Fast computation of Tchebichef moments for binary and gray-scale images

H.Z. Shu, *Senior Member, IEEE*, H. Zhang, B. J. Chen, P. Haigron, and L.M. Luo, *Senior Member, IEEE*

Abstract—**Discrete orthogonal moments have been recently introduced in the field of image analysis. It was shown that they have better image representation capability than the continuous orthogonal moments. One problem concerning the use of moments as feature descriptors is the high computational cost, which may limit their application to the problems where the on-line computation is required. In this paper, we present a new approach for fast computation of the two-dimensional Tchebichef moments. By deriving some properties of Tchebichef polynomials, and using the image block representation for binary images and intensity slice representation for gray-scale images, a fast algorithm is proposed for computing the moments of binary and gray-scale images. The theoretical analysis shows that the computational complexity of the proposed method depends on the number of blocks of the image, thus, it can speed up the computational efficiency as far as the number of blocks is smaller than the image size.**

Index Terms—**Discrete orthogonal moments, Tchebichef polynomials, fast computation, image block representation, intensity slice representation**

I. INTRODUCTION

MOMENTS and moment functions have been extensively used for feature extraction in pattern recognition and object classification [1]-[5]. One important property of the moments is their invariance under affine transformation. The pioneering work on this subject was by Hu [6]. Since then, many applications have been developed, which made use of geometric, complex, rotational and orthogonal moments [7]-[12].

Considerable attention has been paid on the theoretic study and application of the orthogonal moments since they can be easily used to reconstruct the image, and have the minimum information redundancy to represent the image [13]-[17].

Recently, discrete orthogonal moments such as Tchebichef, Krawtchouk, dual Hahn, Racah and Hahn moments have been introduced in image analysis community [18]-[23]. It was shown that they have better image representation capability than the continuous orthogonal moments.

This work was supported by the National Natural Science Foundation of China under Grants 60873048 and 60911130370, the National Basic Research Program of China under Grant 2010CB732503 and the Natural Science Foundation of Jiangsu Province of China under Grants SBK200910055 and BK2008279.

Huazhong Shu, Hui Zhang, Beijing Chen and Limin Luo are with the Laboratory of Image Science and Technology, School of Computer Science and Engineering, Southeast University, and also with Centre de Recherche en Information Biomédicale Sino-Français (CRIBs), 210096 Nanjing, China (e-mail: shu.list@seu.edu.cn, wtian@seu.deu.cn, cbj@nbut.cn,luo.list@seu.edu.cn).

Pascal Haigron is with the INSERM U642, 35000 Rennes, France, with Laboratoire Traitement du Signal et de l'Image, Université de Rennes I, 35000 Rennes, France, and also with Centre de Recherche en Information Biomédicale Sino-Français (CRIBs), 35000 Rennes, France (e-mail: pascal.haigron@univ-rennes1.fr).

One main difficulty concerning the use of moments as feature descriptors is their high computational complexity. To solve this problem, a number of fast algorithms have been reported in the literature [24]-[39]. Most of them concentrated on the fast computation of geometric moments and continuous orthogonal moments. Less attention has been paid on the fast computation of discrete orthogonal moments [35]-[39]. Wang and Wang [35] proposed a recursive algorithm based on Clenshaw's recurrence formula to compute the Tchebichef moments. Kotoulas and Andreadis [36] presented a hardware technique based on FPGA for implementing the calculation of Tchebichef moment values. They further proposed a more flexible architecture [37] dealing with various types of moments. Papakostas *et al.* [38] derived a unified methodology based on the image representation method for efficiently computing the discrete orthogonal moments. Bayraktar *et al.* [39] proposed an approach that consists of calculating the polynomial coefficients with arbitrary precision.

In this paper, we propose an efficient computation of Tchebichef moments for both binary and gray-scale images. For binary images, by using the image block representation proposed by Spiliotis and Merzios [25], the image moments can be obtained from the moments of all blocks. We further derive some properties of Tchebichef polynomials which can be used to efficiently calculate the moments of each block. The proposed method is then extended to gray-scale images by using the so-called 'intensity slice representation' introduced by Papakostas *et al.* [28].

The rest of the paper is organized as follows. In Section II, we review the definition of Tchebichef moments. Section III gives a brief introduction of the image block representation and the intensity slice representation. In Section IV, we first derive some properties of Tchebichef polynomials, and then propose a fast algorithm for computing the Tchebichef moments for both binary and gray-scale images. The computational complexity is analyzed in Section V and some experimental results are also provided. Section VI concludes the work.

II. TCHEBICHEF MOMENTS

The two-dimensional (2-D) Tchebichef moment of order (n+m) of an image intensity function $f(x, y)$ with size $N \times N$ is defined as [18], [19]

$$T_{nm} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} t_n(x) t_m(y) f(x,y), \tag{1}$$

where $t_n(x)$ is the nth order orthonormal Tchebichef polynomial defined by

$$t_n(x) = \frac{(1-N)_n}{\sqrt{\rho(n,N)}} \sum_{k=0}^{n} \frac{(-n)_k(-x)_k(1+n)_k}{(k!)^2(1-N)_k}, n,\ x = 0,\ 1, \ldots,\ N-1. \tag{2}$$

Here $(a)_k$ is the Pochhammer symbol

$$(a)_k = a(a+1)(a+2)\cdots(a+k-1),\ k \geq 1 \text{ and } (a)_0 = 1, \tag{3}$$

and the squared-norm $\rho(n, N)$ is given by

$$\rho(n,N)=\frac{(N+n)!}{(2n+1)(N-n-1)!}. \tag{4}$$

Equation (2) can be rewritten as

$$t_n(x)=\sum_{k=0}^{n}c_{n,k}(-x)_k, \tag{5}$$

where

$$\begin{aligned} c_{n,k} &= \frac{(-1)^k}{\sqrt{\rho(n,N)}}\frac{(n+k)!}{(n-k)!(k!)^2}\frac{(1-N)_n}{(1-N)_k} \\ &= \frac{(-1)^n}{\sqrt{\rho(n,N)}}\frac{(n+k)!(N-k-1)!}{(n-k)!(k!)^2(N-n-1)!}. \end{aligned} \tag{6}$$

The orthogonality property leads to the following inverse moment transform

$$f(x,y)=\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}T_{nm}t_n(x)t_m(y). \tag{7}$$

If only the moments of order up to (M–1, M–1) are computed, equation (7) is approximated by

$$\widetilde{f}(x,y)=\sum_{n=0}^{M-1}\sum_{m=0}^{M-1}T_{nm}t_n(x)t_m(y). \tag{8}$$

III. Representation of binary and gray-scale images

A. Image block representation for binary images

Image block representation (IBR) was introduced by Spiliotis and Mertzios [25] and has been used to achieve a fast computation of geometric moments for binary images.

A binary image, $f(x, y)$, can be represented as a set of blocks, each block corresponding to an object or a part of object. This block is defined as a rectangular area, which gathers a set of connected pixels whose value is included in the same value interval. This area building is briefly reviewed through the following algorithm [25], [28].

Algorithm IBR

Step 1. Consider each line y of the image $f(x, y)$ and find object level intervals in line y.

Step 2. Compare intervals and blocks that have pixels in line y–1.

Step 3. If an interval does not match with any block, this is the beginning of a new block.

Step 4. If a block matches with an interval, the end of the block is in the line y.

After applying the algorithm, the binary image is represented as a set of blocks of level one. This block-represented image is described by the relation

$$f(x, y) = \{b_i, i = 0, 1, \ldots, K-1\}, \tag{9}$$

where b_i is the ith block and K is the total number of blocks. Each block is described by the coordinates of the upper left and down right corner in vertical and horizontal axes.

B. *Partial intensity slice representation for gray-scale images*

Papakostas *et al.* [28] recently introduced a new image representation scheme, known as the intensity slice representation (ISR), which decomposes a gray-scale image $f(x, y)$ into a series of two-level images $f_i(x, y)$, that is

$$f(x, y) = \sum_{i=1}^{L} f_i(x, y), \tag{10}$$

where L is the number of slices (equal to the number of different intensity values) and $f_i(x, y)$ is the intensity function of the ith slice. In the case of a binary image, we have $L = 1$, so that, $f(x, y) = f_1(x, y)$.

Once a gray-scale image is decomposed into several slices according to the ISR scheme, each slice can be considered as a two-level image where the IBR algorithm can be applied. Instead of applying the IBR algorithm to each slice, Papakostas *et al.* [28] proposed to use the partial IBR (PIBR) algorithm. The PIBR algorithm consists of one pass of the image and a bookkeeping process, which can be described as follows.

Algorithm PIBR

Step 1. Consider each line y of the image $f(x, y)$ and find object level intervals for each intensity value that exists in line y.

Step 2. Compare intervals and blocks that have the same intensity line $y-1$.

Step 3. If an interval does not match with any block of the same intensity, this is the beginning of a new block.

Step 4. If a block matches with an interval of the same intensity, the end of the block is in the line y.

After applying the algorithm, the gray-scale image $f(x, y)$ can be redefined in terms of blocks of different intensities as

$$\begin{aligned} f(x, y) &= \{f_i(x, y), i = 1, 2, \ldots, L\} \\ f_i(x, y) &= \{b_{ij}, j = 0, 1, \ldots, K_i - 1\} \end{aligned} \tag{11}$$

where b_{ij} is the jth block of slice i and K_i is the number of image blocks having intensity f_i. Each block is described by the coordinates of the upper left and down right corner in vertical and horizontal axes.

IV. Fast computation of Tchebichef moments

In this section, we first derive some properties of orthonormal Tchebichef polynomials, and then develop an efficient algorithm for computing the Tchebichef moments.

A. *Some properties of orthonormal Tchebichef polynomials*

Theorem 1. Let $P_n(x) = \sum_{k=0}^{n} c_{n,k}(-x)_k$, where $(-x)_k$ is defined by (3) and $c_{n,n} \neq 0$, $n = 0, 1, \ldots, N-1$, be a set of polynomials. Assume for integer numbers a and x, we have

$$P_n(a+x) = \sum_{k=0}^{n} \sum_{l=0}^{n-k} g_l(n,k)(-a)_l P_k(x), \tag{12}$$

then the coefficients $g_l(n, k)$ are given by

$$g_l(n,k) = \sum_{s=k}^{n-l} \binom{l+s}{s} c_{n,l+s} d_{s,k}, \tag{13}$$

where $D_N=(d_{n,k})$, with $0 \le k \le n \le N-1$, is the inverse of the lower triangular matrix $C_N=(c_{n,k})$ of size $N \times N$, i.e., $D_N = C_N^{-1}$.

The proof of Theorem 1 is deferred to Appendix. In order to apply Theorem 1, an essential step consists of finding the inverse matrix D_N when the original matrix C_N is known. In this paper, we are interested in the use of Tchebichef polynomials. For the orthonormal Tchebichef polynomials $t_n(x)$ defined by (5) and (6), we have the following Proposition.

Proposition 1. For the lower triangular matrix C_N whose elements $c_{n,k}$ are defined by (6), the elements $d_{n,k}$ of the inverse matrix D_N are given by

$$d_{n,k} = (-1)^n \frac{\sqrt{\rho(k,N)}(2k+1)(n!)^2(N-k-1)!}{(n+k+1)!(n-k)!(N-n-1)!}. \tag{14}$$

The proof of Proposition 1 is deferred to Appendix. Based on Theorem 1 and Proposition 1, we can easily derive the following result.

Corollary 1. For the orthonormal Tchebichef polynomials, letting

$$t_n(a+x) = \sum_{k=0}^{n} \sum_{l=0}^{n-k} g_l(n,k)(-a)_l t_k(x), \tag{15}$$

then we have

$$g_l(n,k) = (-1)^n \frac{(2k+1)(N-1-k)!}{l!(N-1-n)!} \sqrt{\frac{\rho(k,N)}{\rho(n,N)}} \sum_{s=k}^{n-l} \frac{(-1)^s s!(n+l+s)!(N-1-s-l)!}{(l+s)!(n-l-s)!(s+k+1)!(s-k)!(N-1-s)!}, \quad 0 \le k \le n. \tag{16}$$

For the purpose of this paper, we are particularly interested in the case where a=1 in (15). By the definition of $(a)_k$ given by (3), we have $(-1)_0 = 1$, $(-1)_1 = -1$, and $(-1)_l = 0$ for $l \ge 2$. Using these properties, equation (15) becomes

$$t_n(x+1) = \sum_{k=0}^{n} g_0(n,k)t_k(x) - \sum_{k=0}^{n-1} g_1(n,k)t_k(x), \tag{17}$$

where $g_l(n, k)$, $l = 0, 1$, defined by (16), are given as

$$g_0(n,k) = (-1)^n \frac{(2k+1)(N-1-k)!}{(N-1-n)!} \sqrt{\frac{\rho(k,N)}{\rho(n,N)}} \sum_{s=k}^{n} \frac{(-1)^s (n+s)!}{(n-s)!(s-k)!(s+k+1)!}, \qquad 0 \le k \le n, \tag{18}$$

$$g_1(n,k) = (-1)^n \frac{(2k+1)(N-1-k)!}{(N-1-n)!}\sqrt{\frac{\rho(k,N)}{\rho(n,N)}}\sum_{s=k}^{n-1}\frac{(-1)^s(n+1+s)!}{(s+1)(N-1-s)(n-1-s)!(s+k+1)!(s-k)!}$$
$$= -\frac{(2k+1)(N-1-k)!}{(N-1-n)!}\sqrt{\frac{\rho(k,N)}{\rho(n,N)}}\sum_{s=0}^{n-k-1}\frac{(-1)^s(2n-s)!}{(n-s)(N-n+s)s!(n+k-s)!(n-1-k-s)!} \tag{19}$$
$$= -\sum_{s=0}^{n-k-1} B_{n,k,s}, \qquad 0 \le k \le n-1,$$

where

$$B_{n,k,s} = \frac{(-1)^s(2k+1)(2n-s)!}{(n-s)(N-n+s)s!(n+k-s)!(n-1-k-s)!}\frac{(N-1-k)!}{(N-1-n)!}\sqrt{\frac{\rho(k,N)}{\rho(n,N)}}. \tag{20}$$

For the computation of $g_0(n, k)$, we have the following result.

Theorem 2. For $g_0(n, k)$ given by (18), we have

$$g_0(n,n) = 1, \quad g_0(n,k) = 0 \quad \text{for } 0 \le k \le n-1. \tag{21}$$

The proof of Theorem 2 is deferred to Appendix.

Using (21), equation (17) becomes

$$t_n(x+1) = t_n(x) - \sum_{k=0}^{n-1} g_1(n,k)t_k(x). \tag{22}$$

We are now ready to propose a new approach for efficiently computing the Tchebichef moments defined by (1). This is the subject of the following subsections.

B. *Fast computation of Tchebichef moments for binary images*

For a binary image $f(x, y)$ represented by K blocks, as described in (10), equation (1) can be rewritten as

$$T_{nm} = \sum_{i=0}^{K-1}\sum_{x=x_{1,b_i}}^{x_{2,b_i}}\sum_{y=y_{1,b_i}}^{y_{2,b_i}} t_n(x)t_m(y) = \sum_{i=0}^{K-1} T_{nm}^{b_i}, \tag{23}$$

where (x_{1,b_i}, y_{1,b_i}) and (x_{2,b_i}, y_{2,b_i}) are respectively the left-up and right-bottom coordinates of the block b_i, and $T_{nm}^{b_i}$ is the moment of block b_i given by

$$T_{nm}^{b_i} = \sum_{x=x_{1,b_i}}^{x_{2,b_i}}\sum_{y=y_{1,b_i}}^{y_{2,b_i}} t_n(x)t_m(y)$$
$$= \left[\sum_{x=x_{1,b_i}}^{x_{2,b_i}} t_n(x)\right]\left[\sum_{y=y_{1,b_i}}^{y_{2,b_i}} t_m(y)\right] \tag{24}$$
$$= S_n(x_{1,b_i}, x_{2,b_i})S_m(y_{1,b_i}, y_{2,b_i}),$$

with

$$S_n(x_{1,b_i}, x_{2,b_i}) = \sum_{x=x_{1,b_i}}^{x_{2,b_i}} t_n(x), \quad S_m(y_{1,b_i}, y_{2,b_i}) = \sum_{y=y_{1,b_i}}^{y_{2,b_i}} t_m(y). \tag{25}$$

Equation (23) shows that to obtain the image moments, we need to calculate the moments of each block, so we turn to it in the following. Since $S_n(x_{1,b_i}, x_{2,b_i})$ and $S_m(y_{1,b_i}, y_{2,b_i})$ given in (25) can be calculated in a similar way, we consider only the computation of $S_n(x_{1,b_i}, x_{2,b_i})$.

Assuming that the block b_i contains $\delta_{b_i} = x_{2,b_i} - x_{1,b_i} + 1$ pixels in width, we have

$$S_n(x_{1,b_i}, x_{2,b_i}) = \sum_{j=0}^{\delta_{b_i}-1} t_n(x_{1,b_i} + j). \tag{26}$$

Using (22), we have

$$t_{n+1}(x_{1,b_i} + j + 1) - t_{n+1}(x_{1,b_i} + j) = -\sum_{k=0}^{n} g_1(n+1,k) t_k(x_{1,b_i} + j). \tag{27}$$

Summing the two sides of (27) from $j = 0$ to $\delta_{b_i} - 1$, we obtain

$$\begin{aligned} t_{n+1}(x_{1,b_i} + \delta_{b_i}) - t_{n+1}(x_{1,b_i}) &= -\sum_{j=0}^{\delta_{b_i}-1} \sum_{k=0}^{n} g_1(n+1,k) t_k(x_{1,b_i} + j) \\ &= -\sum_{k=0}^{n} g_1(n+1,k) \sum_{j=0}^{\delta_{b_i}-1} t_k(x_{1,b_i} + j). \end{aligned} \tag{28}$$

Using (26) and making the notation

$$R_n(\delta_{b_i}) = t_{n+1}(x_{1,b_i} + \delta_{b_i}) - t_{n+1}(x_{1,b_i}), \tag{29}$$

Equation (28) becomes

$$R_n(\delta_{b_i}) = \sum_{k=0}^{n} g_1(n+1,k) S_k(x_{1,b_i}, x_{2,b_i}). \tag{30}$$

Let $V_M(x_{1,b_i}, x_{2,b_i}) = (S_0(x_{1,b_i}, x_{2,b_i}), S_1(x_{1,b_i}, x_{2,b_i}), ..., S_{M-1}(x_{1,b_i}, x_{2,b_i}))^T$ and $U_M(\delta_{b_i}) = (R_0(\delta_{b_i}), R_1(\delta_{b_i}), ..., R_{M-1}(\delta_{b_i}))^T$ where the subscript T denotes the transpose and M is the maximal order of Tchebichef moments we want to calculate, we have

$$U_M(\delta_{b_i}) = -A_M V_M(x_{1,b_i}, x_{2,b_i}), \tag{31}$$

where A_M is an $M \times M$ lower triangular matrix given by

$$A_M = \begin{bmatrix} g_1(1,0) & 0 & 0 & 0 \\ g_1(2,0) & g_1(2,1) & 0 & 0 \\ g_1(3,0) & g_1(3,1) & g_1(3,2) & 0 \\ \cdots & \cdots & \cdots & \cdots \\ g_1(M,0) & g_1(M,1) & g_1(M,2) & g_1(M,M-1) \end{bmatrix} \tag{32}$$

The elements $g_1(n, k)$ of the matrix A_M can be computed via (19). In particular, we have

$$g_1(n,n-1) = 2(2n-1)\sqrt{\frac{\rho(n-1,N)}{\rho(n,N)}} = 2\sqrt{\frac{(2n-1)(2n+1)}{(N-n)(N+n)}}. \tag{33}$$

Since all the diagonal elements are not zero, the matrix A_M is non-singular. Thus, we have

$$V_M(x_{1,b_i}, x_{2,b_i}) = -A_M^{-1} U_M(\delta_{b_i}) . \tag{34}$$

The above equation shows that to obtain the values of $V_M(x_{1,b_i}, x_{2,b_i})$, we need only to compute $U_M(\delta_{b_i})$, which can be done via (29). The Tchebichef polynomial values can be calculated by the following recurrence formula [19]

$$t_n(x) = \alpha_1 t_n(x-1) + \alpha_2 t_n(x-2), \text{ for } n=1,2,\ 3,\ \ldots,\ N-1,\ x=2,\ 3,\ \ldots,\ N/2, \tag{35}$$

where

$$\begin{aligned} \alpha_1 &= \frac{-n(n+1)-(2x-1)(x-N-1)-x}{x(N-x)}, \\ \alpha_2 &= \frac{(x-1)(x-N-1)}{x(N-x)}, \end{aligned} \tag{36}$$

and

$$\begin{aligned} t_n(0) &= -\sqrt{\frac{(N-n)(2n+1)}{(N+n)(2n-1)}}\, t_{n-1}(0), \\ t_n(1) &= \left[1 + \frac{n(n+1)}{1-N}\right] t_n(0), \\ t_0(0) &= \sqrt{\frac{1}{N}}. \end{aligned} \tag{37}$$

Note that in (35), the following symmetric property is used

$$t_n(N-1-x) = (-1)^n t_n(x). \tag{38}$$

As indicated by Mukundan [19], the use of (35) and (38) allows avoiding the numerical instability in the calculation of polynomial values.

Because the computation of $g_1(n, k)$ for $0 \le k \le n-2$, when using (19), requires the evaluation of factorial functions, this could be time consuming. To avoid this, we use the following recurrence relations for computing the coefficients $B_{n,k,s}$.

$$B_{n,0,0} = \frac{2(N-n+1)}{n}\sqrt{\frac{(2n-1)(2n+1)}{(N-n)(N+n)}}\, B_{n-1,0,0}, \quad n \ge 2, \tag{39}$$

$$B_{n,k,s} = -\frac{(n-s+1)(n+k-s+1)(n-k-s)(N-n+s-1)}{s(n-s)(2n-s+1)(N-n+s)}\, B_{n,k,s-1}, \quad 0 \le s \le n-k-1, \tag{40}$$

$$B_{n,k,s} = \frac{(n-k-s)}{(n+k-s)}\sqrt{\frac{(2k+1)(N+k)}{(2k-1)(N-k)}}\, B_{n,k-1,s}, \quad 1 \le k \le n-2, \tag{41}$$

$$B_{1,0,0} = 2\sqrt{\frac{3}{N^2-1}} . \tag{42}$$

It is worth noting that the elements $g_1(n, k)$ are independent of the image $f(x, y)$, they can thus be pre-computed and stored in a look-up table for further use.

C. Fast computation of Tchebichef moments for gray-scale images

By using the ISR algorithm, the Tchebichef moments of a gray-scale image *f*(*x*, *y*), which is described by (11), can be computed as

$$\begin{aligned} T_{nm} &= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} t_n(x)t_m(y)\sum_{i=1}^{L} f_i(x,y) \\ &= \sum_{i=1}^{L}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} t_n(x)t_m(y)f_i(x,y) \\ &= \sum_{i=1}^{L} f_i T_{nm}(i), \end{aligned} \tag{43}$$

where $T_{nm}(i)$ is the (*n*+*m*)th order Tchebichef moments of the *i*th binary slice.

Equation (43) shows that the (*n*+*m*)th order Tchebichef moment of a gray-scale image *f*(*x*, *y*) is equal to the intensity-weighted sum of the same order Tchebichef moments of a number of binary slices. The latter moments can be computed using the algorithm presented in the previous subsection.

To summarize, the proposed method for computing the moment values is described as follows.

Algorithm for computing the Tchebichef moments

Step 1. Image block extraction using IBR algorithm for binary image and PIBR algorithm for gray-scale image.

Step 2. Computation of Tchebichef polynomial values at corners of each block using (35), and then the vector $R_n(\delta_{b_i})$ with (29).

Step 3. Calculation of 1-D Tchebichef moments of each block using (34).

Step 4. Computation of image moment values using (23) for binary image and (43) and (23) for gray-scale image.

V. Computational complexity and experimental results

In this section, we give a detailed analysis of computational complexity of the proposed algorithm, and provide some experimental results to validate the theoretic analysis.

A. Computational complexity

The complexity of the proposed algorithm is due to the extraction of the extraction of image blocks with the PIBR algorithm and to the computation of Tchebichef moments based on (23) for binary images or on (43) for gray-scale images. As pointed out by Papakostas *et al.* [28], the procedure of block extraction is performed by simple mathematical and logical operations, and it adds very short time overhead in the overall computation. For this reason, we do not take this part into account.

Since the computation of Tchebichef moments for a gray-scale image based on (43) depends on the algorithm being used to compute the moments of each slice, we first consider the arithmetic complexity of (23) using the proposed algorithm.

Let us consider the case where a binary image contains one rectangular block with level one. For simplicity and without loss of generality, assume a square block with $J \times J$ pixels, and the moments up to order $(M-1, M-1)$ need to be calculated. For the direct method based on (1), the computation of Tchebichef polynomial values $t_n(x)$ using the recursive formula (35) for each given x requires 1 addition and 2 multiplications. The same number of arithmetic operations is needed for $t_m(y)$. Thus, the computation of Tchebichef moments T_{nm} of order up to $(M-1, M-1)$ based on (1), using the direct method, for a block size $J \times J$ pixels needs $M^2(3J^2-1)$ additions and $5M^2J^2$ multiplications.

We then analyze the complexity of the proposed algorithm based on eqs (23) and (34). The computation of the vectors $U_M(\delta_{b_i})$ and $V_M(x_{1,b_i}, x_{2,b_i})$ requires respectively $3M$ additions and $4M$ multiplications, and $M(M-1)/2$ additions and $M(M+1)/2$ multiplications. Thus, the total arithmetic operations required in the computation of $V_M(x_{1,b_i}, x_{2,b_i})$ are $M^2/2+5M/2$ additions and $M^2/2+9M/2$ multiplications. The same number of operations is required for $V_M(y_{1,b_i}, y_{2,b_i})$. Therefore, the computation of M^2 Tchebichef moments using (23) and (34) requires M^2+5M additions and $2M^2+9M$ multiplications. Table I summarizes these results. For comparison purpose, we also list in Table I the arithmetic complexity of the algorithms reported in Refs. [35], [37] and [38]. Note that the algorithm presented in [37] leads to the same number of arithmetic operations as in [36], and the method for computing the block moments reported in [38] is just based on (24), which requires $(4J-2)M^2$ additions and $(4J+1)M^2$ multiplications. It can be seen from this table that among these methods, both the proposed algorithm and the algorithm reported in [37] are independent of the block size, and our method has the lowest computational complexity.

For a gray-scale image $f(x, y)$ with size $N \times N$, suppose that the total number of blocks for all the slices is NB, that is, $NB = \sum_{i=1}^{L} K_i$, where K_i is the number of blocks of the ith slice. Then the computation of Tchebichef moments T_{nm} of order up to $(M-1, M-1)$ based on (1), using the direct method, requires $M^2(3N^2-1)$ additions and $6M^2N^2$ multiplications. The computational complexity of the algorithm reported in [38] is $NB(4J-2)M^2+L-1$ additions and $NB(4J+1)M^2+L-1$ multiplications, and that of the proposed algorithm based on (43) is $NB(M^2+5M)+L-1$ additions and $NB(2M^2+9M)+L$ multiplications.

B. *Experimental results*

Some experimental results are provided in this subsection to validate the theoretical analysis. Since the algorithm presented in [37] was realized by hardware architecture, we compare here the proposed algorithm with the direct method, the recursive algorithm presented in [35] and the fast algorithm reported in [38] in terms of the computational efficiency. We do not provide a full comparison of our method with that reported in [39] for the following reasons: the main advantage of the technique presented by Bayraktar *et al.* [39] is its high precision. As noted by the authors, the computation of polynomial values using arbitrary precision calculator is much slower than the recurrence formula. So, their method is fast only if all the polynomial coefficients are pre-computed and stored in a look-up table. On the contrary, Bayraktar's algorithm is less efficient than the previously reported fast algorithms.

In the first example, four binary images with size 256 × 256 pixels (Fig. 1) selected from the well-known MPEG-7 CE-shape-1 Part B database [40] were used as test images. The number of blocks of these images is *NB* = 171 for Apple, *NB* = 44 for Hammer, *NB* = 388 for Octopus and *NB* = 136 for Tree. Fig. 2 shows the average computation time of moments up to order (120, 120) for these four images using the direct method, the recursive algorithm based on Clenshaw's recurrence formula, the algorithm reported in [38] and the proposed algorithm. Note that the algorithm was implemented in C++ on a PC Dual Core 2.33 GHz, 2GB RAM. Fig. 2 shows that the proposed algorithm is the fastest among all the methods, and the algorithm presented in [38] is more efficient than the other two methods. This is because these binary images have a small number of blocks. Note that the computation time for extracting the blocks of each image is about 1 ms, this time is much less than the computation time required in the calculation of moments.

In the second example, four gray-scale images with size 256 × 256 pixels shown in Fig. 3 have been used. The number of blocks of these images is *NB* = 56211 for Lena, *NB* = 53048 for Pepper, *NB* = 47664 for Women and *NB* = 38561 for House. The computation time for extracting the blocks of each image is about 2 ms. Fig. 4 shows the average computation time up to order (120, 120) for these four images using various methods. The result again indicates that our method has better performance than the other algorithms. But the algorithm presented in [38] is only faster than the direct method due to the large number of blocks in these images, and the computation of 1-D moments based on (24) is time intensive. Fig. 5 shows the reconstructed results using the inverse transform (8).

From the two previous experiments, it can be observed that our algorithm depends on the number of image blocks, which is related to the image content, rather than on the image size. To illustrate this, the images shown in Figs. 1 and 3 were scaled to different sizes (from 320 × 320 to 1024 × 1024) where the nearest interpolation was used. Using such an interpolation, the number of blocks does not change. Fig. 6 shows the average computation time required in the calculation of moments of order up to (40, 40) for different image sizes. It can be seen from this figure that both the proposed algorithm and Papakostas's algorithm are much more efficient than the two other algorithms. To make a full comparison in terms of the efficiency of different methods, we also apply the bilinear interpolation to images shown in Figs. 1 and 3. In such a case, the number of blocks increases with the image size. The average computation time required in the calculation of moments of order up to (40, 40) for different image sizes is shown in Fig. 7. It can be observed from this figure that the computation time required in our method and Papakostas's method increases compared to that of Fig. 6. However, the proposed method remains the most efficient one.

VI. Conclusions

In this paper, by deriving some properties of Tchebichef polynomials, and using the image block representation and intensity slice representation, we have presented a fast algorithm for computing the Tchebichef moments for both binary and gray-scale images. The computation of the moments using the proposed method only depends on the number of blocks, thus, it can significantly decrease the computation time when the number of image blocks is much smaller than the image size.

Acknowledgments

The authors would like to thank the reviewers and Associate Editor Dr. Wu for their insightful suggestions which helped improve the quality of the manuscript.

Appendix A

Proof of Theorem 1. By definition of $d_{n,k}$, we have

$$(-x)_n = \sum_{k=0}^{n} d_{n,k} P_k(x). \tag{A1}$$

Using the following relationship [41]

$$(-a-x)_l = \sum_{s=0}^{l} \binom{l}{s} (-a)_{l-s} (-x)_s, \tag{A2}$$

where $\binom{l}{s} = \frac{l!}{s!(l-s)!}$ is the combination number, we have

$$\begin{aligned} P_n(a+x) &= \sum_{l=0}^{n} c_{n,l}(-a-x)_l = \sum_{l=0}^{n} c_{n,l} \sum_{s=0}^{l} \binom{l}{s} (-a)_{l-s}(-x)_s \\ &= \sum_{s=0}^{n}\sum_{l=s}^{n} c_{n,l} \binom{l}{s} (-a)_{l-s}(-x)_s \\ &= \sum_{s=0}^{n}\sum_{l=0}^{n-s} c_{n,l+s} \binom{l+s}{s} (-a)_l(-x)_s \\ &= \sum_{s=0}^{n}\sum_{l=0}^{n-s} c_{n,l+s} \binom{l+s}{s} (-a)_l \sum_{k=0}^{s} d_{s,k} P_k(x) \\ &= \sum_{k=0}^{n}\sum_{l=0}^{n-k}\sum_{s=k}^{n-l} \binom{l+s}{s} c_{n,l+s} d_{s,k} (-a)_l P_k(x). \end{aligned} \tag{A3}$$

The proof of Theorem 1 is completed. ◆

Proof of Proposition 1. To prove the proposition, we need to demonstrate the following relation

$$\sum_{k=m}^{n} c_{n,k} d_{k,m} = \delta_{nm}. \tag{A4}$$

where δ_{nm} is the Kronecker symbol.

Using (6) and (14), we have

$$\sum_{k=m}^{n} c_{n,k} d_{k,m} = (-1)^n \frac{(2m+1)(N-m-1)!}{(N-n-1)!} \sqrt{\frac{\rho(m,N)}{\rho(n,N)}} \sum_{k=m}^{n} \frac{(-1)^k (n+k)!}{(n-k)!(k+m+1)!(k-m)!}. \tag{A5}$$

For $n = m$, it can be easily deduced from (A5) that

$$c_{n,n} d_{n,n} = (-1)^n (2n+1) \times \frac{(-1)^n (2n)!}{(2n+1)!} = 1. \tag{A6}$$

To prove (A4) for $m < n$, letting

$$G(n,k) = (-1)^{k+1} \binom{n+k+1}{n-k+1} \binom{2k}{k-m} \frac{(n+1-k)(k-m)}{(n+k+1)(n+m+1)(n-m)}. \tag{A7}$$

It can be verified that

$$G(n,k+1) - G(n,k) = \frac{(-1)^k (n+k)!}{(n-k)!(k+m+1)!(k-m)!}. \tag{A8}$$

Substitution of (A8) into (A5), we obtain

$$\begin{aligned} &\sum_{k=m}^{n} c_{n,k} d_{k,m} \\ &= (-1)^n \frac{(2m+1)(N-m-1)!}{(N-n-1)!} \sqrt{\frac{\rho(m,N)}{\rho(n,N)}} \sum_{k=m}^{n} \frac{(-1)^k (n+k)!}{(n-k)!(k+m+1)!(k-m)!} \qquad \text{for } m < n. \\ &= (-1)^n \frac{(2m+1)(N-m-1)!}{(N-n-1)!} \sqrt{\frac{\rho(m,N)}{\rho(n,N)}} \sum_{k=m}^{n} [G(n,k+1) - G(n,k)] \\ &= (-1)^n \frac{(2m+1)(N-m-1)!}{(N-n-1)!} \sqrt{\frac{\rho(m,N)}{\rho(n,N)}} [G(n,n+1) - G(n,m)] = 0, \end{aligned} \tag{A9}$$

The proof is now complete.

Note that the proof of Proposition was inspired by a technique proposed by Petkovsek *et al.* [42].

Proof of Theorem 2. It can be easily verified from (21) that $g_0(n, n) = 1$. To prove $g_0(n, k) = 0$ for $k \le n-1$, it suffices to apply the relationship (A9) to (18).

REFERENCES

[1] S.A. Dudani, K.J. Breeding, and R.B. McGhee, "Aircraft identification by moment invariant," *IEEE Trans. Comput.*, vol. 26, no. 1, pp. 39-46, 1977.

[2] A. Khotanzad, "Invariant image recognition by Zernike moments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 5, pp. 489-497, 1990.

[3] S.O. Belkasim, "Pattern recognition with moment invariants—A comparative study and new results," *Pattern Recognit.*, vol. 24, no. 12, pp. 1117-1138, 1991.

[4] S. Pei and C. Lin, "Normalization of rotationally symmetric shapes for pattern recognition," *Pattern Recognit.*, vol. 25, no. 9, pp. 913-920, 1992.

[5] J. Flusser, “Pattern recognition by affine moment invariants,” *Pattern Recognit.*, vol. 26, no. 1, pp. 167-174, 1993.

[6] M.K. Hu, “Visual pattern recognition by moment invariants,” *IRE Trans. Inform. Theory*, vol. IT-8, no. 2, pp. 179-187, 1962.

[7] R.J. Prokop and A.P. Reeves, “A survey of moment based techniques for unoccluded object representation,” *Graph. Models Image Process.*, vol. 54, no. 5, pp. 438-462, 1992.

[8] R. Mukundan and K.R. Ramakrishnan, *Moment Functions in Image Analysis – Theory and Applications*, Singapore: World Scientific, 1998.

[9] H.Z. Shu, L.M. Luo, and J.L. Coatrieux, “Moment-based approaches in image Part 1: basic features,” *IEEE Eng. Med. Biol. Mag.*, vol. 26, no.5, pp. 70-74, 2007.

[10] H.Z. Shu, L.M. Luo, and J.L. Coatrieux, “Moment-based approaches in image Part 2: invariance,” *IEEE Eng. Med. Biol. Mag.*, vol. 27, no. 1, pp. 81-83, 2008.

[11] H.Z. Shu, L.M. Luo, and J.L. Coatrieux, “Moment-based approaches in image Part 3: computational considerations,” *IEEE Eng. Med. Biol. Mag.*, vol. 27, no. 3, pp. 89-91, 2008.

[12] H.Z. Shu, L.M. Luo, and J.L. Coatrieux, “Moment-based approaches in imaging Part 4: some applications,” *IEEE Eng. Med. Biol. Mag.*, vol. 27, no. 5, pp. 116-118, 2008.

[13] M.R. Teague, “Image analysis via the general theory of moments,” *J. Opt. Soc. Amer.*, vol. 70, pp. 920-930, 1980.

[14] C.H. Teh and R.T. Chin, “On image analysis by the method of moments,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 4, pp. 496-513, 1988.

[15] S.X. Liao and M. Pawlak, “On image analysis by moments,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 3, pp. 254-266, 1996.

[16] Z.L. Ping, R.G. Wu, and Y.L. Sheng, “Image description with Chebyshev-Fourier moments,” *J. Opt. Soc. Am. A*, vol. 19, no. 9, pp. 1748-1754, 2002.

[17] T. Xia, H.Q. Zhu, H.Z. Shu, P. Haigron, and L.M. Luo, “Image description with generalized pseudo-Zernike moments,” *J. Opt. Soc. Am. A*, vol. 24, no. 1, pp. 50-59, 2007.

[18] R. Mukundan, S.H. Ong, and P.A. Lee, “Image analysis by Tchebichef moments,” *IEEE Trans. Image Process.*, vol. 10, no. 9, pp. 1357-1364, 2001.

[19] R. Mukundan, “Some computational aspects of discrete orthonormal moments,” *IEEE Trans. Image Process.*, vol. 13, no.8, pp. 1055-1059, 2004.

[20] P.T. Yap, P. Raveendran, and S.H. Ong, “Image analysis by Krawtchouk moments,” *IEEE Trans. Image Processing*, vol. 12, no. 11, pp. 1367-1377, 2003.

[21] H.Q. Zhu, H.Z. Shu, J. Liang, L.M. Luo, and J.L. Coatrieux, Image analysis by discrete orthogonal dual-Hahn moments, *Pattern Recognit. Lett.*, vol. 28, pp. 1688-1704, 2007.

[22] H.Q. Zhu, H.Z. Shu, J. Liang, L.M. Luo, and J.L. Coatrieux, Image analysis by discrete orthogonal Racah moments, *Signal Process.*, vol. 87, pp. 687-708, 2007.

[23] P.T. Yap, P. Raveendran, and S.H. Ong, “Image analysis using Hahn moments,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 2057-2062, 2007.

[24] L. Yang and F. Albregtsen, “Fast and exact computation of Cartesian geometric moments using discrete Green’s theorem,” *Pattern Recognit.*, vol. 29, no. 7, pp. 1069-1073, 1996.

[25] I.M. Spiliotis and B.G. Mertzios, “Real-time computation of two-dimensional moments on binary images using image block representation,” *IEEE Trans. Image Process.*, vol. 7, no. 11, pp. 1609-1615, 1998.

[26] J. Flusser, “Refined moment calculation using image block representation,” *IEEE Trans. Image Process.*, vol. 9, no. 11, pp. 1977-1978, 2000.

[27] K.L. Chung and P.C. Chen, “An efficient algorithm for computing moments on a block representation of a grey-level image,” *Pattern Recognit.*, vol. 38, no. 12, pp. 2578-2586, 2005.

[28] G.A. Papakostas, E.G. Karakasis, and D.E. Koulourisotis, “Efficient and accurate computation of geometric moments on gray-scale images,” *Pattern Recognit.*, vol. 41, no. 6, pp. 1895-1904, 2008.

[29] R. Mukundan and K.R. Ramakrishnan, “Fast computation of Legendre and Zernike moments,” *Pattern Recognit.*, vol. 28, no.9, pp. 1433-1442, 1995.

[30] H.Z. Shu, L.M. Luo, W.X. Yu, and Y. Fu, “A new fast method for computing Legendre moments,” *Pattern Recognit.*, vol. 33, no. 2, pp. 341-348, 2000.

[31] P.T. Yap and P. Raveendran, “An efficient method for the computation of Legendre moments,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 12, pp. 1996-2002, 2005.

[32] G.Y. Yang, H.Z. Shu, C. Toumoulin, G.N. Han, and L.M. Luo, “Efficient Legendre moment computation for grey level images,” *Pattern Recognit.*, vol. 39, no. 1, pp. 74-80, 2006.

[33] J. Gu, H.Z. Shu, C. Toumoulin, and L.M. Luo, “A novel algorithm for fast computation of Zernike moments,” *Pattern Recognit.*, vol. 35, no. 12, pp. 2905-2911, 2002.

[34] C.W. Chong, P. Raveendran, and R. Mukundan, “A comparative analysis of algorithms for fast computation of Zernike moments,” *Pattern Recognit.*, vol. 36, no.3, pp. 1765-1773, 2003.

[35] G.B. Wang and S.G. Wang, Recursive computation of Tchebichef moment and its inverse transform, *Pattern Recognit.*, vol. 39, no. 1, pp. 47-56, 2006.

[36] L. Kotoulas and I. Andreadis, “Fast computation of Chebyshev moments,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 7, pp. 884-888, 2006.

[37] L. Kotoulas and I. Andreadis, “Fast moment generating architectures,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 4, pp. 533-537, 2008.

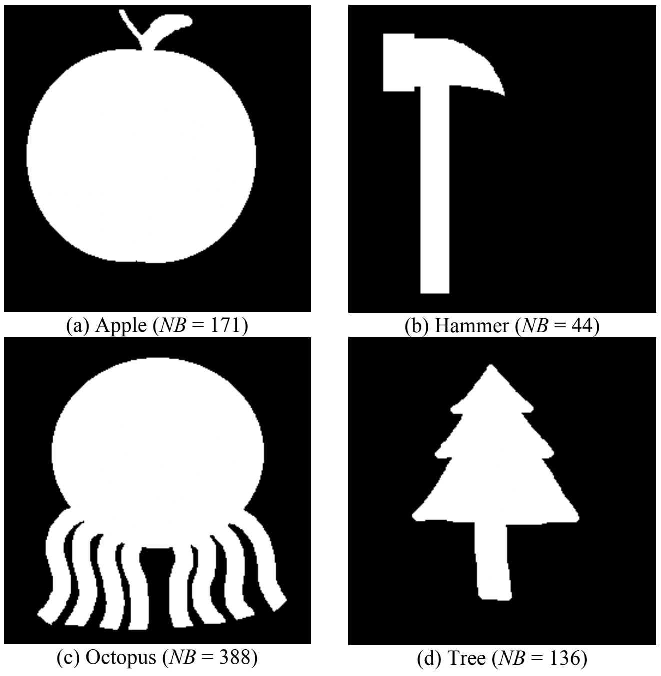
[38] G.A. Papakostas, E.G. Karakasis, and D.E. Koulourisotis, “A unified methodology for efficient computation of discrete orthogonal image moments,” *Information Sciences*, vol. 179, no. 20, pp. 3619-3633, 2009.

[39] B. Bayraktar, T. Bernas, J.P. Robinson, and B. Rajwa, “A numerical recipe for accurate image reconstruction from discrete orthogonal moments,” *Pattern Recognit.*, vol. 40, no. 2, pp. 659-669, 2007.

[40] http://www.imageprocessingplace.com/root_files_V3/image_databases.htm

[41] L. Comet, *Advanced combinatorics: The art of finite and infinite expansions*, Dordrecht, Holland, D. Reidel Publishing Company, 1974.

[42] M. Petkovsek, H. S. Wilf, and D. Zeilberger, *A = B*, AK Peters, Ltd., 1996. (Available on line at the University of Pennsylvania)


(a) Apple (NB = 171) (b) Hammer (NB = 44)

(c) Octopus (NB = 388) (d) Tree (NB = 136)

Fig. 1. Set of test binary images with size 256×256 pixels.

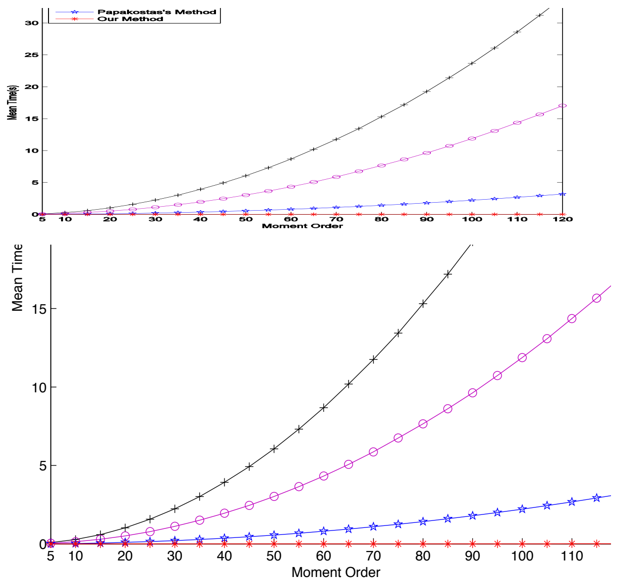
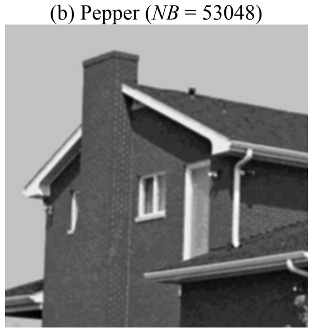


Fig. 2. Average computation time for images shown in Fig. 1 using different methods

(a) Lena (*NB* = 56211)

(b) Pepper (*NB* = 53048)

(c) Woman (*NB* = 47664)



(d) House (*NB* = 38561)

Fig. 3. Set of test gray-scale images with size 256 × 256 pixels.

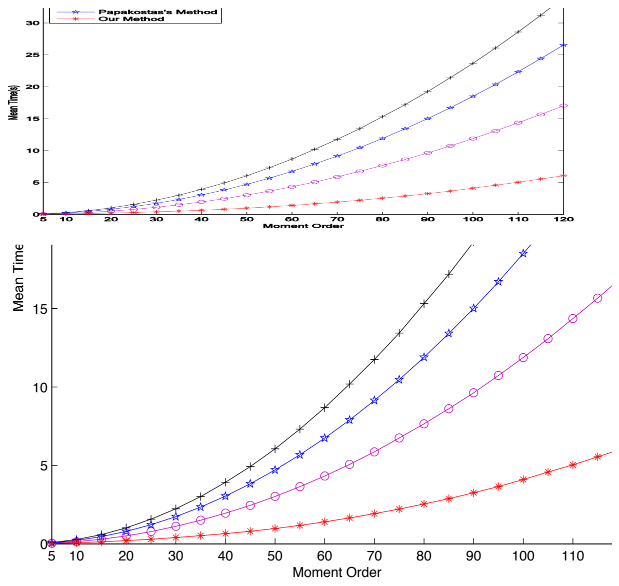
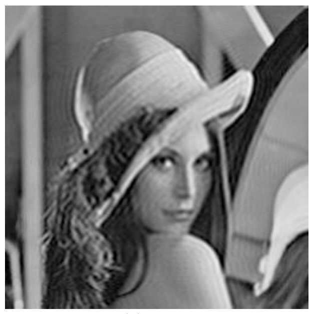
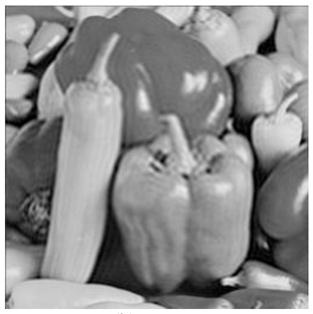
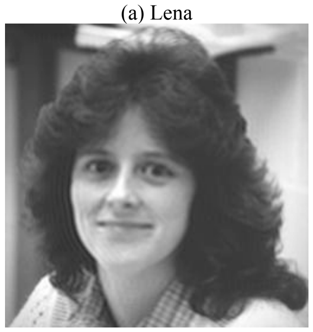


Fig. 4. Average computation time for images shown in Fig. 3 using different methods


(a) Lena


(b) Pepper


(c) Woman

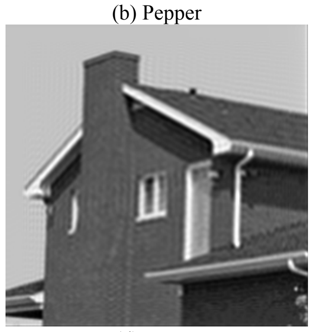

(d) House

Fig. 5. Reconstructed images of Fig. 3 using the inverse moment transform (8) with $M = 120$.

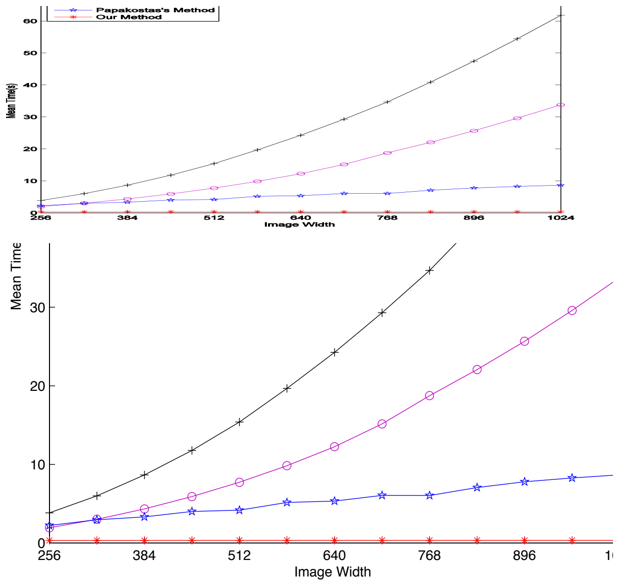


Fig.6. Average computation time for images shown in Fig. 1 and Fig. 3 with varying sizes in the calculation of moments of order up to (40, 40) using nearest interpolation

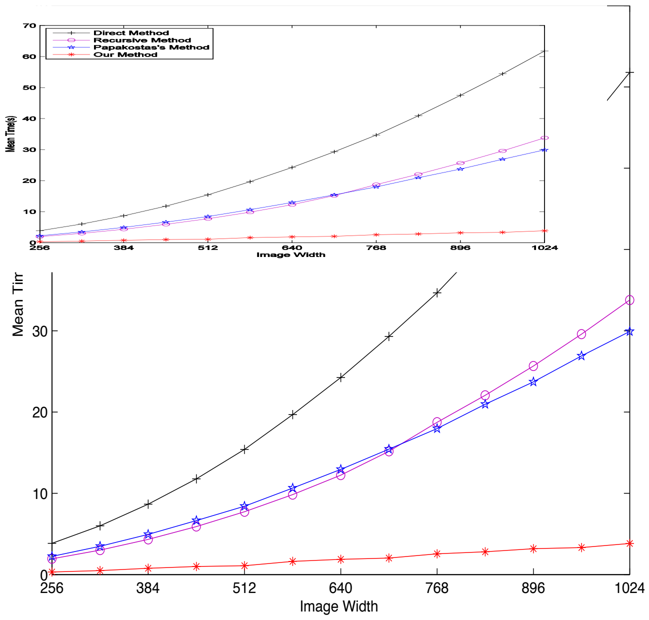


Fig.7. Average computation time for images shown in Fig. 1 and Fig. 3 with varying sizes in the calculation of moments of order up to (40, 40) using bilinear interpolation

TABLE I Computational complexity of the direct method and proposed algorithm for computing the moments of order up to (M–1, M–1) of one block with $J \times J$ pixels

	Additions	Multiplications
Direct method	$M^2(3J^2-1)$	$5M^2J^2$
Recursive method [35]	$M^2(2J^2+2J)$	$M^2(2J^2+3J+1)$
Method resented in [37]	$M^3/3+M^2+2M/3$	$(M+1)(M+2)(M^2+7M+24)/24$
Papakostas's method [38]	$M^2(4J-2)$	$M^2(4J+1)$
Proposed method	M^2+5M	$2M^2+9M$