HHS Public Access

Author manuscript

Proc SPIE Int Soc Opt Eng. Author manuscript; available in PMC 2011 August 11.

Published in final edited form as:

Proc SPIE Int Soc Opt Eng. 2011 ; 7967: . doi:10.1117/12.878371.

Integrating Medical Imaging Analyses through a High-throughput Bundled Resource Imaging System

Kelsie Covington[a], **E. Brian Welch**[b,c], **Ha-Kyu Jeong**[b], and **Bennett A. Landman**[a,b,c]

[a]Electrical Engineering, Vanderbilt University, Nashville, TN, USA 37235

[b]Institute of Imaging Science, Vanderbilt University, Nashville, TN, USA 37235

[c]Radiology and Radiological Sciences, Vanderbilt University, Nashville, TN, USA 37235

Abstract

Exploitation of advanced, PACS-centric image analysis and interpretation pipelines provides well-developed storage, retrieval, and archival capabilities along with state-of-the-art data providence, visualization, and clinical collaboration technologies. However, pursuit of integrated medical imaging analysis through a PACS environment can be limiting in terms of the overhead required to validate, evaluate and integrate emerging research technologies. Herein, we address this challenge through presentation of a high-throughput bundled resource imaging system (HUBRIS) as an extension to the Philips Research Imaging Development Environment (PRIDE). HUBRIS enables PACS-connected medical imaging equipment to invoke tools provided by the Java Imaging Science Toolkit (JIST) so that a medical imaging platform (e.g., a magnetic resonance imaging scanner) can pass images and parameters to a server, which communicates with a grid computing facility to invoke the selected algorithms. Generated images are passed back to the server and subsequently to the imaging platform from which the images can be sent to a PACS. JIST makes use of an open application program interface layer so that research technologies can be implemented in any language capable of communicating through a system shell environment (e.g., Matlab, Java, C/C++, Perl, LISP, etc.). As demonstrated in this proof-of-concept approach, HUBRIS enables evaluation and analysis of emerging technologies within well-developed PACS systems with minimal adaptation of research software, which simplifies evaluation of new technologies in clinical research and provides a more convenient use of PACS technology by imaging scientists.

Keywords

Image Processing; PACS; MRI; JAVA; XML; WDSL

INTRODUCTION

Picture Archiving and Communication Systems (PACS) enables systematic and verifiable tracking, access, visualization, and data providence for medical imaging data throughout the healthcare information management system from acquisition to interpretation, especially for three- and higher- dimensional imaging techniques, such as magnetic resonance imaging [1]. These pervasive informatics platforms serve as ideal targets in which to integrate and evaluate new analyses and techniques for eventual inclusion in patient care [2]. Advanced

integration of PACS system throughout the entire chain of care is an active area of informatics research and engineering development [3].

A variety of imaging frameworks have emerged to fit the needs of research groups both in terms of functionality and visualization capabilities [4–7], while a multitude of specialized image analysis techniques remain largely in the hands of their creators and collaborators. Integrating these techniques, and bridging the gap between these emerging image analysis technologies and those technologies ready for clinical research and patient trials, is a monumental effort [8–11]. Here, we explore a set of technologies which could allow for earlier feedback of clinical collaborators on emerging technologies and facilitate collaborations between imaging researchers, who typically develop methods using custom software platforms, and clinical researchers, who typically prefer established visualization and analysis approaches afforded by PACS-centric systems.

Custom image post-processing in the clinical as well as the research medical imaging setting is often limited by the separation of the processing from the image acquisition hardware platform that connects to the institution's PACS. The separation decreases the chances that the post-processed images will ever be entered into the PACS and creates an obstacle for routine usage and acceptance of the available post-processing algorithms. For example, novel image reconstruction techniques often require substantive changes to the software pipeline within the image reconstruction environment. PACS systems are typically not configured to provide seamless acceptable, transmission, and access the raw data (prior to reconstruction), so it is often not possible to use a PACS system to archive the requisite non-imaging data and meta-data necessary for reconstruction. Hence, an alternative data processing workflow must be constructed, which can involve error-prone human-machine interaction (e.g., numerous clicks). Finally, the alternative workflow must be configured as a DICOM send node and must retain institution specific accession syntax (which requires additional meta-data and programming) and must be authorized to transmit data to the central PACS server (which may conflict with institutional security policies).

To address this problem, we developed a high-throughput bundled resource imaging system (HUBRIS) for integrated medical imaging analysis which directly interfaces with the scanner and injects custom image analysis results into the PACS pipeline at the point of acquisition. HUBRIS extends the Philips Research Imaging Development Environment (PRIDE, Philips Healthcare, Best, The Netherlands). Using HUBRIS, three distinct systems (the image acquisition equipment, image post-processing algorithm workflow server and algorithm execution server) seamlessly communicate to provide advanced image post-processing directly from the medical imaging equipment. The images generated by this framework can be sent to the institution's PACS in the same manner as other acquired images.

METHODS

In this proof-of-concept case, the medical imaging equipment is a magnetic resonance imaging scanner (Philips Healthcare, Best, The Netherlands) running release 2.6.3.4 software or newer. The scanner is equipped with the inline PRIDE functionality that allows

any user-defined executable or script to be invoked automatically upon the completion of an image acquisition. The automatically called process is passed a unique identifier that enables the process to extract the associated image volume from the scanner's database. Additional parameters can be passed using the inline PRIDE process invocation. Upon completion of operation, the invoked inline PRIDE process can write resulting images to a pre-defined folder location. Images found in that location will be read automatically by the scanner's software and imported into the scanner's database to enable the new images to be easily sent to the PACS. Typically, inline PRIDE processes run on the scanner console computer and are restricted in memory and CPU usage. Furthermore, the available post-processing library and tools are limited to those that have been compiled and provided for the scanner console computer's operating system (Windows XP embedded 64-bit).

To increase the power and flexibility of the image post-processing workflow available directly from the scanner, we have integrated the Java Image Science Toolkit (JIST) [6] server with the scanner's inline PRIDE capability to provide access to a large library of algorithms using JIST's open application program interface (API) [12]. JIST is an open-source, platform-independent framework to rapidly develop image analysis tools and distribute them to the scientific community. JIST provides a high level of interoperability, advanced batch processing tools, and requires minimal additional programming or computational overhead. Finally, to address the demands for memory and processing power by many of the algorithms, the JIST server is able to sends jobs to a supercomputer grid.

In this case, the processing power is provided by Vanderbilt's Advanced Computing Center for Research and Education (ACCRE, http://www.accre.vanderbilt.edu/). ACCRE is an education/research grid High Performance Computing facility with a theoretical peak performance of 20 TeraFLOPS using approximately 3,000 interlinked x86-based core processors running 64-bit Linux. Compute nodes all have a disk drive and dual copper gigabit Ethernet ports, but hardware visualization capabilities are disabled. Resource management, scheduling of jobs, and usage tracking are handled by an integrated scheduling system by Moab/Torque.

RESULTS

Figure 1 illustrates the data flow in the HUBRIS system. Running the "Start Hubris" tool in Inline PRIDE opens StartHubris.xml. These tools are defined with a simple, human readable eXtensible Markup Language (XML) definition file as shown in Figure 2. This file configures PRIDE to call StartHubris.bat, which first calls the PRIDE-Leacher tool to pull the image data (or raw data) from the internal PRIDE database and copy it into to the "tempinputseries" folder. While access to raw data is not a natural part of the in-line PRIDE feature, the unique identifier passed to the inline PRIDE invocation is enough information to query the scanner's database and to extract raw data associated with a given MR imaging series. As long as the called script is able to go all the way from raw data to XML/REC image results, the inline PRIDE tool accepts the results even if the XMLRECleacher program is never used. However, in the cases where raw data is saved and there are some scanner produced images, those XML/REC files are a convenient template for stuffing in the new reconstruction results. Different configuration files with different default arguments can

be configured to automatically run for differing modalities. While the scanner's inline PRIDE feature is limited to serial execution, and as such inline PRIDE jobs cannot run simultaneously, multiple images and raw data acquisitions can be simultaneously processed by sending these to the "Stage Hubris" script.

A Web Services Description Language (WSDL) Axis 2 server (Figure 4) provides access to the JIST image processing server routines (Figure 3), and facilitates data transfer via client- and server-side Java methods that use Plain Old Java Objects (POJOs). The Axis2 server automatically converts these POJOs to WSDL XML, which clients use to access server data, and which the server stores on the Internet to be accessed by clients.

Once the data are extracted, a Java class HubrisClient, with WSDL support via Axis 2 libraries, is initialized and loads the data from the tempinputseries folder. HubrisClient is designed to either automatically pass a predefined script along with the tempinputseries data to the server, or in the absence of a script or server location information to load a graphical user interface (GUI).

The GUI simplifies debugging and configuration of the client-server architecture. The GUI allows the user to specify the IP address and port number for the Axis 2 server. Once HubrisClient has successfully contacted the server, it uses Axis 2 WSDL messages to call Jist's discover() method from a Java class HubrisServer, which is called by the Axis 2 server. HubrisServer collects the results from discover(), and builds a list of available algorithms along with their inputs and requirements. This list is returned to HubrisClient, which populates the GUI with the possible algorithms (Figure 5a). Note that HUBRIS/JIST infrastructure detects any custom modules installed on the server and exposes these functionalities to the client — even if the client does not know *a priori* that such modules exist.

When the user clicks on a desired algorithm, a new window is created which lists all available input arguments, notes which arguments are required, and allows the user to set the desired input arguments (Figure 5b). The user can then choose to run the algorithm with the appropriate input arguments, or generate the corresponding script for use in automatic run configurations.

If the user elects to run the algorithm as defined, HubrisClient uses Axis 2 generated WSDL messages to call the RunAlgorithm method in the HubrisServer class on the server, to send the image data, and to provide any other necessary data as arguments as illustrated in Figure 6. RunAlgorithm parses the arguments and sends the request and all necessary data to JIST's run() method on the cluster, which calls the specified algorithm (Figure 7) and returns the output to RunAlgorithm. The server sends the output back to HubrisClient, which writes the output data in an appropriate place. Finally PRIDEXMLRECleacher removes the original image data from tempinputseries to prevent a buildup of data on the server. For grid processing, HUBRIS invokes executables through the X virtual framebuffer (Xvfb, David P. Wiggins, The Open Group, Inc.) framework so that a software X session is available for algorithms which require access to visual elements (e.g., font handles or rendering libraries) even if the elements are not displayed on screen.

Image arithmetic is a simple example of HUBRIS functionality. In order to add two reconstructed images from the MRI console, the user sends the first image to a temporary location via StageHubris, The user can then send the second image via StartHubris, either by sending the image with a predefined HUBRIS script that sends the two images to the server and requests that the two images be added, or by running StartHubris with no commands. The latter will cause the HUBRIS GUI to be initialized, at which point the user can select the ImageArithmetic algorithm, leave the operation argument at its default (add), and input the locations of the images in the "inFirst" and "inSecond" arguments respectively. Finally the user selects "Run Algorithm," at which point the image data and arguments are sent to the server, the server performs the operation, and the output data is returned to the tempoutputseries folder, which the user can then access. Figure 6 demonstrates an example of the potential use of HUBRIS for testing a novel off-line image reconstruction method for diffusion-weighted (DW) sensitivity encoding (SENSE) MR images. It compares fractional anisotropy (FA) map acquired using conventional single-shot (ssh) (Figure 8a) and new multi-shot (msh) EPI (Figure 8b) technique. Usually the evaluation of a new imaging method has been limited to specific users and systems due to relatively large volume of raw data and/or intensive user interactions and computations required for off-line image reconstruction. However, the use of HUBRIS system could provide more convenient environment for imaging scientists and clinicians by incorporating any conventional and emerging technologies within existing PACS system.

CONCLUSIONS

We have presented an advanced proof-of-concept workflow to enable seamless medical image post-processing invoked from a clinical magnetic resonance imaging scanner. The generated images can then easily be transferred to the institution's PACS like any other typically acquired medical image. The system integrates three distinct systems: (1) the medical imaging acquisition hardware, (2) a JIST server to provide access to advanced open-source post-processing algorithms, and (3) a high-powered supercomputer to execute the requested algorithms.

ACKNOWLEDGEMENTS

This project was supported by NIH/NINDS 1R01NS056307 and NIH/NINDS 1R21NS064534.

REFERENCES

1. Schwartz LH, Kijewski P, Lynch K, et al. Review and interpretation of MR imaging studies with PACS: creating uniform series descriptors for radiologists and referring physicians. AJR Am J Roentgenol. 2002; 179(3):575–577. [PubMed: 12185022]
2. Yoshinobu T, Abe K, Sasaki Y, et al. Data Management Solution for Large-Volume Computed Tomography in an Existing Picture Archiving and Communication System (PACS). J Digit Imaging. 2009
3. Fillicelli T. Future of PACS: advanced integration with RIS and workflow management. Radiol Manage. 2001; 23(1):12–13. [PubMed: 14608785]
4. McAuliffe M, Lalonde F, McGarry D, Gandler W, Csaky K, Trus B. Medical image processing, analysis and visualization in clinical research. Proc IEEE Intl Symp on Computer-Based Medical Systems CBMS. 2001:381–386.

5. Lucas, B.; Landman, B.; Prince, J.; Pham, D. Organization for Human Brain Mapping. Australia: Melbourne; 2008. MAPS: A Free Medical Image Processing Pipeline.
6. Lucas B, Bogovic J, Carass A, et al. The Java Image Science Toolkit (JIST) for rapid prototyping and publishing of neuroimaging software. Neuroinformatics. 2010; 8(1):5–17. [PubMed: 20077162]
7. Peiper S, Halle M, Kikinis R. 3D Slicer. Proc IEEE Intl Symp on Biomedical Imaging ISBI. 2004; 1:632–635.
8. Wolfson W. caBIG: seeking cancer cures by bits and bytes. Chem Biol. 2008; 15(6):521–522. [PubMed: 18559260]
9. von Eschenbach AC, Buetow K. Cancer Informatics Vision: caBIG. Cancer Inform. 2007; 2:22–24. [PubMed: 19458755]
10. Kakazu KK, Cheung LW, Lynne W. The Cancer Biomedical Informatics Grid (caBIG): pioneering an expansive network of information and tools for collaborative cancer research. Hawaii Med J. 2004; 63(9):273–275. [PubMed: 15540527]
11. Pieper S, Lorensen B, Schroeder W, Kikinis R. The NA-MIC Kit: ITK, VTK, Pipelines, Grids and 3D Slicer as An Open Platform for the Medical Image Computing Community. Proc IEEE Intl Symp on Biomedical Imaging ISBI. 2006:698–701.
12. Covington, K.; McCreedy, ES.; Chen, M., et al. Interfaces and Integration of Medical Image Analysis Frameworks: Challenges and Opportunities. Biomedical Science and Engineering Conference; Oak Ridge TN. 2010. p. 1-4.

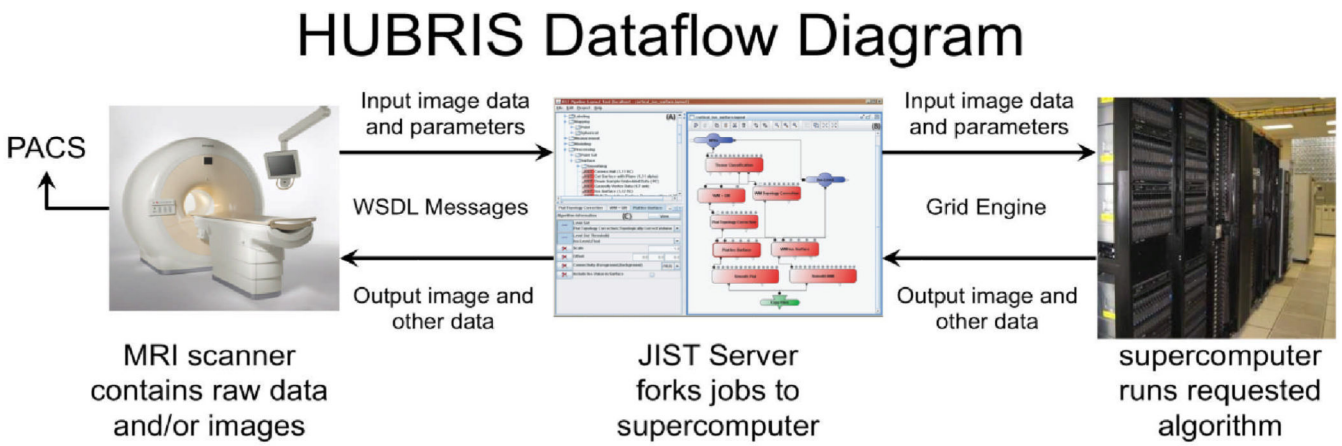

Figure 1.

HUBRIS dataflow diagram. The image acquisition device (left) automatically triggers a processing script with each scan event of a specific type. This script communicates with a JIST server (center) through a standard WSDL connection. In JIST stages the data and manages job submission to a parallel computing facility using the Sun Grid Engine (right).

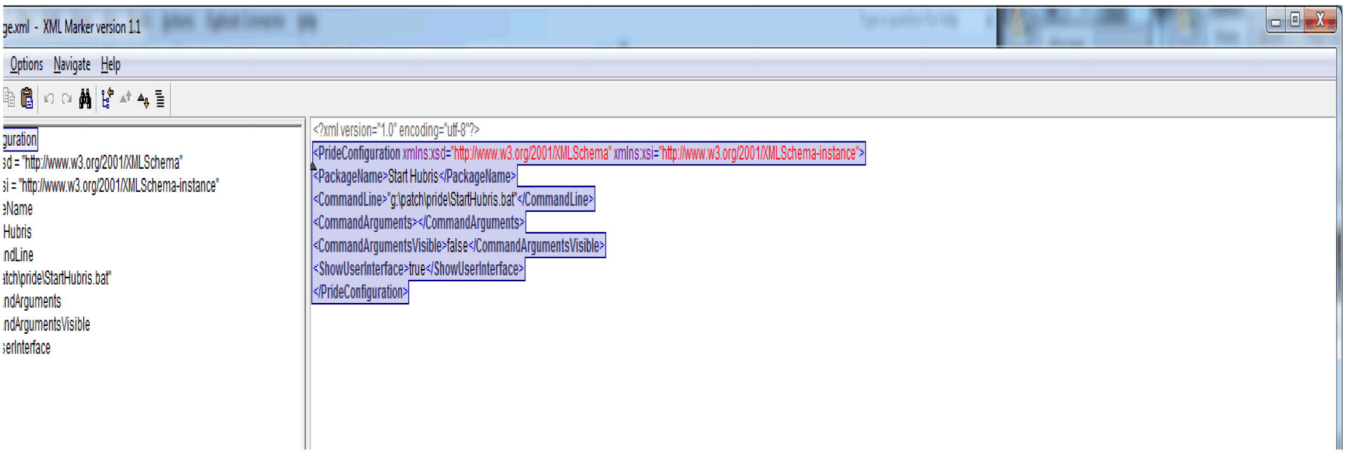

Figure 2.

Inline PRIDE interface configuration. Standard XML scripts define process events and allow for default parameters to be configured.

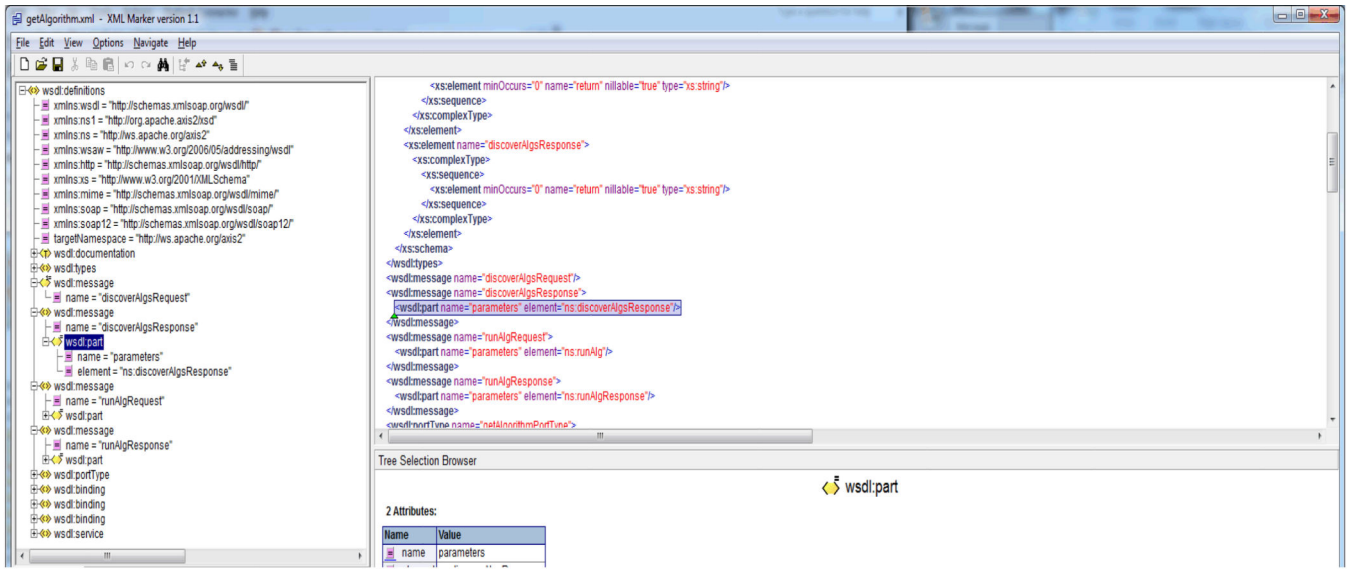

Figure 3.

The JIST WSDL server provides XML encapsulation of algorithm discovery, self-documentation, algorithm submission, and response.

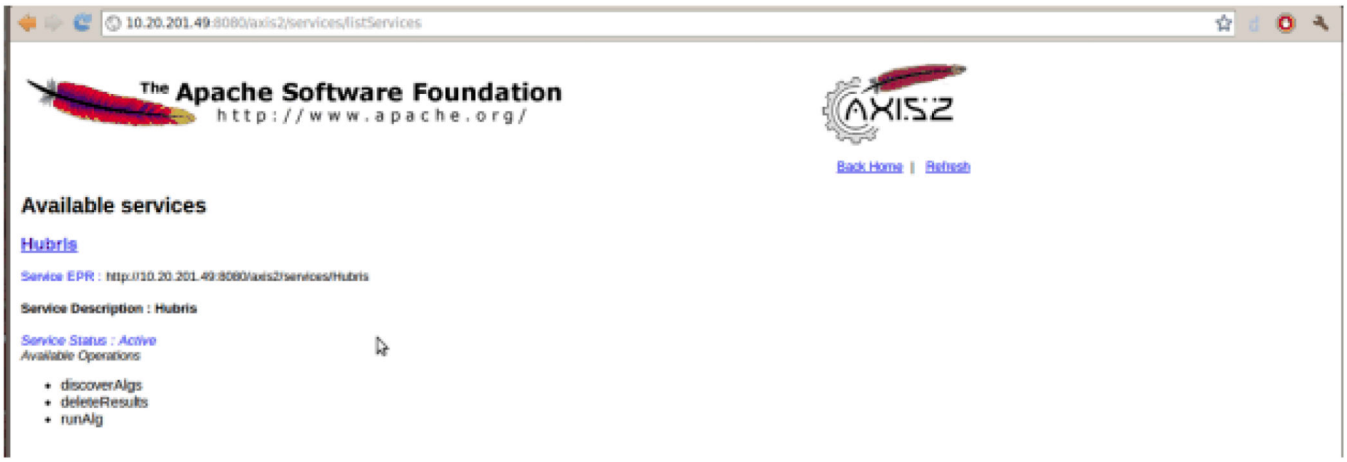

Figure 4.
The Axis 2 Server makes server processes available on the Internet and facilitates message- and data-passing between clients and the server.

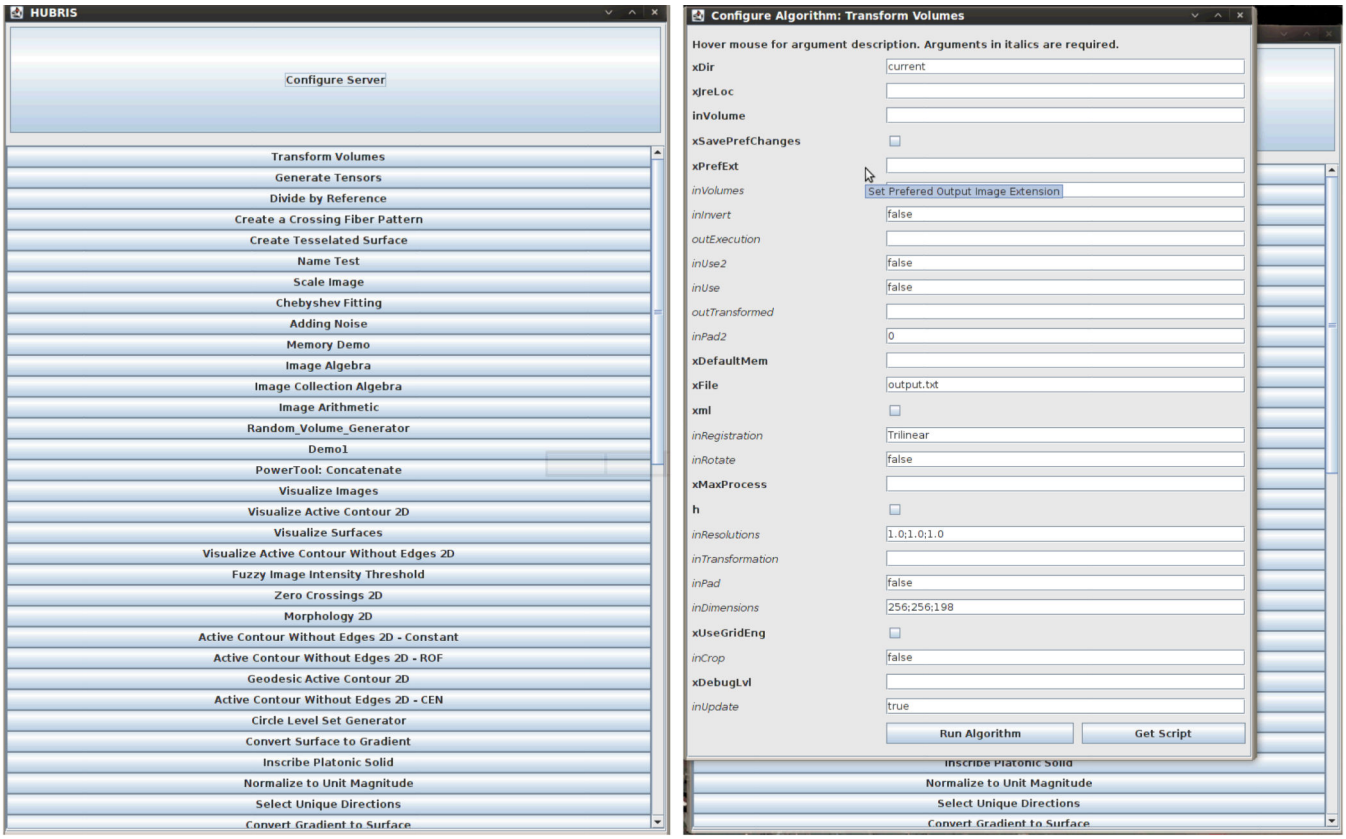

Figure 5.

Client-side Hubris GUI. Server configuration and a clickable listing of algorithms available for processing on the server are made available to the user (left). Arguments to individual algorithm are customized by the user and then either submitted for processing or the corresponding script is saved for future use (right).

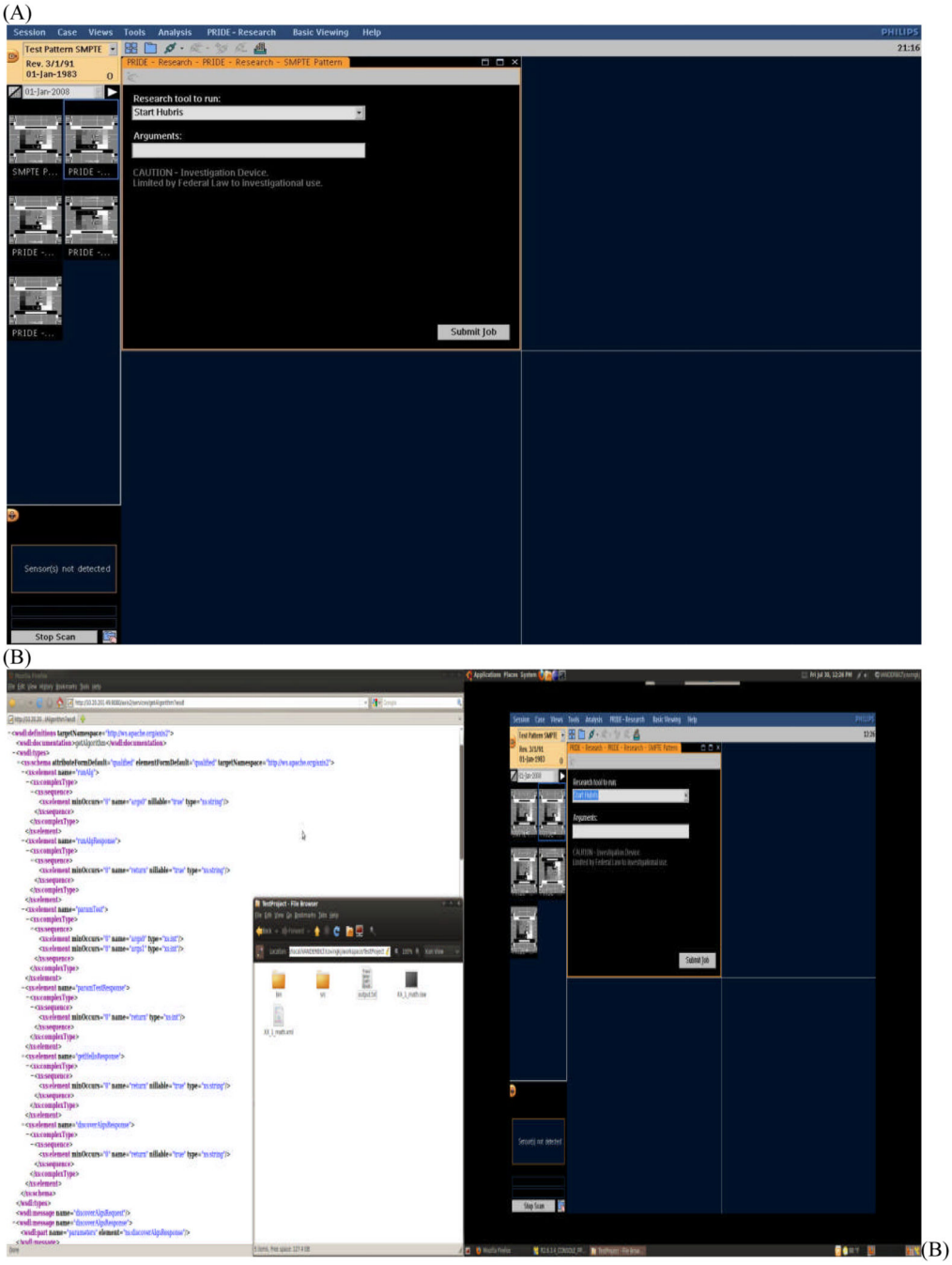

Figure 6.

Inline PRIDE interface on the Philips MRI scanner (top) communicates with a JIST server running on a different computer (bottom) to send, processes, and retrieve imaging data.

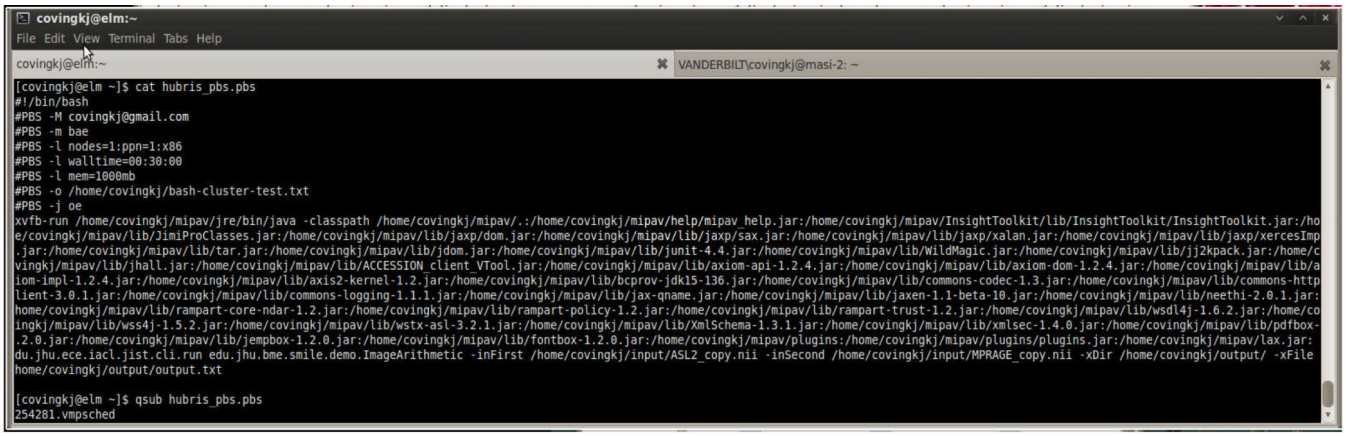

Figure 7.

PBS queuing file for an image addition operation. The results of running this file is returned to RunAlgorithm on exit.

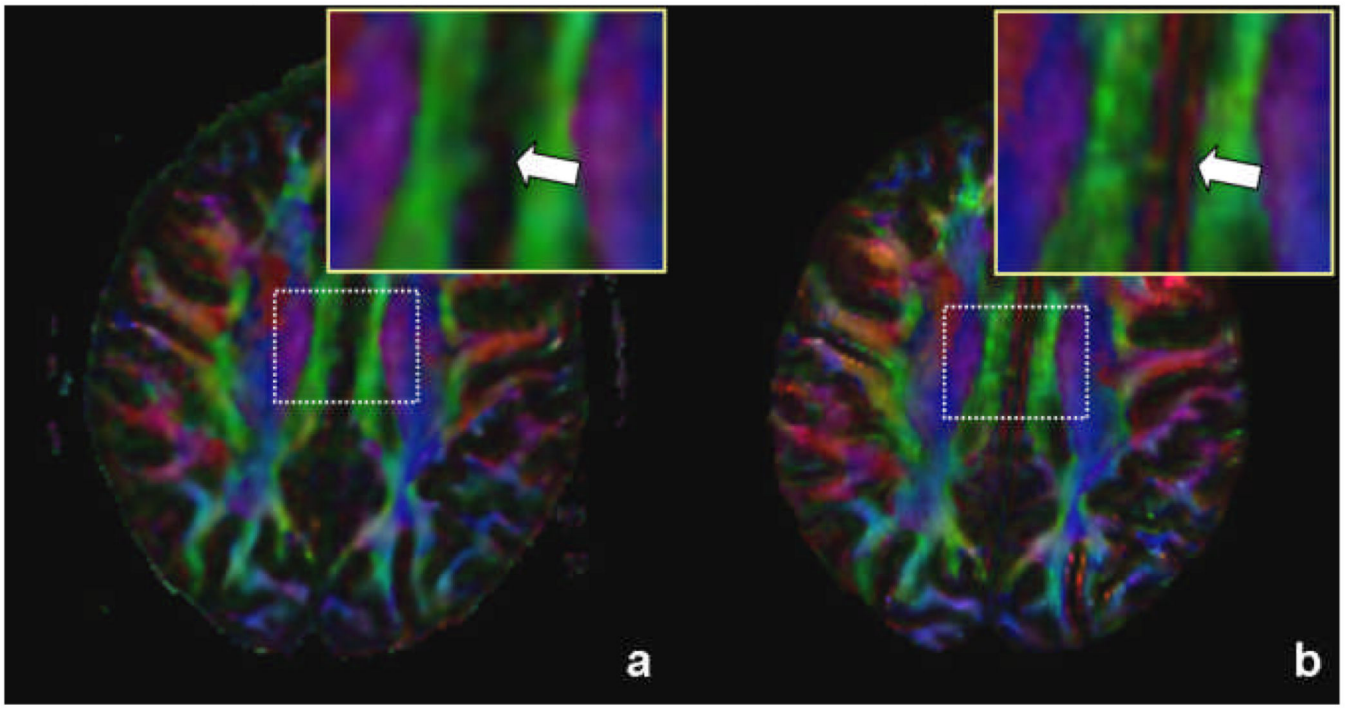

Figure 8.

Example potential application of HUBRIS for the reconstruction of improved diffusion tensor imaging (DTI) fractional anisotropy (FA) maps using a multi-shot acquisition. The presented color maps from single-shot (a) and multi-shot (b) echo-planar imaging (EPI) data show multi-shot EPI exhibits reduced blurring and geometric distortions. The generation of the improved images requires a custom reconstruction algorithm that can be accessed by HUBRIS.