The UltraSound ToolBox

Alfonso Rodriguez-Molares[1,2,*], Ole Marius Hoel Rindal[2,†], Olivier Bernard[‡],
Arun Nair[§], Muyinatu A. Lediju Bell[§], Hervé Liebgott[‡], Andreas Austeng[†], and Lasse Løvstakken[*]

[*]Department of Circulation and Medical Imaging, NTNU, Trondheim, Norway
[†]Department of Informatics, University of Oslo, Oslo, Norway
[‡]Creatis, INSA University of Lyon, Lyon, France
[§]PULSE Lab, Johns Hopkins University, Baltimore, United States

***Abstract*—We present the UltraSound ToolBox (USTB), a processing framework for ultrasound signals. USTB aims to facilitate the comparison of imaging techniques and the dissemination of research results. It fills the void of tools for algorithm sharing and verification, and enables a solid assessment of the correctness and relevance of new approaches. It also aims to boost research productivity by cutting down implementation time and code maintenance.**

USTB is a MATLAB toolbox for processing 2D and 3D ultrasound data, supporting both MATLAB and C++ implementations. Channel data from any origin, simulated and experimental, and using any kind of sequence, e.g. synthetic transmit aperture imaging (STAI) or coherent plane-wave compounding (CPWC), can be processed with USTB.

Here we describe some of the elements of USTB such as: the ultrasound file format, the concept of the general beamformer, and the signal processing pipeline. We also show a minimal code example, and demonstrate that USTB can be used with the most used transmit sequences: STAI, CPWC, diverging wave imaging (DWI), focused imaging (FI), and retrospective transmit beamforming (RTB).

***Keywords*—Beamforming, signal processing, comparison, toolbox**

I. Introduction

Urged by the motto "publish or perish", the Academia faces a challenge of biblical proportions: the universal research-paper flood.

There are two factors that will unleash Armageddon in the ultrasound community. First is our inability to keep track of all the publications relevant to our work. This is due mainly to time constrains, but also because of the increasing specialization required to implement modern methods.

Second is the overloading of the peer-reviewing system. With an ever increasing number of submissions, few reviewers have the luxury of implementing themselves the method described in the manuscript. This makes it very difficult - if not impossible - to assess the correctness and relevance of the proposed method. It reduces the assessment of the method to the observation of some results, often 2D images, that the authors compare to some reference algorithm, often delay-and-sum (DAS).

In this way the question of whether the method is correct and relevant becomes a matter of opinion; a learned opinion perhaps, but an opinion nonetheless.

We ask the reviewers to assess the performance of a nearly black-box by mere observation of a few of its outputs, a request that should set all our scientific alarms off. Not surprisingly, the attention scholars devote to new research is dropping [1] while the proportion of scientific fraud has increased tenfold since 1975 [2]. In 2012, a study published in Nature [3] found that 47 out of 53 medical research papers on cancer research were irreproducible.

As the scientific community becomes aware of this trend reaction starts to take shape. Initiatives such as *open research* [4], [5], and *reproducible research* [6], [7] try to fight this. Using modern tools they aim to reconnect the peer-review system to the spirit that got academic journals started in the XVII century.

In the ultrasound community, a recent attempt to address this issue was made with the PICMUS challenge [8] (Tours, IUS 2016). Some researchers have started to take interest in the comparison of beamforming methods [9], and there seems to be a clear consensus on the need for tools and data to support the verification of all kind of signal processing algorithms.

II. Thesis

We believe this problem can be solved by developing a common set of tools, including:

1) an ultrasound file format, and
2) a framework for ultrasound signal processing.

A common ultrasound file format makes it possible to share processed and unprocessed datasets. Making those datasets publicly available facilitates the definition of a set of standard test cases for verification purposes. Inserting the same input dataset to different algorithms makes for a perfect scenario for comparison. Using a known data format allows for further inspection of the output of a method.

However, the file format alone does not allow a reviewer to check whether a method is correct or relevant for a given application. To address that problem a framework has to be defined so that also the algorithm can be shared. This way reviewers, and other researchers, can test the algorithm on their own terms, inserting new datasets, designing new tests.

Four institutions (Norwegian University of Science and Technology, University of Oslo, University of Lyon, and Johns Hopkins University) have come together to start the development of such a framework: the UltraSound ToolBox (USTB).

The USTB is a free MATLAB toolbox for processing ultrasonic signals, and comes with its own ultrasound file

[1]email:alfonso.r.molares@ntnu.no
[2]These authors have contributed equally to this work.

978-1-5386-3383-0/17/$31.00 ©2017 IEEE

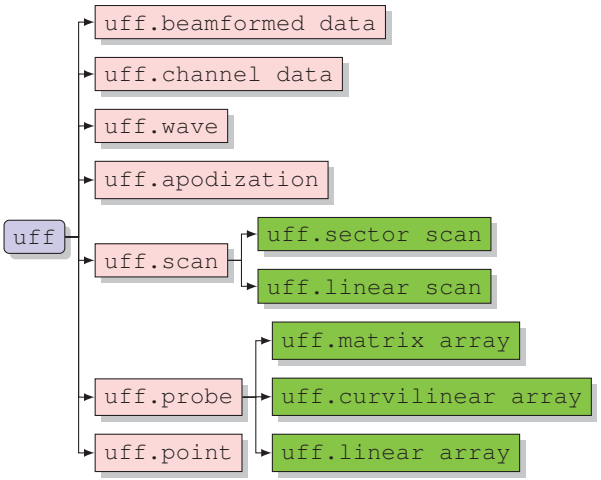

Fig. 1: USTB's UFF class organization.

format (UFF). USTB aims to cover all processing techniques from tissue and flow visualization to other image processing techniques. More information about the USTB initiative can be found at the website https://www.ustb.no. Access to the repository https://bitbucket.org/ustb/ustb is granted for anyone interested in development or testing.

III. Methods

In what follows we review part of the USTB structure and we disclose some of the elements in USTB.

A. The Ultrasound File Format

UFF data structure originates from the data class organization in USTB. A UFF class family (shown in Fig. 1) is defined, composed of a `uff` class and seven children classes. To minimize memory use `uff` is defined as a `handle` class.

All UFF classes can be dumped into HDF5 files (Hierarchical Data Format v5). HDF5 is an open file format to manage extremely large and complex data collections [10]. Originally developed by the National Center for Supercomputing Applications it is now maintained by the HDF Group. Several software platforms support HDF5 such as Java, MATLAB, Python, Octave, Mathematica, etc.

For clarity, we refer to the files dumped by USTB as UFF files (instead of HDF5) and we use the extension `.uff`.

B. Children and grandchildren classes

We follow the policy that general, unconstrained classes must be defined as children classes, while specific, constrained objects are defined as grandchildren classes. For instance we can define an arbitrary probe with `uff.probe` by specifying the position of each element; but with `uff.linear_array` we can directly define a linear array just by specifying the pitch and number of elements.

Processing is greatly simplified by making all processes take only children classes as input, rather than all possible grandchildren classes. The only requirement is then that every grandchild class must know how to define itself as its parent class.

C. The general beamformer

Both UFF and USTB revolve around the concept of the *general beamformer*. The wavefronts in most ultrasound sequences can be fully defined using a single point source P: in focused imaging (FI) and retrospective transmit beamforming (RTB) P is on the transmit focal point in front of the probe, in diverging wave imaging (DWI) P is at the wave origin behind the probe, in synthetic transmit aperture imaging (STAI) P lies on the active element, in coherent plane-wave compounding (CPWC) P is at an infinite distance but in a given direction. Using point sources to define all those waves it is possible to beamform all sequences with a single algorithm.

This is important to reduce the number of beamforming codes in the framework, facilitate intercomparison, and reduce code maintenance.

D. Time zero convention

It is necessary, however, to set an unequivocal convention for what is defined as time zero. This is often defined as the moment the first element in the array is fired. While this is perfectly correct, in USTB we favor a convention that does not depend on the probe geometry. USTB assumes that time zero corresponds to the moment the transmitted wave passes through the origin of coordinates $(0, 0, 0)$.

E. Data dimensions

The data in a `uff.channel_data` structure have four dimensions: `[time, channel, event, frame]`.

The first two dimensions run along the time sample and the channel number. The third is the event dimension whose length will often be equal to the number of waves in the sequence. The fourth dimension is the frame number.

There are two situations where the number of waves in the sequence will differ from the number of events. One is in case of multi-line transmissions (MLT), i.e. when more than one wave is transmitted in a single transmit/receive event. In such case there will be more waves in the sequence than events in `uff.channel_data`. Those are addressed with an event index in the `uff.wave` structure that indicates which event holds the data of a certain wave.

The other case is in packet acquisitions, i.e. when the same wave is transmitted in consecutive events. In this case there are more events than waves in the sequence. These events are addressed by specifying the event indexes in `uff.wave` as an array of integers.

The data in a `uff.beamformed_data` structure have also four dimensions: `[pixel, channel, event, frame]`

The first dimension of the data runs along the pixels in the spatial map. The third dimension separates the contribution of each wave to image formation, a contribution which is also referred to as "low resolution image" in the context of STAI or CPWC. The second dimension separates the contribution of each channel to image formation, equivalently to the contribution of each transmitted wave. The fourth dimension is the frame number.

While the number of dimensions remains constant along the processing pipeline (always four), the number of elements

978-1-5386-3383-0/17/$31.00 ©2017 IEEE

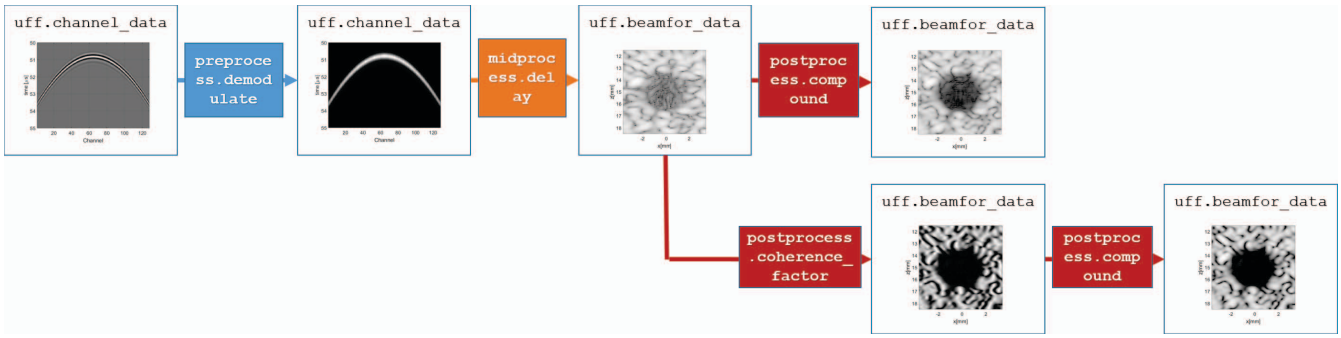

Fig. 2: Example of two processing pipelines.

in any dimensions could vary throughout the processing chain. For instance: after compounding the wave and channel dimensions can become singleton, the number of frames may change after clutter filtering.

F. The signal processing pipeline

An interesting approach to the development of a data processing framework is to set a *processing pipeline* that allows inserting an arbitrary number of processors (filters or gadgets) in a more or less arbitrary order. Such strategy has been successfully implemented in other disciplines of medical imaging [11]. Maximum flexibility is achieved when the processors are "atomic", i.e. as small as possible, so that complex processes can be built by combination of them. This strategy boost productivity by increasing code reutilization and reducing maintenance.

In USTB we define a set of processors that we divide into three types: preprocessors, midprocessors, and postprocessors. A preprocessors takes a `uff.channel_data` class and delivers another `uff.channel_data` class. A midprocessor takes a `uff.channel_data` and delivers a `uff.beamformed_data`. A postprocessor both takes and delivers `uff.beamformed_data` classes. A pipeline is built by connecting the outputs and inputs of a set of processors such as it is shown in Fig. 2. Additional data can be fed into the pipeline as parameters of every processor class. Processors can be implemented both in MATLAB or C++.

Note that the end result of the pipeline does not have to be a B-mode image. Other physical properties, different from scattering intensity, can be estimated through the pipeline and stored in a `uff.beamformed_data` structure: Doppler shift, blood flow, sound speed, elasticity, attenuation, etc.

Several processors are available in USTB implementing adaptive beamforming techniques, such as the coherence factor, phase coherence factor, generalized coherence factor, delay-multiply-and-sum, and short-lag spatial coherence. USTB also features a large set of example scripts including data import from the Verasonics and Alpinion platforms, interaction with the Field II[12] program, as well as examples of acoustic radiation force imaging, multi-line acquisition, etc. Several UFF datasets are available on the USTB website.

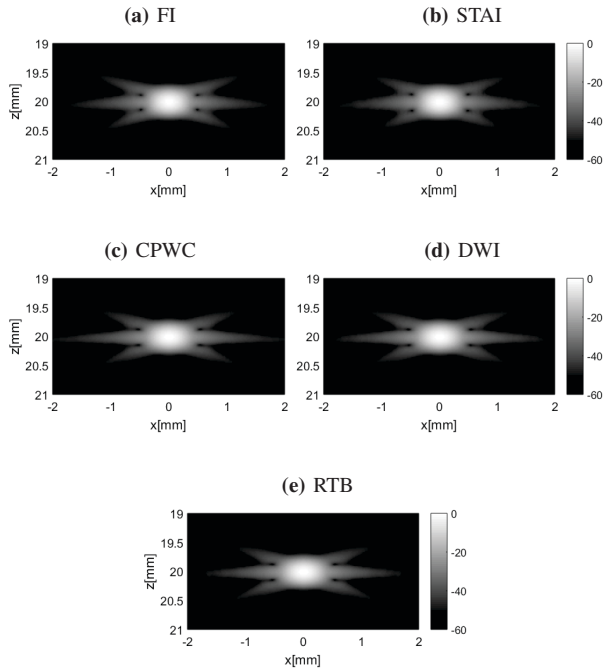

Fig. 3: PSF of the five tested beamforming methods (FI, STAI, CPWC, DWI, and RTB) generated with USTB's general beamformer.

IV. Results and Conclusions

To demonstrate the flexibility of USTB, five datasets (FI, STAI, CPWC, DWI, and RTB) have been simulated with USTB's built-in simulator and reconstructed with the general beamformer approach. It is well known that CPWC becomes equivalent to optimal FI (or STAI) under certain circumstances. Here we use USTB to show that the same result can be extrapolated to DWI and RTB, if the transmit apodization is transformed according to [13]. The equivalence is demonstrated in terms of the full width half maximum (FWHM) and side lobe level (SLL).

Fig. 3 shows the point spread function (PSF) of the tested imaging sequences showing very similar images. Table I displays the quality indexes FWHM and SLL.

We observe a bimodal distribution: nearly identical

978-1-5386-3383-0/17/$31.00 ©2017 IEEE

TABLE I: Quality indexes for the five tested methods.

Method	FWHM [μm]	SLL [dB]
STAI	357.42	-28.92
FI	360.05	-29.02
CPWC	375.73	-27.35
DWI	376.27	-27.31
RTB	375.52	-27.39
mean	369.00	-28.00
std	9.42	0.89

values for STAI and FI (relative standard deviation of σ_{FWHM}=0.52% and σ_{SLL}=0.24%), and for CPWC, DWI, and RTB (σ_{FWHM}=0.10% and σ_{SLL}=0.15%). This is due to the two-step combination of waves that occurs in the latter methods. Between the two distributions we observe a relative error of e_{FWHM}= 4.66% and e_{SLL}=5.75% that can be considered a negligible drop in image quality.

The code used to perform this comparison is available at http://www.ustb.no/code/IUS2017_abstract.m. No data is needed to run the code, but the USTB must be available in MATLAB's path.

Listing 1: USTB minimal code example

```
% download UFF file
tools.download('PICMUS_carotid_long.uff', 'http://
    ustb.no/datasets/', [ustb_path(),'/data/']);

% read channel data from UFF
channel_data=uff.read_object([local_path filename],'
    /channel_data');

% define scan
scan=uff.linear_scan();
scan.x_axis=linspace(-19e-3,19e-3,256).';
scan.z_axis=linspace(5e-3,30e-3,256).';

% initialize pipeline
bmf=beamformer();
bmf.channel_data=channel_data;
bmf.scan=scan;

% set up tx/rx apodization
bmf.receive_apodization.window=uff.window.tukey50;
bmf.receive_apodization.f_number=1.2;
bmf.receive_apodization.origo=uff.point('xyz',[0, 0,
     -Inf]);
bmf.transmit_apodization=bmf.receive_apodization;

% launch pipeline: DAS + coherent compounding
b_data=bmf.go({process.das_mex process.
    coherent_compounding});

% display image
b_data.plot();
```

Listing 1 show a minimal code example for USTB. The code downloads a CPWC dataset from the PICMUS challenge [8], beamforms it, and displays it. It gives a quick overview of the kind of code handling that can be achieved with USTB. The resulting image is shown in Fig. 4.

The UltraSound Toolbox (USTB) aims to facilitate the comparison of imaging techniques and the dissemination of research results. But it may also become a formidable research booster. Imagine how much faster would it be to test new ideas

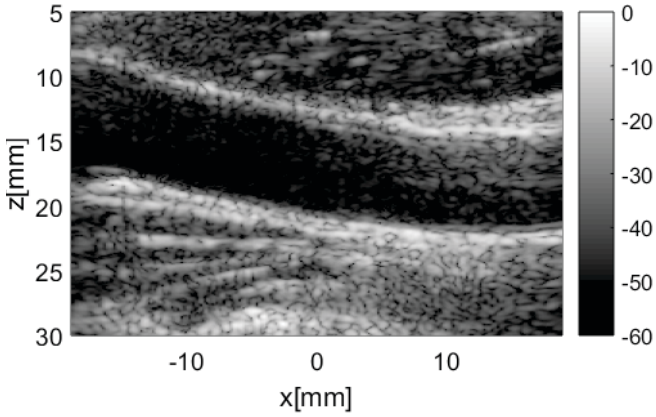

Fig. 4: PICMUS challenge dataset beamformed with the minimal code example in Listing 1.

if we had a plug-and-play implementation of all the methods in the state-of-the-art. Consider how much time could be saved in fruitless recoding and invested in taking the field forward.

REFERENCES

[1] P. D. B. Parolo, R. K. Pan, R. Ghosh, B. A. Huberman, K. Kaski, and S. Fortunato, "Attention decay in science," *Journal of Informetrics*, vol. 9, no. 4, pp. 734 – 745, 2015.

[2] F. C. Fang, R. G. Steen, and A. Casadevall, "Misconduct accounts for the majority of retracted scientific publications," *PNAS*, vol. 109, no. 42, pp. 17 028 – 17 033, 2012, http://dx.doi.org/10.1073/pnas.1212247109.

[3] C. G. Begley and L. M. Ellis, "Drug development: Raise standards for preclinical cancer research," *Nature*, vol. 483, no. 7391, pp. 531–533, 2012, 10.1038/483531a.

[4] M. Nielsen, *Reinventing Discovery*, 3rd ed. Princeton University Press, 2011.

[5] M. Woelfle, P. Olliaro, and M. H. Todd, "Open science is a research accelerator," *Nat Chem*, vol. 3, no. 10, pp. 745–748, 2011, 10.1038/nchem.1149.

[6] J. M. Wicherts and M. Bakker, "Publish (your data) or (let the data) perish! why not publish your data too?" *Intelligence*, vol. 40, no. 2, pp. 73 – 76, 2012.

[7] C. G. Begley, "Reproducibility: Six red flags for suspect work," *Nature*, vol. 497, no. 7450, pp. 433 – 434, 2013.

[8] H. Liebgott, A. Rodriguez-Molares, F. Cervenansky, J. A. Jensen, and O. Bernard, "Plane-wave imaging challenge in medical ultrasound," in *2016 IEEE International Ultrasonics Symposium (IUS)*, Sept 2016, pp. 1–4.

[9] O. M. H. Rindal, A. Austeng, H. Torp, S. Holm, and A. Rodriguez-Molares, "The dynamic range of adaptive beamformers," in *2016 IEEE International Ultrasonics Symposium (IUS)*, Sept 2016, pp. 1–4.

[10] "What is hdf5?" https://support.hdfgroup.org/HDF5/whatishdf5.html, accessed: 2017-08-11.

[11] M. S. Hansen and T. S. Sørensen, "Gadgetron: An open source framework for medical image reconstruction," *Magnetic Resonance in Medicine*, vol. 69, no. 6, pp. 1768–1776, 2013.

[12] J. Jensen and N. B. Svendsen, "Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 39, pp. 262–267, 1992.

[13] A. Rodriguez-Molares, H. Torp, B. Denarie, and L. Løvstakken, "The angular apodization in coherent plane-wave compounding [correspondence]," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 62, no. 11, pp. 2018–2023, November 2015.

978-1-5386-3383-0/17/$31.00 ©2017 IEEE