Implementation of the PLUS open-source toolkit for translational research of ultrasound-guided intervention systems

Release 0.1

Andras Lasso, Tamas Heffter, Csaba Pinter, Tamas Ungi, and Gabor Fichtinger

June 20, 2012

Laboratory for Percutaneous Surgery, School of Computing, Queen's University, Kingston, ON, Canada

Abstract

This document describes the design of the PLUS (Public software Library for Ultrasound) open-source toolkit. The toolkit provides a basic infrastructure for implementing ultrasound-guided intervention systems. Functionalities include collection of synchronized ultrasound and position data from a wide variety of hardware devices, spatial and temporal calibration, volume reconstruction, live streaming to end-user applications, and recording to and replay from file. Source code, documentation, tutorials, application examples are available with a BSD-type license at the project website: www.assembla.com/spaces/plus.

Contents

1. Introduction

Ultrasound (US) has a great potential in guiding medical interventions, as it is capable of acquiring real-time images, it has low cost, small size, and does not use ionizing radiation. Positioning of interventional tools can often be successfully completed by simple free-hand manipulation of the ultrasound transducer and surgical tools. However, there is a wide range of challenging procedures that require continuous tracking of the pose (position and orientation) of the images and surgical tools are required.

Research and development of position tracked (also known as navigated) ultrasound guided intervention systems requires advanced engineering infrastructure, which has not been available in the public domain, and single project-based solutions have not proven to be suitable as reusable platforms.

Gobbi, Boisvert, et al. developed the open-source SynchroGrab software library [1] for the acquisition, synchronization, and volume reconstruction of tracked ultrasound data. SynchroGrab was based on the Visualization Toolkit (VTK, http://www.vtk.org) open-source library and worked with a couple of ultrasound and tracking devices. Later Pace et al. [2] extended the library with electrocardiogram gating to allow reconstruction of time sequences of volumetric images. SynchroGrab was a very valuable contribution to this field and served as a good example with its architecture, utilization of open communication protocols and toolkits. However, the library lacked some essential tools, such as spatial calibration, easy-to-use, documented diagnostic and configuration tools, samples, and tests. The library is not being developed anymore, but several classes are still maintained as part of the Atamai Image Guided Surgery toolkit (https://github.com/dgobbi/AIGS).

The Image-Guided Surgery Toolkit (IGSTK, http://www.igstk.org/) is a generic, open-source, component-based framework for the implementation of image-guided surgery applications. It supports pose tracking using a number of hardware devices and contains some registration functions and a very valuable application infrastructure. However, currently it does not seem to offer essential US guidance features such as tracked image acquisition, temporal and spatial calibration, and volume reconstruction.

Stradwin is a software application developed by Treece et al. [3] at the Medical Imaging group at the Department of Engineering, University of Cambridge, UK, for freehand 3D ultrasound calibration, acquisition, measurement, and visualization. Stradwin has many useful US guidance related features, but the software is designed for performing only a few specific functions, and not for generic tool guidance. Another limitation is that its source code is not publicly available. Therefore the software is not usable as a research platform.

The Medical UltraSound Imaging and Intervention Collaboration (MUSiiC, https://musiic.lcsr.jhu.edu/Research/MUSiiC_Toolkit, [4]) research lab at Johns Hopkins University developed a toolkit for acquiring and processing tracked ultrasound data. MUSiiC addresses many of the common and more advanced problems of US-guided interventions, such as calibration, volume reconstruction, and elastography. The reported characteristics of the toolkit – modular, networked, open interfaces, open-source, real-time, parallel, hardware-agnostic –are very promising. However, the toolkit has yet to be released to the research community as an open-source package.

There are a few other frameworks for prototyping research systems, which support acquisition of real-time tracking and image data. Examples include the Computer-Integrated Surgical Systems and Technology (CISST, https://trac.lcsr.jhu.edu/cisst, [5]) libraries developed at Johns Hopkins University.

The library provides a nice generic framework, many computational algorithms, and interface for several hardware components. However, its component-based framework, which is essential for setting up a complete system, relies on a library (ZeroC ICE) that has a restrictive, non-BSD license. Also, the focus of CISST libraries are not specific to US imaging and therefore its set up and calibration for US guidance applications does not seem to be straightforward.

The goal of our work is to provide a simple-to-use, freely available open-source toolkit to facilitate rapid prototyping of US-guided intervention systems for translational clinical research. The toolkit shall offer access to various US imaging and pose tracking tools using a single, hardware-independent interface. The package shall also include all the software, hardware, and documentation for commonly needed calibration and data processing, visualization, and transfer operations.

2. Data representation

Defining common notations and formats for basic data structures is required for seamless interoperability between different groups that use, maintain, and extend the tracked ultrasound system. The basic information types that have to be represented are image, pose, and time.

2.1. Image

The PLUS toolkit uses the image representation defined in the generic image processing libraries that the toolkit is based on. However, there is an additional property – US image orientation – that has to be specified for each US image slice. In PLUS the US image orientation refers to the spatial relationship between the image slice axes and the transducer's principal axes. The image orientation is defined by a two-letter acronym, the first and second letter specify the directions corresponding to the +x and +y directions in the image coordinate system, respectively. The following directions are defined:

- Marked (M): pointing towards the marked side of the transducer.
- Unmarked (U): pointing towards the unmarked side of the transducer.
- Far (F): normal to the transducer surface, pointing away from the transducer (towards increasing depth direction).
- Near (N): normal to the transducer surface, pointing towards the transducer.

Coordinate system axes definitions for commonly used probes are shown on Figure 1. If the direction of the image +x axis is the same as the marked direction on the transducer and the image +y axis direction is the same as the transducer far direction then the image orientation is MF. If the image is flipped horizontally then the orientation becomes UF, and if flipped vertically then the orientation becomes MN.

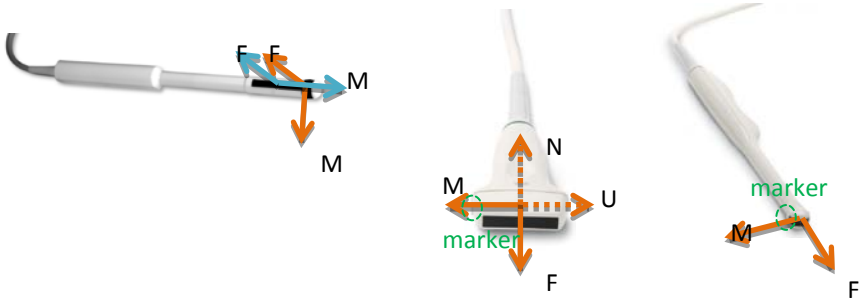

Figure 1. US image coordinate system axes orientation specification for commonly used linear, end-firing endocavity, and biplane endorectal transducers.

2.2. Pose

Pose of the acquired image slices, tools, and other objects are defined by specifying a 3D Cartesian coordinate system (a.k.a. reference frame) for each object and transforms between them. The transform is assumed to be rigid and each transform is represented by 4x4 homogeneous transformation matrix. Each coordinate system is defined by its name, origin position, axis directions, and scaling unit.

In image-guided therapy systems one of the most frequently needed operations is to compute transform between two arbitrary reference frames. Although this operation is a simple multiplication and inversion of selected matrices in a certain order, the manual computation is error-prone and time-consuming. A commonly used technique (see e.g., IGSTK [6]) is to store all the transforms as edges of a graph, where each vertex of the graph corresponds to a coordinate system and compute the transforms automatically. This approach is implemented in PLUS: all the transforms are stored in system-wide transform repository. Each transform is uniquely identified by its name.

PLUS uses a standard naming convention for constructing transform names. The name is constructed from the name of the reference frames that it transforms from (e.g., “FrameA”), the “To” word, and the reference frame name it transforms to (e.g., “FrameB”): <FrameA>To<FrameB>. This naming convention is better than the commonly transform names such as “CalibrationMatrix” or “ReferenceTransform”, because if using ad-hoc names are used then the meaning of the transform must be documented separately, and that separate documentation may be missing, incomplete, or incorrect.

3. Data acquisition

While typically fused pose and image information is needed, the US and tracking data are generally acquired by multiple independent devices. As image frames and object poses are acquired at distinct time points, they have to be resampled before the fusion. Following the implementation in SynchroGrab, in PLUS the pose information is resampled at each time point when an image frame is acquired. The position part is interpolated with weighted averaging; the orientation component is computed by spherical linear interpolation.

3.1. Temporal calibration

Fusion of imaging and pose data requires accurate timestamp information for each item. US-guided intervention systems typically consists of three loosely coupled subsystems: tracker, US scanner, and data collector computer. Not all hardware devices provide accurate timestamp for the acquired data items and even if they do, the timestamps are usually created by hardware clocks, which are not synchronized to each other. Therefore, a temporal calibration method is needed, which can correlate the timestamps provided by the various data sources.

The temporal calibration method in PLUS assumes that data sources attach timestamps on the acquired data with a constant, unknown delay. The output of the temporal calibration is the time offset between the different data sources. The temporal calibration method is similar to the algorithm described in [3]. The technique is based on correlation of the pose and image data that is acquired while moving the transducer with a continuous, quasi-periodic motion, imaging a planar object.

3.2. Spatial calibration

The goal of spatial calibration is to determine the transform between coordinate systems of an object (e.g., image slice, calibration phantom, and stylus) and a marker that is rigidly attached to that object. Pivot calibration, landmark registration, and a multiple N-shaped fiducial based image calibration methods are implemented in Plus [7].

To ensure reproducibility of the method, detailed description of calibration phantom, its CAD model (Figure 2), and assembly instructions are provided in the PLUS project's documentation. The position and size of the N fiducials are configured in a file so that the calibration phantom can be easily modified, without requiring any software change (Figure 3).

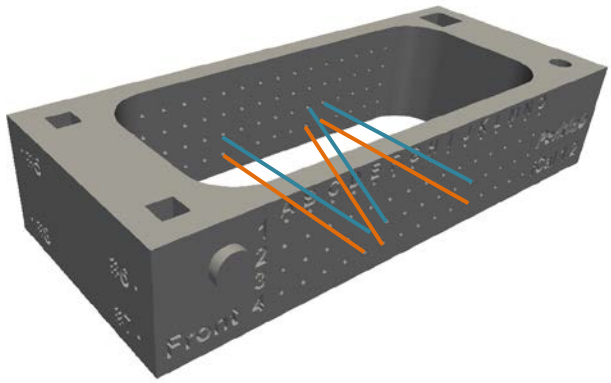

Figure 2. Model of phantom used for spatial calibration of the US guidance system. The phantom body can be produced by printing the 3D model of the phantom (the file is downloadable from the project webpage) and following the step-by-step instructions on the project webpage.

```
<PhantomDefinition>
  <!-- Supported types are: Double-N, U-Shaped-N -->
  <Description
    Name="fCAL"
    Type="Double-N"
  />
  <Geometry>
    <Pattern Type="NWire">
      <Wire Name="1:E3_e3" EndPointFront="20.0 0.0 5.0" EndPointBack="20.0 40.0 5.0" />
      <Wire Name="2:F3_j3" EndPointFront="25.0 0.0 5.0" EndPointBack="45.0 40.0 5.0" />
      <Wire Name="3:K3_k3" EndPointFront="50.0 0.0 5.0" EndPointBack="50.0 40.0 5.0" />
    </Pattern>
    <Pattern Type="NWire">
      <Wire Name="4:E4_e4" EndPointFront="20.0 0.0 0.0" EndPointBack="20.0 40.0 0.0" />
      <Wire Name="5:J4_f4" EndPointFront="45.0 0.0 0.0" EndPointBack="25.0 40.0 0.0" />
      <Wire Name="6:K4_k4" EndPointFront="50.0 0.0 0.0" EndPointBack="50.0 40.0 0.0" />
    </Pattern>
    <Landmarks>
      <Landmark Name="#1" Position="95.0 5.0 15.0" />
      <Landmark Name="#2" Position="95.0 40.0 15.0" />
      <Landmark Name="#3" Position="95.0 40.0 0.0" />
      <Landmark Name="#4" Position="95.0 0.0 0.0" />
      <Landmark Name="#5" Position="-25.0 40.0 15.0" />
      <Landmark Name="#6" Position="-25.0 0.0 10.0" />
      <Landmark Name="#7" Position="-25.0 0.0 0.0" />
      <Landmark Name="#8" Position="-25.0 40.0 0.0" />
    </Landmarks>
  </Geometry>
</PhantomDefinition>
```

Figure 3. Extract from the Device Set configuration file. The customization of the calibration phantom geometry is simple, as it is specified in the XML file.

4. Volume reconstruction

US volume reconstruction methods construct a 3D Cartesian volume from a set of 2D US frames that are sweeping across a region. The volume may be used for many purposes, including 3D visualization, multi-planar reconstruction, and image-based registration.

The basic volume reconstruction method in PLUS is adopted from SynchroGrab [1]. The first step of the procedure is insertion of 2D image slices into a 3D volume. This is implemented by iterating through each pixel of the slice and inserting the pixel value into the corresponding volume voxel. The second step of the volume reconstruction procedure is hole filling. This step is needed as the spacing between the acquired image slices and their orientation may vary, while all the slices of the reconstructed volume must be parallel and equally spaced.

5. Live data streaming

US-guided intervention systems often have to be implemented in a loosely integrated, heterogeneous environment. Data collection, processing, and visualization may need to be performed in different processes, sometimes on different computers. To fulfill these needs PLUS provides live streaming output and accepts live streaming input through the OpenIGTLink protocol. OpenIGTLink is an open, very simple, light-weight, TCP/IP based communication protocol that is specifically developed for image-guided therapy applications. Several hardware devices systems (e.g., Siemens MRI scanners, BrainLab navigation systems, position trackers, robotic devices) and software applications (e.g., 3D Slicer [8]) supports live data sending and receiving using this protocol.

PLUS includes a standalone server application that can acquire data from multiple hardware devices either by direct connection or through OpenIGTLink. The application fuses all data and then transfers to one or more clients that connected to the server.

6. Implementation

2.8.1 System architecture

PLUS consists of a library and is divided into two parts: library and applications. PLUS library contain the implementation of all the algorithms, data collection, interfacing with hardware devices, tests, examples, and basic common classes. PLUS applications include an end-user application (fCal) for performing all of calibration steps of free-hand US system, a standalone application for volume reconstruction, a server that can forward the acquired data to multiple clients using OpenIGTLink, and some diagnostic tools.

PLUS relies on free, open-source libraries that are commonly in the implementation of medical image computing systems: ITK, VTK, OpenIGTLink, and QT. PLUS also uses device drivers and software development kits provided by the device manufacturer for acquiring data from certain hardware devices.

Applications that uses PLUS can be implemented similarly to existing PLUS applications: directly using the PLUS library and other third-party libraries as needed. However, software for image-guided therapy workflows can be much more efficiently implemented by using the 3D Slicer application platform [8]. Using PLUS, 3D Slicer, and a few readily available Slicer extension modules it is possible to set up image guidance application prototypes without the need for any additional software development. The Plus Server acquires the data and forwards it to 3D Slicer application. The application receives the data and the image-guidance module visualizes them (Figure 4).

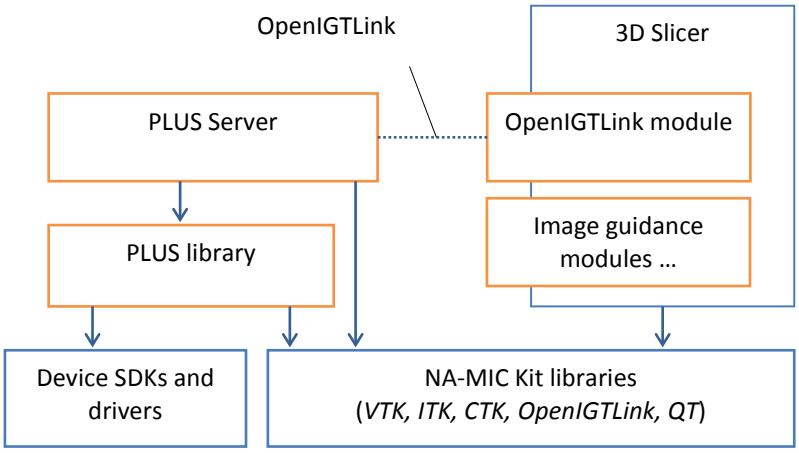

Figure 4. Building 3D Slicer plug-in modules without being directly linked to PLUS, using OpenIGTLink connection.

3D Slicer's core modules already provide many options for visualizing and using the received data in many different ways, and there are several additional modules are available for specifically image-guided therapy as part of the SlicerIGT toolkit (www.assembla.com/spaces/slicerrt). Live Ultrasound is one of the SlicerIGT toolkit's modules, which shows the acquired US slices in the application's 3D viewer in its actual position in real-time. The OpenIGTLink connection allows a very clean separation of the visualization and processing application from the data acquisition process and the data can be even acquired on a different computer. If flexibility is not required and network throughput is a concern then Slicer modules can be implemented that link to the PLUS library and use its functions directly.

2.8.2 File formats

There are several standard, widely used file formats for 2D and 3D medical images, however there were no such commonly used file format for the storage of tracked US data. In PLUS the open standard MetaIO image (http://www.itk.org/Wiki/ITK/MetaIO) format was chosen to store the acquired tracked US data. Reading and writing images in this format is available in many medical image computing software applications and toolkits (including 3D Slicer, ParaView, ImageJ; ITK, VTK). The file header is human-readable, therefore any text editor can be used to view or edit the file contents. Various pixel types can be stored in either uncompressed or lossless compression. Tracking data, status, time, and any other information, such as physiological waveforms can be stored in custom fields. A special type of field – the custom *frame* field – is introduced for storing information for each frame. The custom frame fields start with the "Seq_FrameNNNN_" prefix, where NNNN is the 0-based frame number. Transforms for each frame are stored as "Seq_FrameNNNN_FrameAToFrameBTransform.

```
ObjectType = Image
NDims = 3
DimSize = 820 616 188
ElementType = MET_UCHAR
UltrasoundImageOrientation = MF
…
Seq_Frame0000_ProbeToTrackerTransform  =  -0.0215859  0.20331  0.978876  264.765  -0.156876
     0.966287 -0.204155 16.5199 -0.987382 -0.157969 0.0110363 -8.81807 0 0 0 1
Seq_Frame0000_ProbeToTrackerTransformStatus = OK
Seq_Frame0000_ReferenceToTrackerTransform = -0.106169 0.984401 -0.140294 249.585 -0.076174
     -0.148729 -0.98594 82.5996 -0.991426 -0.0939896 0.0907762 -42.0812 0 0 0 1
Seq_Frame0000_ReferenceToTrackerTransformStatus = OK
Seq_Frame0000_StylusToTrackerTransform  =  -0.32614  0.49856  0.803163  79.1394  0.934506
     0.041887 0.353473 -16.2967 0.142585 0.865843 -0.479569 145.554 0 0 0 1
Seq_Frame0000_StylusToTrackerTransformStatus = OK
Seq_Frame0000_Timestamp = 425.398686
Seq_Frame0000_ImageStatus = OK

Seq_Frame0001_ProbeToTrackerTransform  =  -0.0212408  0.203836  0.978775  264.877  -0.156974
     0.966173 -0.204618 16.5199 -0.987374 -0.157988 0.0114746 -8.81807 0 0 0 1
Seq_Frame0001_ProbeToTrackerTransformStatus = OK
Seq_Frame0001_ReferenceToTrackerTransform  =  -0.105917  0.984393  -0.140544  249.696  -
     0.0759356 -0.148934 -0.985927 82.7112 -0.991471 -0.0937538 0.0905251 -42.0812 0 0 0 1
Seq_Frame0001_ReferenceToTrackerTransformStatus = OK
Seq_Frame0001_StylusToTrackerTransform  =  -0.32614  0.49856  0.803163  79.1394  0.934506
     0.041887 0.353473 -16.2967 0.142585 0.865843 -0.479569 145.554 0 0 0 1
Seq_Frame0001_StylusToTrackerTransformStatus = OK
Seq_Frame0001_Timestamp = 425.477657
Seq_Frame0001_ImageStatus = OK

Seq_Frame0002_ProbeToTrackerTransform  =  -0.0208243  0.204424  0.978661  264.877  -0.157086
     0.966046 -0.205132 16.4083 -0.987365 -0.158006 0.011995 -8.81807 0 0 0 1
…
ElementDataFile = LOCAL
(pixel values)
```

Figure 5. Sample MetaIO image file storing a list of tracked US image slices.

There are numerous configuration parameters in PLUS for the specification of which hardware device to use, acquisition settings, and various options for calibration, volume reconstruction algorithms, and applications. PLUS stores all these configuration settings in a single XML file: the Device Set configuration file. XML format is chosen because XML is a widely supported, open standard, the files are human-readable, self-describing, and new data elements and attributes can be easily added at any time. PLUS also uses an optional XML file – the Preferences configuration file (PlusConfig.xml) – for storing user preferences that are valid for the specific computer where the toolkit is installed. These preferences include information such as logging level, directory paths, preferred editor, and preferred Device Set configuration file.

2.8.3 Development process

PLUS developers use state-of-the-art tools and best practices for maintaining and extending the toolkit. All data, including source code, test data, CAD models, etc. are stored and shared using a third-party cloud-based integrated software development collaboration service, therefore outsourcing the setup, maintenance, and development of the collaboration infrastructure. Implementation of bugfixes and feature requests are tracked using tickets.

Application programming interface documentation is generated from the commented source code using Doxygen. User documentation is provided on the wiki pages of the collaboration suite. Step-by-step tutorial presentations and videos are also provided to explain and demonstrate complex procedures.

PLUS uses the CMake (www.cmake.org) build system to download and build all the required software libraries and build the toolkit itself. CTest is used for automatic testing of all the algorithms and major components of the toolkit. Several computers are configured to run tests automatically after each modification in the repository, publish the results on a CDash (www.cdash.org) dashboard server, and automatically notify developers and administrators about any failures. Testing of the essential graphical user interface elements is implemented using Sikuli (sikuli.org).

7. Results

Acquisition and spatial and temporal calibration of US image and pose tracking data is implemented in PLUS. The functionalities can be linked to any custom software and also available for users with convenient graphical user interface in the fCal (free-hand calibration) application. The user can choose between various data acquisition devices without making any software changes, just by editing the Device Set configuration file. Acquired imaging and tracking data can be stored in standard MetaIO image file format with custom fields. Continuous real-time transfer of the acquired data is provided through OpenIGTLink protocol (Figure 6). A command-line application (VolumeReconstructor) is implemented for reconstructing volumes from tracked US frames. Several simple applications that can be used for testing, diagnostics, and performance optimization of data acquisition and processing algorithms.

The following tracking devices are currently supported in PLUS: Ascension trakSTAR 3DG (same as the SonixGPS system built into Ultrasonix scanners), NDI electromagnetic and optical trackers (Aurora, Polaris, Optotrak), Claron MicronTracker, and Phidgets Spatial 3/3/3 MARG sensor (only for orientation). These image acquisition devices are supported: Ultrasonix US scanners (SonixRP, SonixMDP, SonixTouch, SonixTablet) through Ulterius interface (B-mode and RF-mode acquisition), Epiphan framegrabbers, Imaging Controls framegrabbers, and Video for Windows compatible image sources. In addition to hardware devices, tracking and image information can be acquired from any OpenIGTLink compatible device and can be also replayed from files.

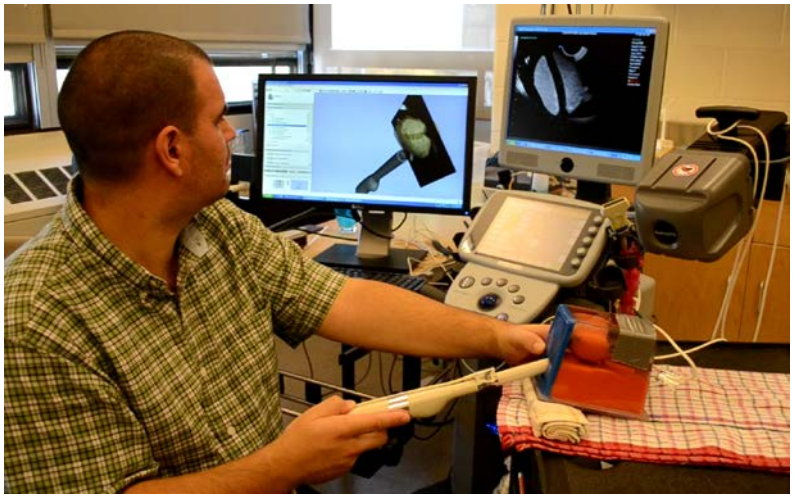

Figure 6. Tracked ultrasound for biopsy navigation with 3D Slicer. US imaging and tracking data are acquired by PLUS and streamed to 3D Slicer for live display.

8. Conclusion

PLUS is open, available to all researchers and system developers. PLUS is distributed with a BSD license, which allows free, unrestricted use, although PLUS is not certified for any particular clinical application. The source code and all documentations and procedures are freely available, without requiring any registration, at the project webpage (https://www.assembla.com/spaces/plus). Only open standards for implementing the toolkit, such as XML and MetaIO for file storage and OpenIGTLink for live streaming. PLUS is easily extensible thanks to the flexibility of the used data representation and its modular design. Maintenance work of the toolkit is supported by the automatic testing suite (which helps early identification of any regressions), the various diagnostic functionalities, and by web-based the collaboration infrastructure.

The first public version of PLUS has been released November 2011 and since then it has been used by several research groups and companies for implementing US-guided intervention systems for translational research and product feasibility studies.

9. Acknowledgments

This work was supported through the Applied Cancer Research Unit program of Cancer Care Ontario with funds provided by the Ontario Ministry of Health and Long-Term Care. Gabor Fichtinger was funded as a Cancer Ontario Research Chair. Tamas Ungi was supported as a Queen's University–Ontario Ministry of Research and Innovation Postdoctoral Fellow.

The PLUS toolkit reused parts of the SynchroGrab library for pose data interpolation, volume reconstruction, and interfacing with Ultrasonix, NDI, and Claron devices. The development of the

SynchroGrab library was funded in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Canadian Institutes of Health Research (CIHR), and Canada Foundation for Innovation (CFI), principal investigators Purang Abolmaesumi and Parvin Mousavi.

The authors wish to thank the students and staff engineers at Queen's University, University of British Columbia, Robarts Research Institute, and Kitware for their contribution to the toolkit.

10. References

[1] Boisvert, J., Gobbi, D., Vikal, S., Rohling, R., Fichtinger, G.,Abolmaesumi, P., 2008. An open-source solution for interactive acquisition, processing and transfer of interventional ultrasound images. In: MICCAI 2008, International Workshop on Systems and Architectures for Computer Assisted Interventions. pp. 1–8.

[2] Pace, D., Gobbi, D., Wedlake, C., Gumprecht, J., Boisvert, J., Tokuda, J., Hata, N., Peters, T., 08 2009. An open-source real-time ultrasound reconstruction system for four-dimensional imaging of moving organs. In: MICCAI 2009, International Workshop on Systems and Architectures for Computer Assisted Interventions. pp. 1–8.

[3] Treece, G. M., Gee, A. H., Prager, R.W., Cash, C. J. C., Berman, L. H., Apr 2003. High-definition freehand 3-d ultrasound. Ultrasound Med Biol 29 (4), 529–546.

[4] Stolka, P. J., Kang, H.-J., Boctor, E., 2010. The musiic toolkit: Modular real-time toolkit for advanced ultrasound research. In: MICCAI 2010, International Workshop on Systems and Architectures for Computer Assisted Interventions.

[5] Kapoor, A., Deguet, A., Kazanzides, P., 2006. Software components and frameworks for medical robot control. In: Proc. IEEE Int. Conf. Robotics and Automation ICRA 2006. pp. 3813–3818.

[6] Gary, K., Ibanez, L., Aylward, S., Gobbi, D., Blake, M. B., Cleary, K., 2006. Igstk: an open source software toolkit for image-guided surgery. Computer 39 (4), 46–53.

[7] T.K. Chen, A.D. Thurston, M.H. Moghari, R.E. Ellis, and P. Abolmaesumi. A real-time ultrasound calibration system with automatic accuracy control and incorporation of ultrasound section thickness. In Proceedings of SPIE Medical Imaging, 2008.

[8] Pieper, S., Lorensen, B., Schroeder, W., Kikinis, R., 2006. The na-mic kit: Itk, vtk, pipelines, grids and 3d slicer as an open platform for the medical image computing community. In: Proceedings of the 3rd IEEE International Symposium on Biomedical Imaging: From Nano to Macro. Vol. 1. pp. 698–701.