2015 2nd International Conference on Information Science and Control Engineering

Algorithms for Computing Zernike Moments and Image Reconstruction in Parallel Process

An-Wen Deng[1] Chia-Hung Wei[1,2,*] Chih-Ying Gwo[1]

[1]Department of Information Management, Chien Hsin University of Science and Technology, Taoyuan, Taiwan

[2]Graduate Institute of Biomedical Informatics, Taipei Medical University, Taipei, Taiwan

awdeng@uch.edu.tw rogerwei@uch.edu.tw ericgwo@uch.edu.tw

Abstract

Zernike moments can be computed in several ways. This paper proposes several coefficient methods based on the recursive relations among multinomial coefficients to compute the Zernike moments. In addition, it proposes a parallel algorithm based on the q-recursive method for image reconstruction. The experimental results show that it takes only 3.965 seconds to reconstruct an image measuring 512×512 from all Zernike moments up to moment order 150.

Keywords: Zernike moments, finite groups, parallel computing

1. Zernike Moments

The Zernike polynomials V_{nm} are a family of one-dimensional complex functions defined over the unit circle D. The Zernike polynomials are based on an orthogonal design for the L^2 function space over the unit circle. For a complex number $z = x + iy$, where x, y are real numbers and $i = \sqrt{-1}$, the Zernike polynomial is given by

$$V_{nm}(z) = R_{nm}(r)e^{im\theta} = R_{nm}(r)(\cos(\mathrm{m}\theta) + i\sin(\mathrm{m}\theta)) \tag{1}$$

where $R_{nm}(r)$ is the Zernike radial polynomial in radius, $r = |z| = \sqrt{x^2 + y^2}$ is the vector length, n is a negative integer, called the order, and m called the repetition, is an integer for which $|m| \le n$, $n - m$ is an even number, and θ is the angle between the vector and the x-axis counterclockwise. The definition of the radial polynomial $R_{nm}(r)$ is given by

$$R_{nm}(r) = \sum_{j=0}^{\frac{n-|m|}{2}} (-1)^j \frac{(n-j)!}{j!(\frac{n+|m|}{2} - j)!(\frac{n-|m|}{2} - j)!} \tag{2}$$

For an even integer N, an image with $N \times N$ pixels is assumed to be embedded into the unit circle D. Let the image pixel data be stored in a two-dimensional table $P(s,t)$ as a function over the square $A: [0, N-1] \times [0, N-1]$. Its circumscribed circle is centered at $({}^{N-1}\!/_2, {}^{N-1}\!/_2)$ with radius $N/\sqrt{2}$. The (s,t)-pixel is projected onto a grid centered at

$$(x,y) = (\frac{2s - N + 1}{N\sqrt{2}}, \frac{2t - N + 1}{N\sqrt{2}}). \tag{3}$$

This results in a corresponding function $f(x,y)$ for which $f(x,y) = P(s,t)$. The Zernike moments Z_{nm} can be regarded as the inner product of $f(x.y)$ with the basis function Zernike polynomials $V_{nm}(x,y)$, and are expressed as

$$Z_{nm} = \frac{n+1}{\pi} \iint_{(x,y)\in D} f(x,y)V_{nm}^*(x,y)dxdy \tag{4}$$

where $V_{nm}^*(x,y)$ denotes the complex conjugation of $V_{nm}(x,y)$. The discrete form of Zernike moments in Equation (4) can be expressed as

$$\hat{Z}_{nm} = \frac{2(n+1)}{\pi N^2} \sum\sum_{(s,t)\in A} P(s,t)R_{nm}(r)(\cos(\mathrm{m}\theta) - i\sin(\mathrm{m}\theta)) \tag{5}$$

The conventional method takes computational complexity $O(N^2M^3)$. For a speed-up of computations, one can consider the symmetry under group actions for certain finite groups [1]: for example, the groups of reflections and rotations, such as

$$D_4 = \{\tau_\theta, \iota_\theta \mid \theta = 0, \tfrac{\pi}{2}, \pi, \tfrac{3\pi}{2}\}, V_4 = \{id, \iota_{\pi/2}, \iota_0, \tau_{3\pi/2}\} \tag{6}$$

where τ_θ denotes the rotation θ in counterclockwise and ι_θ the reflection about the line $\theta/2$. As

$$H = \{(s,t) \mid s,t \in [N/2, N-1], s \ge t\}\ , \tag{7}$$

Equation (5) can be modified as follows

$$\begin{aligned}\hat{Z}_{nm} = \frac{2(n+1)}{\pi N^2}(&\sum\sum_{\substack{(s,t)\in H\\ s\ne t}} \sum_{\sigma\in D_4} P(\sigma(s,t))R_{nm}(r)(\cos(\mathrm{m}\sigma(\theta)) - i\sin(\mathrm{m}\sigma(\theta))) \\ &+ \sum\sum_{\substack{(s,t)\in H\\ s=t}} \sum_{\sigma\in V_4} P(\sigma(s,t))R_{nm}(r)(\cos(\mathrm{m}\sigma(\tfrac{\pi}{4})) - i\sin(\mathrm{m}\sigma(\tfrac{\pi}{4}))))\end{aligned} \tag{8}$$

The mapping values of trigonometric functions are linear combinations of $\cos(\mathrm{m}\theta)$ and $\sin(\mathrm{m}\theta)$. As the pixel (s,s) is projected to the diagonal $x = y$, each point on the diagonal $x = y$ has exactly four conjugates by D_4, except for $(x,y) = (0,0)$. In Equation (2) by substituting $k = n - 2j$ [2], we have

$$R_{nm}(r) = \sum R_{nmk} r^k \qquad (\text{ for } k = |m|, |m|+2, \ldots, n\ ) \tag{9}$$

The coefficients in radial polynomials are integers and can be expressed in terms of combinations of binomial coefficients or multinomial coefficients.

$$R_{nmk} = (-1)^{\frac{n-k}{2}} \frac{(\frac{n+k}{2})!}{(\frac{n-k}{2})!(\frac{k+|m|}{2})!(\frac{k-|m|}{2})!} \tag{10}$$

$$= (-1)^{\frac{n-k}{2}} \binom{\frac{n+k}{2}}{\frac{n-k}{2}, \frac{k+|m|}{2}, \frac{k-|m|}{2}} \tag{11}$$

$$= (-1)^{\frac{n-k}{2}} C_k^{\frac{n+k}{2}} C_{\frac{k-|m|}{2}}^k \tag{12}$$

Using Equation (4), one has

*corresponding author

$$\hat{Z}_{nm} = \frac{2(n+1)}{\pi N^2} \sum_{\substack{k=|m| \\ k-|m|:even}}^{n} R_{nmk}\chi(k,m) \quad (13)$$

where

$$\chi(k,m) = \sum\sum_{(s,t)\in A} P(s,t) r^k (\cos(\mathrm{m}\theta) - i\sin(\mathrm{m}\theta)) \quad (14)$$

Using Equation (10) and (6), a fast way to compute all Zernike moments up to order M is as follows:

(1) develop the methods to compute all coefficients R_{nmk}, which contribute to time complexity of $O(M^3)$.
(2) compute all $\chi(k,m)$ (for *k=0,1,2,...,M, m=0,1,2,...,M*) which contribute to time complexity of $O(N^2M^2)$.
(3) put the computed values R_{nmk} and $\chi(k,m)$ into Equation (13), which takes time complexity of $O(N^2M)$.

The computations above take time complexity of $O(\max(N^2M^2, M^3))$. The exponential function r^k, (for $k = 1, 2, ..., M$), for a fixed pixel (s,t) can be computed as follows.

Algorithm A:

```
    $r^0 \leftarrow 1$
    $r^2 \leftarrow ((2s-N+1)^2 + (2t-N+1)^2)/(2N^2)$
    $r \leftarrow \sqrt{r^2}$
    For k $\longleftarrow$ 3 to M step 1
        $r^k \leftarrow r^{k-2} \cdot r^2$
    Next k
```

The values are stored in a table of space complexity $O(M)$. For a fixed pixel (s,t), the computations of trigonometric functions for $\cos(\mathrm{m}\theta)$, $\sin(\mathrm{m}\theta)$ ($m = 0,1,2,...,M$) can be executed as follows.

Algorithm B:

```
    $\cos 0 \leftarrow 1$
    $\sin 0 \leftarrow 1$
    $\cos\theta \leftarrow x/r$
    $\sin\theta \leftarrow y/r$
    $\tan\theta \leftarrow (2t-N+1)/(2s-N+1)$
    $\cos(2\theta) \leftarrow (1-\tan^2\theta)/(1+\tan^2\theta)$
    $\sin(2\theta) \leftarrow 2\tan\theta/(1+\tan^2\theta)$
    For m $\longleftarrow$ 3 to M step 1
        $\cos(\mathrm{m}\theta) \leftarrow \cos((\mathrm{m}-2)\theta)\cos(2\theta) - \sin((\mathrm{m}-2)\theta)\sin(2\theta)$
        $\sin(\mathrm{m}\theta) = \sin((\mathrm{m}-2)\theta)\cos(2\theta) + \sin(2\theta)\cos((\mathrm{m}-2)\theta)$
    Next m
```

The values are stored in tables, which contributes to the space complexity $O(M)$. For even numbers m and k, the values of $\cos(\mathrm{m}\theta)$, $\sin(\mathrm{m}\theta)$ and r^k are all rational numbers. The computational errors are fully controlled.

If one lists the rational numbers of $\tan\theta = (2t-N+1)/(2s-N+1)$ (for $s,t = {}^{N}\!/_{2}, {}^{N}\!/_{2} +1,...,N-1$ with $s \ge t$) in ascending order, one obtains a sequence of reduced fractions, for which numerators and denominators are all odd numbers. This constitutes a subsequence of Farey sequence (of order N)[3].

2. Coefficient Methods

Apart from directly using the formula in terms of factorials, one can use the Pascal triangle to pre-calculate the binomial coefficients to obtain more accurate numerical results. Here, we develop a coefficient method, which is based on the following recursive relation among multinomial coefficients.

Theorem A:
Let the monomial coefficient be given by $\begin{pmatrix} n \\ a,\ b,\ c \end{pmatrix} = \frac{n!}{a!b!c!}$, then

$$\begin{pmatrix} n \\ a,\ b,\ c \end{pmatrix} = \begin{pmatrix} n-1 \\ a-1,\ b,\ c \end{pmatrix} + \begin{pmatrix} n-1 \\ a,\ b-1,\ c \end{pmatrix} + \begin{pmatrix} n-1 \\ a,\ b,\ c-1 \end{pmatrix} \quad (15)$$

Proof. Let x be a specific object from these n objects. One can put x into group 1, or group 2 or group 3. So, the number for putting x into group 1 is $\begin{pmatrix} n-1 \\ a-1,\ b,\ c \end{pmatrix}$; the number for putting x into group 2 is $\begin{pmatrix} n-1 \\ a,\ b-1,\ c \end{pmatrix}$; the number for putting x into group 3 is $\begin{pmatrix} n-1 \\ a,\ b,\ c-1 \end{pmatrix}$. These three cases are mutually exclusive. By the addition principle, one can obtain the desired identity. Q,E.D.

Theorem B:
The coefficients R_{nmk} in radial polynomials satisfy the following properties:
(1) $R_{kkk} = 1$ ($k = 0,1,....,M$); $R_{n00} = (-1)^{\frac{n}{2}}$ for even n's.

(2) $R_{n,|m|,k} = R_{n-1,|m-1|,k-1} + R_{n-1,|m+1|,k-1} - R_{n-2,|m|,k}$ for $n \ge 2$, $k \ge 1$.

(3) $R_{n,|m|,k} = R_{n-1,|m-1|,k-1} \times (n+k)/(|m|+k)$.

Four different coefficient methods are taken into consideration:

(C1) use factorials to pre-compute and then take the calculated values into the definition of R_{nmk}
(C2) use the recursive monomial relation in Theorem C(2)
(C3) use the multiplicative recursion in Theorem C(3)
(C4) use pre-computations of Pascal triangle identities and then take the calculated values into Equation (12) expressed in terms of binomial coefficients.

Method 1 needs a table of space complexity $O(M)$ for storing the factorials $n!$. Method 4 needs a two-dimensional table for storing the binomial coefficients C_k^n which has a space complexity of $O(M^2)$. Because of pre-computations, method 1 and method 4 can be implemented in parallel algorithms. For illustration, we present a parallel algorithm for method 4 as follows.

Algorithm C:

```
    $C_0^0 \leftarrow 1$
    For i=1 to M step 1
        $C_0^i \leftarrow 1$
        $C_i^i \leftarrow 1$
        For j=1 to $\lfloor i/2 \rfloor$ step 1
            $C_j^i \leftarrow C_j^{i-1} + C_{j-1}^{i-1}$ /*Pascal triangle identity*/
            $C_{i-j}^i \leftarrow C_j^i$
        Next j
    Next i
    Parallel for n $\leftarrow$ 0 to M step 1
        For m $\leftarrow$ n to 0 step -2
            For k $\leftarrow$ n to m step -2
                $R_{nmk} \leftarrow (-1)^{\frac{n-k}{2}} C_k^{(n+k)/2} C_{(k-m)/2}^k$
            Next k
        Next m
    End parallel for
```

In the parallel for the loop, the set of n is divided into disjoint subsets; and all multithreads run over their allocated subsets and independently. For a number of $O(1)$ multithreads, all four methods for computing all coefficients $R_{nmk} = R[n][m][k]$ for $n \le M$ need time complexity $O(M^3)$ and space complexity $O(M^3)$. Method 2 is implemented in a serial algorithm as follows.

```
Algorithm D:
    For $n \leftarrow 0$ to $M$ step 1
        $R[n][n][n] \leftarrow 1$
        For $m \leftarrow n-2$ to 1 step -2
            $R[n][m][n] \leftarrow R[n-1][m-1][n-1]$
                        $+R[n-1][m+1][n-1]$
            For $k \leftarrow n-2$ to $m$ step -2
                $R[n][m][k] \leftarrow R[n-1][m-1][k-1]$
                        $+R[n-1][m+1][k-1] - R[n-2][m][k]$
            Next k
        Next m
        If m=0 then
            $R[n][0][n] \leftarrow 2R[n-1][1][n-1]$
            For $k \leftarrow n-2$ to 1 step -2
                $R[n][0][k] \leftarrow 2R[n-1][1][k-1] - R[n-2][0][k]$
            Next k
            If $R[n][0][2] > 0$ then
                $R[n][0][0] \leftarrow -1$
            Else
                $R[n][0][0] \leftarrow 1$
            End if
        End if
    Next n
```

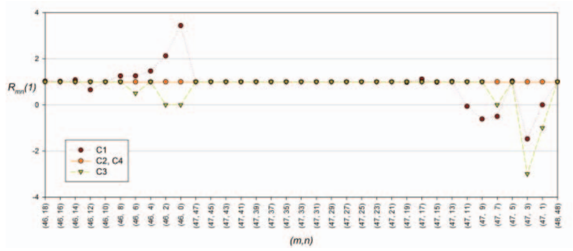


Figure 1: Computation values for $R_{nm}(1)$ of different coefficient methods.

The accuracy of computation values for Zernike radial polynomials plays an important role in computing Zernike moments. Figure 1 shows the computation values for Zernike radial polynomial $R_{nm}(1)$ of order n with repetition m by using coefficient methods. It is known that $R_{nm}(1) = 1$ for all n, m. The computation values in use of method C2 and method C4 are all correct for order $n \le 47$; The computation values by using method C1 and method C3 have significant errors for some $R_{nm}(1)$,where $n \ge 46$. All of computations use the 64-bit double-precision float format. Using the 64-bit long-precision integer format, the method C2, C3 and method C4 can obtain the accurate values for coefficients R_{nmk} , where $n \le 54$. The coefficients in the radial polynomials grow as the order n grows.

Figure 2 shows the function graphs for Zernike radial polynomials $R_{54,0}(r)$ and $R_{54,2}(r)$. The plotting for these graphs is based on the q-recursive relation introduced by Chong et al[4]. One can see that the graphs oscillate in such a way that as r gets closer to one, the difference of x coordinates between two consecutive critical points gets smaller. For large n , the computational error for computing Zernike radial polynomials by using coefficient method near $r = 1$ can be fairly huge.

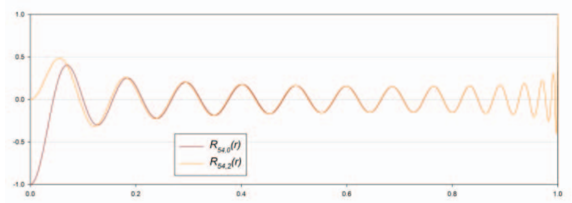


Figure 2: Graphs of Zernike radial polynomials $R_{54,0}(r)$ and $R_{54,2}(r)$..

3. q-Recursive Method

We reproduce the q-recursive relations among the Zernike radial polynomials of the same order introduced by [4, 5].

$$R_{nm}(r) = K_1 R_{n,m+4}(r) + (K_2 + \frac{K_3}{r^2}) R_{n,m+2}, m = n-4, n-6, ..., 1 (or \quad 0) \quad (16)$$

with the initial conditions:

$$R_{nn}(r) = r^n, n \ge 0, \quad (17)$$

$$R_{n,n-2}(r) = nr^n - (n-1)r^{n-2}, n \ge 2 \quad (18)$$

and the coefficients satisfy

$$K_1 = \frac{(m+4)(m+3)}{2} - (m+4)K_2 + \frac{K_3(n+m+6)(n-m-4)}{8}, \quad (19)$$

$$K_2 = \frac{K_3(n+m+4)(n-m-2)}{4(m+3)} + (m+2) \quad (20)$$

and $K_3 = \frac{-4(m+2)(m+1)}{(n+m+2)(n-m)}$, (for $n \ge 4$ and $m \le n-4$) (21)

Using the q-recursive method to compute all Zernike radial polynomial $R_{nm}(r)$ for $n \le M$ needs time complexity $O(M^2)$, where the coefficient method needs time complexity $O(M^3)$. Using the q-recursive method to approximate all Zernike moments Z_{nm} of order $n \le M$ needs time complexity $O(N^2 M^2)$. With the q-recursive method computational values for $R_{nm}(1)$ are all equal to one. The accuracy of the q-recursive method in computing Zernike radial polynomials is the best among the compared methods.

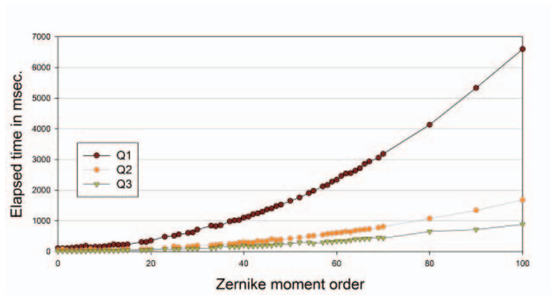


Figure 3: Computation time using q-recursive methods with different speedups to compute the Zernike moments for an image of 512×512 pixels.

Figure 3 shows the elapsed times with the q-recursive method with different speedups to compute the Zernike moments for an image of 512×512 pixels. Q1 denotes the q-recursive method; Q2 denotes the q-recursive method with a speedup using V_4 symmetry; and Q3 denotes the q-recursive method with a speedup using D_4 symmetry. All of these methods use pre-computations for the constants of K_1, K_2 and K_3, exponential functions r^k as in Algorithm A and the trigonometric functions $\cos(m\theta)$, $\sin(m\theta)$ as in Algorithm B. We implement those methods in C/C++ serial programs with double-precision float format and use the Microsoft Visual Studio 2013 C++ 64-bit release compiler. The computer is installed with OS win7 with the processor Intel Pentium B980 dual core which supports two multithreads.

4. Experimental Results and Discussion

To evaluate both the accuracy and the speed of the proposed method, this study reconstructed an image from its Zernike moments. This experiment used the q-recursive method and a speedup by V_4 symmetry and parallelization as follows.

(A) We compute all coefficients of K_1, K_2, K_3 in Equation (19),(20),(21) for $n = 4,5,....,M$, $m = n-4, n-6,....,1(or\ 0)$, and then store them in table of space complexity $O(M^2)$.

(B) Parallel for each pixel $(s,t) \in A \cap \{(s,t) \mid N/2 \le s,t \le N-1\}$:

```
B1. $(r,\theta) =$ polar coordinates for $(x,y) = (\frac{2s-N+1}{N\sqrt{2}}, \frac{2t-N+1}{N\sqrt{2}})$
B2. Compute $r^k$ (for $k = 0,1,...,M$ ) as in Algorithm A
B3. Compute $\cos(m\theta), \sin(m\theta)$ (for $m = 0,1,...,M$ ) as in Algorithm B
B4. For n=0 to M step 1
        For m=n to 2 step -2
B4.1. Compute $R_{mn}(r)$ as in Equation (16),(17),(18) by q-recursive method
B4.2. $\hat{f}(x,y) \leftarrow \sum_{n=0}^{M}(Z_{n0}R_{n0}(r) + 2\sum_{m=0, m-n=even}^{n} R_{nm}(r)(\mathrm{Re}(Z_{nm})\cos(m\theta) - \mathrm{Im}(Z_{nm})\sin(m\theta)))$   (22)
B4.3. $\hat{f}(-x,y) \leftarrow (-1)^n\sum_{n=0}^{M}(Z_{n0}R_{n0}(r) + 2\sum_{m=0, m-n=even}^{n} R_{nm}(r)(\mathrm{Re}(Z_{nm})\cos(m\theta) + \mathrm{Im}(Z_{nm})\sin(m\theta)))$   (23)
B4.4. $\hat{f}(x,-y) \leftarrow \sum_{n=0}^{M}(Z_{n0}R_{n0}(r) + 2\sum_{m=0, m-n=even}^{n} R_{nm}(r)(\mathrm{Re}(Z_{nm})\cos(m\theta) + \mathrm{Im}(Z_{nm})\sin(m\theta)))$   (24)
B4.5. $\hat{f}(-x,-y) \leftarrow (-1)^n\sum_{n=0}^{M}(Z_{n0}R_{n0}(r) + 2\sum_{m=0, m-n=even}^{n} R_{nm}(r)(\mathrm{Re}(Z_{nm})\cos(m\theta) - \mathrm{Im}(Z_{nm})\sin(m\theta)))$   (25)
        Next m
    Next n
End parallel for
```

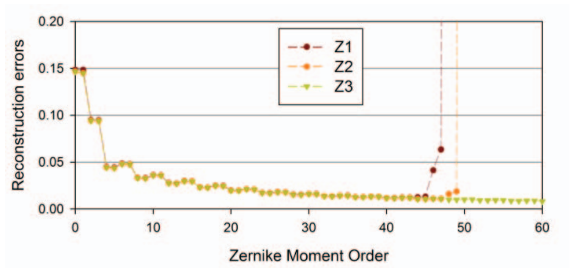


Figure 4: Errors for image reconstruction using different algorithms computing Zernike moments.

Figure 4 shows the errors (normalized mean square error) defined by ε

$$\varepsilon = \frac{\iint_D \left| f(x,y) - \hat{f}(x,y) \right| 2dxdy}{\iint_D \left| f(x,y) \right|^2 dxdy} \tag{26}$$

for the image reconstruction from computed Zernike moments obtained by different algorithms. Z1 denotes the fast algorithm Equation (13) by using the coefficient method in Theorem B(2) and using D_4 symmetry, for which the reconstruction error attains its minimum $\varepsilon = 0.0117$ at the order 41; Z2 denotes the conventional algorithm Equation (8) by using the coefficient method in Theorem B(2) and using D_4 symmetry, for which the reconstruction error $\varepsilon = 0.0105$ attains its minimum at the order 45; Z3 denotes the q-recursive method using D_4 symmetry, for which the reconstruction error attains its minimum $\varepsilon = 0.00149682$ at the order 429. Some experiment results are shown in Table 1.

Figure 5: An image with 512x512 pixels is used for experiments.

Table 1: Reconstructed images using different methods and up to different Zernike moment orders.

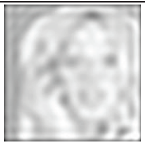
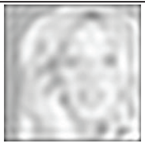
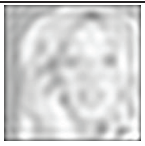
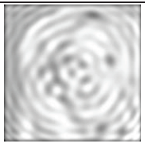
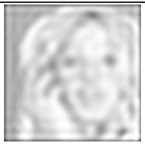
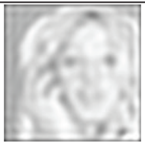
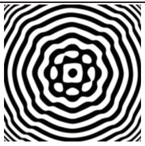
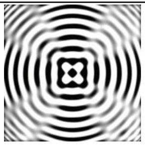
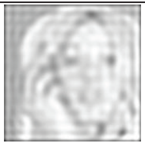
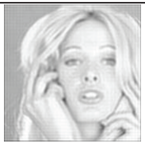
Zernike Moment Order	Z1	Z2	Z3
40	 $\varepsilon = 0.0117726$	 $\varepsilon = 0.0116768$	 $\varepsilon = 0.0116768$
45	 $\varepsilon = 0.0130307$	 $\varepsilon = 0.0105453$	 $\varepsilon = 0.0105463$
50	 $\varepsilon = 79.8521$	 $\varepsilon = 1.61971$	 $\varepsilon = 0.0101561$
150	not applicable	not applicable	 $\varepsilon = 0.00366569$

Table 2: Elapsed times in msec. of image reconstruction using parallel algorithm with two threads/single thread

Order	2 threads	single thread	ratio	Order	2 threads	single thread	ratio
0	91	159	0.57233	130	3118	5743	0.54292
1	91	170	0.53530	140	3492	6643	0.52567
2	93	172	0.54070	150	3965	7764	0.51070
5	133	186	0.71505	160	4700	8787	0.53488
10	147	210	0.70000	170	5132	10214	0.50245
15	169	250	0.67600	180	6886	12146	0.56694
20	192	315	0.60952	190	7811	13393	0.58322
25	228	393	0.58015	200	8002	15201	0.52641
30	270	482	0.56017	250	13220	27282	0.48457
35	320	581	0.55077	300	19116	38460	0.49704
40	379	691	0.54848	350	26360	52498	0.50211
45	460	837	0.54958	360	27880	54092	0.51542
55	603	1172	0.51451	370	29613	64369	0.46005
60	700	1353	0.51737	380	30895	58642	0.52684
70	910	1757	0.51793	390	37237	60819	0.61226
80	1145	2235	0.51230	400	38167	65153	0.58581
90	1574	2827	0.55677	410	40034	67721	0.59116
100	2005	3467	0.57831	420	40879	69973	0.58421
110	2228	4160	0.53558	429	42083	72467	0.58072
120	2618	4910	0.53320				

Our proposed parallel algorithm is also implemented with C/C++ with the openMP library. The openMP [6] uses the fork-join model to handle the parallel mechanism. Some experimental results are shown in Table 2. The table shows the elapsed times in msec. for a reconstruction parallel algorithm with two threads and a single thread.

5. Conclusions

Depending on the specific purpose, one can use fast algorithms or accurate ones to compute Zernike moments. This paper discusses several ways to compute Zernike moments. For a direct method, we proposed three different coefficients to eliminate computing of factorials. These proposed coefficient methods provide more accurate results than direct methods using factorials. Direct methods can be divided into two categories: one is the conventional way and the other is the fast way. According to the experimental results, with the conventional way one can obtain better image reconstructions. For image reconstruction, we proposed a parallel algorithm based on the q-recursive method, implemented with C/C++ with the openMP library. The experimental results show that our proposed reconstruction method is fast; it takes only 3.965 seconds on a laptop for reconstruction of an image 512x512 from all Zernike moments up to moment order 150.

References

1. Li, S. and J.P. Boyd, Symmetrizing grids, radial basis functions, and Chebyshev and Zernike polynomials for the symmetry group; Interpolation within a squircle, Part I. Journal of Computational Physics, 2014. **258**: p. 931-947.

2. Mukundan, R. and K.R. Ramakrishnan, Fast computation of Legendre and Zernike moments. Pattern Recognition, 1995. **28**(9): p. 1433-1442.

3. Apostol, T.M., Modular Functions and Dirichlet Series in Number Theory Graduate Texts in Mathematics. 1997: Springer; 2nd edition (May 22, 1997).

4. Chong, C.-W., P. Raveendran, and R. Mukundan, A comparative analysis of algorithms for fast computation of Zernike moments. Pattern Recognition, 2003. **36**(3): p. 731-742.

5. Singh, C. and E. Walia, Algorithms for fast computation of Zernike moments and their numerical stability. Image and Vision Computing, 2011. **29**(4): p. 251-259.

6. Barbara Chapman, Gabriele Jost and Ruud van der Pas Using OpenMP: Portable Shared Memory Parallel Programming (2008).