Experimental Mechanics (2016) 56:297–309
DOI 10.1007/s11340-015-0091-4

GPU Accelerated Digital Volume Correlation

T. Wang[1] · Z. Jiang[2] · Q. Kemao[1] · F. Lin[1] · S.H. Soon[1]

Received: 17 March 2015 / Accepted: 28 August 2015 / Published online: 6 October 2015

Abstract A sub-voxel digital volume correlation (DVC) method combining the 3D inverse compositional Gauss-Newton (ICGN) algorithm with the 3D fast Fourier transform-based cross correlation (FFT-CC) algorithm is proposed to eliminate path-dependence in current iterative DVC methods caused by the initial guess transfer scheme. The proposed path-independent DVC method is implemented on NVIDIA compute unified device architecture (CUDA) for GPU devices. Powered by parallel computing technology, the proposed DVC method achieves a significant improvement in computation speed on a common desktop computer equipped with a low-end graphics card containing 1536 CUDA cores, i.e., up to 23.3 times faster than the sequential implementation and 3.7 times faster than the multithreaded implementation of the same DVC method running on a 6-core CPU. This speedup, which has no compromise with resolution, accuracy and precision, benefits from the coarse-grained parallelism that the points of interest (POIs) are processed simultaneously and also from the fine-grained parallelism that the calculation at each POI is performed with multiple threads in GPU. The experimental study demonstrates the superiority of the GPU-based parallel computing for acceleration of DVC over the multi-core CPU-based one, in particular on a PC level computer.

✉ Z. Jiang
zhenyujiang@scut.edu.cn

✉ Q. Kemao
mkmqian@ntu.edu.sg

[1] School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore

[2] School of Civil Engineering and Transportation, South China University of Technology, Guangzhou 510640, China

Keywords Digital volume correlation · Parallel computing · Compute unified device architecture · Graphics processing unit · Inverse compositional Gauss-Newton algorithm · Fast Fourier transform

Introduction

Digital volume correlation (DVC) has been adopted as an effective technique for determining internal volumetric deformation within solid materials since it was proposed by Bay and Smith [1, 2]. The DVC technique tracks small motion of points of interest (POIs) by minimizing the difference between the POI-centric subvolumes in the reference (un-deformed) volume image and the target (deformed) volume image acquired using 3D imaging techniques such as X-ray computer tomography (CT). This technique can be considered as a straightforward extension of the well-established digital image correlation (DIC) [3, 4] and shares its simplicity in principles and effectiveness in applications. Nowadays, it has been extensively applied in the characterization of various materials including bones [1, 5], soft materials [6, 7], wood [8] and sand [9]. Compared with DIC, the computational burden of DVC is much heavier. Thus high computation speed of DVC has become one of main issues in this area that challenges researchers in the past decade.

A DVC algorithm can be performed at two levels: integer-voxel level and sub-voxel level. For integer-voxel displacement estimation, fast Fourier transform based cross-correlation (FFT-CC) algorithm [6, 10] is widely used as a classic method, which benefits from the fact that cross-correlation operation in space domain is equivalent to point-wise multiplication in frequency domain. The algorithm can be further accelerated by combining a fast sum-table approach [7]. To achieve a sub-voxel level accuracy, various sub-voxel

registration algorithms have been developed, including the iterative algorithms derive from Newton's minimization method, e.g., Levenberg-Marquardt algorithm [1], Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [2, 11, 10], Newton–Raphson algorithm [7] and iterative least-squares algorithm [12], and non-iterative algorithms such as the correlation coefficients curve-fitting algorithm [6] and the gradient-based algorithm [7]. Recently, Pan et al. introduced three ideas to accelerate the computation of DVC effectively [13] : (1) an inverse compositional Gauss-Newton (ICGN) algorithm [14] was introduced to replace the conventional forward additive Newton–Raphson algorithm for sub-voxel registration. This algorithm eliminates the repeated updates of the Hessian matrix during the iterative procedure; (2) a reliability-guided displacement tracking strategy [15] was employed to provide fast and accurate initial guess for the ICGN algorithm; (3) a global look-up table of cubic interpolation coefficients [16] was pre-established to eliminate the redundant calculation of sub-voxel intensity interpolation coefficients for the construction of warped target image during the iterative procedure. By these means, the computation speed of DVC can surge up from about 0.9 points of interest per second (POI/s) to 41.1 POI/s when using a 19×19×19 -voxel subvolume [13].

The computation speed of DVC could be substantially increased further by processing POIs simultaneously on parallel computing device. However, the application of parallel computing technology requires the independence of calculation at each POI, which conflicts with any path-dependent processing strategy leading to a sequential computation scheme. To overcome this difficulty, our preliminary research proposed a path-independent DIC method [17], which estimated the initial guess for the ICGN algorithm at each POI independently using the FFT-CC algorithm. Elimination of path-dependency allows the application of parallel computing technology to the proposed DIC method. It was found that the parallel computing-powered DIC (paDIC) method, implemented on NVIDIA compute unified device architecture (CUDA) for graphics processing unit (GPU) devices, can reach a speedup of nearly two orders of magnitude without sacrificing high accuracy and precision [18].

Recently, effort has been made to accelerate DVC using GPU devices. Bar-Kochba et al. [19] proposed a DVC method that iteratively compares the reference volume image and the target volume image through FFT-CC algorithm and then adjusts the two images until the displacement increment obtained in iteration reaches a sufficiently small value. In their implementation, the FFT-CC computation at POIs can be performed in parallel on GPU or sequentially on CPU. The GPU implementation gained a speed improvement of about 5.5 times over its CPU counterpart. Gates et al. [10] combined FFT-CC (for coarse search) and BFGS algorithm (for sub-voxel registration), and provided an implementation of their DVC method in a hybrid multi-core CPU and multiple GPU mode. The GPUs were used to estimate the normalized cross-correlation objective function and its gradient at POIs in a refined parallel manner, and the multi-core CPU carries out the BFGS iteration accordingly. In their experiments, the program based on the collaboration of 3 GPUs and a 12-core CPU can be eight times faster than the one running merely on CPU. In addition, GPU was also employed to speed up the computation for a voxel-based global DVC method [20], by solving the resulting system including millions of degrees of freedom.

In this paper, we extend the paDIC to the parallel DVC (paDVC) on GPU devices and assess its performance using computer simulated 3D speckle images. The implementation of the proposed paDVC includes coarse-grained and fine-grained parallelism for all the three major parts, namely FFT-CC algorithm, ICGN algorithm and the precomputation procedure which can further accelerate the ICGN algorithm. The rest of the paper is organized as follows. Section 2 briefly explains the principle of the proposed path-independent DVC algorithm. Subsequently, section 3 describes the implementation of the paDVC on NVIDIA CUDA. In section 4, the accuracy, precision and computational efficiency of the paDVC are verified. Section 5 concludes the paper.

Principle of the paDVC

DVC tracks the displacement from a POI $\mathbf{P}(x_0, y_0, z_0)$ in the reference volume image (Fig. 1(a)) to a point $\mathbf{P}'(x_0', y_0', z_0')$ in the target volume image (Fig. 1(b)). In order to avoid mis-registration, a cubic subvolume centred at $\mathbf{P}(x_0, y_0, z_0)$ is selected as the basic matching unit.

The proposed paDVC can be summarized as a three-step procedure, as illustrated in Fig. 2. The integer-voxel registration is carried out using the 3D FFT-CC algorithm at every POI. The obtained integer-voxel deformation vector $p_0=(u,0,0,0,v,0,0,0,w,0,0,0)^T$ is fed as the initial guess into the 3D ICGN algorithm for the sub-voxel registration. Prior to these two steps, the inverse of the Hessian matrix for each reference subvolume is precomputed and a global look-up table of tri-cubic interpolation coefficients for the target volume image is constructed. This precomputation step helps to eliminate a large number of redundant calculations during iterations of the 3D ICGN algorithm.

Integer-Voxel Registration by the 3D FFT-CC Algorithm

Defining a POI-centric subvolume with $N=(2M+1)\times(2M+1)\times(2M+1)$ voxels, where N and M are integers, the displacement vector $\mathbf{d}=(u,v,w)^T$ can be obtained

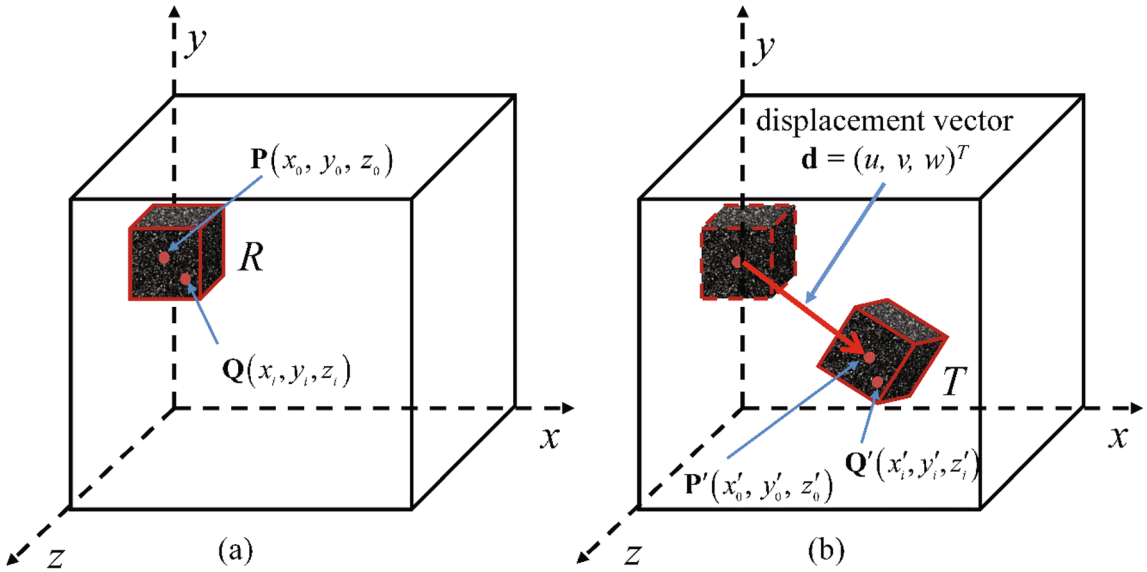

Fig. 1 Schematic illustration of the principle of DVC: the displacement vector $\mathbf{d}=(u,v,w)^T$ from a POI $\mathbf{P}(x_0, y_0, z_0)$ in the reference volume image to $\mathbf{P}'(x_0', y_0', z_0')$ in the target volume image is obtained by tracking the corresponding cubic subvolumes. $\mathbf{Q}(x_i,y_i,z_i)$ and $\mathbf{Q}'(x_i',y_i',z_i')$ indicate the points within the reference subvolume and the target subvolume, respectively

by locating the peak of the zero-mean normalized cross-correlation (ZNCC) coefficients:

$$C_{ZNCC}(u,v,w)=\frac{\sum_i (R_i-R_m)(T_i-T_m)}{\sqrt{\sum_i (R_i-R_m)^2\sum_i (T_i-T_m)^2}} \tag{1}$$

where R_i and T_i are the grey intensity values of the ith voxel in the reference subvolume and the target subvolume, respectively; $R_m=\frac{1}{N}\sum_{i=0}^{N-1} R_i$ and $T_m=\frac{1}{N}\sum_{i=0}^{N-1} T_i$ refer to the mean intensity values within the two subvolumes. According to the Fourier theory, the calculation of $C_{ZNCC}(u,v,w)$ in space domain can be performed as a simple point-wise multiplication in frequency domain, i.e.,

$$C_{ZNCC}(u,v,w)=FFT^{-1}\left\{FFT^{*}\left[\sum_i \frac{R_i-R_m}{\sqrt{\sum_i (R_i-R_m)^2}}\right]\cdot FFT\left[\sum_i \frac{T_i-T_m}{\sqrt{\sum_i (T_i-T_m)^2}}\right]\right\} \tag{2}$$

where FFT and FFT^{-1} are the fast Fourier transform and inverse fast Fourier transform, respectively, and the superscript "*" denotes the complex conjugate. The integer-voxel displacement vector $\mathbf{d}=(u,v,w)^T$ is determined from the peak of $C_{ZNCC}(u,v,w)$ and then transformed to an integer-voxel deformation vector $\mathrm{p}_0=(u,0,0,0,v,0,0,0,w,0,0,0)^T$.

Sub-Voxel Registration by the 3D ICGN Algorithm

Assuming that a point $\mathbf{Q}(x_i,y_i,z_i)$ in the reference subvolume R deforms to a point $\mathbf{Q}'(x_i',y_i',z_i')$ in the target subvolume T (see Fig. 1), a first-order shape function is selected to describe the relationship between $\mathbf{Q}$ and $\mathbf{Q}'$ as

$$\begin{aligned} x_i' &= x_i+u+u_x\Delta x+u_y\Delta y+u_z\Delta z\\ y_i' &= y_i+v+v_x\Delta x+v_y\Delta y+v_z\Delta z\\ z_i' &= z_i+w+w_x\Delta x+w_y\Delta y+w_z\Delta z \end{aligned} \tag{3}$$

where u_x, u_y, u_z, v_x, v_y, v_z, w_x, w_y and w_z are gradients of u, v and w with respect to x, y and z axis; $\Delta x=x_i-x_0$, $\Delta y=y_i-y_0$ and $\Delta z=z_i-z_0$ are the local coordinates of point $\mathbf{Q}$ with respect to the position of POI $\mathbf{P}$ in the reference subvolume. Equation (3) can be rewritten as

$$\mathbf{Q}'=\mathbf{P}+\mathbf{W}(\xi;\mathbf{p}) \tag{4}$$

where $\mathbf{W}(\xi;\mathbf{p})$ is the warp function,

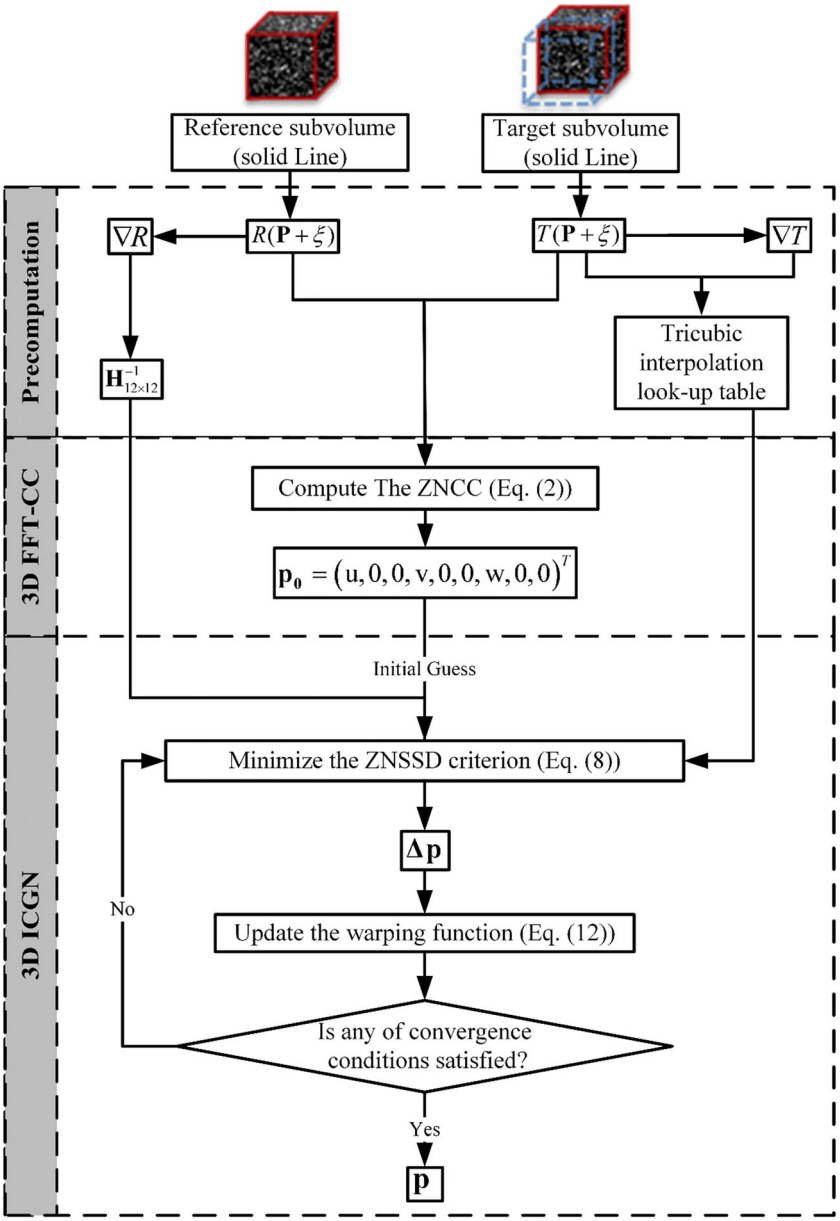

Fig. 2 Flow chart of the proposed paDVC, where R and T refer to the reference subvolume and target subvolume, respectively, ξ is the local coordinates of points within the subvolume

$$\mathbf{W}(\xi;\mathbf{p}) = \begin{bmatrix} 1+u_x & u_y & u_z & u \\ v_x & 1+v_y & v_z & v \\ w_x & w_y & 1+w_z & w \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ 1 \end{bmatrix} \quad (5)$$

where $\xi=(\Delta x,\Delta y,\Delta z,1)^T$ represents the local coordinates of point **Q** within the reference subvolume, and $\mathbf{p}=(u,u_x,u_y,u_z,v,v_x,v_y,v_z,w,w_x,w_y,w_z)^T$ represents the vector of deformation parameters. Besides, the incremental warp function $\mathbf{W}(\xi;\Delta\mathbf{p})$ used to adjust the shape of the reference subvolume can be written as

$$\mathbf{W}(\xi;\Delta\mathbf{p}) = \begin{bmatrix} 1+\Delta u_x & \Delta u_y & \Delta u_z & \Delta u \\ \Delta v_x & 1+\Delta v_y & \Delta v_z & \Delta v \\ \Delta w_x & \Delta w_y & 1+\Delta w_z & \Delta w \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ 1 \end{bmatrix} \quad (6)$$

where $\Delta\mathbf{p}=(\Delta u,\Delta u_x,\Delta u_y,\Delta u_z,\Delta v,\Delta v_x,\Delta v_y,\Delta v_z,\Delta w,\Delta w_x,\Delta w_y,\Delta w_z)^T$ is the incremental vector of deformation parameter. The incremental vector $\Delta\mathbf{p}$ can be determined by the 3D ICGN algorithm which minimizes the zero-mean normalized sum of squared difference (ZNSSD) criterion,

$$C_{ZNSSD}(\Delta\mathbf{p}) = \sum_{\xi}\left\{\frac{R[\mathbf{P}+\mathbf{W}(\xi;\Delta\mathbf{p})]-R_m}{\sqrt{\sum_{\xi}\{R[\mathbf{P}+\mathbf{W}(\xi;\Delta\mathbf{p})]-R_m\}^2}}-\frac{T[\mathbf{P}+\mathbf{W}(\xi;\mathbf{p})]-T_m}{\sqrt{\sum_{\xi}\{T[\mathbf{P}+\mathbf{W}(\xi;\mathbf{p})]-T_m\}^2}}\right\}^2 \tag{7}$$

Then, $\Delta\mathbf{p}$ can be solved through

$$\Delta\mathbf{p} = \mathbf{H}^{-1}_{12x12}\sum_{\xi}\left\{\left(\nabla R\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right)^T_{12x1}\left[\frac{\sqrt{\sum_{\xi}\{R[\mathbf{P}+\mathbf{W}(\xi;\Delta\mathbf{p})]-R_m\}^2}}{\sqrt{\sum_{\xi}\{T[\mathbf{P}+\mathbf{W}(\xi;\mathbf{p})]-T_m\}^2}}(T[\mathbf{P}+\mathbf{W}(\xi;\mathbf{p})]-T_m)-(R[\mathbf{P}+\mathbf{W}(\xi;\Delta\mathbf{p})]-R_m)\right]\right\} \tag{8}$$

where ∇R is the gradient within the reference subvolume,

$$\nabla R = \left(\frac{\partial R(x,y,z)}{\partial x},\frac{\partial R(x,y,z)}{\partial y},\frac{\partial R(x,y,z)}{\partial z}\right) \tag{9}$$

$\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$ is the Jacobian matrix of the warp function,

$$\frac{\partial\mathbf{W}}{\partial\mathbf{p}} = \begin{bmatrix} 1 & \Delta x & \Delta y & \Delta z & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta x & \Delta y & \Delta z & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta x & \Delta y & \Delta z \end{bmatrix} \tag{10}$$

and $\mathbf{H}^{-1}_{12\times12}$ denotes the inverse of the 12×12 Hessian matrix $\mathbf{H}$,

$$\mathbf{H} = \sum_{\xi}\left\{\left[\left(\nabla R\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right)^T_{12\times1}\left(\nabla R\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right)_{1\times12}\right]\right\} \tag{11}$$

The obtained $\Delta\mathbf{p}$ is then substituted back into equation (6) to yield the incremental warp function $\mathbf{W}(\boldsymbol{\xi};\Delta\mathbf{p})$ which is used to update the warp function $\mathbf{W}(\boldsymbol{\xi};\mathbf{p})$ according to

$$\mathbf{W}(\xi;\mathbf{p})\leftarrow\mathbf{W}\left[\mathbf{W}^{-1}(\xi;\Delta\mathbf{p}),\mathbf{p}\right]=\mathbf{W}(\xi;\mathbf{p})\mathbf{W}^{-1}(\xi;\Delta\mathbf{p}) \tag{12}$$

where

$$\mathbf{W}^{-1}(\boldsymbol{\xi};\Delta\mathbf{p}) = \begin{bmatrix} 1+\Delta u_x & \Delta u_y & \Delta u_z & \Delta u \\ \Delta v_x & 1+\Delta v_y & \Delta v_z & \Delta v \\ \Delta w_x & \Delta w_y & 1+\Delta w_z & \Delta w \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\\ 1\end{bmatrix} \tag{13}$$

represents the inverse of the incremental warping function. This procedure is repeated until one of the following two pre-imposed convergence conditions is satisfied, i.e., $\sqrt{(\Delta u)^2+(\Delta v)^2+(\Delta w)^2} < 0.001$ or the maximum iteration number reaches 20.

It is clear that the Hessian matrix $\mathbf{H}$ only depends on the gradient ∇R in the reference subvolume and the Jacobian matrix of the warp function $\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$, thus $\mathbf{H}$ and its inverse can be precomputed, which makes the ICGN algorithm faster than the traditional forward additive Newton–Raphson algorithm.

During the iterations, the reconstructed $T[\mathbf{P}+\mathbf{W}(\xi;\mathbf{p})]$ in equation (8) often locates at a sub-voxel location (see point C in Fig. 3). Therefore, an interpolation scheme is applied to estimate the intensity at sub-voxel locations. In DIC bicubic spline interpolation is usually preferable to polynomial interpolation because it can give more accurate results [21]. However, Tri-cubic interpolation is chosen in the proposed DVC for two reasons: i) The amount of computation in DVC can be several orders of magnitude higher than that of

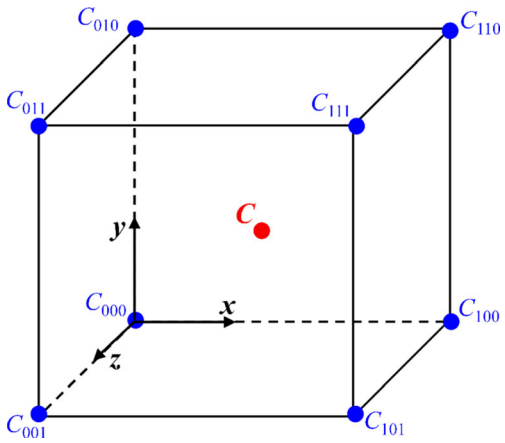

Fig. 3 Diagram of the tri-cubic interpolation at a sub-voxel position surrounded by 8 voxels with integer coordinates

DIC, which makes the computation speed a critical requirement. In this context, the significant speedup gained by using more computationally efficient tri-cubic interpolation, which in particular allows a precomputed global look-up table of interpolation coefficients that can further reduce the interpolation times by orders of magnitude [16], could be highly beneficial to the proposed DVC method at a price of small (sometimes imperceptible) loss of accuracy; ii) Based on our preliminary of the path-independent DIC [17], the proposed DIC method, using bicubic interpolation, gave an excellent accuracy and precision, which is comparable with the Newton–Raphson algorithm using bicubic spline interpolation on similar speckle images reported in Ref. [22]. Moreover, it was found in real experiments that the difference in the bias error of the Newton–Raphson algorithm using quantic spline interpolation and bicubic interpolation can be reduced to a negligible level if a proper Gaussian pre-filtering is performed on the noise contaminated speckle images [23]. Therefore, a tri-cubic interpolation scheme with global look-up table is considered as a good trade-off between computational efficiency and registration accuracy for the paDVC.

Implementation of the paDVC on CUDA

Mapping from CUDA Programming Model to NVIDIA GPU Hardware

Figure 4 illustrates the mapping from the CUDA programming model to NVIDIA GPU hardware. In the CUDA programming model, a parallel problem is first split into coarse sub-problems that can be solved independently by blocks, and each sub-problem is further divided into finer pieces that can be solved cooperatively by all the threads within the block. In the architecture of NVIDIA GPU hardware, the blocks are mapped as components on streaming multiprocessors (SMs).

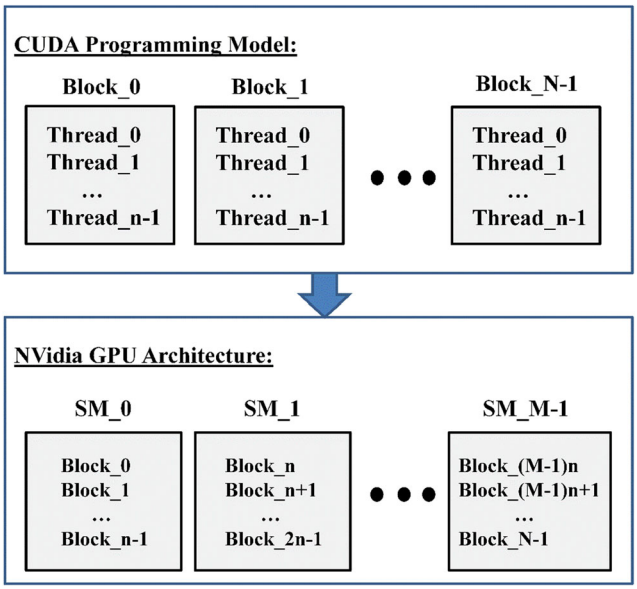

Fig. 4 Mapping from the CUDA programming model to NVIDIA GPU hardware

Parallelization of Precomputation

The gradients ∇R and ∇T used in building the inverse of the Hessian matrix in the reference subvolume and the global look-up table of tri-cubic interpolation coefficients for the target volume image are calculated according to the central differences approximation,

$$\begin{aligned}
\frac{\partial R(x,y,z)}{\partial x} &= \frac{R(x+1,y,z)-R(x-1,y,z)}{2}, & \frac{\partial T(x,y,z)}{\partial x} &= \frac{T(x+1,y,z)-T(x-1,y,z)}{2}\\
\frac{\partial R(x,y,z)}{\partial y} &= \frac{R(x,y+1,z)-R(x,y-1,z)}{2}, & \frac{\partial T(x,y,z)}{\partial y} &= \frac{T(x,y+1,z)-T(x,y-1,z)}{2}\\
\frac{\partial R(x,y,z)}{\partial z} &= \frac{R(x,y,z+1)-R(x,y,z-1)}{2}, & \frac{\partial T(x,y,z)}{\partial z} &= \frac{T(x,y,z+1)-T(x,y,z-1)}{2}
\end{aligned} \tag{14}$$

Equation (14) indicates that the computation of the gradient at a voxel involves the intensity values at 3×3×3 voxels. In this work, instead of processing the gradient at each voxel individually, we introduce a tiled algorithm [24] to calculate the gradients in groups of 6×6×6 voxels using blocks containing 8×8×8 threads. By this way, an optimal occupancy of SMs can be achieved and the latency caused by intensive accesses to global memory can be minimized. During the precomputation of gradients for every 6×6×6 voxels, the intensity values at 8×8×8 voxels are loaded into the shared memory allocated to a block, as shown in Fig. 5. Then the calculation of gradients at 216 voxels is performed by 216 threads in parallel according to equation (14). It is noteworthy that 296 (=512–216) threads around the 216-thread tile are only used to store the intensity values at the voxels along the outer boundary and do not perform any calculation. In this trade-space-for-time strategy, all the data required to calculate the gradients at each voxel in the 6×6×6-voxel unit along the three axes are loaded only once to minimize the access of global memory, which is generally quite time consuming. As

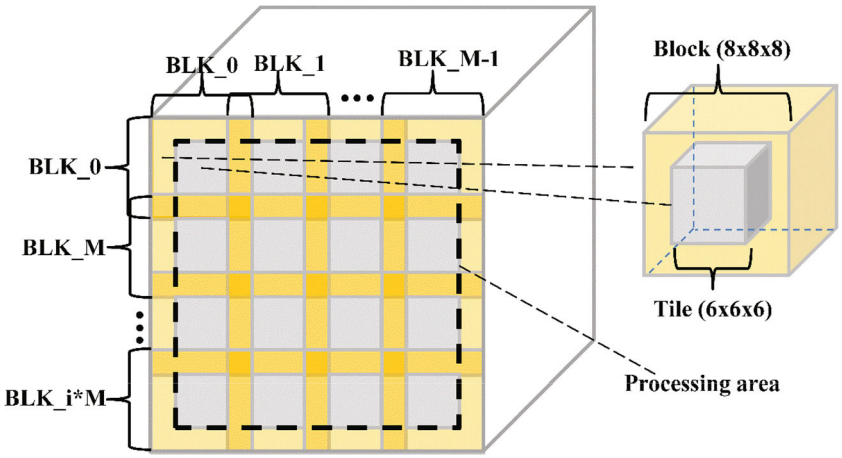

Fig. 5 Illustration of the tiled computation for volume image gradients

the precomputation takes a very small portion of the overall computation time (~3 %, according to Tables 2 and 3 in Section 4.3), this waste of threads only leads to trivial loss of computational efficiency.

The tri-cubic interpolation used in equation (8) is defined as

$$T(x,y,z) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} \alpha_{ijk} x^i y^j z^k \tag{15}$$

where the 64 coefficients α_{ijk} are obtained according to the intensity values at its eight surrounding integer voxels as well as their gradients (see Fig. 3). A look-up table of interpolation coefficients α_{ijk} can be computed using the similar "tiled" strategy, in which each 8×8×8-thread block is assigned to calculate the interpolation coefficients of a 7×7×7 eight-voxel-unit. The constructed look-up table is then stored in global memory and will be accessed during the 3D ICGN algorithm.

Parallelization of 3D FFT-CC Algorithm

The 3D FFT-CC algorithm contains two steps: (a) calculating $C_{ZNCC}(u,v,w)$ using FFT; and (b) searching the peak of $C_{ZNCC}(u,v,w)$.

The calculation of cross correation is accelerated by CUDA FFT (CUFFT) library [25]. To achieve high utilization efficiency of GPU hardware, subvolumes are grouped in batches before they are transferred to the processing line. Although it is not guarenteed that all of the subvolumes are handled simultaneously due to hardware limitation, CUFFT automatically ensures a maximum number of subvolumes being processed in parallel.

The peak searching within each subvolume is also accelerated by parallel computing technology. First, the 3D voxel matrix containing p^3 voxels is transformed into a 1D array. The 3D index (u,v,w) is coded as a 1D index i through

$$i = (w \times p + v) \times p + u \tag{16}$$

which can be decoded according to

$$u = i \bmod p, \quad v = (i/p) \bmod p, \quad w = (i/p)/p \tag{17}$$

Second, a block containing m threads (m=256 in this work) is assigned to handle the subvolume. As in most case p^3 is greater than the thread number in a block, the p^3 C_{zncc} values and their 1D indices are divided into m groups. Each group is processed by a thread within the block to find the maxium C_{zncc} value in the group through a simple maximum element searching algorithm. Third, a classic parallel reduction algorithm [26] is applied to locate the peak C_{zncc} among the m C_{zncc} values obtained by the threads, as illustrated in Fig. 6. In each reduction step, the first half of the threads compare their C_{zncc} values with those in the second half of the threads. If the C_{zncc} of any thread in the first half is smaller than that of its counterpart in the second half, the two threads exchange their data, viz. C_{zncc} values and 1D indices. In the next reduction step, the comparison and data exchange only occur in the active half of the threads that containing larger C_{zncc} values. This procedure continues until there remains only one active thread. Finally, the 3D index of the peak C_{zncc} in a subvolume can be located by decoding the remained 1D index according to equation (17).

Compared with the sequential searching algorithm, which results in a computation complexity of $O(p^3)$ for a $(p\times p\times p)$-voxel subvolume, the parallel reduction algorithm lowers the computation complexity down to $O(p^3/m)$.

Parallelization of the 3D ICGN Algorithm

Figure 7 illustrates the parallelization of the 3D ICGN algorithm. The computation at each POI is carried out by a block containing m threads (m=256 in this work). Initially, the integer-voxel deformation vector estimated using the 3D FFT-CC algorithm is loaded into the shared memory of the block by one thread. Then, all the N threads within the block participate in the construction of the warped target subvolume

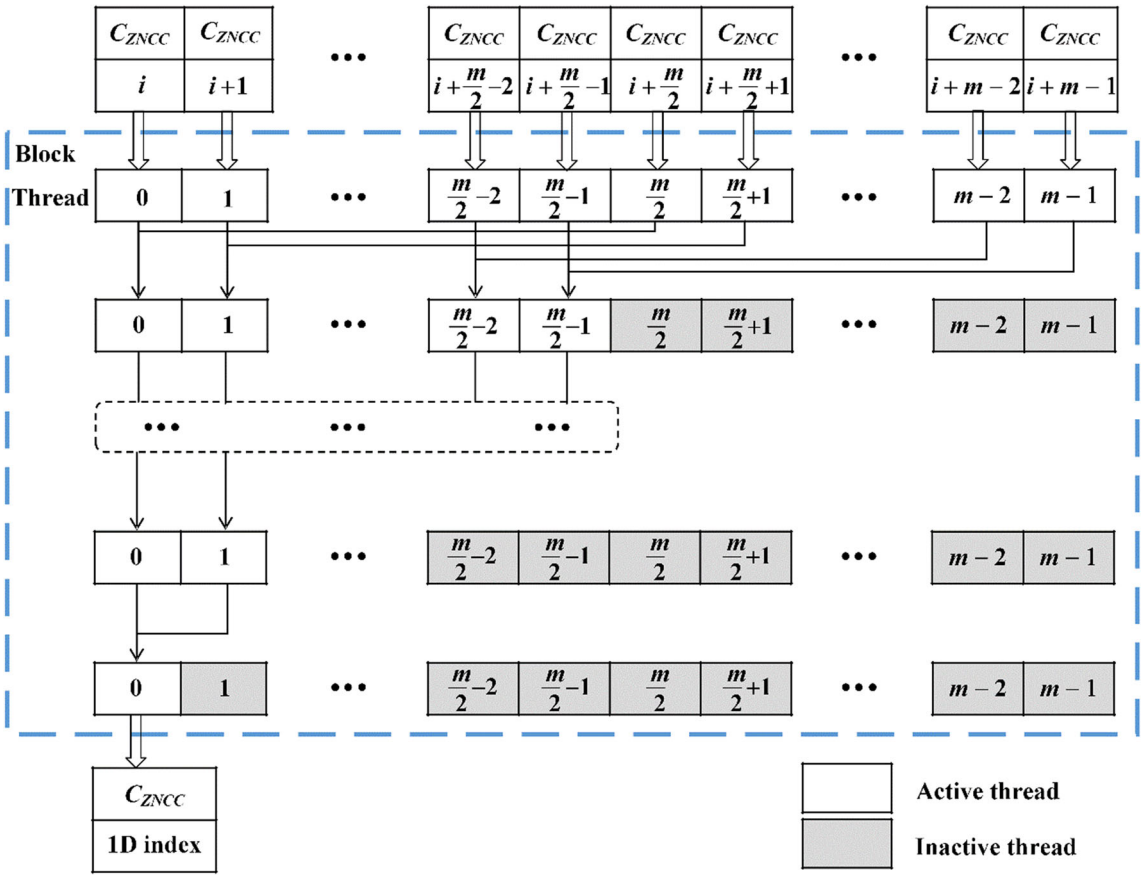

Fig. 6 Flow chart of the parallel reduction algorithm within an individual block

$T[\mathbf{P}+\mathbf{W}(\xi;\mathbf{p})]$, referring to the look-up table of tri-cubic interpolation coefficients. Afterwards, 12 threads are involved in updating the warp function $\mathbf{W}(\xi;\mathbf{p})$ according to equation (12), while the other threads become inactive. Finally, one thread remains active to check whether any of the two convergence conditions is satisfied and output the sub-voxel deformation in case the iteration is completed.

Batch Processing Mechanism

It would be ideal that all the POIs can be processed simultaneously on GPU. However, the degree of parallelism is restricted in practice due to the limited hardware capability, namely limited computing cores and memory on a GPU device. In particular, for a middle size volume image containing 512×512×512 voxels, the data generated during the DVC method can be tens of gigabytes, which is even far more than the primary memory equipped in a desktop computer. The paDVC, therefore, is implemented in a batch processing manner. It means that the volume images are first divided into cubes. And then the cubes are processed in turn, namely the POIs distributed in the whole volume image are processed in batches of a certain number. The batch size (or cube size) is variable, depending on the spatial sampling rate (viz. number of POIs within the cube) and the size of subvolume configured in the experiment. A basic rule in the batch planning is to achieve a full use of the available global memory on graphics card, and to ensure all the data required during the computation for a batch of POIs can be loaded into the global memory only once. In this work, the batch size is roughly estimated and set in the program. A dynamic and precise batch planning will be developed in the future. It is noteworthy that within the cube the calculation at POIs is also carried out in batches of an integer N, where N is dependent on the available blocks for the parallel computation in a GPU (see Fig. 4).

Obviously, the efficiency of the paDVC may be influenced to some extent by this batch processing mechanism due to more exchange of data between the global memory on graphics card and the primary memory on mainboard, and slightly redundant computation to treat the voxels along the boundary of cubes. However, this mechanism offers the paDVC a good adaptability to large scale of data and flexible computing power of GPU devices. An immediate boost of performance can be achieved by increasing the POI number in each batch

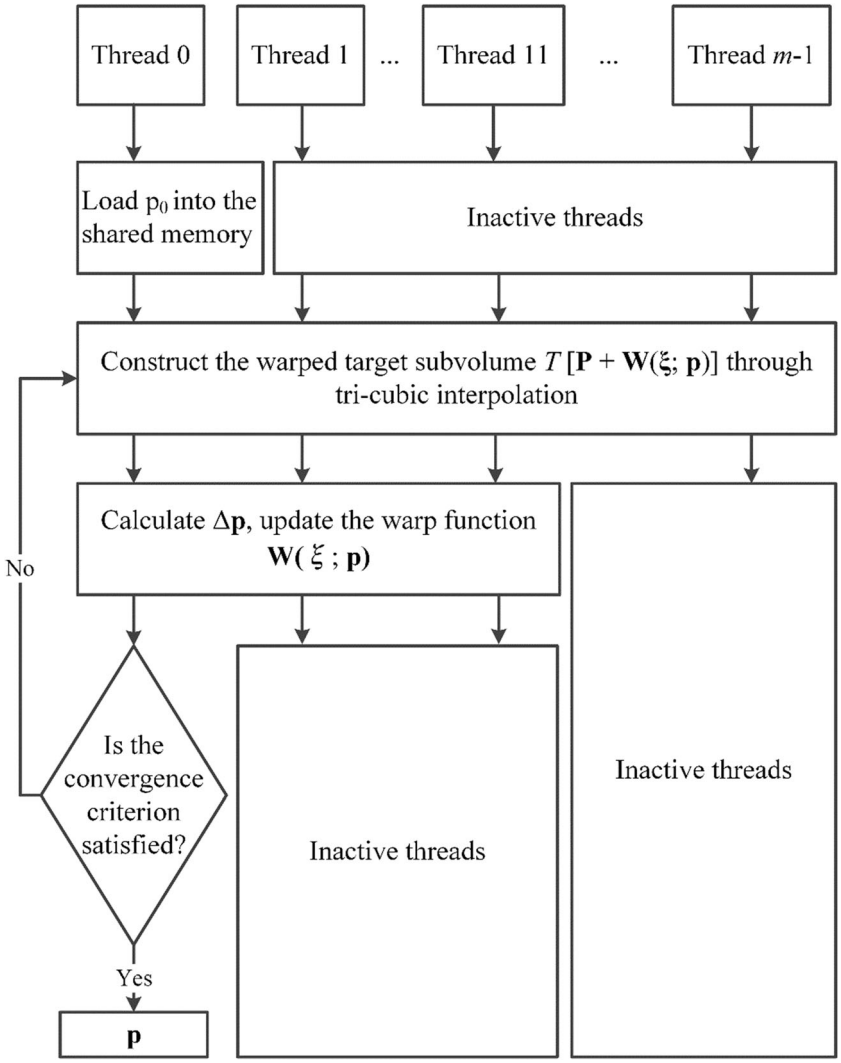

Fig. 7 Schematic flow chart of the computation procedure within one thread block of the 3D ICGN algorithm in the paDVC

when using a high-end graphics card equipped with more computing cores and memory.

Experimental Verification

Experimental Specification

The proposed paDVC method is programmed on CUDA 6.5 using C++ language, and tested on a desktop computer equipped with an Intel® Xeon® CPU E5-1650 (6 cores, 3.20 GHz main frequency), 16.0 GB RAM and a NVIDIA GeForce GTX 680 graphics card (8 SMs with 1536 CUDA cores and 2GB 256-bit RAM). It is worth mentioning that, although a low-end GPU is used in this work, the paDVC is applicable to any platforms supporting CUDA, and higher performance in terms of computation speed can be expected on more powerful platforms.

In order to quantitatively evaluate the accuracy and precision of the paDVC method, an 8-bit grayscale reference volume image $R(x,y,z)$ (see Fig. 8) is generated according to Ref. [13]:

$$R(x,y,z) = \text{round}\left\{\sum_{i=1}^{s} I_i \exp\left[-\frac{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}{r^2}\right]\right\} \tag{18}$$

where s represents the total number of speckles, I_i and (x_i,y_i,z_i) represents the random peak intensity value of the ith speckle and its center position, respectively. r is the radius of the speckle. Function round(x) returns the nearest integer to x. Afterwards, ten target volume images are generated by translating $R(x,y,z)$ in the Fourier domain according to the shift theorem [21], with pre-set sub-voxel displacement along z-axis ranging from 0 to 1 voxel. The step between every two successive images is set to be 0.1 voxels. To simulate the influence of noise in real experiments, a white Gaussian noise field with a zero mean value and a variance of 4 gray levels is then

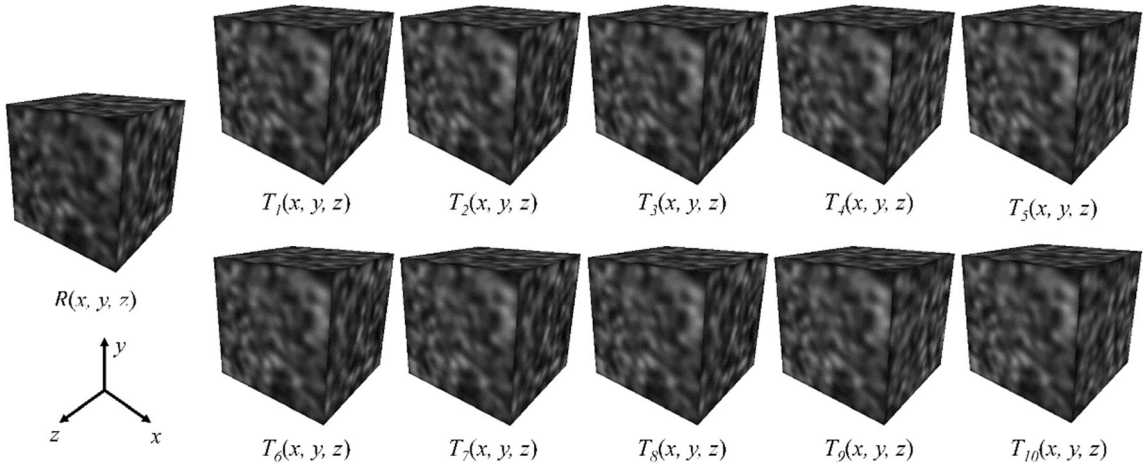

Fig. 8 Computer simulated reference volume image $R(x, y, z)$ and the 10 translated target volume images from $T_1(x, y, z)$ to $T_{10}(x, y, z)$

superposed on these volume images. In this work, 11 volume images with a dimension of 512×512×512 voxels were generated, each of which contains approximately 1.5×10^6 speckles. In each volume image 7.29×10^5 (=90×90×90) POIs are evenly distributed with a space of 5 voxels between the neighbouring POIs. Three kinds of subvolumes with different sizes are selected to carry out the DVC calculation, namely 17×17×17 voxels, 25×25×25 voxels and 33×33×33 voxels.

Verification of the Accuracy and Precision

The accuracy and precision of the paDVC are studied according to two performance indicators: mean bias error and standard deviation. The mean bias error of the w-component in z-direction is expressed by

$$e_w = \frac{1}{N}\sum_{i=1}^{N}(w_i - w_{set}) \tag{19}$$

where N is the number of POIs, w_i is the calculated displacement at the ith POI and w_{set} denotes the pre-set sub-voxel displacement. The standard deviation of the measured displacement is defined as

$$\sigma_w = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(w_i - \overline{w}\right)^2} \tag{20}$$

where $\overline{w} = \frac{1}{N}\sum_{i=1}^{N} w_i$ denotes the expectation of the measured w-component.

Figure 9 shows the mean bias errors and standard deviations of sub-voxel displacement calculated by the paDVC using the three kinds of subvolumes, respectively. It can be seen in Fig. 9(a) that the mean bias error of the calculated w-component falls in a narrow band from -0.43×10^{-3} to 0.29×10^{-3} voxels. The accuracy of the paDVC seems not sensitive to the variation of subvolume size in a certain range. In Fig. 9(b), the standard deviations of the measured w-component are below 1.42×10^{-3} voxels. A larger subvolume size leads

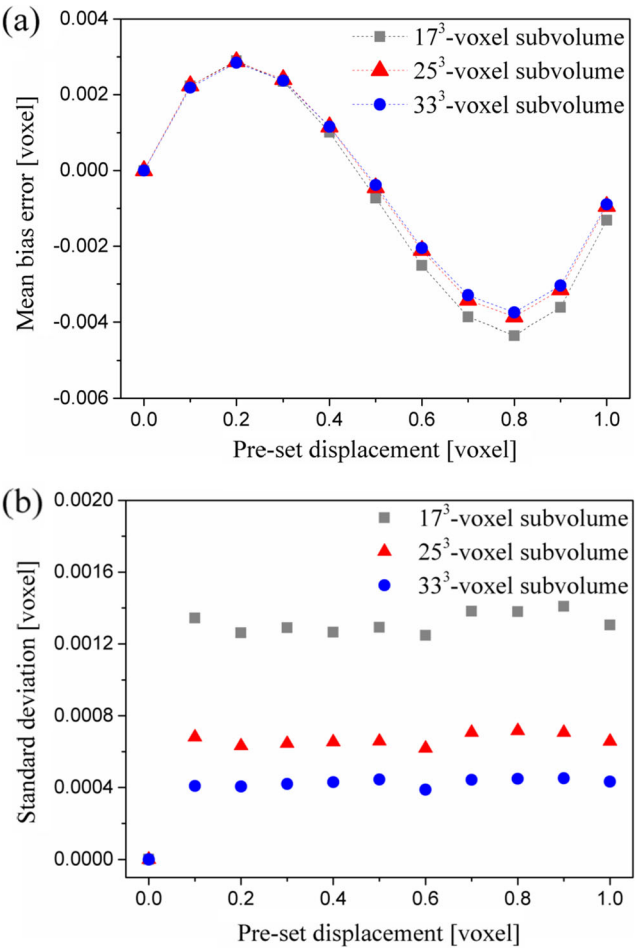

Fig. 9 (**a**) Mean bias errors and (**b**) standard deviation of the measured w-components on a series of noise-contaminated volume images, calculated using different subvolume sizes

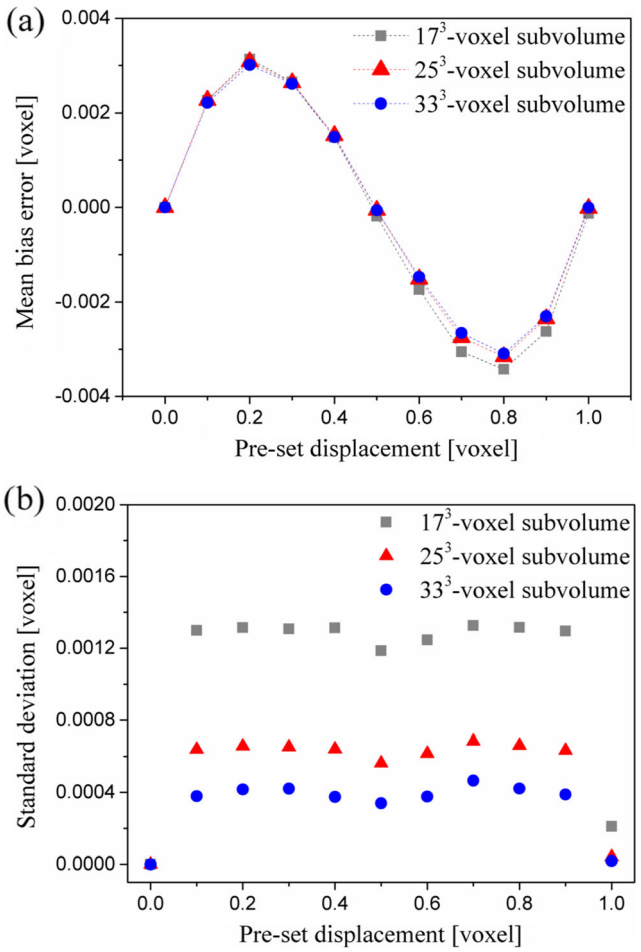

Fig. 10 (**a**) Mean bias errors and (**b**) standard deviation of the measured *w*-components on a series of noise-free volume images, calculated using different subvolume sizes

to a smaller standard deviation for the measurement of sub-voxel displacements, but at a price of significantly increased computation time, as will be discussed in the next subsection. Figure 10 gives the mean bias errors and standard deviations calculated on the same series of volume images without noises. Comparing the results in the two figures, it can be found that the white Gaussian noises reduce the accuracy and precision of the proposed paDVC method. But this adverse effect remains at an insignificant level.

Verification of the Computational Efficiency

To assess the computational efficiency of the proposed paDVC, the same DVC algorithm with sequential implementation (seDVC) and multithreaded implementation (muDVC) based on OpenMP [27] running merely on CPU are used as benchmarks.

In the seDVC, the POIs are processed one by one. The FFTW library [28] is employed to perform the fast Fourier transform, and the peak C_{zncc} as well as its index are obtained by a simple maximum element searching algorithm. The implementation of muDVC is almost identical to the seDVC except that a coarse-grained parallelization is applied. The number of threads is set to be equal to twice the number of physical CPU cores, since the Intel hyper-threading technology allows each CPU core operates at most two simultaneous threads. It is found that a larger thread number does not increase the computation speed further, because these threads actually share the time slices of up to 6 CPU cores. The batch processing scheme is also adopted in the seDVC and the muDVC due to the huge amount of data. To make the comparison fair, the three DVC programs use the same batch size during the experimental study. The Intel single instruction multiple data (SIMD) instruction is enabled for the compiling of all the programs.

Table 1 compares the computation time and computation speed among the paDVC, the muDVC and the seDVC with different subvolume sizes. The paDVC achieves a significant speedup, approximately 3.0×~3.7× over the muDVC and 18.3×~23.3× over the seDVC. Compared with another sequential implementation of ICGN algorithm-based DVC with path-dependent strategy, which was run on a desk computer equipped with similar grade CPU [13], the paDVC also demonstrates its clear advantage in computational efficiency. It can be observed that the increase in subvolume size effectively reduces the iteration number required by the 3D ICGN algorithm. However, a larger subvolume does not lead to a higher computation speed. The computation time taken by all the three DVC programs is substantially increased due to the surge of voxel number involved in calculation.

Table 1 Comparison of the average computation time and speed among the paDVC, the muDVC and the seDVC

Subvolume [voxel]	Average consumed time [s]			Average iteration number			Average computation speed [POI/s]		
	paDVC	muDVC	seDVC	paDVC	muDVC	seDVC	paDVC	muDVC	seDVC
17×17×17	416.5	1261.5	7617.5	4.12	4.12	4.12	1750.4	577.9	95.7
25×25×25	861.9	2887.9	17862.6	2.91	2.91	2.91	845.8	252.4	40.8
33×33×33	1641.4	6061.0	38186.6	2.64	2.64	2.64	444.1	120.3	19.1

Table 2 Comparison of the average computation time consumed per batch in the precomputation step among the paDVC, the muDVC and the seDVC. Batch size means the number of POIs processed in one batch (or in a cube)

Subvolume [voxel]	Cube Size [voxel]	Batch size	Average time consumed in the precomputation step [ms/batch]		
			paDVC	muDVC	seDVC
17×17×17	92×92×92	3375	52.33	498.84	2776.43
25×25×25	75×75×75	1000	26.75	261.28	1482.43
33×33×33	63×63×63	216	15.31	152.95	862.77

Table 2 further explore the average time per batch consumed by the the three DVC programs in the precomputation stage. The paDVC is approximately 9.8× and 54.9× faster than the muDVC and the seDVC, respectively.

Table 3 gives comparisons of average computation time per POI in the stage 2 (3D FFT-CC) and stage 3 (3D ICGN) among the three programs. The calculation of the 3D FFT-CC powered by GPU leads to a speedup ranging from 4.6× to 22.0× over the two CPU-based implementations. And the GPU accelerates the operation of 3D ICGN algorithm in the paDVC for about 3.2 times to 19.8 times, as compared with the muDVC and seDVC. According to the results, it can be found that the most remarkable acceleration of the paDVC is achieved in the precomputation step, which is performed in a highly parallel manner. In the paDVC, each CUDA block can calculate the gradients at 216 voxels and the interpolation coefficients for 147 eight-voxel-units simultaneously, whereas the degree of parallel computing in the muDVC is 12 at most. In the other two steps, the speedup ratios achieved by the paDVC decrease markedly due to the limited CUDA cores in the GPU used in this work, which may allow only a couple of POIs to be processed in parallel. However, the fine-grained parallel computing applied in the calculation at each POI makes the paDVC demonstrate considerably superior computational efficiency over the muDVC.

Table 3 Comparison of computation efficiency for the 3D FFT-CC algorithm and the 3D ICGN algorithm among the paDVC, the muDVC and the seDVC

Subvolume [voxel]	Average computation time consumed per POI by the 3D FFT-CC algorithm [ms/POI]			Average computation time consumed per POI by the 3D ICGN algorithm [ms/POI]		
	paDVC	muDVC	seDVC	paDVC	muDVC	seDVC
17×17×17	0.005	0.030	0.088	0.501	1.55	9.02
25×25×25	0.025	0.114	0.550	1.104	3.40	21.18
33×33×33	0.042	0.235	0.740	2.146	7.31	47.75

Conclusion

In this work, a parallel digital volume correlation (paDVC) method is proposed and implemented on GPU devices. The path-dependence of the conventional reliable initial guess transferring schemes widely used in iterative DVC algorithms is eliminated in the proposed paDVC method by introducing the 3D FFT-CC algorithm to estimate the integer-voxel initial guess for the 3D ICGN algorithm at each POI independently. Then a GPU-based parallel implementation of the proposed DVC method has been developed on CUDA. The proposed paDVC employs a variety of techniques and strategies to combine parallel computing and batch processing at coarse-grained and fine-grained level, and demonstrates a significantly enhanced computational efficiency more than 20 times higher than the CPU-based sequential implementation of the same DVC method, without sacrificing accuracy and precision. In comparison with the multithreaded implementation on multi-core CPU, the paDVC also gains an approximately 3× improvement in speed. Therefore, one may reach a conclusion that GPU-based parallel computing could be a superior option to accelerate DVC over the parallel computing based on current multi-core CPU, unless an explosive increase of CPU cores is realized. Moreover, since a high-end CPU with more than 8 cores is still quite expensive, the paDVC provides the end-users the possibility to carry out large scale DVC computation efficiently on their desktop computers equipped with low or medium level graphics cards for measurement or research work.

Acknowledgments The work is partially supported by a grant, MOE2011-T2-2-037 (ARC 4/12), Ministry of Education, Singapore, the Multi-plAtform Game Innovation Centre (MAGIC) funded by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority, and National Natural Science Foundation of China (NSFC Nos. 11202081 and 11272124). Z Jiang would acknowledge the support of the Project sponsored by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

References

1. Bay BK, Smith TS, Fyhrie DP, Saad M (1999) Digital volume correlation: three-dimensional strain mapping using X-ray tomography. Exp Mech 39(3):217–226
2. Smith TS, Bay BK, Rashid MM (2002) Digital volume correlation including rotational degrees of freedom during minimization. Exp Mech 42(3):272–278
3. Peters WH, Ranson WF (1982) Digital imaging techniques in experimental stress analysis. Opt Eng 21(3):427–431
4. Sutton MA, Orteu JJ, Schreier H (2009) Image correlation for shape, motion and deformation measurements: basic concepts, theory and applications. Springer, New York
5. Zauel R, Yeni Y, Bay B, Dong X, Fyhrie D (2006) Comparison of the linear finite element prediction of deformation and strain of human cancellous bone to 3D digital volume correlation measurements. J Biomech Eng 128(1):1–6
6. Franck C, Hong S, Maskarinec S, Tirrell D, Ravichandran G (2007) Three-dimensional full-field measurements of large deformations in soft materials using confocal microscopy and digital volume correlation. Exp Mech 47(3):427–438
7. Huang J, Pan X, Li S, Peng X, Xiong C, Fang J (2011) A digital volume correlation technique for 3-D deformation measurements of soft gels. Int J Appl Mech 3(2):335–354
8. Forsberg F, Sjödahl M, Mooser R, Hack E, Wyss P (2010) Full three-dimensional strain measurements on wood exposed to three-point bending: analysis by use of digital volume correlation applied to synchrotron radiation micro-computed tomography image data. Strain 46(1):47–60
9. Hall S, Bornert M, Desrues J, Pannier Y, Lenoir N, Viggiani G, Bésuelle P (2010) Discrete and continuum analysis of localised deformation in sand using X-ray μCT and volumetric digital image correlation. Geotechnique 60(5):315–322
10. Gates M, Heath MT, Lambros J (2015) High-performance hybrid CPU and GPU parallel algorithm for digital volume correlation. Int J High Perform Comput Appl 29(1):92–106
11. Gates M, Lambros J, Heath M (2011) Towards high performance digital volume correlation. Exp Mech 51(4):491–507
12. Pan B, Wu D, Wang Z (2012) Internal displacement and strain measurement using digital volume correlation: a least-squares framework. Meas Sci Technol 23(4):045002
13. Pan B, Wang B, Wu D, Lubineau G (2014) An efficient and accurate 3D displacements tracking strategy for digital volume correlation. Opt Laser Eng 58:126–135
14. Baker S, Matthews I (2004) Lucas-kanade 20 years on: a unifying framework. Int J Comput Vision 56(3):221–255
15. Pan B (2009) Reliability-guided digital image correlation for image deformation measurement. Appl Optics 48(8):1535–1542
16. Pan B, Li K (2011) A fast digital image correlation method for deformation measurement. Opt Laser Eng 49(7):841–847
17. Jiang Z, Kemao Q, Miao H, Yang J, Tang L (2015) Path-independent digital image correlation with high accuracy, speed and robustness. Opt Laser Eng 65:93–102
18. Zhang L, Wang T, Jiang Z, Kemao Q, Liu Y, Liu Z, Tang L, Dong S (2015) High accuracy digital image correlation powered by GPU-based parallel computing. Opt Laser Eng 69:7–12
19. Bar-Kochba E, Toyjanova J, Andrews E, Kim K-S, Franck C (2015) A fast iterative digital volume correlation algorithm for large deformations. Exp Mech 55(1):261–274
20. Leclerc H, Périé J-N, Hild F, Roux S (2012) Digital volume correlation: what are the limits to the spatial resolution? Mechanics Industry 13(6):361–371
21. Schreier HW, Braasch JR, Sutton MA (2000) Systematic errors in digital image correlation caused by intensity interpolation. Opt Eng 39(11):2915–2921
22. Pan B, Xie H, Xu B, Dai F (2006) Performance of sub-pixel registration algorithms in digital image correlation. Meas Sci Technol 17(6):1615–1621
23. Pan B (2013) Bias error reduction of digital image correlation using Gaussian pre-filtering. Opt Laser Eng 51(10):1161–1167
24. Iwamura M, Hondo T, Noguchi K, Kise K (2007) An attempt of CUDA implementation of PCA-SIFT (International Session 6). Technical Report IEICE PRMU 107(281):149–154
25. http://docs.nvidia.com/cuda/cufft/
26. Cook S (2013) CUDA programming: a developer's guide to parallel computing with GPUs. Applications of GPU Computing. Morgan Kaufmann, Waltham
27. http://www.openmp.org/
28. Frigo M, Johnson SG (2005) The design and implementation of FFTW3. Proc IEEE 93(2):216–231