Published in final edited form as:

IEEE Trans Ultrason Ferroelectr Freq Control. 2017 April ; 64(4): 694–705. doi:10.1109/TUFFC.2017.2661821.

A GPU-accelerated 3D Coupled Sub-sample Estimation Algorithm for Volumetric Breast Strain Elastography

Bo Peng,
Department of Biomedical Engineering, Michigan Technological University, Houghton, MI, 49931, USA. School of Computer Science, Southwest Petroleum University (SWPU), Chengdu, China

Yuqi Wang,
Medical Physics Department at the University of Wisconsin, Madison, WI 53705, USA

Timothy J Hall, and
Medical Physics Department at the University of Wisconsin, Madison, WI 53705, USA

Jingfeng Jiang
Department of Biomedical Engineering, Michigan Technological University, Houghton, MI, 49931, USA

Abstract

Our primary objective of this work was to extend a previously published 2D coupled sub-sample tracking algorithm for 3D speckle tracking in the framework of ultrasound breast strain elastography. In order to overcome heavy computational cost, we investigated the use of a graphic processing unit (GPU) to accelerate the 3D coupled sub-sample speckle tracking method. The performance of the proposed GPU implementation was tested using a tissue-mimicking (TM) phantom and *in vivo* breast ultrasound data. The performance of this 3D sub-sample tracking algorithm was compared with the conventional 3D quadratic sub-sample estimation algorithm. On the basis of these evaluations, we concluded that the GPU implementation of this 3D sub-sample estimation algorithm can provide high-quality strain data (*i.e.* high correlation between the pre- and the motion-compensated post-deformation RF echo data and high contrast-to-noise ratio strain images), as compared to the conventional 3D quadratic sub-sample algorithm. Using the GPU implementation of the 3D speckle tracking algorithm, volumetric strain data can be achieved relatively fast (approximately 20 seconds per volume [2.5 cm × 2.5 cm × 2.5 cm]).

Index Terms

Speckle Tracking; Motion Tracking; Ultrasound Elastography; Graphic Processing Unit; Strain Elastography

I. Introduction

Ultrasound strain elastography (SE) [1] is an imaging method that can be used to non-invasively estimate tissue strains. Strain images can be formed by acquiring a frame (in 2D;

Contact: jjiang1@mtu.edu; Contact Phone Number: 1-906-487-1943.

volume in 3D) of radio-frequency (RF) echo signal data before and after a small deformation of the tissue and tracking the motion that occurred between the acquisitions. If the stress distribution is relatively uniform, local strain is largely inversely proportional to tissue stiffness. Ultrasound SE has shown promise in differentiating breast tumors [2], [3]. Recent advancements in SE can be found in recent review articles [4], [5]. The potential of elastography has resulted in several major vendors (*e.g.*, General Electric, Siemens, Philips, Hitachi, Toshiba, Samsung Medison *etc.*) releasing commercially-available SE packages. The majority of speckle tracking algorithms that are used to estimate tissue deformation is correlation-based and therefore, motion tracking suffers from tracking errors due to echo signal de-correlation. A major source of signal de-correlation in 2D SE is out-of-plane motion (motion that is perpendicular to the imaging plane) because only 2D ultrasound data are available. Strain artifacts due to tracking errors have been a major confounding factor in the interpretation of strain images [6].

Consequently, tracking tissue deformation through 3D ultrasound data to obtain 3D strain data is highly desirable. Reports in the literature have clearly demonstrated that 3D tracking minimizes motion tracking errors due to out-of-plane motion, as compared to results obtained from 2D speckle tracking [7]–[13]. Furthermore, during radiological evaluation, viewing 3D strain data in multiple parallel planes and/or perpendicular planes can often further aid diagnosis.

3D ultrasound data are becoming readily available [14]. Developments of plane wave ultrasound data acquisition [15], [16] may further accelerate this process. As well articulated by Yang *et al.* [17], there are great needs to preserve image quality in clinical workflow while achieving high frame rates for real-time or nearly real-time ultrasound SE. This study investigates whether or not a more sophisticated sub-sample estimation algorithm [18] extended to 3D data can further enhance the image quality of SE.

Toward this end, our main objectives of this study were twofold. First, we intended to demonstrate whether, in addition to tracking in 3D, the quality of strain data could be further improved by adoption of an accurate sub-sample displacement estimation algorithm. Second, we wanted to realistically assess the feasibility of introducing this high-quality sub-sample estimation algorithm, which is computationally expensive, into the clinical workflow. In order to do so, a graphic processing unit (GPU) was chosen to improve its computational efficiency through massive parallelization. It is worth noting that GPU-based elastography has drawn interests in recent years, through prior work was concentrated on 2D SE applications [17], [19]–[21] or accumulation of fairly small echo strain due to thermal expansion [22], [23]. The work reported here is the first to use a GPU card to accelerate 3D strain imaging formation in breast applications where the tissue volume underwent modest (1–2%) average strain from frame/volume to frame/volume. In this study, the performance of the 3D coupled sub-sampled tracking algorithm was compared to that of a conventional 3D quadratic sub-sample tracking strategy. Different GPU implementation strategies of the proposed 3D sub-sample speckle tracking were also compared against each other. Those performance evaluations included assessments of both accuracy of the displacement estimates and computational efficiency.

II. Materials and Methods

A. Computer Hardware and Software

Implementation and subsequent testing of the 3D speckle tracking algorithm in this work were performed on a professional Tesla K20 GPU card (Nvidia Corp., Santa Clara, CA). The K20 card has 13 Stream Multiprocessors (SMs), each containing 192 cores for a total of 2496 cores operating at 0.706GHz. This GPU card was installed in a WINDOWS workstation with a 3.10-GHz Xeon CPU E3-1220 V2 and 16GB memory (Intel Corp.; Santa Clara, CA). The K20 card had 5 GB of on-board global memory.

Compute Unified Device Architecture (CUDA, NVIDIA Inc., Santa Clara, CA) was used to program the K20 card. All GPU codes were compiled using the nvcc compiler from CUDA 7.0 (Nvidia Corp., Santa Clara, CA) with the aid of the CUDA Toolkit 7.0 (Nvidia Inc.). All GPU codes were called from MATLAB (version 2015a, Mathworks Inc., MA, USA) using a mex-interface. Throughout this paper, we attempted to make technical terms related GPU consistent with CUDA manuals. These GPU terms were capitalized for clarity.

B. Introduction to GPU

In CUDA, a KERNEL is a (computing) function that executes on the GPU and, tasks defined within the KERNEL can be performed in parallel through multi-threading. As shown in Fig. 1, each KERNEL can invoke one GRID typically consisting of several (*e.g. n*) BLOCKs; each BLOCK can initiate a number of THREADs.

In order to further grasp how massive parallel computing is being executed on the GPU, it is important to note that the basic programming unit here is a streaming multi-processor (SM). Each SM is a collection of many streaming processors (SPs; also known as CUDA cores). Although a single THREAD is executed on a SP, memory access and GPU scheduling are largely managed at the SM level. As shown in Fig. 2, a GPU has its own on-board memory, which can be divided into two categories: off-chip memory and on-chip memory. On the one hand, the off-chip memory includes global memory, TEXTURE and L2 cache on the device and accessible for all SMs. The difference between the global memory and TEXTURE is that TEXTURE is read-only while global memory can be used both for read and write. The global memory is the main data storage but has high latency for data access. Latency of TEXTURE is lower than the global memory but higher than any on-chip memory described below.

On the other hand, the on-chip memory is integrated into SMs and private to each SM. Consequently, each SM can only access to a small amount of on-chip memory, though the latency for data access (both read or write) is low. On-chip memory is designed to reduce demands for the global memory in order to achieve rapid data access. As also shown in Fig. 2, in each SM, there are four types of on-chip memory: register, shared memory, L1 cache and read-only memory. The actual configuration of on-chip memory may be hardware-dependent. The fastest on-chip memory is the register while the other three types have similar latency values. For instance, on the K20 card, the combination of shared memory and L1 cache is 64 KB for each SM: 48 KB shared memory + 16 KB L1 cache or 16 KB shared memory + 48 KB L1 cache. Each SM can access to 65536 (32-bit) registers and the amount of registers is equivalent to 262KB data storage.

C. Need for Optimization of On-chip Memory Access

As well understood in the literature [17], [24], the most time-intensive component of a correlation-based speckle tracking algorithm, such as the Block-Matching Algorithm (BMA) [25], [26], is the calculation of correlation values. Fortunately, this process can be massively parallelized using GPUs [17], [19]–[21]. Typically, such a parallelization is done using a single instruction, multiple thread (SIMT) mode. This mode in the context of speckle tracking is referred as to an execution style in which a single BLOCK within a KERNEL function (see Fig. 1) can call a large number SPs (within a SM) to simultaneously calculate correlation values. In order to do so, each SM has to load two large chunks of RF data segments so that a large number of THREADs (*e.g.* 192 SPs in one SM for the K20) can use them to calculate correlation values. Considering the following scenario: If all RF data are not loaded onto on-chip memory private to the SM, either reloading RF data or using RF data in the off-chip memory would significantly increase wait-time.

One concrete but simplified example is provided below so that the memory requirement for 3D speckle tracking can be better put into perspective. In order to avoid potential confusion, it is important to make a distinction between the block matching patch of data for correlation analysis (the motion tracking "kernel") and the CUDA executable function (a KERNEL; capitalized). Under the framework of a 3D BMA, one direction search ranges are A, B and C for axial, lateral and elevational directions, respectively. Of note, the one direction search range above was defined as the maximal translation of the center of the tracking kernel along one given direction (*e.g.* upward or downward). Given a 3D tracking kernel with a size of I(axial) × J(lateral) × K(elevation) samples, we need to get access to RF data whose size is at least (2A+I+1) × (2B+J+1) × (2C+K+1) RF samples for a single THREAD. Simply put, such a RF segment is approximately 40 Kilobytes (KB) if we assume the following: $A = 20$ samples, $B = 5$ samples, $C = 5$ samples, $I = 40$ samples, $J = 5$ samples, $K = 5$ samples and, RF echo data are digitized in at least 12 bits. To our knowledge, several high-end clinical or research scanners provide 2 byte digitized RF data [27], [28]. The above-chosen parameters lead to a tracking kernel whose axial length is roughly 4 wavelengths long at 7.5 MHz, which are appropriate for tracking 1% strain for breast tissue at the 3-cm depth. Recall that the calculated 40KB above was only for one THREAD. Thus, it is difficult to deal with the memory requirement using the limited on-chip memory for 100+ THREADs which can simultaneously execute on a single SM. Recall that, as described in Section II-B, the total on-chip memory for each SM is approximately 300KB. Under the same conditions for 2D tracking (a 2D kernel and a 2D search region), the memory requirement reduces to 2 KB per THREAD, which can be easily managed by the private on-chip memory of a single SM.

D. Description of the 3D Speckle Tracking Method Using GP-GPU

The 3D speckle tracking algorithm follows the framework of BMA [25], [26]. Basically, one tracking kernel is selected in the pre-deformation RF echo data (*i.e.* reference data) for each location. The motion tracking process finds the most similar data in the post-deformation echo data (*i.e.* target data), given a pre-determined search region. As stated above, this 3D motion tracking strategy uses a 3D kernel and a 3D search region. It is worth noting, however, that the pre-deformation field is defined by a 3D Cartesian grid of kernel location centers, and the 3D displacement estimates are estimated one plane at a time. Therefore, we first obtain 3D displacement vectors in a frame by frame fashion and, then form a volume of 3D displacement vectors by stacking all frames of displacement data. Consequently, the overall algorithm design is similar to what has been presented in an early publication for 2D coupled speckle tracking [18]. In order to keep our presentation concise, our emphasis here is on GPU-implementation. As shown in Fig. 3, in order to estimate displacements on an image plane, the proposed algorithm executes the following five major steps. Tracking parameters (*e.g.* search range and median filter size) in each step are summarized in Table I. Those five steps will continue until displacement estimates from all image planes of interest are completed.

<u>In the first step</u>, tracking in multiple locations first determined a gross direction of the tissue motion — downward or upward. Then, a brute-force BMA was used to obtain integer displacements along the center line of this region of interest (ROI) [26]. The size of the search region was determined by the deformation of the tissue being imaged (*e.g.* 1 or 2%). Once all displacement vectors were obtained along the center line, a predictive search scheme [25], [26] was then be used to track integer displacement on a 2D computing grid (M (axial) × N(lateral)) that covered a region of interest (ROI) on an image plane. The predictive search reduced the search region to a small size (see Table I) based on the displacement estimates around the center line.

Given a 3D search range of A(axial) × B(lateral) × C(elevation) samples, we needed to calculate (2A+1) × (2B+1) × (2C+1) cross-correlation values for each displacement vector. Consequently, (2A+1) × (2B+1) × (2C+1) × M × N computing THREADs were needed to compute all required correlation values. As discussed in Section II-E, once the search region was reduced, we could possibly load all data into the on-chip memory to improve computational efficiency. At that point, our GPU program invoked a number of KERNELs. Because each KERNEL contained 8 BLOCKs and each BLOCK started 256 THREADs, our GPU program initiated 2048 THREADs for each KERNEL. The total number of THREADs used per KERNEL was based on the available SPs (*i.e.* 2496) on the K20 so that a good occupancy of the GPU could be achieved. The occupancy was typically 80%, meaning that the GPU was utilized for computing about 80% of the time. Recall that the GPU can only execute one physical THREAD at a given time on a single SP. CUDA automatically passes the requested THREADs counts to the GPU for execution; there is no need for user interventions.

<u>In the second step</u>, a simple median filter was used to remove large outliers (also known as peak-hopping errors [29]) from the initial (integer) displacement field. This process requires only $M \times N$ THREADS — one THREAD for each displacement vector. After large errors were removed for each location on the M×N computing grid, motion compensation was first performed using the corrected integer displacements. Then, a re-tracking was performed in a reduced search region (3 samples in each direction; see Table I), eliminating potential biases induced by the median filter. Consequently, normalized correlation coefficient (NCC) values

are estimated on a 7 × 7 × 7 grid. The same parallelization using on-chip memory described in **Step 1** is used in this step to calculate this 7 × 7 × 7 correlation map.

In the fourth step, the 3D correlation map was first up-sampled through spline interpolations to increase its resolution. Then, a pre-determined threshold $0.95\rho_{max}$, where ρ_{max} was the maximum of the local correlation map and used to extract an iso-surface. Extraction of an iso-surface on the correlation map falls into image segmentation and the classic Marching-cubes method [30] was used.

In the fifth step, the coordinates of all points located on the iso-surface were fit to an ellipsoid model, as follows,

$$\frac{(X-\delta_x)^2}{C_1}+\frac{(Y-\delta_y)^2}{C_2}+\frac{(Z-\delta_z)^2}{C_3}=1 \quad (1)$$

In Eqn. 1, six constants (Δ_x, Δ_y, Δ_z, C_1, C_2 and C_3) were solved in a least-squares fashion. Validity of Eqn. 1 can be mathematically proved based on theory of linear medical ultrasound systems [31]. The 2D case was formally proved by Jiang and Hall [18] and an extension to 3D tracking is straightforward. In short, through the surface fitting [32], the fitted center of the ellipsoid (Δ_x, Δ_y and Δ_z) represents the unknown displacement vector. Finally, three "corrected" integer displacements and three sub-sample estimates were combined to obtain the full displacement vector for each point on the M × N computing grid.

In **Steps 4–5**, only $M \times N$ THREADs were needed and thus, the level of parallelization was significantly lower, as compared to that in **Steps 1 and 3**.

For the sake of completeness, two additional CUDA programming strategies were employed to improve computational efficiency across all five steps when applicable. First, in order to increase memory bandwidth of GPUs, TEXTURE (memory) access [33] was used for storing 3D RF volumes prior to the calculation of cross-correlation. The TEXTURE technique (NVIDIA Inc., Santa Clara, California, USA) utilizes hardware to accelerate interpolations which are required to calculate the correlation function beyond the sampling rate of the original RF echo data in **Step 3**. Second, programming variables that require frequent access (*e.g.* axial and lateral search ranges) were locked in read-only memory (see Fig. 2) for rapid access.

E. Strategies toward Improvement of On-chip Memory Access

As discussed in Section II-C, if the search region is large (*e.g.* a brute-force search for 1–2% strain), the RF data requirements become too large to be accommodated within the on-chip memory (see Section II-B). Referring to **Steps 1 and 3** in Section II-D, the search region of the guided search (see Table I) can be significantly reduced (*e.g.* to a 5 × 5 × 5 grid). Given that and compression of RF data to 1-byte, the memory requirement for the RF data was reduced to 5.5 KB, thereby becoming manageable for on-chip memory access. Four variations have been implemented to test how the memory access may influence the computational efficiency. All RF data are compressed into 1 byte data in all four versions.

1. **GPU-Version 1:** In Steps 1 and 3, RF echo data were only stored and accessed using global memory. As noted before, global memory has the largest amount among the off-chip memory and can be used both for read and write.
2. **GPU-Version 2:** In Steps 1 and 3, RF echo data were stored and accessed only using TEXTURE. As noted, TEXTURE is a special type of global memory because hardware accelerations are available [33].
3. **GPU-Version 3:** In Steps 1 and 3, we loaded RF data from the target echo frames to the shared memory, while the RF data from the reference echo frames were left in TEXTURE. Of note, shared memory belongs to on-chip memory.
4. **GPU-Version 4:** In Steps 1 and 3, all required RF echo data were loaded into the shared memory.

F. Comparative Tracking Implementations

In order to demonstrate the performance (both in computational efficiency and strain/ displacement quality), the proposed 3D sub-sample estimation algorithm was compared to a 3D quadratic sub-sample estimation algorithm [34]. Mathematically, 3D quadratic sub-sample estimates can be calculated by using correlation values on a 3×3×3 grid to fit the following quadratic function [34]:

$$\rho(x,y,z)=a_1+a_2x+a_3y+a_4z+a_5xy+a_6xz+a_7yz+a_8x^2+a_9y^2+a_{10}z^2 \quad (2)$$

where $\rho(x, y, z)$ is a discrete correlation function estimated through the BMA as described above. Eqn. 2 can be solved in a least-squares approach.

This quadratic sub-sample displacement estimation algorithm [34] has also been implemented on the K20 GPU and is hereafter referred to as **GPU-Version Quadratic**. The implementation details from Steps 1–3 were identical to those illustrated in Fig. 3. Then, Steps 4–5 were replaced by the solution of Eqn. 2. Data available in the literature [34] and our unpublished testing results suggested that this 3D quadratic algorithm out-performed both 1D and 2D quadratic sub-sample displacement estimation algorithms used in [13]. Consequently, comparisons of the proposed 3D sub-sample algorithm with inferior 1D and 2D sub-sample algorithms [34] are not included.

The coupled sub-sample displacement estimation algorithm [18] as illustrated in Fig. 3 has also been implemented as a standalone CPU version using standard C to verify all versions of GPU implementation and was used as a benchmark. Hereafter, we refer the CPU implementation as to **CPU-version**.

In **GPU-Versions 1–4** and the **CPU-Version**, the speckle tracking parameters used are summarized in Table I. Other tracking kernels were also used but were solely for the purpose of assessments of computational efficiency.

G. Validation Experiments

The GPU implementation was tested using a tissue-mimicking phantom and one set of *in vivo* breast tissue data. Details of the phantom experiment have been reported elsewhere [13] and a brief description is included for the sake of completeness. In both the phantom experiment and *in vivo* breast scan, Axius Direct Ultrasound Research Interface software [28] was used for data acquisition. The RF echo sampling frequency was 40MHz, and ultrasound data was downloaded for off-line analysis.

During the phantom experiment, a prototype 9-MHz 2D CMUT array connected to a Siemens SONOLINE Antares (Siemens Health Care, Inc. Ultrasound Division, Mountain View, CA) was used to acquire RF echo data from a 100mm × 100mm × 70mm oil-in-gelatin phantom containing two 10mm diameter spherical inclusions (one center target and one corner target) that have a 5:1 elastic contrast with the background [35]. The CMUT transducer manufacture has been described by Daft *et al.* [36]. Ultrasound data was acquired using a robotic arm. Dimensions of acquired ultrasound data were 40mm (axial) × 37mm (lateral) × 30mm (elevation), resulting in 140 image planes. In each image plane, there were 312 lines. Overall, voxel size of volumetric ultrasound data was 19 μm (axial) × 119 μm (lateral) × 214 μm (elevation). There was approximately 1% strain between two sequential volumes of RF echo data.

Under a protocol approved by the institutional review board (IRB) at the University of Wisconsin, 3D volumetric ultrasound data were collected from a human subject using a Siemens S2000 automated breast volume scan system (ABVS; Siemens Medical (USA) Solution Inc., Mountain View, California). The ABVS system included a high-frequency 1D array ultrasound transducer (14L5) that was excited with an 11MHz pulse. The ultrasound transducer moved along elevation direction with a stepper-motor to obtain a sequence of 2D ultrasound echo frames. More details about this modified ABVS system can be found in [37]. The human subject had a biopsy-confirmed fibroadenoma (FA). During the ultrasound scan, the subject was instructed to hold her breath so that unintended physiological motion was minimal. Two volumes of ultrasound data were acquired; each volume covered 6cm (axial)×7.6cm (lateral)×4cm (elevation). Volume-averaged strain estimated from the distance the ABVS arm dropped was approximately 2.5%. The correlation cell size (full width at half maximum) for the ABVS (roughly 0.25 mm [axial]× 0.9 mm [lateral] × 2.0 mm [elevational]) was approximated with the auto-correlation of the echo signals from a uniform phantom with a high concentration of sub-wavelength spherical scatterers [37].

H. Data Analysis

In addition to assessments of computational efficiency, two metrics were chosen to compare the performance among six implementation strategies, namely, **GPU-Versions 1–4, GPU-Version Quadratic and CPU-Version**. The first metric was adopted from the "displacement quality metric" (DQM) method [6]. We use the normalized cross correlation (NCC) between the pre-deformation and motion-compensated post-deformation RF echo fields as a quantitative measure of motion tracking accuracy. The NCC gives a single summary measurement for the entire region of interest, with 1 indicating the best motion tracking accuracy.

The second metric was the weighted contrast-to-noise ratio (CNR) [38]:

$$CNR = \frac{|I_t - I_b|}{\sqrt{w_t {\sigma_t}^2 + w_b {\sigma_b}^2}} \quad (3)$$

where I and σ denote means and variances of signals, and subscripts b and t represent the background and target, respectively. The CNR (Eqn. 3) is weighted by areas of the background w_b and the target w_t, respectively. In each strain image, the lesion was manually segmented (representing the target) and the rest of strain image represented the background. Song et al. [38] demonstrated that consideration of the weighted area is necessary because the target and the background contribute differently to the noise estimates.

III. Results

In Subsections III-A, III-B and III-C, phantom data acquired by the CMUT transducer were used to evaluate computational efficiency. Results (Subsection III-D) obtained from the *in vivo* breast data were intended to demonstrate that the proposed GPU implementation can potentially be used in a clinical workflow. Both the phantom and *in vivo* breast data were used for image quality analysis (NCC and CNR).

A. Tests of Computational Efficiency

The influence of RF data formats was investigated using four different tracking kernels as shown in Fig. 4. All tracking parameters were identical to those shown in Table I except that the size of tracking kernel varied as stated in those figures. For reference, 61 and 69 axial samples are equivalent to 1.16 mm and 1.35 mm, respectively, while 7,9 and 11 beam lines are 1.05 mm, 1.35 mm and 1.65 mm, respectively. 3 elevation planes equal to 0.87 mm in space. Using the computing time required for Step 3 as an example, we found that using 4 byte RF data significantly increased (by roughly 50%) the required computing time, while the computing time for 1 byte and 2 byte RF data was comparable.

Overhead needed for transferring data between the computer CPU and GPU were also analyzed. In general, the overhead of data transferring (*i.e.* RF data from CPU to GPU and displacement estimates from GPU back to CPU) were similar among four GPU versions and only the results related to **GPU Version 2** are shown in Table II. As compared to the computational costs for cross-correlation calculations (see Fig. 4), the overhead involving data transfer were considerably small (approximately 2–4%).

The computational efficiency of **GPU-Versions 1–4** were examined using 1 byte RF data. As explained in Section II-C, RF data with high bit-depth (2 bytes and 4 bytes) are too large to be loaded into the shared memory so that implementation to **GPU-Versions 3&4** was not feasible for the current mainstream GPU cards (e.g. Nvidia K20 and K40). As shown in Fig. 5, the required time to complete Step 3 of the proposed speckle tracking algorithm (see Fig. 3) increased from **GPU-Version 1** to **GPU-Version 4**. We also compared the performance between the **CPU-version** and the **GPU-version 4** for the entire speckle tracking process and the Step 3, respectively. Fig. 6 clearly suggested that **GPU-Version 4** accelerated speckle tracking process roughly by a factor of 100. This observation was also valid both for the entire speckle tracking process and Steps 3 – 5.

(**GPU-Version 4**) was also compared to **GPU-Version Quadratic** for each step in the process of speckle tracking. The results are summarized in Table III. It is interesting to note that, in our implementation, the 3D quadratic sub-sample estimation (*i.e.* step 4 in Table III) requires neglectable time to complete. However, the computational load required for Step 4 was generally small (approximately 10% of the total time).

B. Comparisons Among Different Implementations

Using the same configuration, displacement estimates obtained from using the **CPU-Version** of the proposed algorithm were compared to **GPU-Versions 1–4**. Differences among those five versions of the same algorithm are neglectable, as shown by one example in Fig. 7. Consequently, the differences between the CPU version and any of GPU versions in terms of DQM and CNR were very small (< 0.1%) both for the TM phantom and *in vivo* breast data investigated.

C. Results from A Tissue Mimicking Phantom

Results from the center target contained in the tissue-mimicking (TM) phantom under an approximately 1.0% axial strain obtained with **GPU-Version 4** are shown in Fig. 8. Visually, estimated displacements obtained with the coupled sub-sample displacement estimation algorithm, particularly, the lateral and elevation displacements, appeared to be smoother and contained fewer tracking errors (see arrows in Fig. 8), as compared to their counterparts estimated by the **GPU-Version Quadratic**. Consequently, as shown in Fig. 8, the axial strain image obtained by the coupled sub-sample method contained slightly lower noise. The CNR values from axial strain images obtained from both methods for 40+ frames covering both the corner target (P#1) and the center targets (P#2) are displayed in Fig. 9(a). Similarly, we examined the NCC for motion tracking accuracy for both methods. Overall, the coupled sub-sample displacement estimation method outperformed the 3D quadratic sub-sample estimation algorithm both in terms of CNR and NCC, though differences are relatively small. Results are summarized in Table IV.

D. Results from One Set of in vivo Breast Data

Representative results from an *in vivo* fibroadenoma (FA) were used to demonstrate the performance of the 3D coupled tracking method. The volume-average strain was approximately 1%. Fig. 10 clearly showed that the 3D quadratic sub-sample method (*i.e.* **GPU-Version Quadratic**) made some suspected errors (see arrows in Fig. 10), while the 3D coupled tracking method was able to avoid them. The general appearance of the axial, lateral and elevation displacement images obtained using the 3D coupled tracking method is smoother as compared to results from the 3D quadratic method.

The NCC and CNR values were also calculated for this breast data set as shown in Fig. 11. Similar to results obtained with the TM phantom, the coupled 3D sub-sample speckle tracking algorithm outperformed the 3D quadratic sub-sample estimation algorithm in terms of CNR (0.86 ± 0.13 vs. 0.85 ± 0.13; mean ± one standard deviation) and NCC values (0.78 ± 0.09 vs. 0.76 ± 0.09).

IV. Discussion

An important contribution of this work is the extension of a 2D algorithm [18] to 3D coupled sub-sample estimation. The initial results (see Fig. 10) showed that the coupled tracking method can further enhance strain image quality (*i.e.* lower noise and higher motion tracking accuracy) using conventional ultrasound signals. Furthermore, this advantage can be achieved with only slightly higher computational demands, *i.e.*, an approximately 10% increase based on the data shown by Table III. We expect that the benefits of using a high-quality sub-sample estimation algorithm will be even more pronounced if a displacement accumulation scheme (also known as multiple compression technique [6], [18], [39]) is used to estimate large tissue deformation. Recall that, in the framework of multi-compression tracking, we first track tissue deformations through a long sequence of echo data and then map all deformations back to the initial reference state through interpolations before accumulation. It is easy to see that lower quality displacements from the previous step affect the quality of the current step and, this effect will carry forward. Hence, we anticipate that the availability of the 3D coupled sub-sample tracking algorithm may help for estimating nonlinear elastic parameters in breast tissues [40], [41] and tissue-mimicking materials [42], potentially in 3D. Algorithmic optimization for tracking 3D large tissue deformation is on-going.

In this work, GPU-accelerated motion tracking was applied to analysis of relatively large tissue deformation (*e.g.* 1–2%), while early work focused on either 2D SE [17], [19]–[21] or tracking thermal expansion [22], [23]. Particularly, we explored the utility of on-chip memory to further accelerate motion tracking. Recall that the storage capacity of on-chip memory is fairly limited (*e.g.* approximately 300KB for the K20 GPU). Thus, in order to do so, we chose two strategies: (1) bit compression for RF data and (2) reduction of the search range by a predictive search, similar to that described by Jiang and Hall [26]. By the combination of the bit compression and predictive search, we have achieved 5–8 frames/second as indicated by Table III, equivalent to 1 volume of strain data in 20–30 seconds for an approximately 2.5 cm × 2.5cm × 2.5cm volume of interest. We stipulate that the above-mentioned frame/volume rate is probably sufficient so that the proposed method may be used as an on-line post-processing method in a clinical setting: 20–30 seconds are on par with the time required to reconstruct a 3D ultrasound volume for visualization in the clinical workflow. The computational timing results of **GPU-Versions 1–4** provided in Section III-A were based on a specific computer workstation configuration described in Section II-A. We repeated same calculations reported in Section III-A using a different computer workstation (NVIDIA K40 under a Dell 5600 workstation with 64GB memory and an INTEL E2560 12-core CPU). Because the K40 GPU has 15% more SPs, the computational efficiency improved roughly by 10–20%, indicating our implementation can run faster given a GPU with higher clock speed, greater memory bandwidth, and more RAM.

In this work, we described the options chosen related to the construction of the GPU-accelerated 3D sub-sample estimation algorithm (see Fig. 3) and provided justification for our choices as a part of this feasibility study. However, the current GPU implementation can be further optimized. First, a well-known multi-resolution (coarse-to-fine) approach can be adopted to further reduce the computational cost, while maintaining the low rate of peak-hopping errors. Second, our current approach was built on the single instruction, multiple data (SIMD) mode. Currently, a multiple instructions, multiple data (MIMD) mode is available. In the MIMD mode, additional instructions can be executed while one instruction is waiting for data loading to be completed. Therefore, MIMD mode may further improve the computational efficiency as demonstrated by others [19]. Third, a sum-table scheme developed by Luo and Konofagou [43] showed a significant speedup of 1D correlation calculation. If their work can be extended to the 3D correlation analysis on GPU our algorithm can benefit from such a sum-table scheme.

Since speckle tracking is a common element in ultrasound elastography methods, the 3D sub-sample algorithm may be applicable to other elastography methods such as the shear wave elastography (SWE) and acoustic radiation force impulse (ARFI) [1], [44]. Incorporation of a more accurate 3D sub-sample estimation method like ours will, perhaps, improve the estimation accuracy of shear wave speed or ARFI parameters (*e.g.* peak-displacement).

It is also interesting to note that we used a median filter combining with a relatively large (6 wavelengths long) tracking kernel for motion tracking. This is a reasonable choice because, with the increase of kernel length, the probability of peak-hopping errors became low [29], [45]. Thus, those sparse peak-hopping errors can be corrected using a median filter. Particularly, results by Chen, Shi and Varghese [45] have already shown that peak-hopping errors were minimal after a large (10 wavelengths long) window had been applied to track 1D displacements. Certainly, general applicability of this approach requires more investigations.

Median filter has also been used for speckle tracking by Boctor *et al.* [22]. One notable distinction between theirs and ours is that the method by Boctor *et al.* applied the median filter after the sub-sample estimation had been completed. Consequently, the median filter was used to remove noise with a possibility of introducing biases to the resultant displacement estimates. Our method removes large errors at the integer level, as shown in Fig. 3. In order to avoid the potential biases after applying the median filter, the speckle tracking was redone at a reduced but still adequately large (*i.e.* a $7 \times 7 \times 7$) grid in the Step 3 (see Fig. 3 and Table I). Our observation was that, without this "re-tracking" process, displacement estimation errors will likely occur, particularly, in *in vivo* data where tissue motion is complex. As an example shown in Fig. 12, noticeable displacement estimation errors would have occurred (see the left column) if the the re-tracking process was not implemented. Furthermore, the errors were more pronounced in the lateral and elevation displacements (see Fig. 12).

V. Conclusion and Future Work

We have demonstrated that the improvement in speckle tracking can be accomplished by the adoption of a more accurate 3D sub-sample estimation method, as compared to the conventional 3D quadratic sub-sample estimation algorithm. Significant accelerations (as much as a factor of 100) were achieved after we had migrated the coupled 3D sub-sample estimation method from CPU to GPU. Consequently, this work demonstrated that 3D data acquisition using an ABUS system can be used to generate SE data, potentially in a clinical workflow, given the reasonably high volume rate (20 seconds per volume covering a 2.5 cm × 2.5 cm × 2.5 cm volume of interest.

We also found that, among all four GPU implementations, computational efficiency was the highest once we loaded all required RF data into the shared memory space of the GPU (*i.e.* **GPU-Version 4**). Furthermore, such computational efficiency came with no compromise in strain image quality for data tested. Further developments will be focused on strategies for accumulations of large tissue deformation so that reliable nonlinear modulus inversion can be tested and, hopefully, validated in clinical studies.

Acknowledgments

This study is partially funded by NIH grants (R15CA179409 and R01CA140271) and is also benefited from an academic hardware program by NVIDIA Inc. Bo Peng was supported in part by an oversea post-doctoral fellowship from the Chinese Scholarship Council in 2015. Dr. Bo Peng is currently also receiving funding from SWPU (No. 2014QHZ023) and an applied basic research program (Science and Technology Commission, Sichuan, China, No. 2016JY0201).

References

1. Shiina T, Nightingale KR, Palmeri ML, Hall TJ, Bamber JC, Barr RG, Castera L, Choi BI, Chou YH, Cosgrove D, Dietrich CF, Ding H, Amy D, Farrokh A, Ferraioli G, Filice C, Friedrich-Rust M, Nakashima K, Schafer F, Sporea I, Suzuki S, Wilson S, Kudo M. WFUMB Guidelines and Recommendations for Clinical Use of Ultrasound Elastography: Part 1: Basic Principles and Terminology. Ultrasound in Medicine and Biology. May; 2015 41(5):1126–1147. [PubMed: 25805059]
2. Itoh A, Ueno E, Tohno E, Kamma H, Takahashi H, Shiina T, Yamakawa M, Matsumura T. Breast disease: Clinical application of us elastography for diagnosis. Radiology. 2006; 239(2):341–350. [PubMed: 16484352]
3. Burnside ES, Hall TJ, Sommer AM, Hesley GK, Sisney GA, Svensson WE, Fine JP, Jiang J, Hangiandreou NJ. Differentiating benign from malignant solid breast masses with us strain imaging. Radiology. 2007; 245(2):401–410. [PubMed: 17940302]
4. Hall TJ, Barboneg PE, Oberai AA, Jiang J, Dord JF, Goenezen S, Fisher TG. Recent results in nonlinear strain and modulus imaging. Current Medical Imaging Reviews. 2011; 7(4):313–327. [PubMed: 22754425]
5. Wells PNT, Liang HD. Medical ultrasound: imaging of soft tissue strain and elasticity. Journal of The Royal Society Interface. 2011; 8(64):1521–1549.
6. Jiang J, Hall TJ, Sommer AM. A novel performance descriptor for ultrasonic strain imaging: a preliminary study. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Jun; 2006 53(6):1088–1102.
7. Insana, MF., Chaturvedi, P., Hall, TJ., Bilgen, Mg. 3-d companding using linear arrays for improved strain imaging. Ultrasonics Symposium, 1997. Proceedings., 1997 IEEE; Oct 1997; p. 1435-1438.
8. Konofagou EE, Ophir J. Precision estimation and imaging of normal and shear components of the 3d strain tensor in elastography. Physics in Medicine and Biology. 2000; 45(6):1553. [Online]. Available: http://stacks.iop.org/0031-9155/45/i=6/a=311. [PubMed: 10870710]
9. Chen X, Xie H, Erkamp R, Kim K, Jia C, Rubin JM, O'Donnell M. 3-d correlation-based speckle tracking. Ultrasonic Imaging. 2005; 27(1):21–36. [PubMed: 16003924]

10. Patil AV, Garson CD, Hossack JA. 3d prostate elastography: algorithm, simulations and experiments. Physics in Medicine and Biology. 2007; 52(12):3643. [PubMed: 17664564]
11. Rivaz, H., Fleming, I., Matinfar, M., Ahmad, O., Khamene, A., Choti, M., Hager, G., Boctor, E. Ablation monitoring with a regularized 3d elastography technique. 2008 IEEE Ultrasonics Symposium; Nov 2008; p. 308-312.
12. TGM, LJE, GAH, PRW. Freehand ultrasound elastography with a 3-d probe. Ultrasound Med Biol. 2008; 34(3):463–74. [PubMed: 17993244]
13. Fisher TG, Hall TJ, Panda S, Richards MS, Barbone PE, Jiang J, Resnick J, Barnes S. Volumetric elasticity imaging with a 2-d cmut array. Ultrasound in Medicine and Biology. 2010; 36(6):978–990. [PubMed: 20510188]
14. Solberg OV, Lindseth F, Torp H, Blake RE, Hernes TAN. Freehand 3d ultrasound reconstruction algorithmsa review. Ultrasound in Medicine and Biology. 2007; 33(7):991–1009. [PubMed: 17512655]
15. Tseng, L-Y., Li, P-C. 3d cardiac strain imaging using plane wave excitation and feature tracking. 2011 IEEE International Ultrasonics Symposium; Oct 2011; p. 740-743.
16. Hollnder B, Hendriks GA, Mann RM, Hansen HH, de Korte CL. Plane-wave compounding in automated breast volume scanning: A phantom-based study. Ultrasound in Medicine and Biology. 2016; 42(10):2493–2503. [PubMed: 27401958]
17. Yang X, Deka S, Righetti R. A hybrid CPU-GPGPU approach for real-time elastography. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Dec; 2011 58(12):2631–2645.
18. Jiang J, Hall TJ. A coupled subsample displacement estimation method for ultrasound-based strain elastography. Physics in Medicine and Biology. 2015; 60(21):8347. [Online]. Available: http://stacks.iop.org/0031-9155/60/i=21/a=8347. [PubMed: 26458219]
19. Deshmukh NP, Kang HJ, Billings SD, Taylor RH, Hager GD, Boctor EM. Elastography Using Multi-Stream GPU: An Application to Online Tracked Ultrasound Elastography, In-Vivo and the da Vinci Surgical System. PLoS ONE. 2014; 9(12):1–32.
20. Verma P, Doyley MM. Synthetic aperture elastography: a GPU based approach. Proc SPIE. 2014; 9040:90 401A–6.
21. Idzenga T, Gaburov E, Vermin W, Menssen J, Korte CLD. Fast 2-d ultrasound strain imaging: the benefits of using a gpu. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Jan; 2014 61(1):207–213.
22. Boctor EM, Deshmukh N, Ayad MS, Clarke C, Dickie K, Choti MA, Burdette EC. Three-dimensional heat-induced echo-strain imaging for monitoring high-intensity acoustic ablation. 2009:72 650R–12.
23. Liu D, ESE. Real-time 2-d temperature imaging using ultrasound. IEEE Transactions on Biomedical Engineering. Jan; 2010 57(1):12–16. *. [PubMed: 19884075]
24. Hein IA, O'Brien WD. Current time-domain methods for assessing tissue motion by analysis from reflected ultrasound echoes-a review. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Mar; 1993 40(2):84–102.
25. Zhu Y, Hall TJ. A modified block matching method for real-time freehand strain imaging. 2002; 24(3):161–176.
26. Jiang J, Hall TJ. A parallelizable real-time motion tracking algorithm with applications to ultrasonic strain imaging. Physics in Medicine and Biology. 2007; 52(13):3773–3790. [PubMed: 17664576]
27. Wilson T, Zagzebski J, Varghese T, Chen Q, Rao M. The ultrasonix 500rp: A commercial ultrasound research interface. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. 2006; 53(10):1772–1782.
28. Brunke SS, Insana MF, Dahl JJ, Hansen C, Ashfaq M, Ermert H. An ultrasound research interface for a clinical system. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Jan; 2007 54(1):198–210.
29. Walker WF, Trahey GE. A fundamental limit on delay estimation using partially correlated speckle signals. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Mar; 1995 42(2):301–308.

30. Lorensen WE, Cline HE. Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH Comput Graph. Aug; 1987 21(4):163–169. [Online]. Available: http://doi.acm.org/10.1145/37402.37422.
31. Meunier J, Bertrand M. Ultrasonic texture motion analysis: theory and simulation. IEEE Transactions on Medical Imaging. Jun; 1995 14(2):293–300. [PubMed: 18215833]
32. Yury, P. Ellipsoid fit. [Online]. Available: \url{https://www.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit
33. CUDA. C programming guide v7. 0. NVIDIA Corporation; Sep. 2015
34. Azar RZ, Goksel O, Salcudean SE. Sub-sample displacement estimation from digitized ultrasound rf signals using multi-dimensional polynomial fitting of the cross-correlation function. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Nov; 2010 57(11):2403–2420.
35. Madsen EL, Hobson MA, Shi H, Varghese T, Frank GR. Tissue-mimicking agar/gelatin materials for use in heterogeneous elastography phantoms. Physics in Medicine and Biology. 2005; 50(23): 5597. [Online]. Available: http://stacks.iop.org/0031-9155/50/i=23/a=013. [PubMed: 16306655]
36. Daft, C., Wagner, P., Bymaster, B., Panda, S., Patel, K., Ladabaum, I. cmuts and electronics for 2d and 3d imaging: monolithic integration, in-handle chip sets and system implications. IEEE Ultrasonics Symposium, 2005; Sept 2005; p. 463-474.
37. Wang Y, et al. Three-dimensional ultrasound elasticity imaging on an automated breast volume scanning system. Ultrasonic imaging. 2016 under review, no. XX, pp. XX–XX.
38. Song X, Pogue BW, Jiang S, Doyley MM, Dehghani H, Tosteson TD, Paulsen KD. Automated region detection based on the contrast-to-noise ratio in near-infrared tomography. Appl Opt. Feb; 2004 43(5):1053–1062. [PubMed: 15008484]
39. Du H, Liu J, Pellot-Barakat C, Insana MF. Optimizing multicompression approaches to elasticity imaging. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Jan; 2006 53(1):90–99.
40. Oberai AA, Gokhale NH, Goenezen S, Barbone PE, Hall TJ, Sommer AM, Jiang J. Linear and nonlinear elasticity imaging of soft tissue in vivo: demonstration of feasibility. Physics in Medicine and Biology. 2009; 54(5):1191. [PubMed: 19182325]
41. Goenezen S, Dord JF, Sink Z, Barbone PE, Jiang J, Hall TJ, Oberai AA. Linear and nonlinear elastic modulus imaging: An application to breast cancer diagnosis. IEEE Transactions on Medical Imaging. Aug; 2012 31(8):1628–1637. [PubMed: 22665504]
42. Pavan TZ, Madsen EL, Frank GR, Jiang J, Carneiro AAO, Hall TJ. A nonlinear elasticity phantom containing spherical inclusions. Physics in Medicine and Biology. 2012; 57(15):4787. [Online]. Available: http://stacks.iop.org/0031-9155/57/i=15/a=4787. [PubMed: 22772074]
43. Luo J, Konofagou EE. A fast normalized cross-correlation calculation method for motion estimation. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. Jun; 2010 57(6):1347–1357.
44. Palmeri ML, Nightingale KR. Acoustic radiation force-based elasticity imaging methods. Interface Focus. 2011
45. Chen H, Shi H, Varghese T. Improvement of elastographic displacement estimation using a two-step cross-correlation method. Ultrasound in Medicine and Biology. Jan; 2007 33(1):48–56. [PubMed: 17189046]

Biographies

Bo Peng received his B.E degree in computer science from Southwest Petroleum University (SWPU), China, in 2003. He received his MS and PhD degrees in computer science both from Sichuan University, China, in 2007 and 2014, respectively. He joined SWPU as a lecturer in 2007 and now is an Associate Professor of Computer Science. He is currently visiting Michigan Technological University as a post-doctoral researcher. His research interests include ultrasound elastography, GPU computation, and medical image analysis.

Yuqi Wang received his B.S. degree in Flight Vehicle Propulsion Engineering from Civil Aviation University of China, Tianjin, China, in 2003. He later joined the PhD program in Fluid Mechanics of Beihang University, Beijing, China. He was awarded the Ph.D. degree in 2010 from the Beihang University. He is currently a research associate with Medical Physics Department at the University of Wisconsin, Madison, WI, USA. His research interests include ultrasound measurement of biological tissue and *in vivo* assessments of blood pressure.

Timothy Hall received his B.A. degree in physics from the University of Michigan-Flint in 1983. He received his M.S. and Ph.D. degrees in Medical Physics from the University of Wisconsin-Madison in 1985 and 1988, respectively. From 1988 to 2002 he was in the Radiology Department at the University of Kansas Medical Center where he worked on measurements of acoustic scattering in tissues, primarily kidneys, on contrast-detail analysis in ultrasound imaging and on developing elasticity imaging techniques including early development of materials for elasticity imaging phantoms. In 2003 he returned to the University of Wisconsin-Madison where he is a Professor in the Medical Physics Department. At the University of Wisconsin, he has been working on characterization of breast and cervical tissues using medical ultrasound technology. Overall, his research interests continue to center on developing new image formation strategies based on acoustic scattering and tissue viscoelasticity and the development of test objects for system performance evaluation.

Jingfeng (JJ) Jiang received the BS and MS degrees in structural engineering from Zhejiang University, China, in 1995 and 1998, respectively. He obtained his MS in Computer Science and PhD in Civil Engineering in 2002 and 2003 from University of Kansas, respectively. He was with Department of Medical Physics at University of Wisconsin-Madison from 2003 to 2012, first as a post-doctoral associate and then as a research scientist. He is now an Assistant Professor of Biomedical Engineering at Michigan Technological University, Houghton, Michigan. His overall research interests stride the borders among imaging, biology, and computational sciences. At the University of Wisconsin, he mainly worked on algorithm developments for ultrasound elastography. More recently, he has expanded his research into advanced open-source elastography simulations (https://github.com/jjiang-mtu/virtual-breast-project), cardiovascular flow analytics and medical imaging analysis.

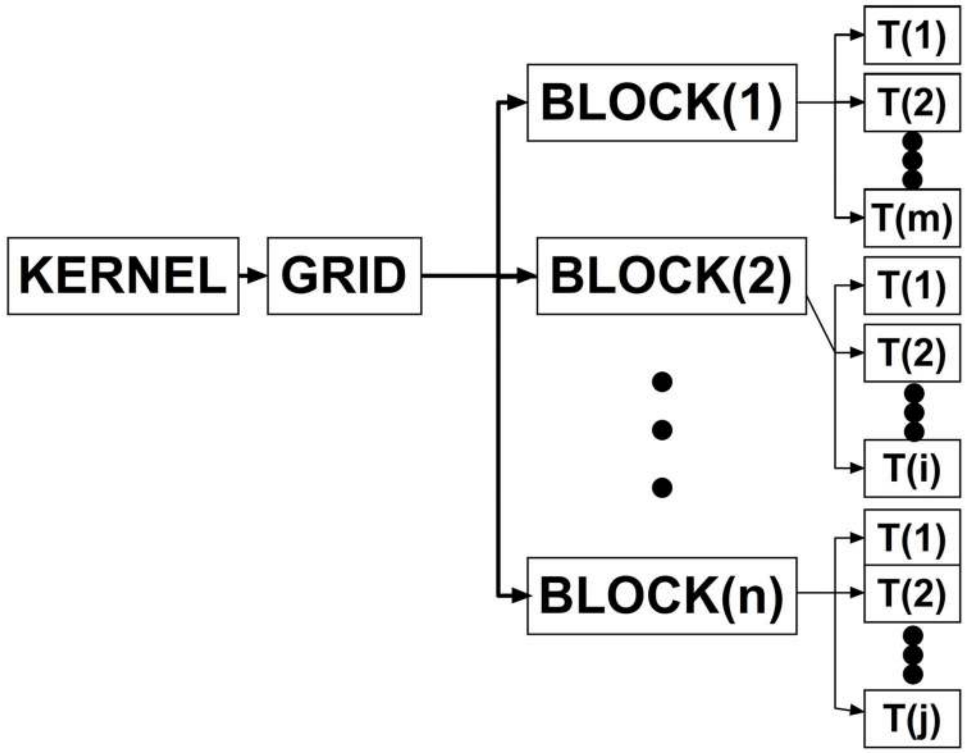

Fig. 1.
An graphic illustration of the relationship among KERNEL, BLOCK and THREAD in CUDA. BLOCK(X) and T(X) stand for the X^{th} BLOCK and THREAD, respectively.

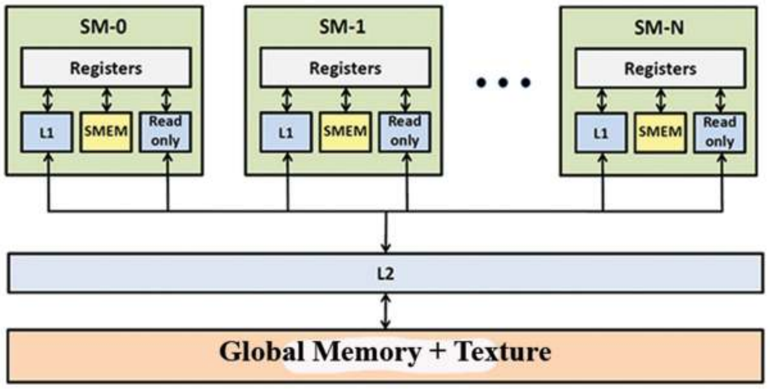

Fig. 2.
An pictorial illustration of memory hierarchy of typical GPU cards. SM-X denotes the X^{th} streaming multiprocessor and SMEM means the shared memory. L1 and L2 are Levels 1 and 2 caches following standard definitions in computer architecture, respectively.

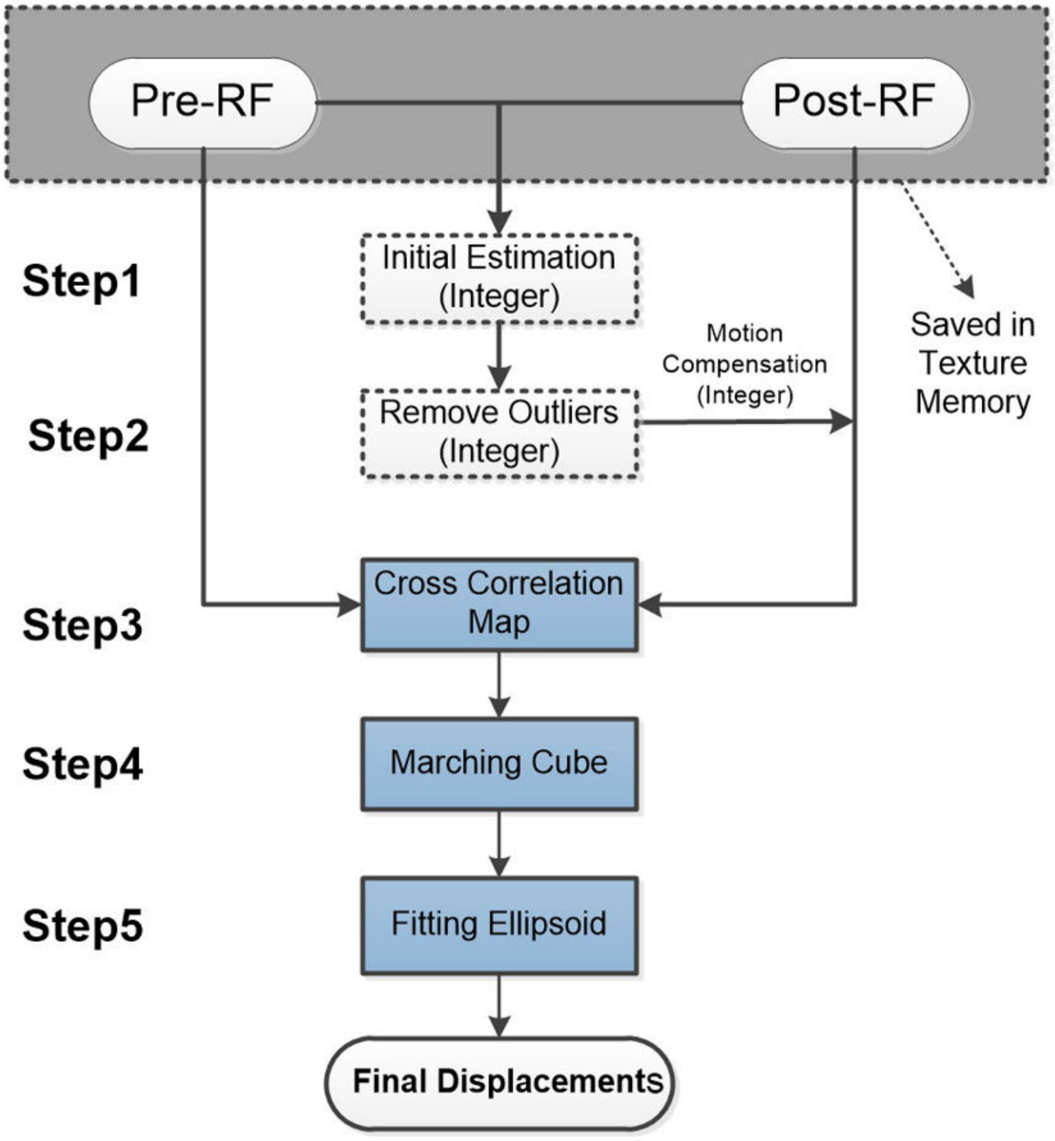

Fig. 3.
An illustration of the proposed speckle tracking estimation for estimating a plane of 3D displacements

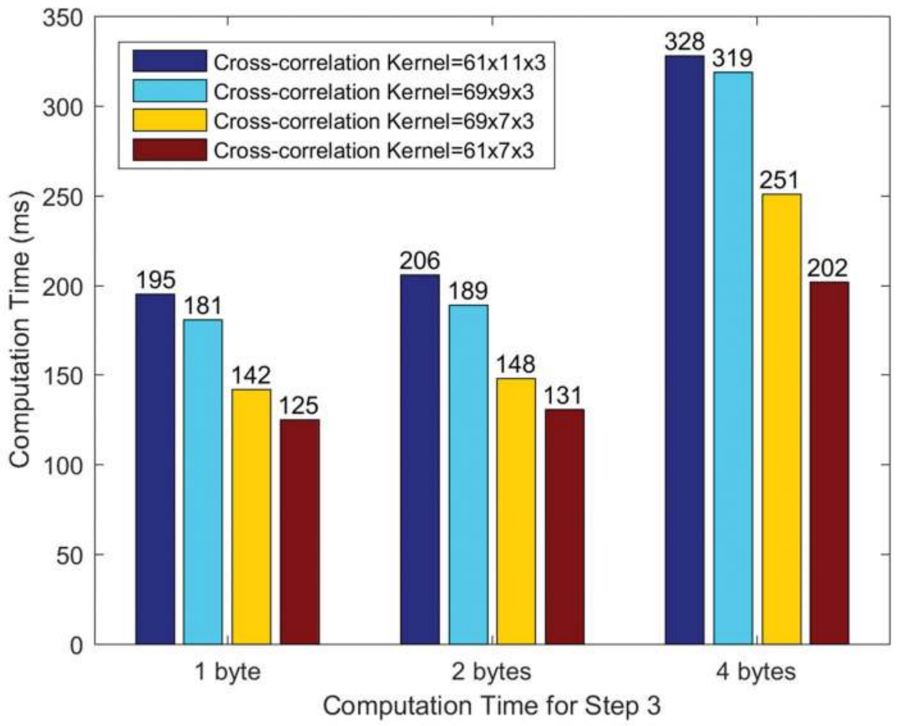

Fig. 4.

A plot comparing computational time needed for Step 3 in the **GPU-Version 2** algorithm. Four different tracking kernels were used and their sizes were in RF samples. Timing information was based on estimation of a single plane of (100 × 100) 3D displacements and RF samples were represented in three formats: 1, 2, and 4 bytes.

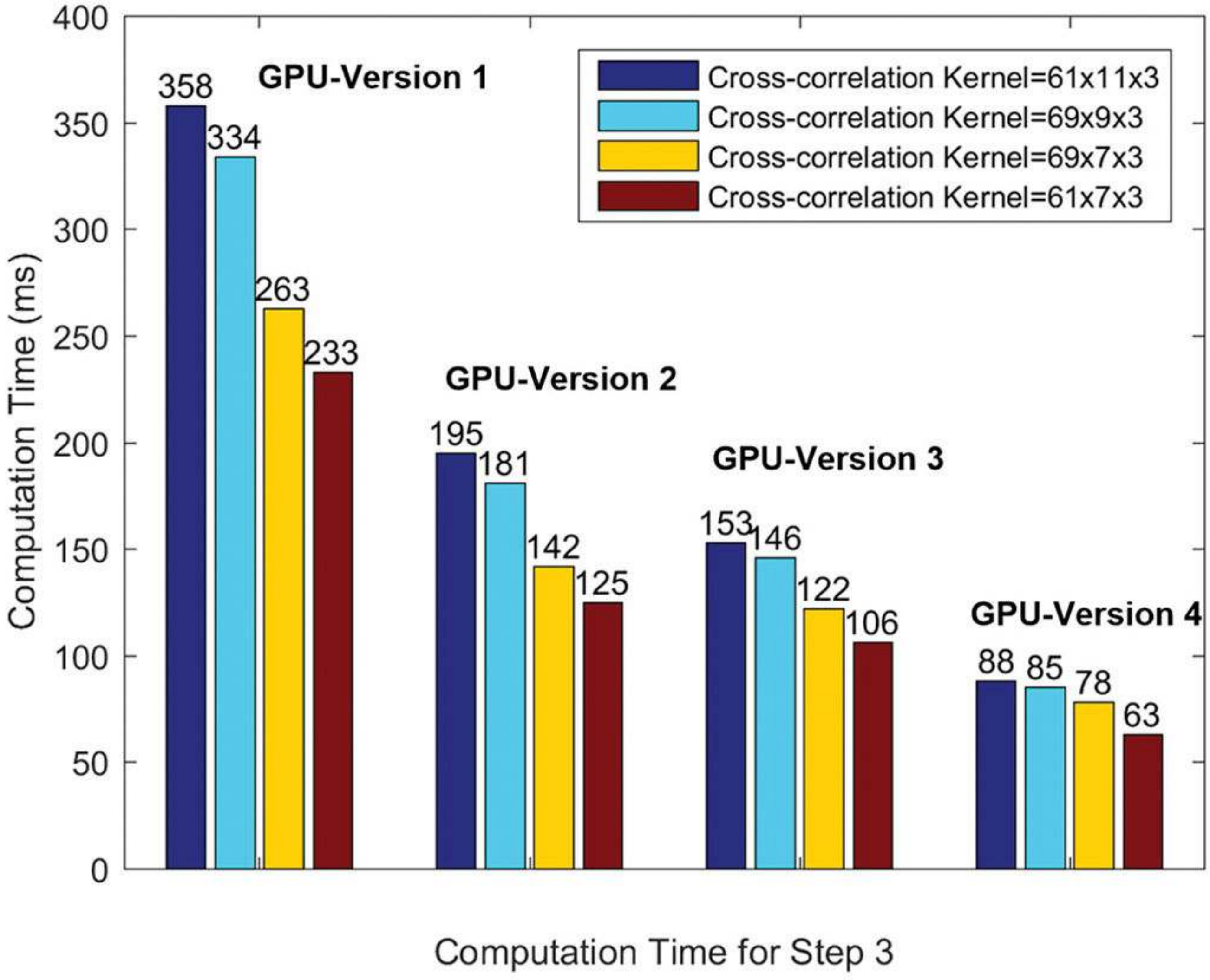

Fig. 5.

A plot comparing computational time needed for Step 3 using four different GPU-based variations: **GPU-Versions 1–4**. Testing was done to estimate a single plane of 3D displacements and four different tracking kernel sizes.

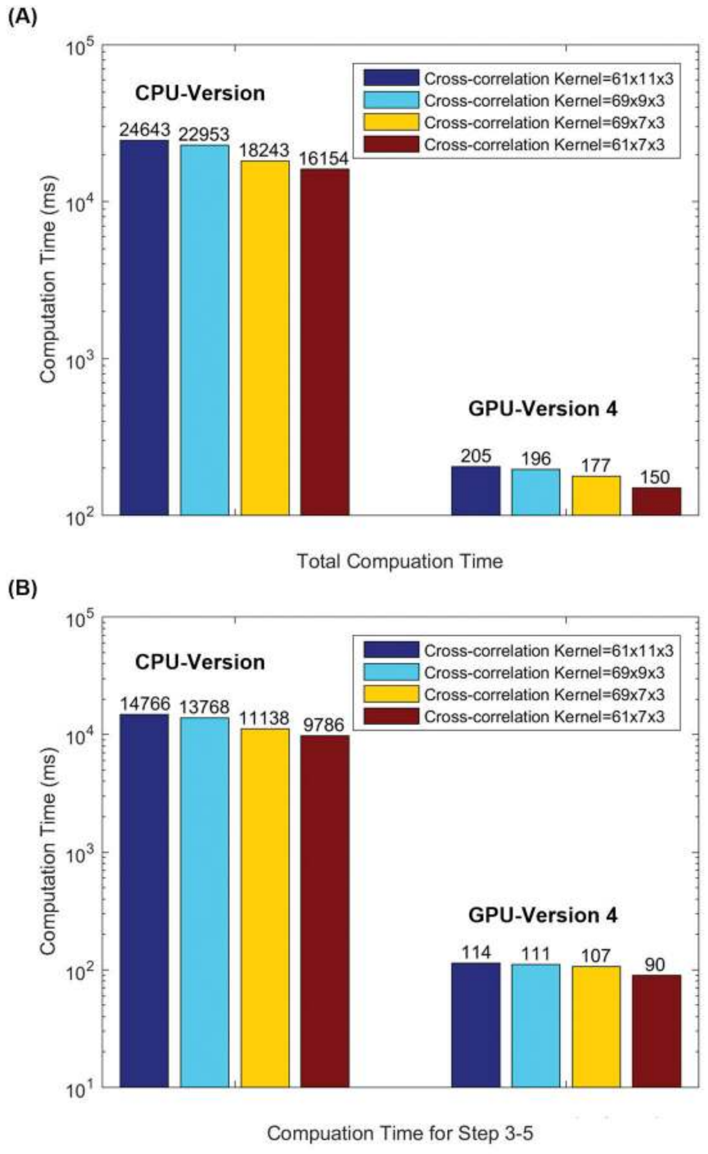

Fig. 6.

Plots comparing computational time between **CPU-version** and **GPU-Version 4** for: (a) the entire speckle tracking process and (b) Steps 3–5 of the proposed algorithm (see Fig. 3). Testing was done using four different tracking kernel sizes and the computational time was estimated for 100×100 displacement vectors on the TM phantom under an 1% compression.

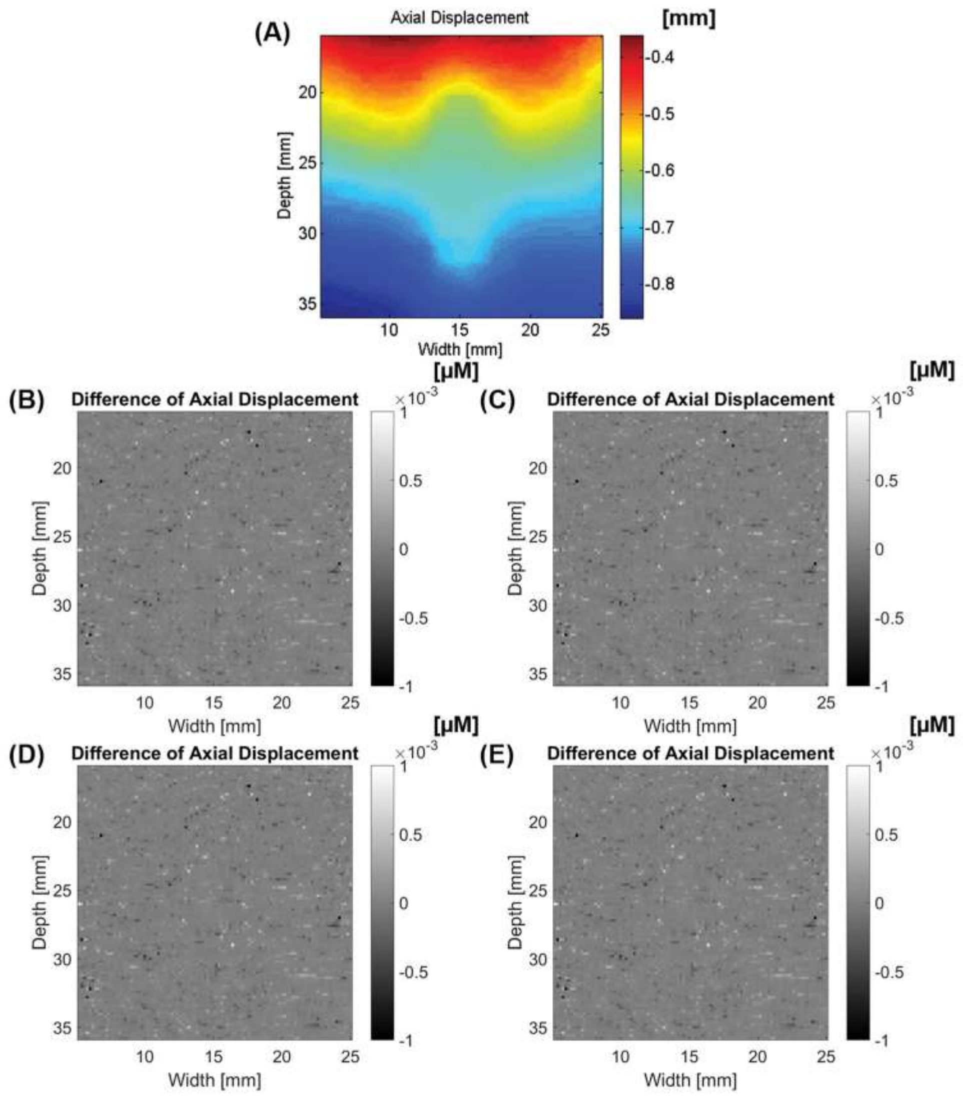

Fig. 7.

Axial displacement results from a TM phantom using (a) CPU-Version. (b)-(e) shows the displacement estimate difference (in micrometer) between the CPU-Version and the 4 GPU-Versoins: (b)GPU-Version 1, (c)GPU-Version 2, (d)GPU-Version 3; and (e) GPU-Version 4.

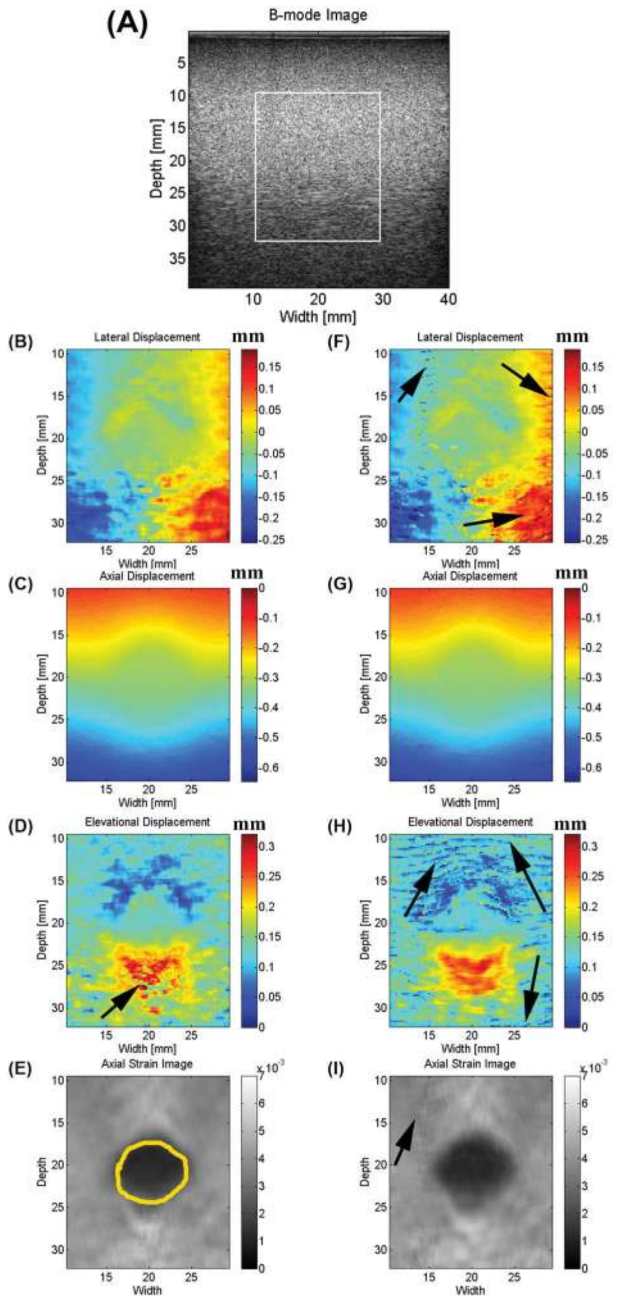
(A)
B-mode Image
Depth [mm]
Width [mm]
(B) Lateral Displacement mm
(F) Lateral Displacement mm
(C) Axial Displacement mm
(G) Axial Displacement mm
(D) Elevational Displacement mm
(H) Elevational Displacement mm
(E) Axial Strain Image x10^-3
(I) Axial Strain Image x10^-3
Depth
Width

Fig. 8.

Elastographic results from a TM phantom using two different sub-sample estimation algorithms (3D coupled [left column] vs. 3D quadratic [right column]): (a) A B-mode image; (b) and (f) are lateral displacement images, (c) and (g) are axial displacement images; (d) and (h) are elevation displacement images; (e) and (i) are axial strain elastograms. The white box in (a) indicates the ROI for displacement and strain estimation and, is the same as the image size in (B–I). Arrows point to visible displacement estimation errors and their impact on strain images. The manually-segmented contour in (e) was used to calculate CNR values based on Eqn. 3.

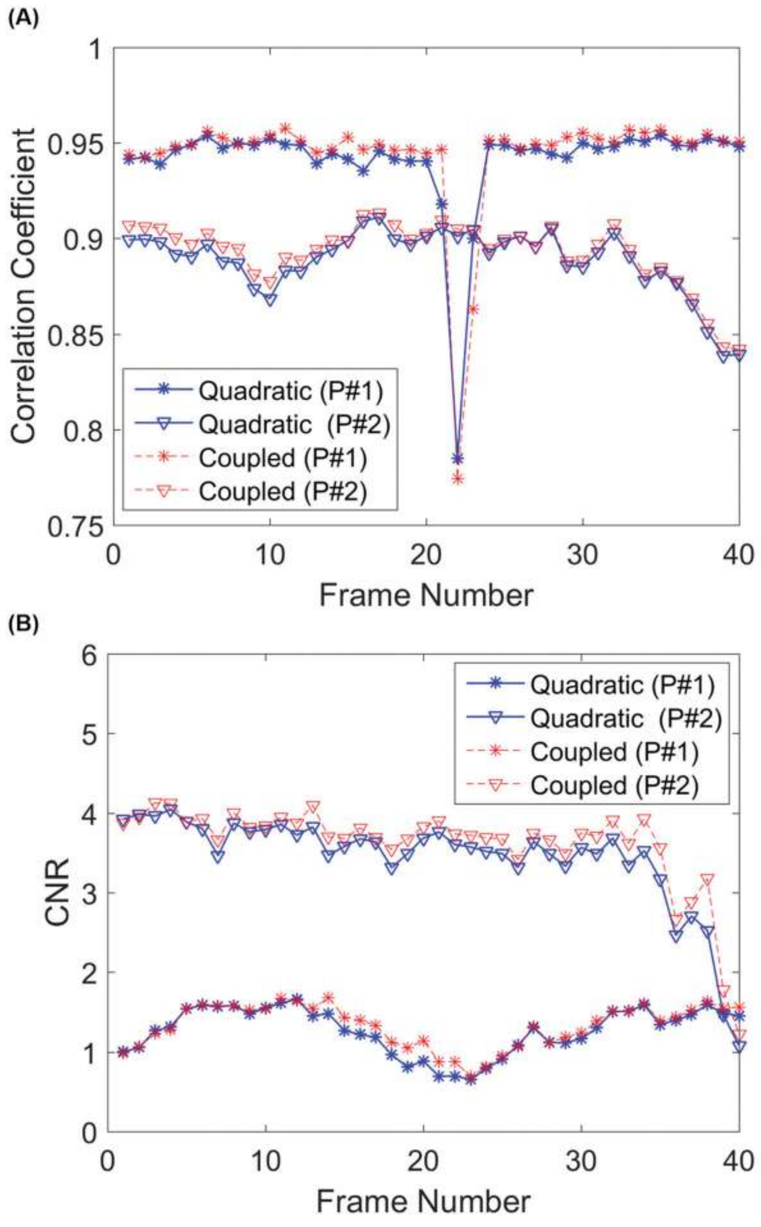

Fig. 9.
Estimated (a) NCC and (b) CNR values from the TM phantom using two different sub-sample estimation algorithms: 3D coupled vs. 3D quadratic. The estimated NCC and CNR values (y-axis) were displayed with respect to frame numbers (x-axis). P#1 and P#2 stand for the corner target and the center target, respectively.

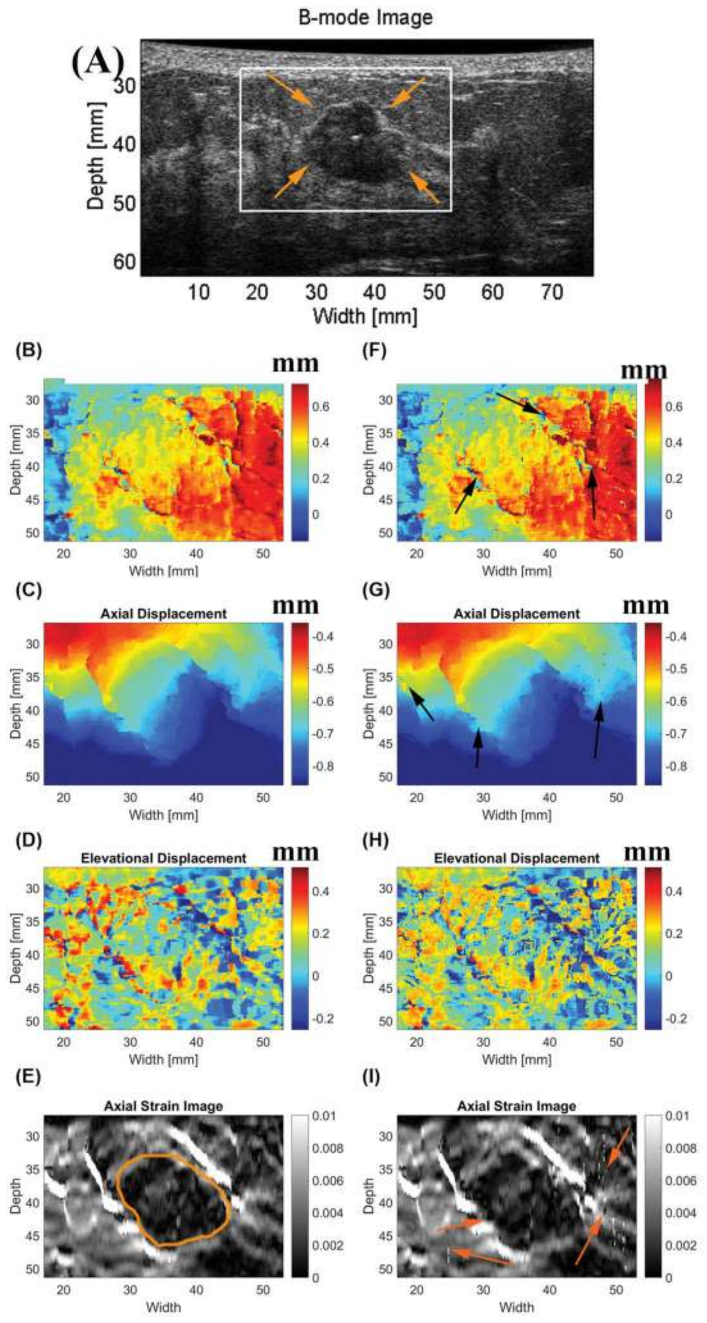

Fig. 10.

Elastographic results from a biopsy-confirmed breast fibroadenoma using two different sub-sample estimation algorithms (3D coupled [left column] vs. 3D quadratic [right column]): (a) A B-mode image; (b) and (f) are lateral displacement images, (c) and (g) are axial displacement images; (d) and (h) are elevation displacement images; (e) and (i) are axial strain elastograms. The white box in (a) indicates the area from which displacements and strains were estimated and has the same size as sizes of plots (b) (i). Arrows on (a) point to the FA, while arrows on other plots point to visible tracking noise. The manually-segmented contour in (e) was used to calculate CNR values based on Eqn. 3.

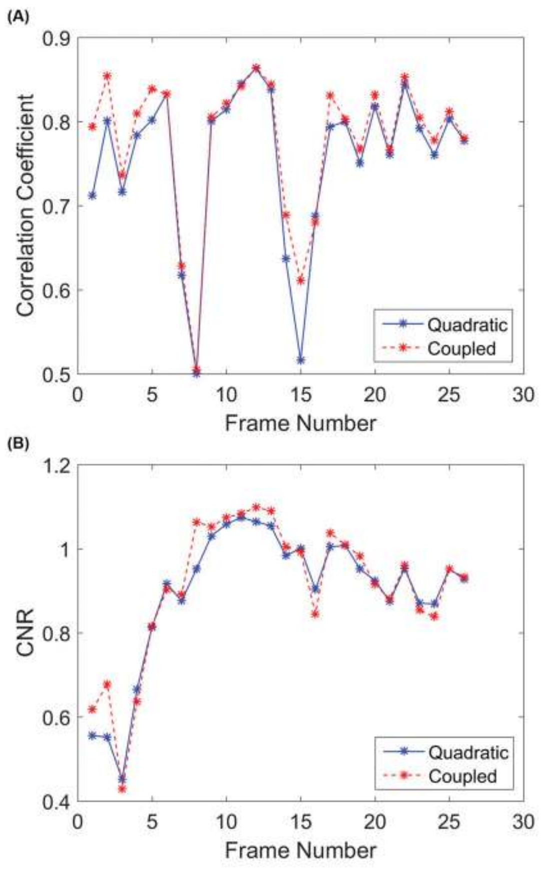

Fig. 11.
Estimated (a) NCC and (b) CNR values from the FA breast lesion using two different sub-sample estimation algorithms: 3D coupled vs. 3D quadratic. The estimated NCC and CNR values (y-axis) were displayed with respect to frame numbers (x-axis).

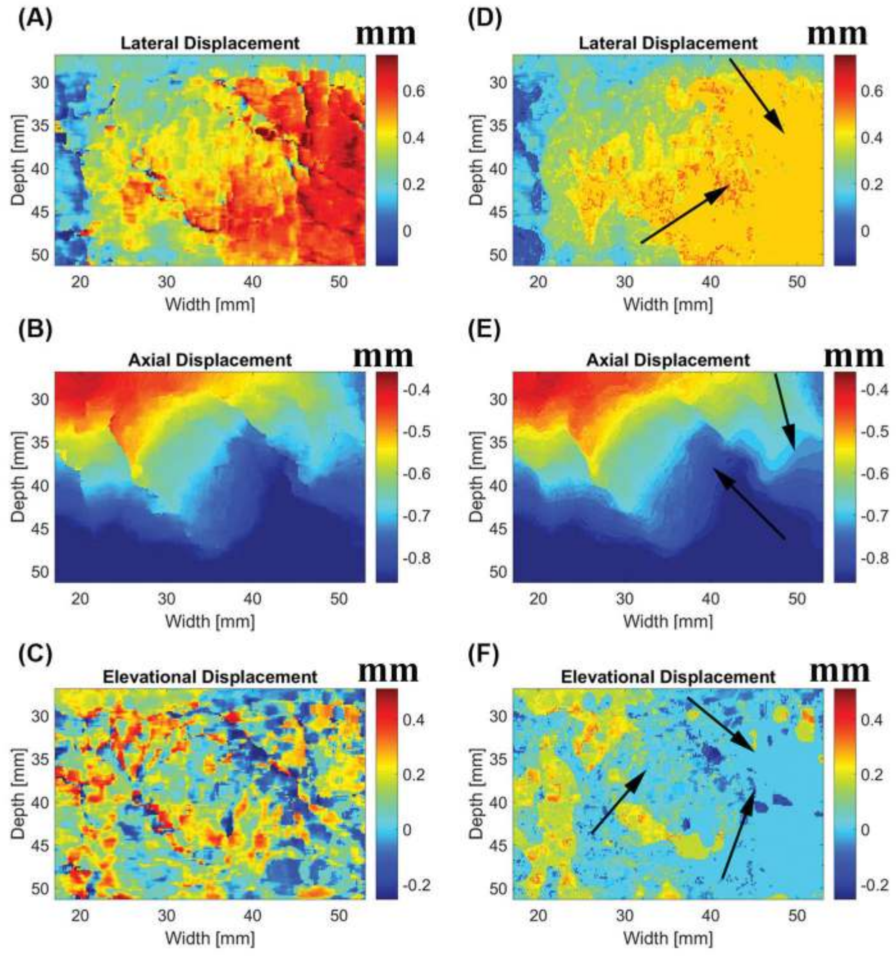

Fig. 12.

Displacement results from the *in vivo* FA using the 3D coupled tracking with and without the re-tracking strategy in Step 3: (a) and (d) are lateral displacement images, (b) and (e) are axial displacement images; and (c) and (f) are elevation displacement images. Images in the left and right columns are results with and without the re-tracking strategy. The tracking parameters used were same as those shown in Table I. Arrows point to visible tracking noise.

TABLE I

Tracking parameters used in the tissue-mimicking and in vivo breast data. AL = axial length of the tracking kernel; LL = lateral length of the tracking kernel; EL = elevation length of the tracking kernel; ASR = axial search range; LSR = lateral search range; ESR = elevation search range; All numbers are in samples. "ASR= −4~3" means that we searched 4 samples downward and 3 samples upward along the axial direction. Search regions in the other two directions followed the same convention, except different search directions.

		Data Set	
		Phantom Data	**In Vivo Data**

Tracking Kernel		AL=69
		LL=9
		EL=3
			AL=69
			LL=9
			EL=3

Step 1	Centerline Tracking	ASR=−43~0
		LSR=−2~2
		ESR=−2~2
			ASR=−51~0
			LSR=−2~2
			ESR=−2~2

	Predictive Tracking	ASR = −5~5
		LSR=−2~2
		ESR=−2~2
			ASR=−5~5
			LSR=−2~2
			ESR=−2~2

Step 2	Median Filter	7 samples in each direction	7 samples in each direction

Step 3	Search Range	ASR=−3~3
		LSR=−3~3
		ESR=−3~3
			ASR=−3~3
			LSR=−3~3
			ESR=−3~3

TABLE II

A summary of computational overhead (in millisecond) comparing 3 different data formats. GPU Version 2 was used for the testing and the tracking kernel size was 69 samples×9 samples×3 samples.

Data Format	CPU to GPU (millisecond)	GPU to CPU (millisecond)
1 byte	3.6	<0.1
2 bytes	7.1	<0.1
4 bytes	14.0	<0.1

TABLE III

A summary of computational cost (in millisecond) comparing the proposed algorithm with 3D quadratic sub-sample algorithm.

Method	Kernel Size	Step1	2	3	4	5	Total
Coupled	61 × 11 × 3	83	8	88	20	6	205
	69 × 9 × 3	77	8	85	20	6	196
	69 × 7 × 3	62	8	78	22	7	177
	61 × 7 × 3	52	8	63	21	6	150
Quadratic	61 × 11 × 3	83	8	89	<1	N/A	180
	69 × 9 × 3	77	8	86	<1	N/A	171
	69 × 7 × 3	62	8	79	<1	N/A	149
	61 × 7 × 3	52	8	64	<1	N/A	124

TABLE IV

A Summary of CNR and NCC values (mean ± one standard deviation) comparing the 3D quadratic sub-sample estimation (**GPU-Version Quadratic**) and the proposed 3D coupled sub-sample tracking method (**GPU-Version 4**) in the tissue-mimicking phantom

	Corner Target		**Center Target**	
	Quadratic	**Coupled**	**Quadratic**	**Coupled**
CNR	1.27±0.29	1.33±0.27	3.43±0.61	3.61±0.57
NCC	0.889±0.017	0.893±0.017	0.940±0.027	0.944±0.031