhang H, Luo L. General Form for Obtaining Discrete Orthogonal Moments. IET Image Processing. 2010; 4(5):335–352. https://doi.org/10.1049/iet-ipr.2009.0195

29. Abdulhussain SH, Ramli AR, Mahmmod BM, Saripan MI, Al-Haddad SAR, Jassim WA. A New Hybrid form of Krawtchouk and Tchebichef Polynomials: Design and Application. Journal of Mathematical Imaging and Vision. 2019; 61(4):555–570. https://doi.org/10.1007/s10851-018-0863-4

30. Joodi M, Saleh M, Kadhim D. Increasing validation accuracy of a face mask detection by new deep learning model-based classification. Indonesian Journal of Electrical Engineering and Computer Science. 2023; 29:304–314. https://doi.org/10.11591/ijeecs.v29.i1.pp304-314

31. Joodi MA, Saleh MH, Khadhim DJ. Proposed Face Detection Classification Model Based on Amazon Web Services Cloud (AWS). Journal of Engineering. 2023; 29(4):176–206. https://doi.org/10.31026/j.eng.2023.04.12

32. Spiliotis IM, Mertzios BG. Fast Algorithms for Basic Processing and Analysis Operations on Block-represented Binary Images. Pattern Recognition Letters. 1996; 17(14):1437–1450. https://doi.org/10.1016/S0167-8655(96)00112-2

33. Shu H, Zhang H, Chen B, Haigron P, Luo L. Fast Computation of Tchebichef Moments for Binary and Grayscale Images. IEEE Transactions on Image Processing. 2010; 19(12):3171–3180. https://doi.org/10.1109/TIP.2010.2052276 PMID: 20542765

34. Abdulhussain SH, Mahmmod BM. Fast and efficient recursive algorithm of Meixner polynomials. Journal of Real-Time Image Processing. 2021. https://doi.org/10.1007/s11554-021-01093-z

35. Rivera-Lopez JS, Camacho-Bello C, Vargas-Vargas H, Escamilla-Noriega A. Fast computation of 3D Tchebichef moments for higher orders. Journal of Real-Time Image Processing. 2021;(0123456789). https://doi.org/10.1007/s11554-021-01152-5

36. Mahmmod BM, Abdulhussain SH, Suk T, Hussain A. Fast Computation of Hahn Polynomials for High Order Moments. IEEE Access. 2022; 10:48719–48732. https://doi.org/10.1109/ACCESS.2022.3170893

37. Foncannon JJ. Irresistible integrals: symbolics, analysis and experiments in the evaluation of integrals. The Mathematical Intelligencer. 2006; 28(3):65–68. https://doi.org/10.1007/BF02986888