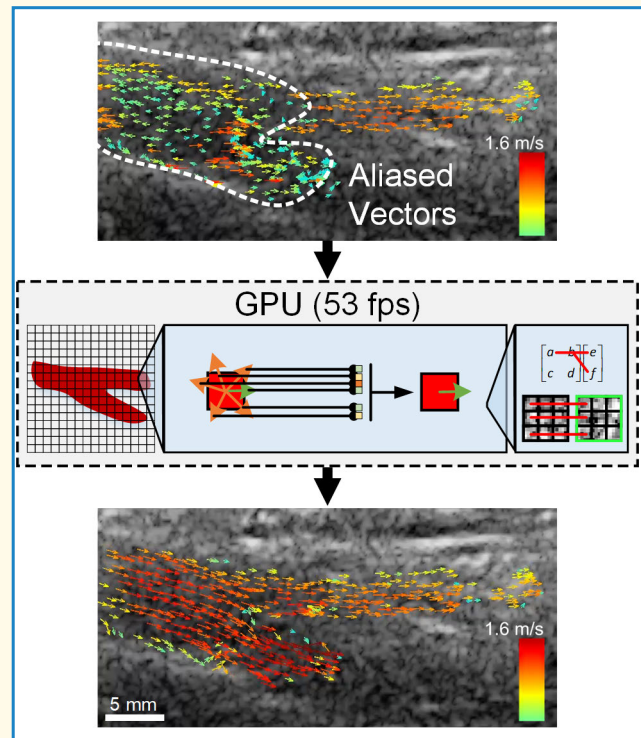


A GPU-Based, Real-Time Dealiasing Framework for High-Frame-Rate Vector Doppler Imaging

Hassan Nahas¹, Member, IEEE, Takuro Ishii², Member, IEEE, Billy Y. S. Yiu³, Member, IEEE, and Alfred C. H. Yu⁴, Senior Member, IEEE

Abstract—Vector Doppler is well regarded as a potential way of deriving flow vectors to intuitively visualize complex flow profiles, especially when it is implemented at high frame rates. However, this technique's performance is known to suffer from aliasing artifacts. There is a dire need to devise real-time dealiasing solutions for vector Doppler. In this article, we present a new methodological framework for achieving aliasing-resistant flow vector estimation at real-time throughput from precalculated Doppler frequencies. Our framework comprises a series of compute kernels that have synergized: 1) an extended least squares vector Doppler (ELS-VD) algorithm; 2) single-instruction, multiple-thread (SIMT) processing principles; and 3) implementation on a graphical processing unit (GPU). Results show that this new framework, when executed on an RTX-2080 GPU, can effectively generate aliasing-free flow vector maps using high-frame-rate imaging datasets acquired from multiple transmit–receive angle pairs in a carotid phantom imaging scenario. Over the entire cardiac cycle, the frame processing time for aliasing-resistant vector estimation was measured to be less than 16 ms, which corresponds to a minimum processing throughput of 62.5 frames/s. In a human femoral bifurcation imaging trial with fast flow (150 cm/s), our framework was found to be effective in resolving two-cycle aliasing artifacts at a minimum throughput of 53 frames/s. The framework's processing throughput was generally in the real-time range for practical combinations of ELS-VD algorithmic parameters. Overall, this work represents the first demonstration of real-time, GPU-based aliasing-resistant vector flow imaging using vector Doppler estimation principles.

Index Terms—Dealiasing, extended least squares, graphical processing unit (GPU), multiple-thread (SIMT) computing, real time, single instruction, vector Doppler.



Manuscript received 8 June 2023; accepted 5 August 2023. Date of publication 7 August 2023; date of current version 9 November 2023. This work was supported in part by the Natural Sciences and Engineering Council of Canada under Grant CREATE-528202-2019 and Grant RGPIN-2022-04157, in part by the Canadian Institutes of Health Research under Grant PJT-153240, in part by the E.W.R. Steacie Memorial Fellowship under Grant SMFSU-556263-2021, and in part by the Canadian Space Agency under Grant 21FAWATA08. (Corresponding author: Alfred C. H. Yu.)

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by Clinical Research Ethics Committee of the University of Waterloo under Application No. 31694.

Please see the Acknowledgment section of this article for the author affiliations.

This article has supplementary downloadable material available at <https://doi.org/10.1109/TUFFC.2023.3303349>, provided by the authors.

Digital Object Identifier 10.1109/TUFFC.2023.3303349

I. INTRODUCTION

HIGH frame-rate vector flow imaging is an emerging ultrasound technique for visualizing complex and transient in vivo flow dynamics in real time [1]. One recognized way of realizing this technique is to perform multi-angle Doppler estimation, where a set of plane wave pulses are transmitted sequentially from multiple steering angles [2]. For such an approach, which is often referred to as vector Doppler, the flow vector at each pixel position in the imaging view is derived via a two-step process. First, for each steering angle, pulsed Doppler estimation is carried out on every pixel position by individually processing the corresponding slow-time ensemble. Second, at each pixel position, the Doppler frequency values derived from different

Highlights

- This paper showcases a new GPU-powered, extended least-squares vector Doppler estimation framework that is resilient against multicycle aliasing artifacts.
- Real-time throughput of at least 53 frames/s can be achieved in vitro and in vivo in imaging scenarios with two-cycle Doppler aliasing and a wide velocity dynamic range from 0 to 1.6 m/s.
- With GPU computing, vector Doppler images without aliasing errors can be generated in real time to enable more intuitive visualization of blood flow in clinical diagnostics.

steering angles are used as the input to a least-squares fitting operation that computes the resulting flow vector from triangulation arguments and multiple instances of the Doppler equation [3].

Although performing vector flow imaging via Doppler principles has demonstrated robust estimation performance [4], it is known to be susceptible to Doppler aliasing errors, where, for any steering angle at a given pixel position, the Doppler frequency estimate may not correspond to the actual value induced by blood flow. The root cause for the emergence of Doppler aliasing is insufficient slow-time sampling, in which the pulse repetition frequency (PRF) used to acquire slow-time ensembles is lower than two times the maximum Doppler frequency induced by blood flow [5], thereby violating the Nyquist–Shannon sampling theorem. If left unaddressed, aliased Doppler frequency estimates at certain steering angles would ultimately lead to the derivation of spurious flow vectors due to erroneous least-squares fitting. In turn, they would significantly distort the appearance of the flow profile rendered in the imaging view [1].

Aliasing is particularly prone to arise in live implementations of high-frame-rate vector Doppler due to various physical and system-level constraints. From an imaging physics standpoint, the effective PRF used for vector Doppler processing is inevitably reduced by the use of multiple steering angles for robust vector flow estimation [4], because the time interval between frames with the same steering angle is concomitantly increased. Another important PRF constraint is the depth of the imaging target. Specifically, when performing pulse-echo sensing, the PRF might need to be kept low to allow sufficient time in between pulse firings, so that the transmitted pulse can realize two-way propagation to and from the maximum imaging depth. As such, aliasing artifacts tend to emerge more easily when performing vector Doppler mapping of fast blood flow in deep vessels, especially those in stenosed arteries [6], the carotid bifurcation [7], the urinary tract [8], and the heart [9]. From a system-level standpoint, the maximum usable PRF may be constrained by the system's raw data streaming bandwidth [10] and computational power [11] in some hardware implementations. In particular, it is known that PRF directly scales the number of data samples acquired by the array transducer [12]. If the PRF is too high, the size of the acquired raw data volume may be beyond what the system can stream in real time to the computing back-end and beyond what the available on-board computing resources can handle

to execute beamforming and signal processing operations for live imaging.

Resolving aliasing errors in vector Doppler is after all a challenging task, particularly if it needs to be done in real time. One solution is to resolve Doppler aliasing artifacts from each steering angle independently through the use of a dealiasing algorithm that is applied to every pixel position in the image frame [13], [14]. Based on a similar anglewise dealiasing approach, staggered transmissions [15], [16] and dual-wavelength [17] processing strategies have also been developed to resolve Doppler aliasing artifacts in cross-beam flow vector estimation (the baseline implementation of vector Doppler). Another approach to aliasing correction for vector Doppler is to holistically take into account all Doppler frequency estimates from all steering angles [18]. Transcending beyond these dealiasing approaches, an extended least squares vector Doppler (ELS-VD) algorithm [19] was introduced as a more robust dealiasing framework for vector Doppler. It works by synergizing least squares fitting [4] and speckle tracking via block matching [20] to effectively identify and resolve aliased Doppler frequency estimates. This solution has demonstrated efficacy in suppressing aliasing artifacts from in vivo vector Doppler maps acquired from the carotid bifurcation. However, the algorithm's reported processing throughput of 5 s per frame [19] is seemingly inadequate for real-time realization. As ultrasound is typically regarded as a real-time imaging modality that may be used in point-of-care settings [21], excessive offline processing would hinder clinical translation of vector Doppler as a potent vector flow imaging technique that is resilient against aliasing artifacts.

In this article, we present a new real-time dealiasing framework for vector Doppler estimation performed on high-frame-rate imaging datasets acquired from multiple plane wave steering angles. In devising our framework, we have worked with the guiding proposition that an effective dealiasing algorithm with real-time feasibility can be developed by harnessing the theoretical merit of the ELS-VD method and reformulating its execution using parallel computing principles and graphical processing unit (GPU) processing. We have particularly sought to exploit the multidimensional data independency inherent in the ELS-VD method to devise a single-instruction, multiple-thread (SIMT) parallel computing pipeline that can be readily executed with high processing throughput on the GPU that has many compute cores in its hardware. We shall demonstrate our framework's real-time

performance while reaping the theoretical benefits of ELS-VD to achieve robust vector flow imaging via Doppler principles. Through our algorithmic innovations, we aim to bring reliable vector Doppler with expanded velocity range closer to clinical application by resolving a prohibitive computational bottleneck in its processing pipeline.

II. THEORETICAL PRINCIPLES

A. Background: Multi-Angle Doppler Estimation

The multi-angle Doppler estimation approach is generally regarded as the prevailing way of deriving flow estimates for high-frame-rate vector Doppler imaging [4]. For this estimation technique, each flow vector \mathbf{v} (i.e., a 2×1 velocity vector estimate at a given pixel position) is computed as the solution that yields the minimum mean-squared error to an N -equations, two-unknowns algebra computation problem that is formed from N instances of the Doppler equation obtained from multiple transmit (Tx)–receive (Rx) angle pairs. Specifically, for a set of N Tx–Rx angle pairs where the n th Tx and Rx angles are denoted, respectively, as θ_n and ϕ_n , the following set of Doppler equations can be obtained in matrix form [4]:

$$\begin{bmatrix} \cos \theta_1 + \cos \phi_1 & \sin \theta_1 + \sin \phi_1 \\ \vdots & \vdots \\ \cos \theta_N + \cos \phi_N & \sin \theta_N + \sin \phi_N \end{bmatrix} \begin{bmatrix} v_z \\ v_x \end{bmatrix} = \mu \begin{bmatrix} f_{D,1} \\ \vdots \\ f_{D,N} \end{bmatrix} \quad (1)$$

where v_z and v_x are the unknown axial and lateral flow velocity components of the flow vector \mathbf{v} , respectively, $f_{D,n}$ is the Doppler frequency (normalized to the PRF) from the n th Tx–Rx angle pair, and μ is a scaling factor of the following form:

$$\mu = \frac{c_o f_{\text{PRF}}}{2f_o} \quad (2)$$

for c_o , f_o , and f_{PRF} , respectively, denoting the speed of sound, ultrasound frequency, and PRF. The $N \times 2$ left matrix in (1) is commonly referred to as the angle matrix \mathbf{A} , while the right-hand side is the $N \times 1$ measurand vector $\mu\mathbf{f}$. It is known that \mathbf{v} in (1) can be solved by the following pseudoinverse operation that yields the least-squares solution [4]:

$$\mathbf{v} = \mu(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{f} = \mu \mathbf{A}^\dagger \mathbf{f} \quad (3)$$

where T , -1 , and \dagger superscripts, respectively, denote the matrix transpose, square matrix inverse, and nonsquare matrix pseudoinverse operations. Note that $\mathbf{A}^\dagger = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$.

In (3), \mathbf{v} is known to be erroneous if aliasing has occurred when obtaining the components of \mathbf{f} . Specifically, the actual Doppler frequency for the n th Tx–Rx angle pair $\hat{f}_{D,n}$ may exceed the maximum unaliased frequency by multiple folds of the unaliased frequency range. In this case, the measured Doppler frequency $f_{D,n}$ may differ from the actual value as follows for a scenario with $l_{D,n}$ wraparound cycles at that Tx–Rx angle pair:

$$f_{D,n} = \hat{f}_{D,n} - l_{D,n} \quad (4)$$

where $l_{D,n}$ is an integer, and it may be different for other angle pairs. As such, the objective of our new algorithmic framework is to ensure that aliasing-resistant vector flow maps with real-time processing throughput can still be derived in the presence of Doppler aliasing. It achieves so by making use of SIMT-based parallel computing principles, which form the essence of GPU programming, to derive aliasing-corrected velocity estimates via ELS-VD's signal processing chain.

B. Overview of the ELS-VD Method

One important theoretical merit of the ELS-VD algorithm is that it can in principle resolve aliasing artifacts with higher aliasing orders (i.e., multiple wraparound cycles) [19]. Within this algorithm, a maximum aliasing order L (a natural number) is specified, such that $l_{D,n}$ in (4) is assumed to be within the range from $-L$ to L for a given angle pair. This algorithm then works to derive, for each pixel, a dealiased velocity vector estimate through two computational stages, which will be described in Sections II-B1 and II-B2. Note that our methodological description is not intended to restate all the theory behind the ELS-VD algorithm [19]. Instead, a concise summary is provided here to facilitate the subsequent description in Section III about our algorithmic innovations for realizing real-time ELS-VD.

1) *Derivation of Pen-Optimal Aliasing Correction Factor*: This first stage of ELS-VD seeks to identify, at each pixel, the pen-optimal (i.e., almost optimal) aliasing correction vector \mathbf{p} with N entries based on the set of slow-time ensembles acquired from all N Tx–Rx angle pairs [19]. Each entry of \mathbf{p} indicates the number of frequency unwrapping cycles (within the integer range $-L$ to $+L$) to be applied to the Doppler frequency estimate (normalized to PRF) of a given Tx–Rx angle pair. To perform this task, the raw Doppler measurand vector \mathbf{f} (with the normalized Doppler frequencies for all N angle pairs) is first derived from conventional Doppler processing. Subsequently, aliasing correction vector candidates \mathbf{p}_k are formed, each of which is one combination of the set of frequency unwrapping cycle values to be applied to $N - 1$ Tx–Rx angle pairs relative to the last Tx–Rx angle pair [19]. Note that, by deduction, the total number of aliasing correction vector candidates K is equal to $(2L + 1)^{N-1}$. For each instance of \mathbf{p}_k , its fitting residue (i.e., squared error) r_k in least-squares velocity vector estimation is computed as follows:

$$r_k = \|\mathbf{A}\mathbf{v} - \mu(\mathbf{f} + \mathbf{p}_k)\|_2 \quad (5)$$

where $\|\mathbf{x}\|_2 = (\mathbf{x}^T \mathbf{x})^{1/2}$ is the l_2 norm. The instance of \mathbf{p}_k with the minimum fitting residue is then set as the pen-optimal aliasing correction vector \mathbf{p} . In other words

$$\mathbf{p} = \arg \min_{\mathbf{p}_k} (r_k). \quad (6)$$

2) *Computation of Optimal Velocity Vector*: After obtaining the pen-optimal aliasing correction vector \mathbf{p} , the ELS-VD algorithm proceeds to determine the optimal dealiased velocity vector \mathbf{v}_{opt} using speckle tracking that is realized via a block matching algorithm [19]. This task is operationalized via multiple steps. First, since \mathbf{p} covers the frequency unwrapping cycle values for $N - 1$ Tx–Rx angle pair [19], a set of

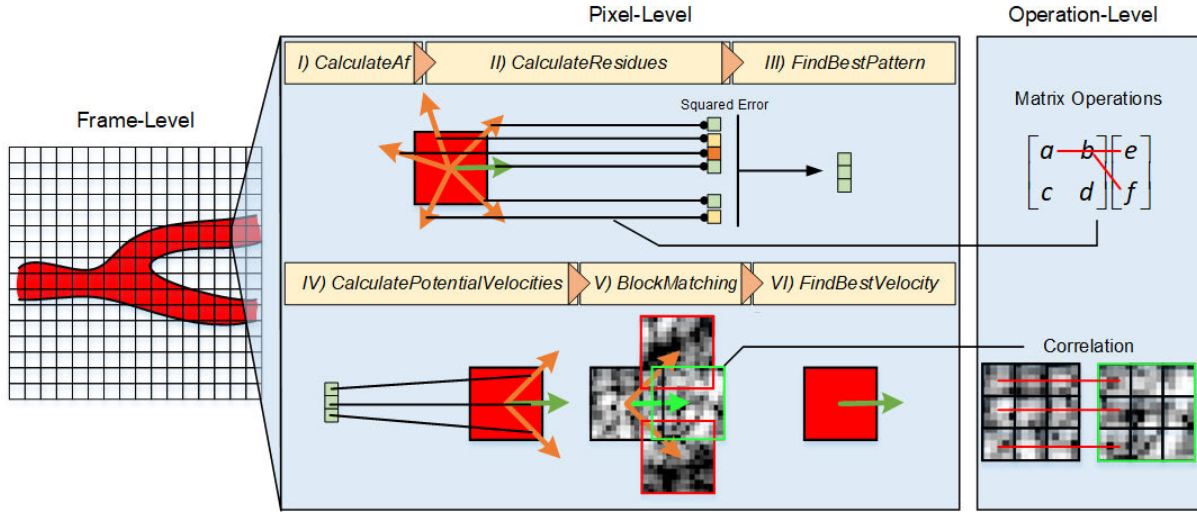


Fig. 1. Multilevel parallelization of our real-time ELS-VD framework. At the frame level, each pixel can be processed independently. Within each pixel, the execution of the ELS-VD algorithm involves sequential execution of six compute kernels (see Table I for each kernel's function). Within each kernel, various matrix arithmetic and computational operations can also be parallelized.

$2L + 1$ dealiased velocity vector candidates are derived to cover possible aliasing cases (spanning from $-L$ to $+L$) that may arise for the measured Doppler frequency of the remaining Tx–Rx angle pair not covered in the previous ELS-VD algorithmic stage. Modifying from (3), each dealiased velocity vector candidate \mathbf{v}_l can be found from the least-squares solution to the system of Doppler equations as

$$\mathbf{v}_l = \mu \mathbf{A}^\dagger (\mathbf{f} + \mathbf{q}_l) \quad (7)$$

where \mathbf{q}_l is a candidate for the optimal aliasing correction vector, and it is one instance of the pen-optimal vector \mathbf{p} with a specific frequency unwrapping cycle value (i.e., an integer from $-L$ to L) for the unaccounted Tx–Rx angle pair. In the second step, for each dealiased velocity vector candidate, interframe block matching [20] was performed to determine the corresponding sum of absolute difference (SAD) value. At last, the velocity vector candidate with the minimum SAD is defined as the optimal velocity vector \mathbf{v}_{opt} for that pixel.

III. REAL-TIME ALIASING-RESISTANT VECTOR DOPPLER FRAMEWORK DESIGN

A. Key Parameters of Interest

From a computational standpoint, real-time execution of ELS-VD is bottlenecked by some major parameters, including the following:

- 1) number of pixels C that requires ELS-VD to be performed in each image frame;
- 2) number of Tx–Rx angle pairs N , since a larger N yields more Doppler frequency measurands, each of which may be prone to aliasing that needs to be rectified;
- 3) aliasing order L , because a larger L naturally creates more instances of aliasing correction vector candidates that need to be analyzed in the ELS-VD algorithm; and
- 4) spatial window dimension B and slow-time window size M used for block matching, since larger window sizes

involve the processing of more samples during block matching.

In our framework, various steps in ELS-VD's processing pipeline have been parallelized to reduce the computational burden imposed by these major influencing parameters. Specific parallelization strategies will be presented in the following subsections.

B. Overall Parallelization Strategy

Recognizing the key ELS-VD parameters of interest, our real-time computing framework has been organized in a three-level (frame, pixel, and operation) hierarchy, as illustrated in Fig. 1, to parallelize various ELS-VD computing stages. On an image frame level, ELS-VD is concurrently applied to estimate individual pixel position's dealiased flow vector estimate. Such a pixel-by-pixel parallel processing approach is inherently executed without interpixel dependence. Note that frame-based pixelwise parallelization itself is inadequate to achieve real-time processing, as we have discovered in our in-house preliminary tests. It is necessary to further parallelize the computational tasks when processing the set of slow-time ensembles for each pixel.

In our framework, pixel-level parallelization was achieved by exploiting the concurrence of various steps within ELS-VD when processing each pixel. Specifically, it is known that ELS-VD involves two algorithmic stages [19], in which different instances of the same computational operation are required and some precalculated constants are repeatedly accessed. As such, our framework has sought to achieve parallelization by allocating multiple threads to handle different instances of calculations and by making frequently accessed data values conveniently accessible to all threads that need them.

The third level of parallelization in our framework pertains to specific computational operations in the algorithm. For instance, the execution of least-squares fitting calculations [4] involves matrix operations that can be accelerated via parallel computing. Accordingly, these operations have been

TABLE I
KERNELS USED IN THE FRAMEWORK

Kernel	Purpose
1. CalculateAf	Performs the matrix multiplication operation $\mathbf{A}^P \mathbf{f}$
2. CalculateResidues	Calculates all K instances of the residue r_k
3. FindBestP	Finds the pen-optimal aliasing correction vector \mathbf{p} according to minimum squared residue
4. CalculatePotentialVelocities	Generates the potential velocity candidates from the best pattern
5. BlockMatching	Performs block matching comparisons
6. FindBestVelocity	Determines the final velocity estimation

parallelized through elementwise operations that can run concurrently via an SIMT approach.

With three levels of parallelization, our framework for SIMT has been structured into six compute kernels, each of which leverages a hierarchical organization of threads to achieve different parallel computation tasks. The function of each kernel is summarized in Table I. Note that Kernels 1–3 focus on handling the first algorithmic stage of ELS-VD, while Kernels 4–6 address the second stage. Specific parallelization strategies are discussed in Sections III-C and III-D.

C. Stage One: Parallelized Derivation of Pen-Optimal Aliasing Correction Vector

In pursuit of a parallel implementation of (5) and (6), we note from linear algebra [22] that r_k is essentially equal to the l_2 norm of the product between the orthogonal projection of \mathbf{A} and the dealiased measurand vector ($\mathbf{f} + \mathbf{p}_k$). This operation can be mathematically expressed as

$$r_k = \|\mathbf{A}^P \mathbf{f} + \mathbf{A}^P \mathbf{p}_k\|_2 \quad (8)$$

where \mathbf{A}^P is the $N \times N$ projection matrix that is equal to $(\mathbf{A}\mathbf{A}^\dagger - \mathbf{I})$, with \mathbf{A}^\dagger as defined in (3) and \mathbf{I} being an identity matrix ($N \times N$). Note that, in (8), \mathbf{A}^P and $\mathbf{A}^P \mathbf{p}_k$ are the algebraic quantities that remain constant for all pixels and can thus be precalculated. In contrast, $\mathbf{A}^P \mathbf{f}$ is the only part of the equation that needs to be calculated differently for each pixel. Based on this notion, we have devised three SIMT compute kernels in our framework to handle specific computational tasks, as elaborated in Sections III-C1–III-C3.

1) *CalculateAf Kernel*: The compute kernel that we have devised to perform the $\mathbf{A}^P \mathbf{f}$ operation for each pixel is illustrated in Fig. 2(a). This kernel uses N^2 threads per pixel to parallelize the prescribed computation process on an elementwise basis. Specifically, each thread with a unique identification (i, j) is responsible for calculating the product between one matrix element and the corresponding element in

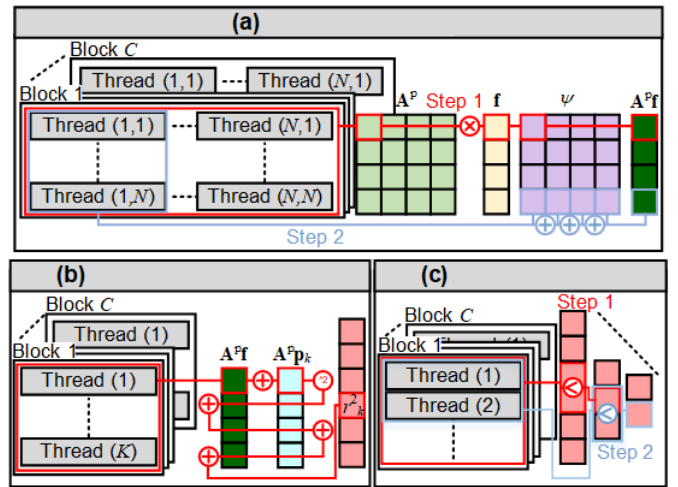


Fig. 2. Thread organization for the kernels in Stage 1 of the proposed framework. For each kernel, the steps that require different thread groupings are highlighted using colored pathways. (a) *CalculateAf* Kernel: matrix-vector multiplication is performed between \mathbf{A}^P and \mathbf{f} to form $\mathbf{A}^P \mathbf{f}$. (b) *Calculate Residues* Kernel: summation of $\mathbf{A}^P \mathbf{f}$ and $\mathbf{A}^P \mathbf{p}_k$, and calculation of the fitting residue for each aliasing pattern candidate \mathbf{p}_k . (c) *FindBestP* Kernel: minimization across the fitting residues to find the pen-optimal aliasing correction vector.

the vector \mathbf{f} as follows:

$$\psi_{i,j} = a_{i,j}^P f_j \quad (9)$$

where $(\cdot)_{i,j}$ denotes the element in the i th row and the j th column in that matrix, while $(\cdot)_j$ denotes the j th element in the vector. Subsequently, N threads are used to form the vector $\mathbf{A}^P \mathbf{f}$, whose i th element is the result of the following summation that is handled by the i th thread:

$$(\mathbf{A}^P \mathbf{f})_i = \sum_{j=1}^N \psi_{i,j} = \sum_{j=1}^N a_{i,j}^P f_j. \quad (10)$$

2) *CalculateResidues Kernel*: The output from the first kernel is fed as an input into a second kernel [see Fig. 2(b)] that aims to compute the fitting residue for all aliasing correction vector candidates. The latter process involves a parallelized solution to (8), whereby parallel addition of $\mathbf{A}^P \mathbf{f} + \mathbf{A}^P \mathbf{p}_k$ and computation of the resulting l_2 norm are performed for each aliasing correction vector candidate. For this computation, K threads per pixel are used to perform the elementwise vector addition tasks needed to compute the residue for all aliasing correction vector candidates. The k th thread is specifically responsible for calculating the following squared sum:

$$r_k^2 = \sum_{i=1}^N (\mathbf{A}^P \mathbf{f} + \mathbf{A}^P \mathbf{p}_k)_i^2. \quad (11)$$

Note that since finding the minimum r_k is equivalent to find the minimum r_k^2 , the square root operation is omitted after obtaining (11) to reduce the computational burden.

3) *FindBest P Kernel*: Given the calculated residues, the third kernel [see Fig. 2(c)] then proceeds to search for the pen-optimal aliasing correction vector \mathbf{p} that has the minimum residue (among a set of K candidates). It achieves so by

performing tree-based parallel reduction with multiple threads. Specifically, the set of K residue values is first partitioned into equally sized subsets, and each thread is used to search for the minimum of one subset. These local minima are then repartitioned into another cluster of subsets, and parallelized search for subset minima is again performed. This reduction process is repeated until the global minimum is obtained as per (6).

D. Stage Two: Parallelized Computation of Optimal Dealiased Velocity Vector

In our framework, one compute kernel is devised for each of the three steps involved in computing the optimal, aliasing-free velocity vector \mathbf{v}_{opt} . Their specific operations are presented as follows.

1) *CalculatePotential Velocities Kernel*: To derive the set of dealiased velocity vector candidates \mathbf{v}_l , this kernel [see Fig. 3(a)] first allocates a block of threads to execute all $2L + 1$ instances of (7) on an elementwise basis. Given the dimensions of the matrix and vectors in (7), $2N$ groups of threads are assigned per pixel, each group of which is responsible for handling the matrix multiplication between one element of the $2 \times N$ matrix \mathbf{A}^\dagger and the corresponding element in the $N \times 1$ vector $(\mathbf{f} + \mathbf{q}_l)$. Each thread group is allocated with $2L + 1$ threads to parallelize the calculations for all instances of $(\mathbf{f} + \mathbf{q}_l)$. Accordingly, the two elements of the i th dealiased velocity vector candidate (i.e., its axial and lateral velocity components) are found to equal to the following sum of elementwise products corresponding to the same matrix row of \mathbf{A}^\dagger :

$$(\mathbf{v}_l)_1 = \sum_{j=1}^N a_{1,j}^\dagger [f_j + (q_l)_j] \quad (12)$$

$$(\mathbf{v}_l)_2 = \sum_{j=1}^N a_{2,j}^\dagger [f_j + (q_l)_j]. \quad (13)$$

2) *Block Matching Kernel*: This kernel [see Fig. 3(b)] takes the $2L + 1$ instances of dealiased velocity vector candidates \mathbf{v}_l as an input to perform block matching to determine which instance of \mathbf{v}_l yields the highest interframe flow speckle similarity [19]. For each velocity vector candidate, the kernel first calculates its corresponding interframe displacement (equals to \mathbf{v}_l multiplied by interframe interval) across a sequence of M consecutive flow speckle frames in a given time window of interest (with M typically smaller than or equal to the slow-time ensemble size used to calculate Doppler frequencies). For each pair of adjacent frames, a block of $B \times B$ pixels centered about the pixel of interest is formed in the former frame, and the corresponding displaced block location is identified in the latter frame. After that, the flow speckle similarity is evaluated between the former frame's $B \times B$ block and the displaced $B \times B$ block in the latter frame. This measure is calculated as the SAD of the two blocks, and the cumulative SAD over all $M - 1$ adjacent frame pairs is evaluated. Such a sequence of calculations for the entire set of velocity vector candidates is handled in parallel by the kernel via the use of B^2 threads. The (i, j) th thread is responsible for

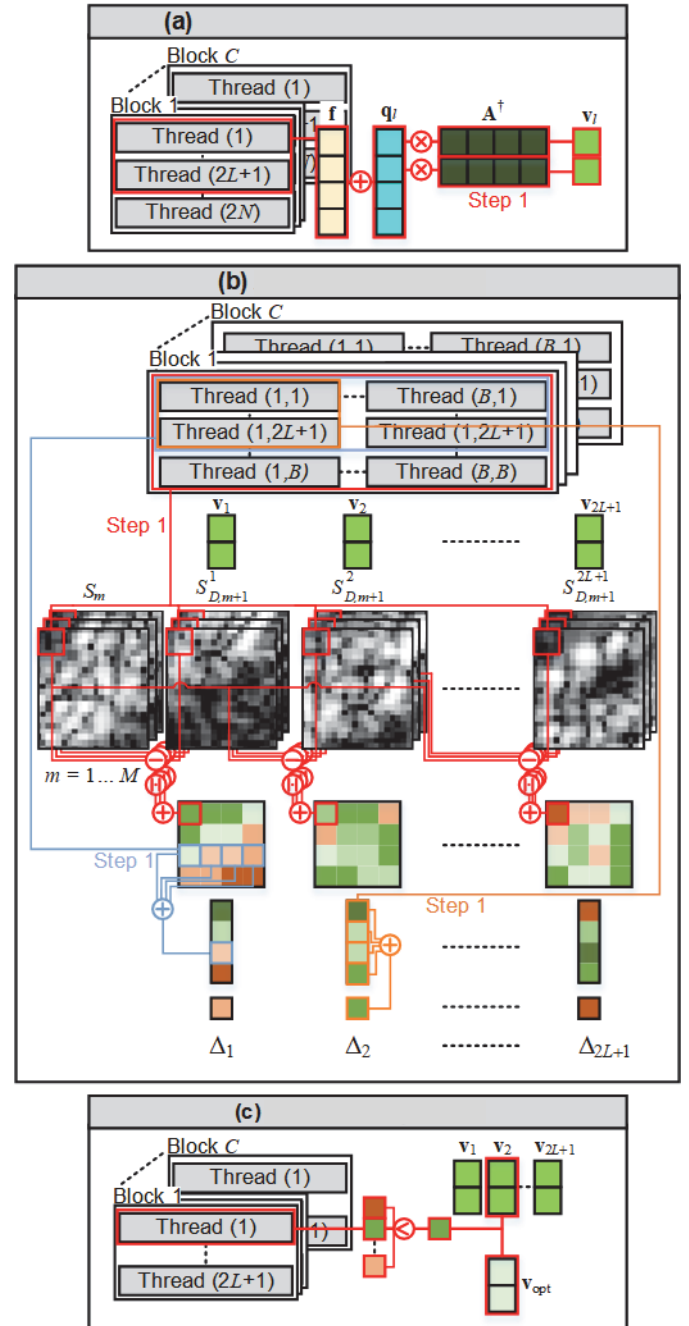


Fig. 3. Thread organization of the kernels in Stage 2 of the proposed framework. For each kernel, the steps that require different thread groupings are highlighted using colored pathways. (a) Calculate Potential Velocities Kernel: matrix-vector multiplication to form $2L + 1$ velocity candidates from the pen-optimal aliasing correction vector. (b) Block Matching Kernel: flow speckle similarity score computation via block matching for each velocity candidate. $B \times B$ blocks across the flow speckle ensemble (length M) are formed based on the interframe displacement due to the velocity vector. The SAD between time consecutive blocks is computed as the flow speckle similarity score for that velocity candidate l . (c) FindBestPVelocity Kernel: minimization across flow speckle similarity score to identify aliasing free velocity estimate.

finding the cumulative SAD of one pixel in the block [denoted as $\Delta_l(i, j)$] over all $M - 1$ adjacent frame pairs

$$\Delta_l(i, j) = \sum_{m=1}^{M-1} |S_{D,m+1}^l(i, j) - S_m(i, j)| \quad (14)$$

where $S_m(i, j)$ denotes the (i, j) th block pixel in the former flow speckle frame in the m th adjacent frame pair, while $S_{+1}^{lD,m}(i, j)$ denotes the same block pixel in the latter frame's displaced block for the l th velocity vector candidate \mathbf{v}_l . It follows that the SAD for the l th dealiased velocity vector candidate (denoted as SAD_l) is given by

$$\text{SAD}_l = \sum_{i=1}^B \sum_{j=1}^B \Delta_l(i, j). \quad (15)$$

When calculating (14) and (15), nonflow pixels in the block, as determined via power thresholding akin to power Doppler imaging, are excluded from the SAD computation. Also, tree-based parallel reduction, similar to what has been described in Section III-C3, is implemented in the kernel to accelerate the addition process in (14). Note that, in lieu of SAD, cross correlation (i.e., sum of squares) may also be used to evaluate interframe flow speckle similarity. Yet, it has not been used in this kernel because it is more computationally intensive due to the involvement of multiplications, and it has been suggested that SAD outperforms sum of squares in block matching [23].

3) FindBestVelocity Kernel: This last kernel [see Fig. 3(c)] in our real-time ELS-VD framework works to identify the optimal velocity vector \mathbf{v}_{opt} that is ideally not subject to aliasing. It achieves so by analyzing the $2L + 1$ values of SAD calculated for all dealiased velocity vector candidates. \mathbf{v}_{opt} is then defined as the dealiased velocity vector with the minimum SAD, which corresponds to the highest interframe flow speckle similarity. Computationally, this task is similar to the one in the *FindBestP* kernel (see Section III-C3). The only difference is that the minimum SAD search process can be executed trivially without the need to implement parallel reduction strategies since L is typically not large.

E. Implementation of Real-Time ELS-VD

The compute unified device architecture (CUDA) platform (NVidia, Santa Clara, CA, USA) was used in this investigation to implement our parallelized ELS-VD framework. CUDA is an application programming interface (API) developed for general purpose computing on GPUs, and it relies on a SIMT architecture. The ELS-VD framework was realized as a standalone processing pipeline on a computing platform that was equipped with a Xeon E5-2620 central processing unit (Intel, Santa Clara, CA, USA) and an RTX-2080 GPU (NVidia). Our framework was coded in C++ using the CUDA API (ver. 8.0.60). It was compiled for use with MATLAB (ver. 2020a; Mathworks Inc., Natick, MA, USA) through the `mexcuda` function that was configured to invoke the `nvcc` and Microsoft Visual C++ compilers, respectively, for compiling parallelized and sequential C++ instructions. This compilation approach generated a parallelized MATLAB executable with dynamic memory allocation operations that directly interfaced with the computing platform's operating system.

In implementing our real-time ELS-VD framework on the GPU, we have optimized its computational operations for single-floating point precision. Specifically, all variables have been stored and processed as `floats` data type. Also,

we have leveraged CUDA's `fast maths` compilation setting whenever possible to execute arithmetic operations, such as division and power.

Besides optimizing the arithmetic operations, we have taken into consideration various key features of the GPU when implementing real-time ELS-VD. These features include the following.

- 1) Each GPU is equipped with thousands of compute cores (4352 on the GTX-2080 model) to facilitate concurrent execution of multiple threads.
- 2) CUDA-based computing kernels are executed with thread blocks (indexed in 2-D entities) that have access to the GPU's on-chip memory, whose access latency is much shorter than the GPU's off-chip global memory.
- 3) There is a maximum thread block size that is supported by the GPU (1024 for the RTX-2080 model), beyond which run-time overhead arises and it negatively impacts the computing throughput.
- 4) GPU's on-chip memory is limited in capacity; on the RTX-2080 GPU, between a group of 64 threads, only 96 kB of on-chip memory (64-kB shared memory and 32-kB L1 cache) is available for sharing.
- 5) Texture memory is available on the GPU to cache frequently accessed data in the GPU's global memory.

F. Kernel-Level Computational Optimization Strategies

1) Fast Memory Access: Various kernels in our real-time ELS-VD framework have suitably leveraged the GPU's fast on-chip memory by executing their SIMT computations on a one thread block per pixel basis. For example, the *CalculateAf* kernel (see Section III-C1) requires N^2 threads for each pixel with N (number of Tx-Rx angle pairs) typically less than 32, so accordingly, an $N \times N$ thread block is allocated to execute this kernel for each pixel. In doing so, the matrix elements of \mathbf{A}^P and \mathbf{f} , which are repeatedly accessed according to (5), are copied to the GPU's shared memory during run time to significantly shorten the resulting data access latency. A similar strategy for thread block organization is used in the *CalculatePotentialVelocities* kernel (see Section III-D1) and the *FindBestVelocity* kernel (see Section III-D3).

2) Optimization of the CalculateResidues Kernel: For compute kernels that require more processing threads than the maximum block size of 1024 threads, we have adopted alternative implementation strategies to ensure that only one thread block is allocated to process each pixel. Specifically, for the *CalculateResidues* kernel (see Section III-C2) that in principle uses K threads (where K is typically much larger than 1024 as per Section II-B1), it has adopted a $\min(1024, K)$ thread block structure, so that the number of threads per block would not exceed 1024. If K was greater than 1024, the threads were instructed to compute (7) for multiple instances of \mathbf{p}_k (with an even distribution of computing load between threads). This implementation strategy was chosen in lieu of initiating more thread blocks, so that thread organization can be simplified to reduce the number of read-write operations between the GPU's global memory and on-chip memory.

3) *Optimization of the FindBestP Kernel*: For the *FindBestP* kernel (see Section III-C3), depending on the actual values of N and L , it is also prone to use more threads than the maximum block size supported by the GPU. To alleviate this issue, we have frontloaded parts of the minimum search operation to the *CalculateResidues* kernel. In particular, since each thread within *CalculateResidues* is already tasked to calculate r_k^2 for multiple instances of \mathbf{p}_k , we have revised its operation further, so that it would also directly identify the instance with the minimum r_k^2 value within its subgroup. This strategy is computationally effective, because the values for multiple instances of r_k^2 already exist within the thread block's allocated on-chip memory. In turn, the number of memory access operations is reduced to improve the processing throughput, and the *FindBestP* kernel's parallel reduction task is naturally simplified. Note that, for the *FindBestP* kernel's minimum search among the remaining candidates, we have divided them into subsets of 32, so that at most 32 subsets can be processed by a single block of 1024 threads. The key benefit of doing so is that the same thread block's on-chip memory can be leveraged to effectively reduce memory access latency of repeated r_k^2 value access requests during the minimum search. Only the final global minimum result would be copied back to the GPU's global memory for further processing.

4) *Optimization of the BlockMatching Kernel*: For the *BlockMatching* kernel (see Section III-D2), there is repeated access of flow speckle image values in the current frame when executing (13), and some flow speckle pixels are common between the spatial block matching window of adjacent pixels. Accordingly, the kernel has been devised to copy these frequently accessed flow speckle image values to the GPU's texture memory to reduce the data access overhead involved in retrieving data from the GPU's global memory. Note that the number of threads required to compute (14) and (15) depends on the size of the $B \times B$ spatial window used for block matching. As such, give a maximum thread block size of 1024, the window dimension B was limited to 32 or smaller, so that a single thread block can be allocated to handle the block matching calculations for one dealiased velocity vector candidate.

IV. EXPERIMENTAL METHODS

A. Phantom Imaging Studies

Our framework's ability to achieve real-time derivation and seamless visualization of aliasing-resistant vector Doppler images was first tested through a series of in vitro phantom experiments. We used a wall-less carotid bifurcation model with 50% stenosis, which was chosen because of the prevalence of aliasing in stenosed conditions. It was fabricated using a previously established investment casting protocol [24]. Its tissue mimicking material was composed of a mixture of 10% polyvinyl alcohol (341584; SigmaAldrich, St. Louis, MO, USA), 3% graphite scatterers of particle diameter less than 20 μm (282863; Sigma-Aldrich), and 0.3% potassium sorbate preservatives (85520; Sigma-Aldrich). The phantom was connected to a pulsatile flow pump that generated

TABLE II
IMAGING PARAMETERS USED IN THE EXPERIMENTS

Parameter	Value
Imaging Platform	
Number of Tx/Rx channels	128
Array pitch (mm)	0.3048
Data sampling rate (MHz)	40
Data sample resolution (bits)	12
Data Acquisition	
Transmission frequency (MHz)	5
Tx pulse duration (number of cycles)	3
Tx steering angles ($^\circ$)	-10 and +10
Pulse repetition frequency (Hz)	6,666 Hz (3,333 Hz/angle)
Data acquisition duration (s)	2
Beamforming	
RF filter passband (MHz)	2.5–7.5
RF filter order	52
Filter design method	Equiripple
(Tx, Rx) angle pair combinations	(-10 $^\circ$, -10 $^\circ$), (-10 $^\circ$, -3 $^\circ$), (-10 $^\circ$, 6 $^\circ$), (-10 $^\circ$, 10 $^\circ$), (10 $^\circ$, -6 $^\circ$), (10 $^\circ$, 3 $^\circ$), (10 $^\circ$, 10 $^\circ$)
Image dimension (cm)	Lateral: -1.9 to 1.9 Axial: 0.0 to 4.0
Image dimension (number of pixels)	256 (lateral) \times 256 (axial)
Slow-Time Data Processing	
Clutter filter cut-off (normalized)	0.05
Clutter filter order	75
Filter design method	Equiripple
Slow-time ensemble size (samples)	30 (sliding window processing)
Sliding window step size (samples)	1

carotid pulses with a peak flow rate of 5 mL/s and a pulse rate of 72 beats/min. The flow rate was chosen to achieve one-cycle aliasing artifacts that could be resolved through manual phase unwrapping. The blood mimicking fluid was matched to the acoustic and viscous properties of human blood. Details on the fabrication of the flow pump and the blood mimicking fluid have been reported elsewhere [24].

Our imaging platform was a research-purpose ultrasound scanner (SonixTouch; Analogic Corporation, Peabody, MA, USA) equipped with a linear array transducer (L14-5; Analogic Ultrasound). It was configured for plane wave transmission, and a prebeamformed data acquisition system [25] was used to acquire the channel-domain radio frequency (RF) data needed for processing. Imaging and system parameters used in this experiment are shown in Table II. In brief, transmission was performed with two steering angles (-10 $^\circ$, +10 $^\circ$) and the raw data were then beamformed on receive to form $N = 7$ Tx-Rx steering angle combinations. This multi-angle Doppler configuration was chosen to maintain a compromise between: 1) achieving robust flow vector estimation, which is realized through the use of multiple Rx angles for a given Tx angle; and 2) obtaining a kilohertz-range effective PRF for each Tx-Rx angle pair, as can be

TABLE III
IMAGING PARAMETERS USED IN THE IN VIVO DEMONSTRATION

Parameter	Value
Imaging Platform	
Number of Tx/Rx channels	192
Array pitch (mm)	0.245
Data sampling rate (MHz)	25
Data Acquisition	
Pulse repetition frequency (Hz)	6000 Hz (3000 Hz/angle)
Beamforming	
RF filter passband (MHz)	2.5–7.5 (Speckle), 4.5 – 5.5 (Doppler)
RF filter order	52
Slow-Time Data Processing	
Clutter filter cut-off (normalized)	0.2
Clutter filter order	37
Slow-time ensemble size (samples)	60 (sliding window processing)

achieved using a limited number of Tx angles (two in this case). Note that a relatively low-effective PRF (3333 Hz for each Tx–Rx angle pair) was used to maintain a fine Doppler resolution for the detection of slow velocities during diastole. These parameters have been similarly used in another Doppler dealiasing investigation [19].

B. In Vivo Imaging Trial

In addition to phantom studies, an in vivo pilot experiment was conducted to demonstrate our framework’s performance in monitoring arterial hemodynamics with fast flow velocities. This pilot trial, approved by the Clinical Research Ethics Committee of the University of Waterloo (Protocol No. 31694), was in the form of an exercise intervention, where a 25-years old healthy male volunteer was tasked to perform a set of 20 consecutive body squats. A long-axis imaging scan was then performed at the subject’s femoral bifurcation, which is known to have flow characteristics with large variations in flow velocity over a cardiac cycle [26]. Note that the bodyweight squats effectively led to an acute increase in blood flow velocities in the femoral bifurcation [27], [28], because muscular contraction is known to induce active hyperemia that increases blood flow in nearby arteries [29]. Accordingly, the resulting vector Doppler images are particularly prone to single- and double-cycle aliasing.

For this human imaging trial, raw vector Doppler data were acquired using another ultrasound research scanner (US4US; Warsaw, Poland) that was equipped with a linear array transducer (SL1543; Esaote, Genoa, Italy). Plane wave transmission was used and the prebeamformed data were transferred to the computational platform for offline processing in the same way as that described in Section IV-A. Table III shows the system, transmission, and processing parameters for the in vivo experiment. Parameters that are not listed in Table III were identical to those in Table II. Note that a narrower bandpass filter was used for the Doppler frequency

estimation to reduce spectral broadening, and a longer ensemble size of 60 was used to improve Doppler resolution.

C. Data Processing and Dealiasing Performance Analysis

All channel-domain RF data samples were acquired as fixed-point values at 12-bit resolution. There were loaded onto our computational platform, converted into floating-point variables at the software level, and processed via a custom MATLAB and CUDA-based pipeline for beamforming [30] and Doppler processing [31]. Accordingly, Doppler frequency measurands, flow detection maps, and flow speckle maps were derived on a frame-by-frame basis, and they were used as an input to analyze the efficacy of our real-time ELS-VD framework. Sliding window processing (with one-sample steps) was applied to each slow-time ensemble to derive flow vector estimates on a pixelwise basis at different times. Vector Doppler cineloops with dynamic projectiles [32] were generated accordingly.

The vector Doppler maps obtained from our real-time ELS-VD framework were copied back to the CPU for visualization. Flow vector estimation was also performed using the conventional least-squares vector Doppler estimator [4] to determine the impact of aliasing on the resulting vector flow images if these artifacts were not resolved. To facilitate quantification of our framework’s efficacy in achieving dealiasing, we have generated a set of reference flow vector maps by performing inspection-based manual dealiasing on the flow vector maps obtained via least-squares vector Doppler estimation. The procedure for manual dealiasing has been presented elsewhere [29] and was used to keep all parameters identical between the reference and the output. The root-mean-squared difference (RMSD) between the ELS-VD-derived flow vectors and the manually dealiased flow vectors was then calculated as a metric for our framework’s aliasing resistance efficacy, especially for the image frames corresponding to flow systole when fastest flow velocities emerged.

D. Throughput Analysis

The throughput of our real-time ELS-VD framework was evaluated in terms of the processing time per frame under different imaging resolutions and different parameters. Each kernel’s execution time was tracked individually using CUDA timers to identify the framework’s computational bottlenecks. The measured time only considered the GPU processing time, while the time for auxiliary operations (e.g., data transfer between CPU and GPU, and CPU functions) was excluded because the practical use of our framework would involve an end-to-end GPU processing pipeline.

We specifically investigated the impact of the following factors on processing time: 1) number of flow pixels being processed C ; 2) maximum aliasing order L ; 3) block matching spatial dimension B ; and 4) slow-time window size M used for block matching. In a practical application, the number of flow pixels processed depends on the imaging resolution and the flow dynamics of the scene, making it an important parameter to investigate. The maximum aliasing order L is a

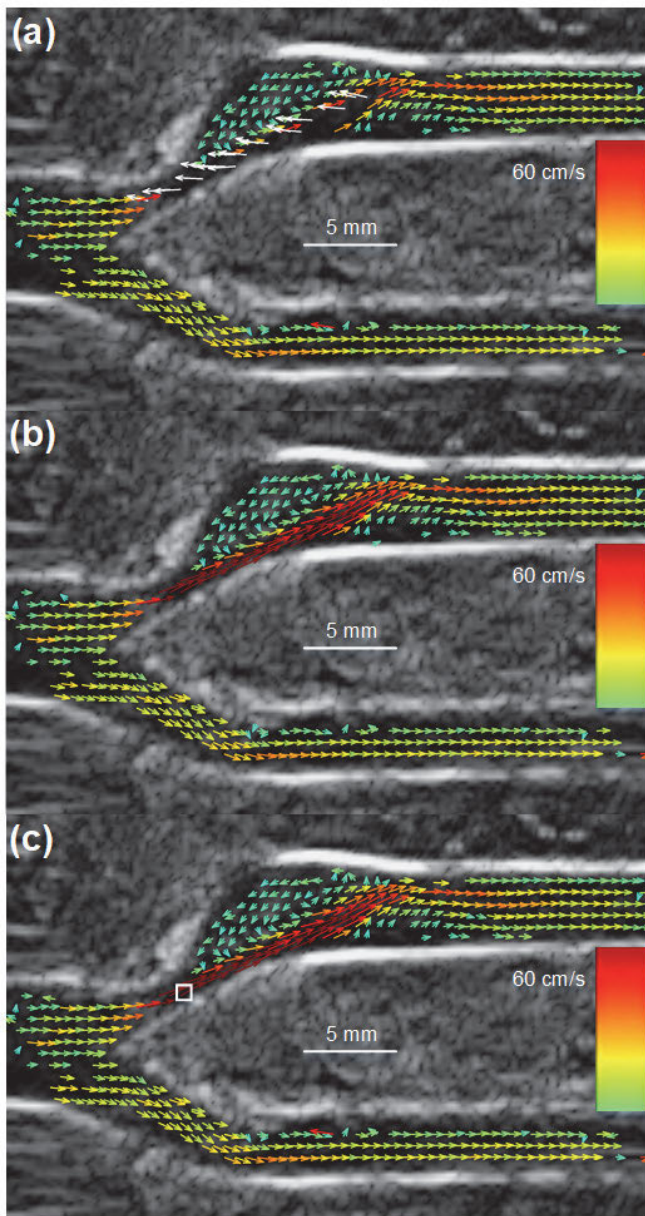


Fig. 4. Peak-systolic flow vector profiles obtained from a carotid bifurcation phantom with 50% stenosis. Results are derived using (a) least squares vector Doppler estimation, with aliasing artifacts highlighted in white, (b) real-time ELS-VD framework, and (c) manually dealiased (shown for reference).

function of the imaging scenario, with faster flow conditions requiring higher aliasing order. Moreover, this parameter has a significant impact on the number of aliasing patterns to investigate. The block matching parameters (B and M), which influence the fidelity of the block matching analysis, may well impact the processing time, and thus, their effects should be analyzed.

V. RESULTS

A. Real-Time Aliasing-Resistant Flow Vector Mapping Achieved in a Stenosed Carotid Bifurcation

Aliasing-resistant flow vector estimation was demonstrated successfully in the in vitro stenosed carotid bifurcation

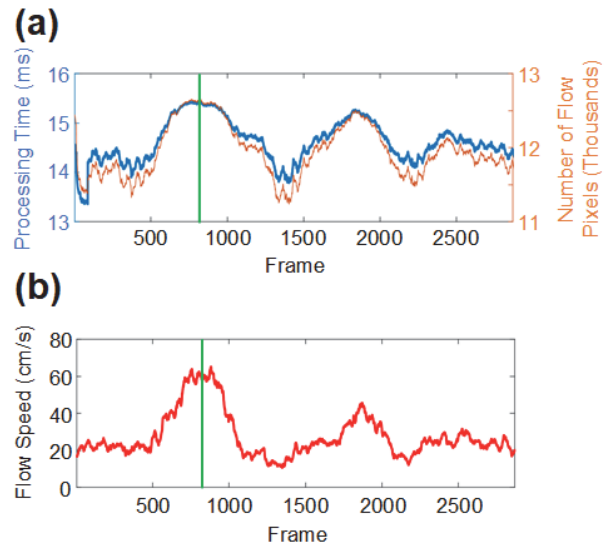


Fig. 5. Real-time feasibility of the proposed framework. (a) Frame processing time and the number of flow pixels, shown over all 2775 frames in a single cardiac cycle. (b) Reference flow speed profile in the white square denoted in Fig. 2(c).

phantom with real-time slow-motion playback of the acquired high-frame-rate data. Fig. 4(a)–(c), respectively, shows the peak-systolic flow vector maps derived using the least-squares vector Doppler estimator, our real-time ELS-VD framework, and manual dealiasing. The ELS-VD estimation results were obtained for $L = 1$, $B = 20$, and $M = 30$. Note that, in Fig. 4(a), the white-colored flow vectors indicate ones with aliasing artifacts as determined from the manually dealiased reference. The key observation to be noted from Fig. 4(a) is that, during peak systole, aliasing expectedly emerged in the carotid stenosis region because the narrowed lumen had induced faster flow velocities. These artifacts have been largely resolved by our real-time ELS-VD framework, as shown in Fig. 4(b). Indeed, over the 371 systolic frames where aliasing artifacts emerged, real-time ELS-VD merely yielded an RMSD of 5 cm/s in the stenotic jet region with respect to the manually dealiased flow vector reference maps that have a systolic jet speed of 60 cm/s [as shown in Fig. 5(b)]. In contrast, the least-squares vector Doppler estimator was found to yield a significantly higher RMSD of 44 cm/s in the stenosis zone.

Movie 1 shows the dealiasing performance of the framework throughout the cardiac cycle. This cinelooop shows a slow-motion real-time playback of vector flow images with dynamic visualization [8] rendered at a real-time display rate of 25 frames/s for the least-squares vector Doppler estimator (Movie 1a), the GPU-accelerated ELS-VD (Movie 1b), and the manually dealiased reference (Movie 1c). It can be observed that after resolving the aliasing artifacts with our framework, jet formation in the stenosed region and flow recirculation in the carotid bulb can be more intuitively visualized in the bifurcation phantom.

As a substantiation of the notion of the real-time playback processing, Fig. 5(a) plots the frame processing time over one

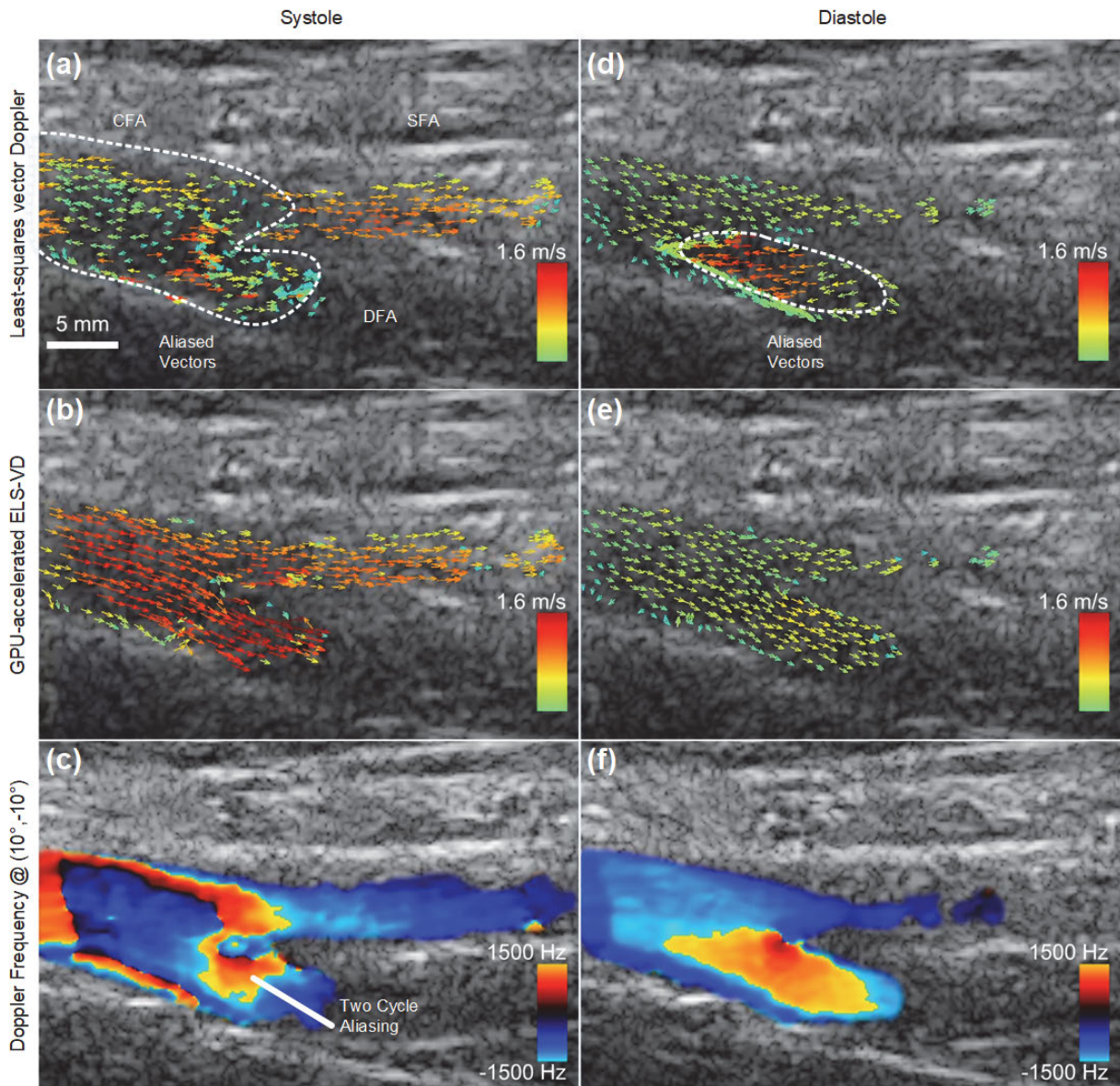






Fig. 6. Vector flow profiles obtained from a femoral bifurcation after exercise showing the common femoral artery (CFA) as it splits into the superficial femoral artery (SFA) and deep femoral artery (DFA). Flow profiles are shown for peak systole (left) and start of diastole (right). Results are derived using (a) and (d) least squares vector Doppler estimation, with aliasing artifacts highlighted in white, and (b) and (e) real-time ELS-VD framework. A visualization of the Doppler frequency measurement from the $(10^\circ, -10^\circ)$ Tx-Rx combination using color flow principles is shown in (c) and (f) to emphasize the presence of two cycle aliasing due to elevated blood flow.




full cardiac cycle. For reference, the mean flow speed profile is also plotted in Fig. 5(b) for the white square zone indicated in Fig. 4(c). Note that the number of flow pixels per frame varied between a minimum of 11 239 and a maximum of 12 731, with a mean of 11 971. It can be observed that real-time processing throughput was achieved throughout the cardiac cycle that spanned 2775 frames in this dataset. Specifically, regardless of the cardiac cycle phase, the frame processing time was under 16 ms, which corresponds to a minimum processing throughput of 62.5 frames/s. The frame processing time was found to vary slightly over different cardiac cycle phases. During systole, more flow pixels emerged because of flow-mediated dilation of the vessel lumen, and thus, ELS-VD needed to be applied to more pixels to detect (and

possibly correct for) aliasing artifacts. Nonetheless, real-time processing throughput was still achieved in this scenario.

B. In Vivo Demonstration of Real-Time Aliasing-Resistant Flow Vector Mapping

In the exercise intervention imaging trial, our framework was found to yield performance similar to the phantom experiments in generating aliasing-resistant vector flow maps at real-time processing throughputs. The corresponding in vivo imaging results are shown in Movie 2  for the least-squares vector Doppler method (Movie 2a ) and the proposed framework (Movie 2b ). Note that two-cycle aliasing was present in this scenario, as evident in the color Doppler image

with 10° Tx steering and -10° Rx steering (Movie 2c ). The proposed framework has resolved these aliasing artifacts and has properly rendered the femoral bifurcation's anterograde flow pattern, whereas the original vector Doppler formulation has erroneously depicted the presence of retrograde flow. This key observation is summarized in Fig. 6, which shows the flow visualization at two temporal snapshots (peak systole and start of diastole) in the cardiac cycle.

The framework's real-time processing performance in the human imaging scenario is summarized in Movie 2d . These results were obtained for $L = 2$, $B = 20$, and $M = 30$. Throughout this dataset with 1690 frames, the compute time of the proposed framework reached a peak of 19 ms (53 frames/s) as the number of flow pixels processed per frame fluctuated between 6581 and 9994 with a mean of 7079. Movie 2e  shows the flow speed profile for the sample volume marked in Movie 2b . The systolic flow velocity in this scenario was approximately 150 cm/s. Spurious velocity errors were observed in the framework output during the transitory phases of the cardiac cycle and near tissue regions.

C. ELS-VD Processing Throughput Influenced by Multiple Parameters

For practical combinations of processing parameters, our ELS-VD framework was generally found to be capable of achieving real-time processing throughput (with at least a video-range frame rate of 25 frames/s). Fig. 7 shows a series of corresponding results. Specifically, Fig. 7(a) shows the frame processing time as a function of the number of flow pixels C being processed and the maximum aliasing order L (for $B = 20$ and $M = 30$). This plot confirms that the frame processing time expectedly increased when there were more flow pixels or with the use of a larger aliasing order L . For an aliasing order of $L = 1$, our ELS-VD framework can achieve real-time performance for frames with just over 40 000 pixels or an image grid of 200×200 . In contrast, for an aliasing order of $L = 3$, our framework can yield real-time throughput for frames with 8100 pixels.

As additional insight into the real-time performance of our ELS-VD framework, Fig. 7(b) shows the number of pixels C that can be processed at a 25 frames/s throughput under different block matching spatial window sizes B and aliasing orders L (for $M = 30$). This plot shows that the use of a smaller spatial window (i.e., smaller B) for block matching tends to allow more pixels to be processed at real-time throughput. Fig. 7(c) shows the pixel processing throughput (with 25 frames/s rate) as a function of the slow-time block matching window size M that sets the number of frames to be included in each block matching operation (for $B \times B = 20 \times 20$ windows). As expected, the number of pixels C that can be accommodated with real-time throughput was decreased at larger slow-time block matching window sizes M .

D. Block Matching Is the Primary Computational Bottleneck

As shown in Fig. 8, the *BlockMatching* kernel was found to be the bottleneck in every tested configuration of our real-time ELS-VD framework. In particular, Fig. 8(a) shows the

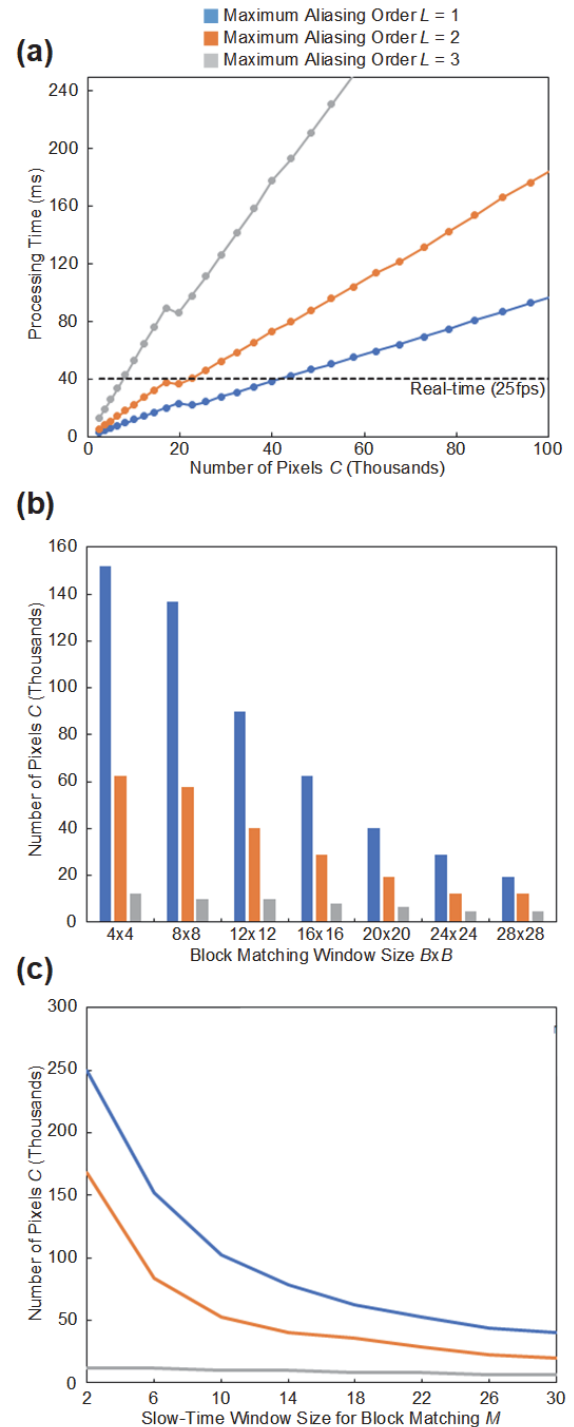


Fig. 7. Computing performance of the real-time ELS-VD framework as a function of different parameters, including numbers of flow pixels C , maximum aliasing order L , block matching window dimension B , and slow-time block matching window size M . (a) Frame processing time as a function of C for $L = 1, 2$, and 3 (with $B = 20$ and $M = 30$). (b) Number of flow pixels that can be processed in real time (25 frames/s) for different $B \times B$ block matching windows ($L = 1, 2$, and 3; $M = 30$). (c) Number of pixels C that can be processed in real time (25 frames/s) as a function of M ($L = 1, 2$, and 3; $B = 20$).

percentage of processing time occupied by each GPU kernel as a function of slow-time block matching window size M for

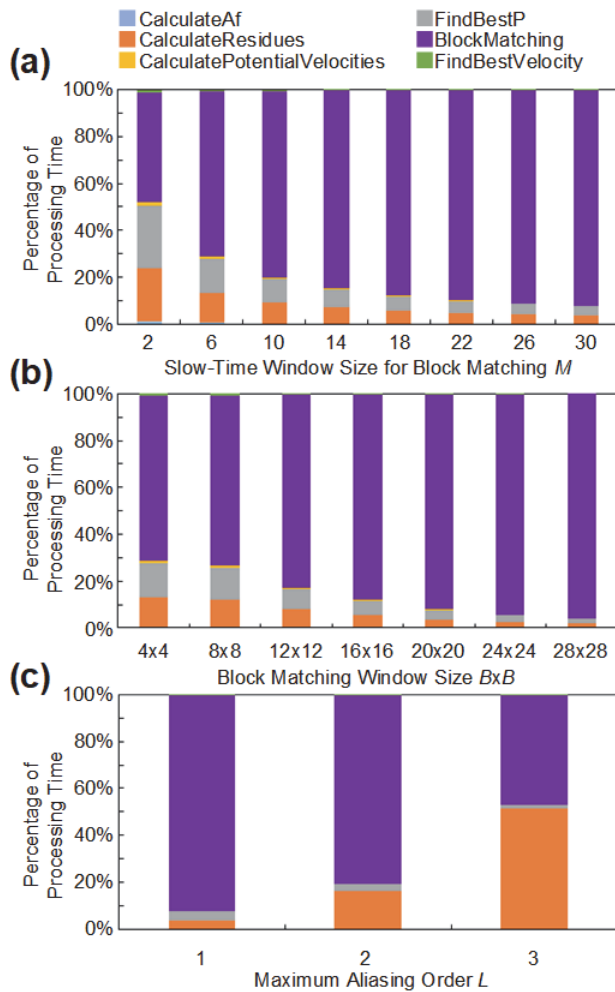


Fig. 8. Processing time for each kernel in the real-time ELS-VD framework. Results are shown (in percentages) as a function of (a) slow-time block matching window size M ($L = 1$ and $B = 20$), (b) block matching spatial window size $B \times B$ ($L = 1$ and $M = 20$), and (c) maximum aliasing order L ($B = 20$ and $M = 20$). Number of flow pixels C was kept constant at 65 536 (256×256) in all cases.

$L = 1$ and $B = 20$. As the slow-time block matching window size M increased to 30, the *Blockmatching* kernel apparently dominated nearly 90% of the frame processing time. When a smaller spatial window dimension B was used for block matching, the *Blockmatching* kernel was found to occupy a shorter fraction of the frame processing time, as shown in Fig. 8(b) for $L = 1$ and $M = 20$. The only instance where another kernel, *CalculateResidues*, was identified as a major computational bottleneck was when using a high aliasing order of $L = 3$. As shown in Fig. 8(c) (for $B = 20$ and $M = 20$), in the case of $L = 3$, 98% of the frame processing time was attributed to two kernels: *CalculateResidues* (51%) and *BlockMatching* (47%).

VI. DISCUSSION

A. Summary of Contributions

There is significant interest in translating high-frame-rate vector flow imaging toward use in the clinic to facilitate the

imaging of complex and transient phenomena [1], [33], [34]. Success in such a clinical translation drive can have ramifications on cardiovascular health monitoring, especially in the diagnosis of diseased conditions such as atherosclerosis where complex hemodynamics are prone to emerge and are known to contribute to disease progression [35], [36], [37]. For vector flow imaging to be favorably considered as a useful clinical diagnostic tool, it is important for this modality to be realized in real time without aliasing artifacts. In response to such a technical need, we have sought to devise a GPU-based ELS-VD framework (see Fig. 1) with a set of SIMT computing kernels (see Table I) to derive aliasing-free velocity vector maps at real-time processing throughput. This framework has transcended beyond the original ELS-VD algorithm [19] by introducing parallel computing principles to enable its real-time realization. It has been implemented with the CUDA API and has been executed on the latest generation of NVidia GPUs (RTX-2080).

Experimental results have shown that our GPU-based ELS-VD framework is effective in resolving aliasing artifacts in an anthropomorphic carotid bifurcation phantom with 50% stenosis (see Fig. 4) with short frame processing time that corresponds to real-time display rates (see Fig. 5). We particularly targeted real-time playback performance, such that slow-motion vector flow cinelops can be computed and visualized at a target of 25 frames/s. Our framework has also succeeded in achieving real-time dealiasing functionality for higher aliasing orders L in vivo (see Fig. 6) and for practical frame sizes with a reasonable number of flow pixels (see Fig. 7). Among all the computational operations of our ELS-VD framework, we found that block matching is generally the most time-consuming (except in the case with an aliasing order of $L = 3$) and occupies the majority of processing time needed to derive each frame of dealiasing vector flow map (see Fig. 8).

B. Significance to Vector Doppler Technology

Aliasing removal in Doppler ultrasound has recently received renewed attention with the advent of advanced ultrasound techniques, such as high-frame-rate ultrasound and vector Doppler [16], [17]. Our approach to dealiasing is conceptually founded upon the rational use of speckle tracking via block matching to overcome vector Doppler's inherent proneness to aliasing errors. Its real-time applicability in this investigation was established by designing a new GPU computing framework that involves parallel processing principles. From an engineering standpoint, our framework can be regarded as an enabling solution for achieving aliasing-resistant flow vector estimation in real time.

Compared to other dealiasing algorithms, our GPU-based ELS-VD technique is a flow vector estimation scheme that can remain resilient against multicycle aliasing artifacts. It does not require a priori knowledge of aliasing characteristics (i.e., no data training is needed), so it is fundamentally different from inference-based dealiasing solutions that rely on deep learning principles to correct for aliasing artifacts [38], [39]. Also, it is distinguished from dealiasing techniques that

are based on a staggered transmission strategy [15], whose dual-PRF configuration has demonstrated real-time feasibility in achieving aliasing-resistant cross-beam vector Doppler in single-cycle aliasing scenarios [16]. Because our GPU-based ELS-VD technique does not require reconfiguration of the transmit firing scheme, it is not subjected to uneven slow-time data sampling that is inherent to the staggered transmission approach and that adds challenge to the clutter filtering step of Doppler processing [15]. Our dealiasing framework is also different from other dealiasing methods that are based on broadband transmissions [17]. The efficacy of the latter approach is often degraded by Doppler spectral broadening that considerably increases the difficulty to perform clutter filtering effectively and, in turn, increases the variance of Doppler frequency estimation.

It should be emphasized that our GPU-based ELS-VD method does not seek to expand the range of measurable velocity by implementing a special transmission sequence, such as one that alternates between packets of the same steering angle to preserve the effective PRF for each angle [40], [41]. Instead, our framework tackles aliasing in vector Doppler by considering Doppler frequency measurements from all steering angles in one cohesive framework. It supports more transmit steering angles with no constraints on the bandwidth of the transmitted pulses, clutter filtering, or the size of the ensemble. It can be readily applied to other Tx-Rx angle pair combinations besides the seven-angle configuration used in our experiments. Hence, it can provide more robust velocity estimation as per previously reported least squares vector Doppler findings [4]. Perhaps, one limitation of our framework is its sensitivity to speckle correlation, which may be low in conditions of in vivo flow. Nonetheless, time-domain techniques have been similarly reported in early work on aliasing removal [14], so it is after all feasible to leverage speckle motion information for aliasing correction purposes.

C. Insights on GPU Acceleration of ELS-VD

In this work, GPU acceleration of the ELS-VD framework was realized via SIMT parallelization with a three-level hierarchy of threads: frame level, pixel level, and operational level (see Fig. 1). This parallelization strategy naturally emerged from the data independencies of the ELS-VD framework, and it complemented the organization of threads used in GPU architecture. This thread organization, however, is dependent on specific ELS-VD computational steps being accelerated, and thus, dedicated GPU kernels need to be devised for each part of the algorithm. In addition to thread organization, memory management is deemed to be another important consideration. Achieving real-time performance requires optimized use of the GPU's on-chip memory, texture memory, and global memory when executing each kernel. While global memory is the largest in size, it is also the slowest to access. Variables that need to be accessed multiple times, such as the projection matrix \mathbf{A}^P in (8) and (9), should be transferred to the shared memory first before the matrix-vector operations are performed.

It is interesting to note that although our ELS-VD framework has achieved real-time performance (see Fig. 7), we were not able to achieve full parallelization in some compute kernels because of practical constraints in the GPU's hardware resources. Compromises need to be made during implementation (see Section III-F). Nonetheless, with the continuing development of more powerful GPUs, the impact of making these implementation compromises will likely be mitigated, and it is reasonable to expect that our framework can achieve an even higher processing throughput in the future.

As shown in our computing time analysis (see Fig. 8), the *BlockMatching* kernel was the primary computational bottleneck in our ELS-VD framework under almost all parameter combinations. Block matching is important in the ELS-VD framework for disambiguating velocity candidates that cannot be differentiated by least-squares optimization alone, and thus, it is ultimately responsible for selecting the aliasing free velocity vector. From a computational standpoint, the block matching bottleneck stems from the repeated and random access of flow speckle image values from the GPU's off-chip memory. The number of these off-chip data accesses increased for larger block matching windows (B , M) and slow-time window sizes M . After all, both a relatively large block matching window size ($B = 20$) and a relatively large ensemble size ($M = 30$ samples) were needed to achieve robust dealiasing performance in the stenosed carotid phantom imaging scenario (or else significant spurious errors would arise, as we have discovered in our in-house preliminary tests). These findings are in accordance with the conclusions made in the original ELS-VD algorithm paper [19], and they are the typical limitations of block matching in speckle tracking applications. Note that research has been done on the performance of speckle tracking in different flow conditions [42], [43]. In the future, it would be worthwhile to incorporate these results to refine the performance of our real-time ELS-VD framework.

D. Future Integration of Proposed Framework Into a System

In principle, our GPU-based ELS-VD framework can be integrated into any software-oriented ultrasound scanner that seeks to realize an end-to-end GPU processing pipeline from beamforming to display [10]. This framework should be readily generalizable to most CUDA-compliant GPUs without any theoretical or design modifications aside from, perhaps, some performance fine-tuning if GPUs with vastly different specifications are used. Note that the approach of accelerating various aspects of the ultrasound processing pipeline using GPU processing has been gaining momentum in the ultrasound community [44]. This processing strategy has been successfully applied to beamforming [26], [45], Doppler processing [46], and strain imaging [23]. Our current work, which is the first to achieve aliasing-resistant vector Doppler imaging in real time, can be regarded as a further advancement of GPU-based medical ultrasound technology.

In integrating our proposed framework into an end-to-end GPU processing pipeline for vector Doppler imaging, other

processing steps need to be included. These auxiliary steps include: 1) transfer of raw RF data to the GPU; 2) filtering and analytic signal conversion; 3) beamforming for multiple steering angles; 4) Doppler estimation for multiple steering angles; and 5) image rendering and display. As described in our previous work [26], [27], [47], these auxiliary steps altogether have an estimated GPU computation time of at most 20 ms performed image for imaging parameters similar to those used in this work. Since our framework's ELS-VD processing time is less than 20 ms/frame for the processing parameters used in our human experiments, we anticipate that a complete GPU pipeline of aliasing-resistant vector Doppler imaging would need a total computing time of at most 40 ms to generate each vector flow image from beamforming to display. Accordingly, it should be feasible to realize live vector Doppler imaging with a real-time navigation frame rate of at least 25 frames/s. The frame rate may be further improved by refining the ELS-VD framework to only process slow-time ensembles that are deemed to belong to the flow region [48].

E. Implications for Practical Applications

Since ultrasound is widely expected to be a point-of-care modality [21], it is of critical importance to engineer a reliable, real-time vector Doppler implementation that has the same accessibility as color Doppler. With the advent of such an implementation, various clinical applications can be pursued, such as the routine and point-of-care visualization of flow dynamics to identify abnormal flow [49], [50]. Whereas an experienced sonographer may be accustomed to occasional aliasing in color Doppler imaging [51], aliasing in vector Doppler imaging is more prevalent and manifests as more complex artifacts that are difficult to interpret [see Fig. 6(a)]. Relevant scenarios with vector Doppler aliasing are those with limited PRF due to imaging considerations (such as velocity resolution and depth) or system requirements. Aliasing is also relevant in vascular stenosis [6] or active hyperemia [29] (see Fig. 6), where flow speeds may be high.

By deriving both flow speed and direction, our real-time, aliasing-resistant vector Doppler imaging framework may be suitable for capturing true velocity changes with minimal operator dependence, wherein beam-to-flow angle does not need to be specified. For instance, it may enable more consistent estimation of systolic velocity, which is an indicator for stenosis severity [6], throughout the imaging view concurrently. Nevertheless, to truly capture peak systolic velocity, it may be necessary to investigate the integration of maximum Doppler frequency estimators [52] for vector Doppler, as this framework was developed on the basis of mean frequency estimation.

VII. CONCLUSION

Aliasing is a known problem that affects the performance of vector Doppler estimation. It results in spurious rendering of velocity fields in vector flow imaging, especially when performed at high frame rates. To address such a problem, this article has presented a real-time, GPU-based ELS-VD framework that can facilitate the derivation of aliasing-resistant

flow vector maps. From a diagnostic efficacy standpoint, deriving aliasing-free vector flow images is critical to clinical cardiovascular diagnostics because complex flow patterns are typically marked by the existence of a wide range of flow speeds, for which slow speeds would emerge in recirculation zones, while fast speeds would arise during systole especially in a stenosed artery. Through more intuitive and comprehensive visualization of blood flow, real-time aliasing-resistant vector Doppler may eventually enable on-site vascular diagnostics directly at the bedside. In turn, the clinical value of vector flow imaging can be better substantiated.

ACKNOWLEDGMENT

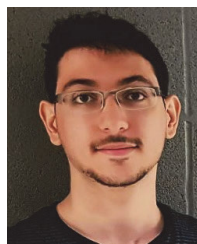
Hassan Nahas, Billy Y. S. Yiu, and Alfred C. H. Yu are with the Schlegel Research Institute for Aging and the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: alfred.yu@uwaterloo.ca).

Takuro Ishii is with the Frontier Research Institute for Interdisciplinary Sciences, Tohoku University, Sendai 980-8578, Japan.

REFERENCES

- [1] A. Goddi et al., "High-frame rate vector flow imaging of the carotid bifurcation," *Insights Imag.*, vol. 8, no. 3, pp. 319–328, Jun. 2017.
- [2] J. A. Jensen, S. I. Nikolov, A. C. H. Yu, and D. Garcia, "Ultrasound vector flow imaging—Part II: Parallel systems," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 63, no. 11, pp. 1722–1732, Nov. 2016.
- [3] J. A. Jensen, S. I. Nikolov, A. C. H. Yu, and D. Garcia, "Ultrasound vector flow imaging—Part I: Sequential systems," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 63, no. 11, pp. 1704–1721, Nov. 2016.
- [4] B. Y. S. Yiu and A. C. H. Yu, "Least-squares multi-angle Doppler estimators for plane-wave vector flow imaging," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 63, no. 11, pp. 1733–1744, Nov. 2016.
- [5] A. A. Pellett, W. G. Tolar, D. G. Merwin, and E. K. Kerut, "Doppler aliasing," *Echocardiography*, vol. 22, no. 6, pp. 540–543, Jun. 2005.
- [6] M. D. Kronick et al., "Peak systolic velocity and color aliasing are important in the development of duplex ultrasound criteria for external carotid artery stenosis," *J. Vascular Surg.*, vol. 72, no. 3, pp. 951–957, Sep. 2020.
- [7] J. S. Au et al., "Ultrasound vector projectile imaging for detection of altered carotid bifurcation hemodynamics during reductions in cardiac output," *Med. Phys.*, vol. 47, no. 2, pp. 431–440, Feb. 2020.
- [8] T. Ishii, H. Nahas, B. Y. S. Yiu, A. J. Y. Chee, and A. C. H. Yu, "Contrast-enhanced urodynamic vector projectile imaging (CE-UroVPI) for urethral voiding visualization: Principles and phantom studies," *Urology*, vol. 140, pp. 171–177, Jun. 2020.
- [9] F. Mehregan et al., "Doppler vortography: A color Doppler approach to quantification of intraventricular blood flow vortices," *Ultrasound Med. Biol.*, vol. 40, no. 1, pp. 210–221, Jan. 2014.
- [10] E. Boni, A. C. H. Yu, S. Freear, J. A. Jensen, and P. Tortoli, "Ultrasound open platforms for next-generation imaging technique development," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 65, no. 7, pp. 1078–1092, Jul. 2018.
- [11] C. Basoglu, R. Managuli, G. York, and Y. Kim, "Computing requirements of modern medical diagnostic ultrasound machines," *Parallel Comput.*, vol. 24, nos. 9–10, pp. 1407–1431, Sep. 1998.
- [12] G. York and Y. Kim, "Ultrasound processing and computing: Review and future directions," *Annu. Rev. Biomed. Eng.*, vol. 1, no. 1, pp. 559–588, Aug. 1999.
- [13] S. Muth, S. Dort, I. A. Sebag, M.-J. Blais, and D. Garcia, "Unsupervised dealiasing and denoising of color-Doppler data," *Med. Image Anal.*, vol. 15, no. 4, pp. 577–588, Aug. 2011.
- [14] X. Lai, H. Torp, and K. Kristoffersen, "An extended autocorrelation method for estimation of blood velocity," *IEEE Trans. Ultrason., Ferroelectr. Freq. Control*, vol. 44, no. 6, pp. 1332–1342, Nov. 1997.
- [15] D. Posada et al., "Staggered multiple-PRF ultrafast color Doppler," *IEEE Trans. Med. Imag.*, vol. 35, no. 6, pp. 1510–1521, Jun. 2016.
- [16] S. Ricci, L. Bassi, A. Dallai, R. Matera, and P. Tortoli, "Real-time staggered PRF for vector Doppler blood velocity assessment," in *Proc. IEEE Int. Ultrason. Symp. (IUS)*, Sep. 2017, pp. 1–4.

- [17] J. Porée et al., “Dealiasing high-frame-rate color Doppler using dual-wavelength processing,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 68, no. 6, pp. 2117–2128, Jun. 2021.
- [18] J. Flynn, R. Daigle, L. Pflugrath, K. Linkhart, and P. Kaczkowski, “Estimation and display for vector Doppler imaging using planewave transmissions,” in *Proc. IEEE Int. Ultrason. Symp.*, Oct. 2011, pp. 413–418.
- [19] I. K. Ekroll, J. Avdal, A. Swillens, H. Torp, and L. Løvstakken, “An extended least squares method for aliasing-resistant vector velocity estimation,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 63, no. 11, pp. 1745–1757, Nov. 2016.
- [20] L. N. Bohs, B. J. Geiman, M. E. Anderson, S. C. Gebhart, and G. E. Trahey, “Speckle tracking for multi-dimensional flow estimation,” *Ultrasonics*, vol. 38, nos. 1–8, pp. 369–375, Mar. 2000.
- [21] C. L. Moore and J. A. Copel, “Point-of-care ultrasonography,” *New England J. Med.*, vol. 364, no. 8, pp. 749–757, Feb. 2011.
- [22] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.
- [23] T. Idzenga, E. Gaburov, W. Vermin, J. Menssen, and C. L. De Korte, “Fast 2-D ultrasound strain imaging: The benefits of using a GPU,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 61, no. 1, pp. 207–213, Jan. 2014.
- [24] C. K. Ho, A. J. Y. Chee, B. Y. S. Yiu, A. C. O. Tsang, K. W. Chow, and A. C. H. Yu, “Wall-less flow phantoms with tortuous vascular geometries: Design principles and a patient-specific model fabrication example,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 64, no. 1, pp. 25–38, Jan. 2017.
- [25] C. C. P. Cheung et al., “Multi-channel pre-beamformed data acquisition system for research on advanced ultrasound imaging methods,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 59, no. 2, pp. 243–253, Feb. 2012.
- [26] K. L. Hansen, J. Udesen, F. Gran, J. A. Jensen, and M. B. Nielsen, “Fast blood vector velocity imaging using ultrasound: In-vivo examples of complex blood flow in the vascular system,” in *Proc. IEEE Ultrason. Symp.*, Nov. 2008, pp. 1068–1071.
- [27] J. K. Shoemaker, S. M. Phillips, H. J. Green, and R. L. Hughson, “Faster femoral artery blood velocity kinetics at the onset of exercise following short-term training,” *Cardiovascular Res.*, vol. 31, no. 2, pp. 278–286, Feb. 1996.
- [28] G. Rådegran, “Ultrasound Doppler estimates of femoral artery blood flow during dynamic knee extensor exercise in humans,” *J. Appl. Physiol.*, vol. 83, no. 4, pp. 1383–1388, Oct. 1997.
- [29] B. Saltin, “Hemodynamic adaptations to exercise,” *Amer. J. Cardiol.*, vol. 55, no. 10, pp. D42–D47, Apr. 1985.
- [30] B. Y. S. Yiu, I. K. H. Tsang, and A. C. H. Yu, “GPU-based beamformer: Fast realization of plane wave compounding and synthetic aperture imaging,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 58, no. 8, pp. 1698–1705, Aug. 2011.
- [31] B. Y. S. Yiu and A. C. H. Yu, “High-frame-rate ultrasound color-encoded speckle imaging of complex flow dynamics,” *Ultrasound Med. Biol.*, vol. 39, no. 6, pp. 1015–1025, Jun. 2013.
- [32] B. Y. S. Yiu, S. S. M. Lai, and A. C. H. Yu, “Vector projectile imaging: Time-resolved dynamic visualization of complex flow patterns,” *Ultrasound Med. Biol.*, vol. 40, no. 9, pp. 2295–2309, Sep. 2014.
- [33] K. L. Hansen, H. Møller-Sørensen, J. Kjaergaard, M. B. Jensen, J. A. Jensen, and M. B. Nielsen, “Aortic valve stenosis increases helical flow and flow complexity: A study of intra-operative cardiac vector flow imaging,” *Ultrasound Med. Biol.*, vol. 43, no. 8, pp. 1607–1617, Aug. 2017.
- [34] K. Hansen, P. Hansen, C. Ewertsen, L. Lönn, J. Jensen, and M. Nielsen, “Vector flow imaging compared with digital subtraction angiography for stenosis assessment in the superficial femoral artery—A study of vector concentration, velocity ratio and stenosis degree percentage,” *Ultrasound Int. Open*, vol. 05, no. 02, pp. E53–E59, Mar. 2019.
- [35] H. F. Younis et al., “Hemodynamics and wall mechanics in human carotid bifurcation and its consequences for atherogenesis: Investigation of inter-individual variation,” *Biomech. Model. Mechanobiol.*, vol. 3, no. 1, pp. 17–32, Sep. 2004.
- [36] D. N. Ku, D. P. Giddens, D. J. Phillips, and D. E. Strandness, “Hemodynamics of the normal human carotid bifurcation: In vitro and in vivo studies,” *Ultrasound Med. Biol.*, vol. 11, no. 1, pp. 13–26, Jan. 1985.
- [37] D. A. Steinman and C. A. Taylor, “Flow imaging and computing: Large artery hemodynamics,” *Ann. Biomed. Eng.*, vol. 33, no. 12, pp. 1704–1709, Dec. 2005.
- [38] H. Nahas, J. S. Au, T. Ishii, B. Y. S. Yiu, A. J. Y. Chee, and A. C. H. Yu, “A deep learning approach to resolve aliasing artifacts in ultrasound color flow imaging,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 67, no. 12, pp. 2615–2628, Dec. 2020.
- [39] H. Nahas, S. Huver, B. Y. S. Yiu, C. M. Kallweit, A. J. Y. Chee, and A. C. H. Yu, “Artificial-intelligence-enhanced ultrasound flow imaging at the edge,” *IEEE Micro*, vol. 42, no. 6, pp. 96–106, Nov. 2022.
- [40] J. A. Jensen, “Estimation of high velocities in synthetic-aperture imaging—Part I: Theory,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 66, no. 6, pp. 1024–1031, Jun. 2019.
- [41] J. A. Jensen, “Estimation of high velocities in synthetic-aperture imaging—Part II: Experimental investigation,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 66, no. 6, pp. 1032–1038, Jun. 2019.
- [42] H. Gao, N. Bijnens, D. Coisne, M. Lugiez, M. Rutten, and J. D’Hooge, “2-D left ventricular flow estimation by combining speckle tracking with Navier–Stokes-based regularization: An in silico, in vitro and in vivo study,” *Ultrasound Med. Biol.*, vol. 41, no. 1, pp. 99–113, Jan. 2015.
- [43] F. Yeung, S. F. Levinson, and K. J. Parker, “Multilevel and motion model-based ultrasonic speckle tracking algorithms,” *Ultrasound Med. Biol.*, vol. 24, no. 3, pp. 427–441, Mar. 1998.
- [44] K. H. So, J. Chen, B. Y. S. Yiu, and A. C. H. Yu, “Medical ultrasound imaging: To GPU or not to GPU?” *IEEE Micro*, vol. 31, no. 5, pp. 54–65, Sep. 2011.
- [45] B. Y. S. Yiu and A. C. H. Yu, “GPU-based minimum variance beamformer for synthetic aperture imaging of the eye,” *Ultrasound Med. Biol.*, vol. 41, no. 3, pp. 871–883, Mar. 2015.
- [46] L.-W. Chang, K.-H. Hsu, and P.-C. Li, “Graphics processing unit-based high-frame-rate color Doppler ultrasound processing,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 56, no. 9, pp. 1856–1860, Sep. 2009.
- [47] B. Y. S. Yiu, M. Walczak, M. Lewandowski, and A. C. H. Yu, “Live ultrasound color-encoded speckle imaging platform for real-time complex flow visualization in vivo,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 66, no. 4, pp. 656–668, Apr. 2019.
- [48] H. Nahas, B. Y. S. Yiu, A. J. Y. Chee, J. S. Au, and A. C. H. Yu, “Deep-learning-assisted and GPU-accelerated vector Doppler imaging with aliasing-resistant velocity estimation,” *Ultrasonics*, vol. 134, Sep. 2023, Art. no. 107050.
- [49] A. A. Oqlat, M. Z. Matjafri, N. Suardi, M. A. Oqlat, M. A. Abdelrahman, and A. A. Oqlat, “A review of medical Doppler ultrasonography of blood flow in general and especially in common carotid artery,” *J. Med. Ultrasonol*, vol. 26, no. 1, pp. 3–13, Jan. 2018.
- [50] J. Y. Hwang, “Doppler ultrasonography of the lower extremity arteries: Anatomy and scanning guidelines,” *Ultrasonography*, vol. 36, no. 2, pp. 111–119, Apr. 2017.
- [51] N. C. Wunderlich, R. Beigel, and R. J. Siegel, “Management of mitral stenosis using 2D and 3D echo-Doppler imaging,” *JACC, Cardiovascular Imag.*, vol. 6, no. 11, pp. 1191–1205, Nov. 2013.
- [52] S. Ricci, D. Vilkomerson, R. Matera, and P. Tortoli, “Accurate blood peak velocity estimation using spectral models and vector Doppler,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 62, no. 4, pp. 686–696, Apr. 2015.



Hassan Nahas (Member, IEEE) received the B.S. degree in electrical engineering from New York University Abu Dhabi, Abu Dhabi, United Arab Emirates, in 2017, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2021.

He is currently a Research Associate and an Associate Scientist with the Laboratory on Innovative Technology in Medical Ultrasound (LITMUS), University of Waterloo. He has specific research interests in Doppler ultrasound, flow estimation, machine learning, and ultrasound systems.

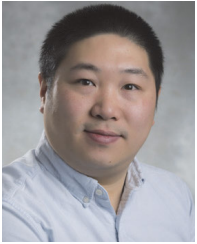


Takuro Ishii (Member, IEEE) received the B.Eng., M.Eng., and Ph.D. degrees in medical systems engineering from Chiba University, Chiba, Japan, in 2009, 2011, and 2014, respectively.

He then worked for two years with the Center for Frontier Medical Engineering, Chiba University. In 2016, he joined the University of Hong Kong, Hong Kong, as a Research Associate. Subsequently, as a JSPS Overseas Research Fellow recipient, he joined the Laboratory on

Innovative Technology in Medical Ultrasound (LIMTUS), University of Waterloo, Waterloo, ON, Canada, as an Associate Scientist, from 2017 to 2019. Since 2020, he has been an Assistant Professor with the Frontier Research Institute for Interdisciplinary Sciences, Tohoku University, Sendai, Japan. He is interested in research on ultrasound flow imaging and urodynamics.

Dr. Ishii is a Principal Investigator of the MEXT LEADER Program in Japan.

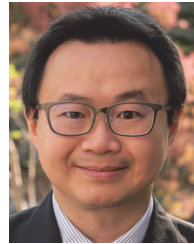


Billy Y. S. Yiu (Member, IEEE) received the B.Eng. degree (Hons.) in medical engineering and the M.Phil. degree in electrical and electronic engineering from the University of Hong Kong, Hong Kong, in 2007 and 2010, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2019.

Before joining the University of Waterloo, he was a Research Staff with the Biomedical Ultrasound Laboratory, University of Hong Kong,

from 2010 to 2016. He is currently a Research Assistant Professor in biomedical engineering with the University of Waterloo, where he is the Co-Principal Investigator of the Laboratory on Innovative Technology in Medical Ultrasound (LITMUS). His research interests are in advanced ultrasound imaging techniques and systems.

Dr. Yiu was a past recipient of the USE Young Scientist Award, the KSUM Young Investigator Gold Prize Award, and the ASA Biomedical Acoustics Best Student Paper Award. He is the Co-Instructor for the "Ultrasound Signal Processing with GPUs" short course at the IEEE Ultrasonics Symposium.



Alfred C. H. Yu (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Calgary, Calgary, AB, Canada, in 2002, and the M.A.Sc. and Ph.D. degrees in biomedical engineering from the University of Toronto, Toronto, ON, Canada, in 2004 and 2007, respectively.

From 2007 to 2015, he was the Founding Principal Investigator of the Biomedical Ultrasound Laboratory, University of Hong Kong, Hong Kong. In 2015, he relocated to the

University of Waterloo, Waterloo, ON, Canada, where he is currently the Assistant Vice-President (Research and International) and a Professor in biomedical engineering. He has long-standing research interests in ultrasound imaging and therapeutics.

Dr. Yu has been named as an NSERC Steacie Memorial Fellow. He is a fellow of the American Institute of Ultrasound in Medicine and the Engineering Institute of Canada. Earlier in his career, he was a recipient of the IEEE Ultrasonics Early Career Investigator Award, the Frederic Lizzi Award, and the Ontario Early Researcher Award. He was the Program Chair of 2023 IEEE Ultrasonics Symposium, the Past Chair of the Medical Ultrasound Group of the IEEE Ultrasonics Symposium, and an Elected AdCom Member of the IEEE UFFC Society. He is currently the Editor-in-Chief of the IEEE TRANSACTIONS ON ULTRASONICS, FERROELECTRICS, AND FREQUENCY CONTROL.