Optical Engineering

OpticalEngineering.SPIEDigitalLibrary.org

Fast digital image correlation using parallel temporal sequence correlation method

Chen Xiong
Jiatao Chen
Feng Li
Ming Cai

SPIE.

Chen Xiong, Jiatao Chen, Feng Li, Ming Cai, "Fast digital image correlation using parallel temporal sequence correlation method," *Opt. Eng.* **58**(12), 124101 (2019), doi: 10.1117/1.OE.58.12.124101.

Fast digital image correlation using parallel temporal sequence correlation method

Chen Xiong,[a,b] Jiatao Chen,[a,b] Feng Li,[c] and Ming Cai[a,b,*]
[a]Sun Yat-sen University, School of Intelligent Systems Engineering, Guangzhou, China
[b]Sun Yat-sen University, Guangdong Provincial Key Laboratory of Intelligent Transportation System, Guangzhou, China
[c]Guangdong Polytechnic Normal University, School of Automotive and Transportation Engineering, Guangzhou, China

Abstract. Digital image correlation (DIC) is a noncontact technique that is widely used for deformation measurement, but improving the calculation efficiency to achieve real-time DIC calculation has always been a big concern. A parallel temporal sequence DIC method is proposed, which chooses seed points to determine the integer-pixel displacement and applies the moving least-squares fitting technique to acquire the subpixel displacement. This method avoids traditional complex iterations and takes full advantage of the GPU parallel computing. Results of a simulation experiment and an actual experiment demonstrate the accuracy and efficiency of the proposed algorithm. The calculation speed in the simulation experiment of the proposed method achieved 463,320 POI/s, whereas the speed in the actual experiment was 432,866 POI/s, when the speed of the ICGN method was 2700 POI/s and 2074 POI/s under the same accuracy, respectively. Also, the subpixel displacement calculation made up less than 1% of the entire calculation. The computational efficiency could be further enhanced if a faster integer-pixel displacement calculation method is discovered or a parallel algorithm is used. [DOI: 10.1117/1.OE.58.12.124101]

Keywords: digital image correlation; graphics processing unit; parallel computing; moving least-squares fitting.

Paper 191358 received Sep. 27, 2019; accepted for publication Nov. 15, 2019; published online Dec. 5, 2019.

1 Introduction

Due to its incomparable advantages, such as low environment requirement and easy data processing, digital image correlation (DIC)[1,2] has been applied in many noncontact measurement areas.[3–6] Now with the development of high-speed cameras and computer performance, improving the computation efficiency of the DIC method has become a major problem. Continuous improvements have been made to the DIC algorithm.

Originally, the Newton–Raphson[7,8] algorithm was widely accepted and used to determine the subpixel displacement due to its high accuracy. This method requires several iterations to optimize coefficients and to get the correlational function maximum. Later, the inverse compositional Gauss–Newton[9,10] algorithm was proposed, omitting the repeated calculation of Hessian matrix and greatly improving the computation efficiency. Also, a reliability-guided displacement scanning strategy and a precomputed global interpolation coefficient look-up table are employed to accelerate the integer-pixel displacement searching and subpixel displacement calculation, respectively.[11] The calculation of a reliability-guided digital image correlation (RGDIC) method begins with a seed point and is then guided by ZNCC coefficients of the computed points.[9] This ensures that the calculation path is always along the most reliable direction, and error propagation is avoided. Then, a seed point-based parallel method was proposed to improve the calculation speed, which can maximize the computing speed using an improved initial guess.[12]

In addition, the parallel computing based on graphics processing units (GPU)[13,14] has been found suitable for image processing, including in the DIC field. The compute unified device architecture (CUDA) as a parallel computing platform, developed by NVIDIA, has been widely used due to its flexibility. Through programming, GPU can be assigned different computational tasks. Points of interest can be divided into many groups, and points in a single group can be calculated at the same time. Considering that the path-dependent tracking strategy wastes the parallel computing power of modern computers with multicore processors, the RGDIC method is improved by using a two-section tracking scheme.[15] These calculated points with correlation coefficients higher than a preset threshold are taken as reliable computed points and are given the same priority to extend the correlation analysis to their neighbors. Thus, DIC calculation can be initially executed in parallel at multiple points by separate independent threads. A parallel DIC (paDIC) method powered by GPU parallel computing is proposed.[16] This method combines the IC-GN algorithm for subpixel registration with a path-independent fast Fourier transform-based cross correlation algorithm for integer-pixel initial guess estimation, achieving a superior computation efficiency over the DIC method, purely running on the CPU. Based on paDIC method, a pipelined system framework unifying five variants of combinations of CPU and GPU was proposed, which can be flexibly applied to various practical applications with different requirements of measurement scales and speeds.[17] A fast method was introduced for estimation of dense two- and three-dimensional displacement fields from image correlation. It is based on a local, or window-based, optical flow algorithm, which is ideally suited for parallel processors. This method adopted a coarse-to-fine strategy on a multiresolution image pyramid helping to propagate a good initialization for the optimization throughout the scales; hence, convergence could be reached even for

*Address all correspondence to Ming Cai, E-mail: caiming@mail.sysu.edu.cn

displacements significantly larger than the subset size.[18] However, the repeated iteration is not avoided and is only simplified through these methods.

A DIC algorithm combining the spatial correlation with temporal continuity is proposed.[19] The time-consuming iteration operation during subpixel displacement calculation is replaced by the fitting method based on the moving least-squares technique. Based on this method, a parallel temporal sequence DIC method is proposed. A fast integer-pixel searching algorithm based on the seed points is used to calculate the integer-pixel displacement. The integer-pixel displacement results of seed points are calculated along the time axis, and these results are taken as the initial displacement guess of other points. Thus, the searching field of other points can be greatly reduced and calculated together by separate independent threads. The fitting method based on the moving least-squares technique[20] is applied to determine the subpixel displacement, avoiding complex iterations and improving computation efficiency. Also, the fitting process of every point is fully independent, without any hinderance from the neighboring points and the overlapping subsets, such that the advantage of GPU parallel computing can be exploited.

In this paper, GPU parallel computing is used to speed up a temporal sequence DIC method, which chose seed points to determine the integer-pixel displacement and applied the moving least-squares fitting technique to acquire the subpixel displacement. The result shows a super-fast calculation speed under the equivalent precision.

2 Principle of the Parallel Temporal Sequence DIC Method

The parallel temporal sequence DIC algorithm is divided into three steps, and it mainly considers a continuous deformation process. The first step is to calculate the integer-pixel displacement of seed points along the time axis. The second step is to use the results of seed points as the initial values to determine the integer-pixel displacement of other points, and the calculation of other points takes the advantage of parallel computing to accelerate. At last, the moving least-squares technique is applied to determine the subpixel displacement, also using parallel computing.

2.1 *Integer-Pixel Displacement Calculation*

In order to decrease the computation time, the integer-pixel displacement of seed point at every deformation moment is calculated in advance. Along the time axis, the result of the last moment is also taken as the initial guess of the current moment considering the temporal continuity. As Fig. 1 shows, the point P is chosen as the seed point. In each speckle image, the rectangles in different colors stand by the search field, and the search of every moment is centered on the result of the last moment. Since the deformation is continuous in space, once the integer-pixel displacement of one point $p(x, y)$ is acquired, the searching scale of its neighboring points $p_n(x_n, y_n)$ can be reduced greatly. The searching scales of the neighboring points are depended on the complexity of the deformation field. Considering the large deformation and deformation discontinuity in space, more seed points are needed, so the searching scale of neighboring points can be much smaller.

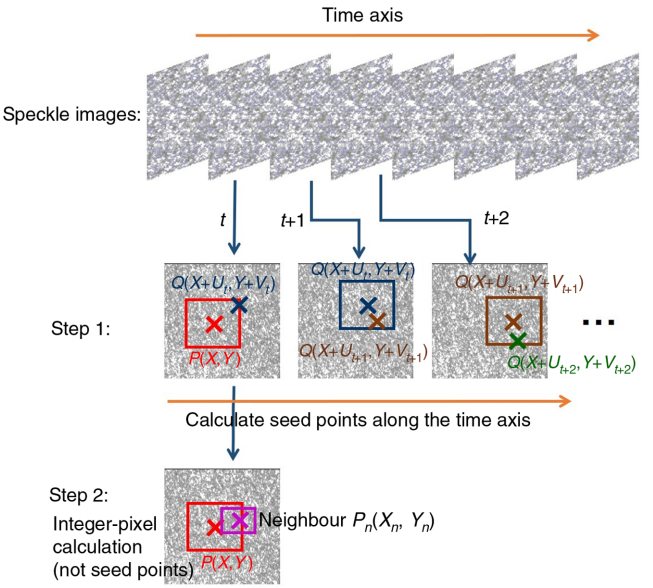

Fig. 1 Integer-pixel displacement calculation schematic diagram.

It is remarkable that the method is not suitable when there is a jump during the deformation process. Large deformation gradient will cause a large computation error, but this kind of computation error can be effectively decreased or avoided by increasing the search scale or by improving the exposure speed of the camera.

Other calculation points usually are chosen in a checkerboard pattern (if the surface of the specimen is irregular such as a hole, the calculation points can then be chosen according to the real situation). The whole points in different locations and in different deformation moments are calculated together. However, if the grid is too tight, the data conflict will still exist when reading gray data from the same speckle image. The strategy is to move the calculation grid. At first, a suitable vertical spacing and horizontal spacing are chosen to achieve a reasonable grid layout to avoid the data conflict, and then the grid is separately moved vertically and horizontally. Thus, the grid can now become tighter within several iterations and all the points of interest can be calculated.

Once the calculation task was determined, the CUDA programming is used to finish the job. As Fig. 1 shows, a multicore GPU usually consists of multiple steaming multiprocessors (SMs), and each SM includes a certain number of steaming processors (SPs). In the CUDA programming, the calculation task was assigned to grids of blocks, and each block contains a number of threads. In order to ensure maximum utilization of the GPU resource and hiding the latency effectively, the number of threads usually required is more than 64 (in this case around 512), and the number of blocks cannot be too small either,[9] which is according to the quantity of the speckle image analyzed once (in this case around 100). During the calculation process, a kernel consisting of a series of data and instructions is assigned to blocks and then distributed to threads, which run on every SP independently within each block.

2.2 *Subpixel Displacement Calculation*

Figure 2 shows the subpixel displacement calculation process at one point. A weighted moving least-squares method is

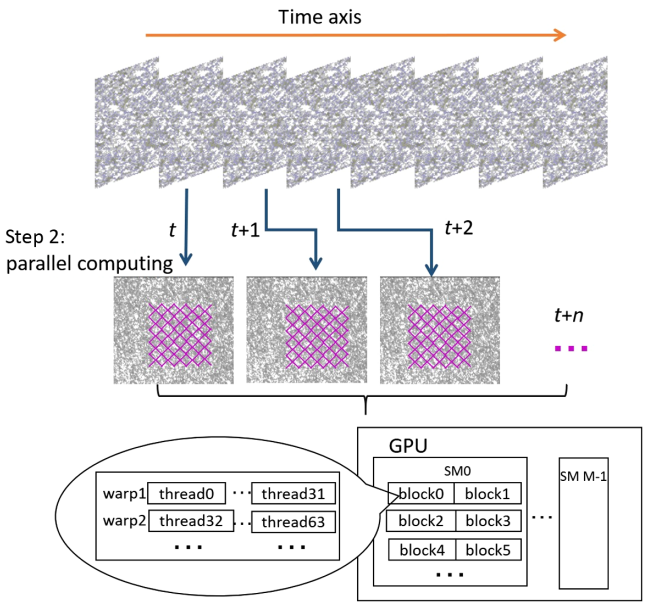

Fig. 2 Weighted moving least-squares method schematic diagram.

used to fit the integer-pixel displacement along the time axis to obtain the subpixel displacement. The influence domain is defined by the central point and the influence radius, and the central point is the calculated point. A cubic polynomial is adopted as the basic function. However, the integer-pixel displacements happen to be stair-stepping due to the neighboring results being the same. In order to overcome the stair-stepping effect and improve the fitting accuracy, the integer-pixel displacement is weighted according to the correlation coefficient in advance. Previous studies have generally considered all integer-pixel displacements to have equal precision, hence the least squares algorithm is employed. However, accuracy differs among points in practice. The correlation coefficient is a simple representation of reliability, where larger correlation coefficient indicates that the integer-pixel displacement is a better approximation of the actual displacement. Therefore, a weighted function is generated based on the correlation coefficient, such that a point's weight is larger for larger correlation coefficient, and vice versa. The weighted fitting principle is described as follows:

$$J(a_0, a_1, \ldots, a_n) = \sum_{i=1}^{m} k_i (f - y)^2$$

$$= \sum_{i=1}^{m} k_i (a_0 + a_1 x_i + \ldots + a_n x_i^n - y_i)^2, \quad (1)$$

where $J(a_0, a_1, \ldots, a_n)$ represents the distance function, y represents the integer-pixel displacement results at different moments, and k represents the weight coefficients. The function $f(a_0, a_1, \ldots, a_n)$ is the adopted polynomial, and $a_0, a_1, \ldots, a_n$ represents the unknown parameters. m represents the number of integer-pixel displacement results involved in fitting. In order to get the minimum of $J(a_0, a_1, \ldots, a_n)$, $a_0, a_1, \ldots, a_n$ is substituted into Eq. (2):

$$\frac{\partial J}{\partial a_j} = 2 \sum_{i=1}^{m} k_i (x_i)^j (a_0 + a_1 x_i + \ldots + a_n x_i^n - y_i) = 0, \quad (2)$$

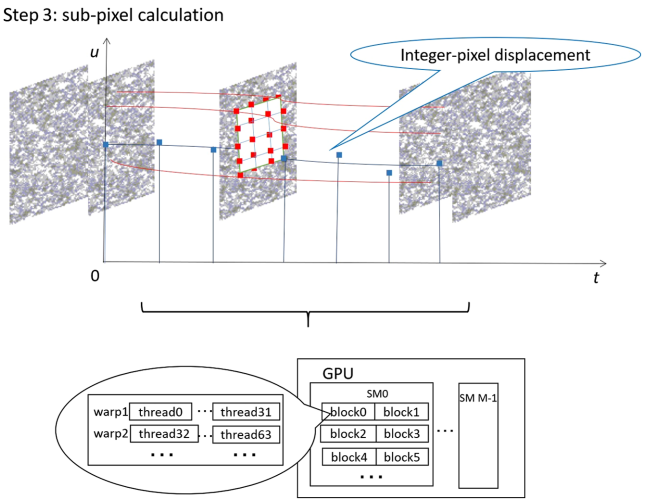

Fig. 3 Subpixel displacement calculation schematic diagram.

$$\begin{bmatrix} \sum_{i=1}^{m} k_i & \sum_{i=1}^{m} k_i x_i & \ldots & \sum_{i=1}^{m} k_i x_i^n \\ \sum_{i=1}^{m} k_i x_i & \sum_{i=1}^{m} k_i x_i^2 & \ldots & \sum_{i=1}^{m} k_i x_i^{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{m} k_i x_i^n & \sum_{i=1}^{m} k_i x_i^{n+2} & \ldots & \sum_{i=1}^{m} k_i x_i^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} k_i y_i \\ \sum_{i=1}^{m} k_i x_i y_i \\ \vdots \\ \sum_{i=1}^{m} k_i x_i^n y_i \end{bmatrix}. \quad (3)$$

The parameters can be obtained according to Eq. (3), then the subpixel displacement of the central point can be calculated. It is remarkable that this method may not work well at the few first moments because the location in the fitting curve of these moments is at the beginning, which may cause large computation error. For example, when the fitting data number is 33, the central number is 16 ($t = 16$), but the first to the fifth moment must be calculated through the same curve ($t = 0, 1, 2, \ldots, 15$) (Fig. 3).

Moreover, the fitting process of different points is fully independent, without any hinderance from the neighboring points and the overlapping subsets, such that the advantage of GPU parallel computing can be exploited. Similar to the integer-pixel displacement calculation, the computation task was divided into many groups and was distributed to every thread within every block. Also, the number of threads required is more than 64 (in this case around 512), and the number of blocks cannot be too small either (in this case around 100).

3 Experimental Verification

The parallel fitting DIC method was programmed using the C++ language based on CUDA 8.0 and was run on a desktop computer equipped with Intel i7-7700 CPU and a GeForce GTX 1080Ti graphics card. A simulation duralumin fatigue tensile experiment was used to illustrate the accuracy and efficiency of the method. As Fig. 4 shows, the speckle image was obtained as the reference image from a real experiment. Other speckle images were generated with preset subpixel

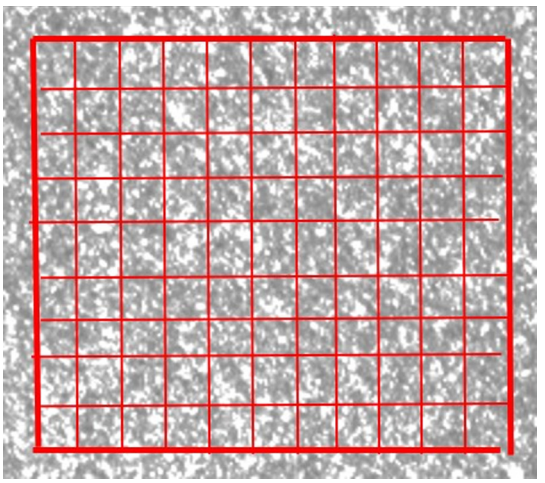

Fig. 4 Simulation speckle image.

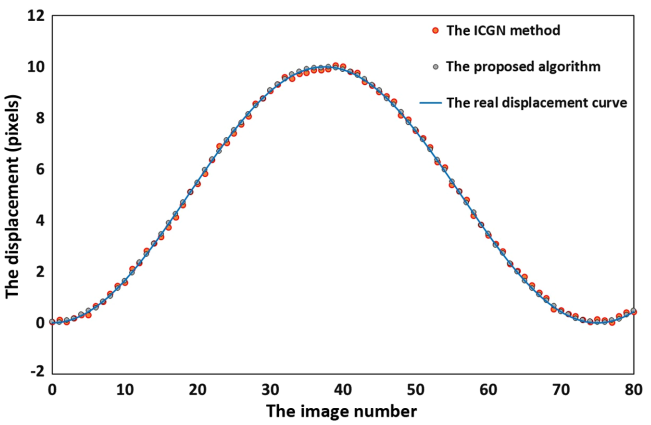

Fig. 5 Displacement result curve.

displacements. The subpixel displacements of the translation experiment including 800 deformation speckle images were set according to Eq. (4) as follows:

$$v(n,h) = 5.0 + 5.0 \cdot \sin\left(\frac{2\pi}{75} \cdot n - \frac{\pi}{2}\right) + \left[1.5 \times 10^{-3} + 1.5 \times 10^{-3} \cdot \sin\left(\frac{2\pi}{75} \cdot n - \frac{\pi}{2}\right)\right] \cdot h, \tag{4}$$

where h is the row number in the pixel matrix for each image, $n(n = 1,2,\ldots,375)$ is the time series number for the image, and the unit for v is pixels. The coefficient in this equation was acquired from the ANSYS simulation result. A model of the duralumin specimen under load was established and simulated in the ANSYS software. The displacement curve was then obtained to calculate the coefficient. Since the plastic stage was relatively complex in the tensile experiment, the results were only calculated for the elastic stage in this study. The computation points were arranged as a checkerboard.

Note that the errors of integer-pixel displacements are all within 1 pixel. In order to evaluate the calculation accuracy and speed of the proposed algorithm, the ICGN method was used to compare. As shown in Fig. 5, the green line was the real displacement curve of a random calculation point along the deformation time axis, and the blue and the red points stood by the calculation results of the proposed algorithm and the ICGN method at each deformation time, respectively. These points agreed well with the actual sine curve, thereby exhibiting that the calculation of the proposed algorithm was correct and highly accurate.

The average error[7] and standard error[7] at every deformation time were used to illustrate the computation accuracy. Figure 6 shows the average error of the proposed algorithm and the ICGN method at different deformation times, and the result showed that the average error of the proposed algorithm is almost at the same level with the ICGN method.

Tables 1 and 2 show that the calculation speed of the proposed algorithm increased when the calculation image number or the calculation point number at a single deformation time increased. With the calculation requirement increasing, the computing units of GPU device can be fully unitized and the calculation speed increases. At last, it will reach the limit, and the speed becomes stable. According to Table 2, the maximum calculation speed achieved 463,320 POI/s (points of interest per second) when the calculation speed of the ICGN method was 2700 POI/s under the same precision, which verified that the proposed algorithm could greatly improve the computation efficiency compared with the ICGN method. Also, the result showed that the subpixel fitting method was suitable for parallel computing, and the subpixel displacement calculation made up less than 1% of the entire calculation.

In order to verify the accuracy and efficiency of the proposed method in the actual experiment, a real low carbon steel tensile experiment was used. The tensile sample size is shown in Fig. 7(a). The loading rate was 0.02 mm/s, and

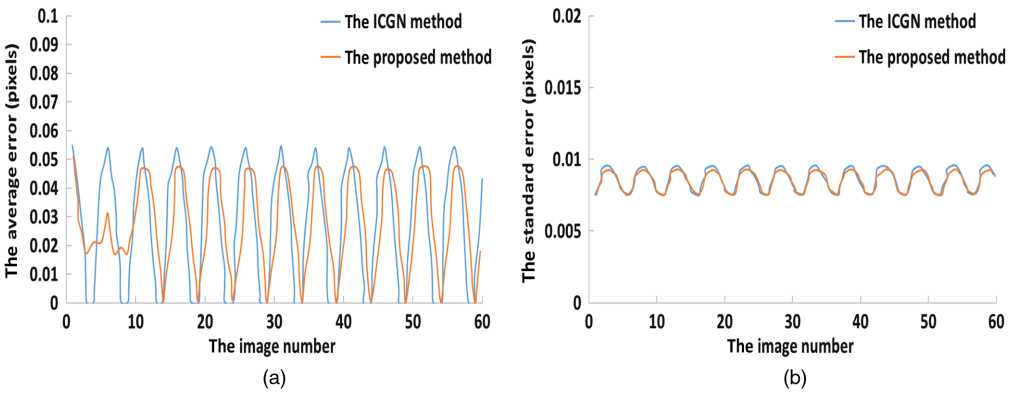

Fig. 6 Comparison between two methods: (a) average error curve and (b) standard error curve.

Table 1 Calculation speed under different calculated image numbers.

Calculation points		Calculation time of the proposed algorithm (ms)				Calculation time of the ICGN method (ms)		
Image number	Total number	Seed points	Integer-pixel	Subpixel	Speed (POI/s)	Seed points	Subpixel	Speed (POI/s)
50	45,000	31	121	6	284,810	62	16,723	2681
100	90,000	78	176	12	338,346	125	35,240	2545
225	202,500	156	387	22	358,407	281	75,567	2670
400	360,000	265	771	42	333,952	515	133,459	2688
625	562,500	437	968	61	383,697	796	211,308	2652

Table 2 Calculation speed under different calculated image numbers.

Calculation points		Calculation time of the proposed algorithm (ms)				Calculation time of the ICGN method (ms)		
Image number	Total number	Seed points	Integer-pixel	Subpixel	Speed (POI/s)	Seed points	Subpixel	Speed (POI/s)
100	10,000	78	78	2	63,291	125	3744	2585
100	40,000	78	117	5	200,000	140	14,833	2663
100	90,000	78	186	11	327,273	125	33,212	2700
100	360,000	78	658	41	463,320	125	135,315	2658

200 speckle images were obtained from the experiment, one of which is shown in Fig. 7(b). Also, the average error[7] and standard error[7] at every deformation time were used to illustrate the computation accuracy. (The real displacement of every point at every deformation time was obtained and calculated from the extensometer.)

Figure 8 shows the average error and standard error of the proposed algorithm and the ICGN method at different deformation times. The result showed that the average error and standard error of the proposed algorithm are little more than those of the ICGN method. But the average error was within 0.04 pixels, whereas the standard error was within 0.03 pixels, which shows that the computation accuracy of the proposed algorithm was at the same level with the accuracy of present algorithms (which were usually within 0.005 to 0.1 pixels).[14]

According to Table 3, in the actual tensile experiment, the calculation speed achieved 432,866 POI/s when the calculation speed of the ICGN method was 2052 POI/s under the same precision, which verified that the proposed algorithm could greatly improve the computation efficiency compared with the ICGN method. Also, in the simulation experiment, the subpixel displacement calculation made up less than 1% of the entire calculation. The computational efficiency could be further enhanced if a faster integer-pixel displacement calculation method is discovered. Moreover, it can be found that the calculation speed of the ICGN method clearly reduced in the actual experiment, whereas the speed of the proposed method almost remained unchanged. In the stage of subpixel displacement calculation using the ICGN method, more iterations are always needed in the actual experiment due to the lower quality speckle image compared with the simulation

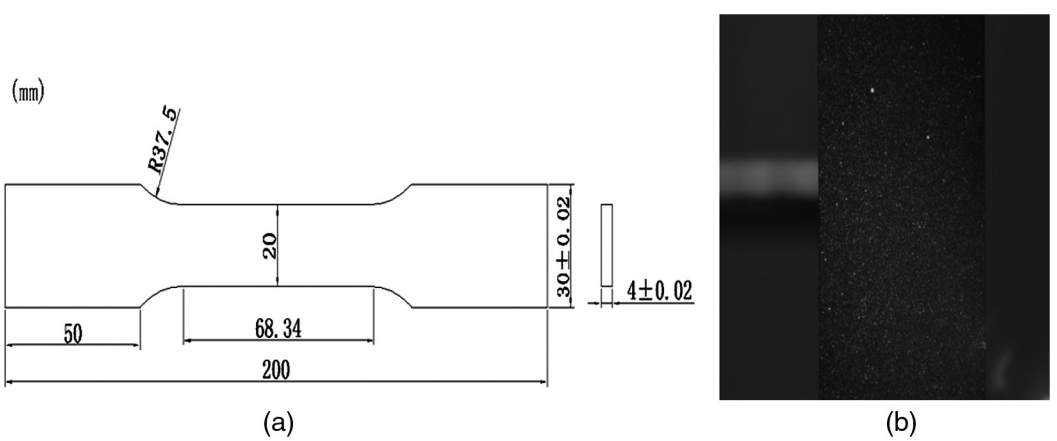

Fig. 7 The experimental specimen: (a) specimen size and (b) speckle image.

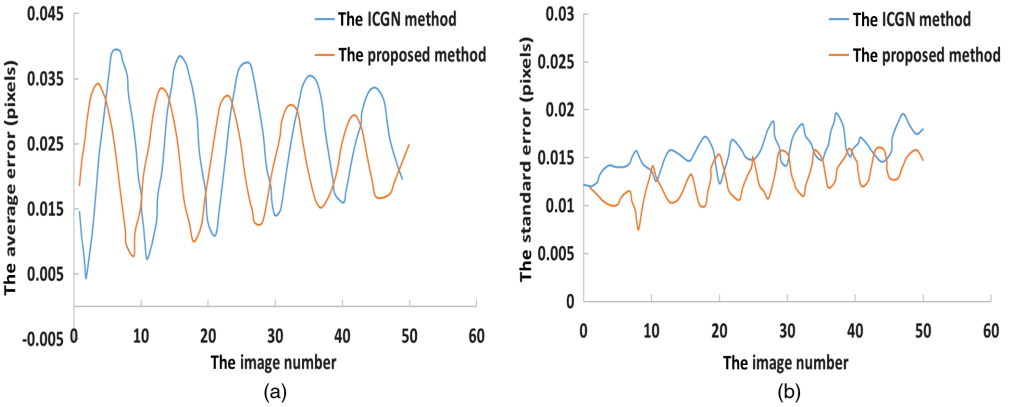

Fig. 8 Comparison between two methods: (a) average error curve and (b) standard error curve.

Table 3 Calculation speed.

Calculation points		Calculation time of the proposed algorithm (ms)				Calculation time of the ICGN method (ms)		
Image number	Total number	Seed points	Integer-pixel	Subpixel	Speed (POI/s)	Seed points	Subpixel	Speed (POI/s)
50	45,000	31	260	6	151,514	78	21,621	2074
100	90,000	78	373	12	194,469	125	44,164	2033
180	162,000	109	606	21	206,658	234	78,858	2049
180	288,000	109	1066	38	222,918	234	141,102	2038
180	648,000	109	2611	83	231,143	234	315,652	2052

experiment, but its effect on the speed of the proposed method can be ignored because there were no iteration progress during the proposed calculation method.

4 Conclusion

In this study, a new parallel temporal sequence DIC method was described. This method chose seed points to determine the integer-pixel displacement and applied the weighted moving least-squares fitting technique to acquire the subpixel displacement. The complex iterations were completely avoided, and the computation efficiency was further improved by parallel computing. The computational accuracy and efficiency of this proposed method were analyzed, and the conclusions are as follows.

- In the actual experiment, the calculation speed achieved 432,866 POI/s when the calculation speed of the ICGN method was 2052 POI/s under the same precision, which greatly improved the computation efficiency and did help in achieving real-time DIC.
- The subpixel fitting method was suitable for parallel computing, and the subpixel displacement calculation made up less than 1% of the entire calculation. If a faster integer-pixel displacement is discovered, the computation speed can be further improved.
- Under the restriction of data reading conflict, the grid cannot be too dense. If too many interest points were to be calculated, the grid can be separately moved vertically and horizontally, and the calculation grid can then be converged within several iterations.

Acknowledgments

We acknowledge the financial support provided by China Postdoctoral Science Foundation under Contract No. 2019M653148, the Fundamental Research Funds for the Central Universities under Contract No. 19lgpy290, the Guangdong Natural Science Foundation under Contract No. 2019A1515011779, and the National Natural Science Foundation of China under Contract No. U1811463.

References

1. W. Peters and W. Ranson, "Digital imaging techniques in experimental stress analysis," *Opt. Eng.* **21**(3), 213427 (1982).
2. M. A. Sutton, "Digital image correlation for shape and deformation measurements," in *Springer Handbook of Experimental Solid Mechanics*, W. N. Sharpe, Jr., Ed., pp. 565–600, Springer, Boston, Massachusetts (2008).
3. H. W. Schreier, J. R. Braasch, and M. A. Sutton, "Systematic errors in digital image correlation caused by intensity interpolation," *Opt. Eng.* **39**, 2915–2921 (2000).
4. B. Pan et al., "Two-dimensional digital image correlation for in-plane displacement and strain measurement: a review," *Meas. Sci. Technol.* **20**(6), 062001 (2009).
5. B. Pan et al., "Full-field strain measurement using a two-dimensional Savitzky–Golay digital differentiator in digital image correlation," *Opt. Eng.* **46**(3), 033601 (2007).
6. D. S. Zhang, M. Luo, and D. D. Arola, "Displacement/strain measurements using an optical microscope and digital image correlation," *Opt. Eng.* **45**(3), 033605 (2006).
7. P. Bing et al., "Performance of sub-pixel registration algorithms in digital image correlation," *Meas. Sci. Technol.* **17**(6), 1615–1621 (2006).
8. D. Lecompte et al., "Quality assessment of speckle patterns for digital image correlation," *Opt. Lasers Eng.* **44**(11), 1132–1145 (2006).
9. B. Pan, "An evaluation of convergence criteria for digital image correlation using inverse compositional Gauss–Newton algorithm," *Strain* **50**(1), 48–56 (2014).
10. Z. Jiang et al., "Path-independent digital image correlation with high accuracy, speed and robustness," *Opt. Lasers Eng.* **65**, 93–102 (2015).

11. B. Pan, K. Li, and W. Tong, "Fast, robust and accurate digital image correlation calculation without redundant computations," *Exp. Mech.* **53**(7), 1277–1289 (2013).
12. X. Shao, X. Dai, and X. He, "Noise robustness and parallel computation of the inverse compositional Gauss–Newton algorithm in digital image correlation," *Opt. Lasers Eng.* **71**, 9–19 (2015).
13. G. Pratx and L. Xing, "GPU computing in medical physics: a review," *Med. Phys.* **38**(5), 2685–2697 (2011).
14. J. Michalakes and M. Vachharajani, "GPU acceleration of numerical weather pr ediction," *Parallel Process. Lett.* **18**(4), 531–548 (2008).
15. B. Pan and L. Tian, "Superfast robust digital image correlation analysis with parallel computing," *Opt. Eng.* **54**(3), 034106 (2015).
16. L. Zhang et al., "High accuracy digital image correlation powered by GPU-based parallel computing," *Opt. Lasers Eng.* **69**, 7–12 (2015).
17. T. Wang et al., "A flexible heterogeneous real-time digital image correlation system," *Opt. Lasers Eng.* **110**, 7–17 (2018).
18. G. Le Besnerais, Y. Le Sant, and D. Lévêque, "Fast and dense 2-D and 3-D displacement field estimation by a highly parallel image correlation algorithm," *Strain* **52**(4), 286–306 (2016).
19. W. Hu and H. Miao, "Sub-pixel displacement algorithm in temporal sequence digital image correlation based on correlation coefficient weighted fitting," *Opt. Lasers Eng.* **110**, 410–414 (2018).
20. P. Lancaster and K. Salkauskas, "Surfaces generated by moving least squares methods," *Math. Comput.* **37**(155), 141–158 (1981).

Chen Xiong received his doctorate in mechanics in the Department of Modern Mechanics from the University of Science and Technology of China. Currently, he works at Sun Yat-Sen University and is employed as an associate researcher. His research interests include optical measurement and intelligent transportation systems.

Jiatao Chen is a BE candidate in the School of Intelligent Systems Engineering at Sun Yat-Sen University. His research interests include optical measurement and data analysis.

Feng Li received his doctorate in mechanics at Sun Yat-Sen University. Currently, he works at Guangdong Polytechnic Normal University and is employed as an associate professor. His research interests include optical measurement and intelligent transportation systems.

Ming Cai received his doctorate in mechanics at Sun Yat-Sen University. Currently, he works at Sun Yat-Sen University and is employed as a professor. His research interests include intelligent transportation systems and transportation big data.