A PLATFORM INDEPENDENT INFRASTRUCTURE FOR REAL-TIME MRS/MRI DATA STREAMING, PROCESSING AND STORAGE

Brian J. Soher[1]
[1]*Radiology, Duke University Medical Center, Durham, NC, United States*

Purpose: The dynamic flow of MR data from scanners to external computers during data acquisition is complicated by the hierarchical study/series organization of data acquisition and processing and by the variety of hardware, software and network infrastructures inherent to MR platforms. There are various applications where 'real-time' data access is desirable, such as: custom setup optimization, interactive regions of interest and physiologic monitoring during interventions. Most MR systems offer limited real-time capabilities with complex interfaces, complicating application prototyping. We introduce a platform/OS independent software package, RTView, that simplifies dynamic data transfer and demonstrate its use on a Siemens platform.

Architecture and Methods: RTView is an open source package that allows users to transfer, store, process and visualize MR data on an external computer during MR acquisition. It is written in Python using standard libraries making it hardware and OS independent and is easily installed on any platform. Data transfer uses the XML-RPC standard, a remote procedure call protocol that uses XML to encode data and HTTP as a transport mechanism. The key to our network communication is the well-documented Python standard library module SimpleXMLRPCServer, which enabled us to establish our backbone network data transfer connectivity with less than 30 lines of code.

The RTView architecture consists of an independent background process, 'rtview-listener', that runs an XML-RPC server on the external computer. This listens for data being transferred from an XML-RPC client on the MR scanner. On receipt of data it stores a time labeled copy of the raw data in XML format in a default directory and then broadcasts the data to a second XML-RPC server in the main RTView application framework. This is shown in Fig.1. The RTView application has a GUI interface structured as a tabbed notebook, where users can develop GUI/algorithms in tabs for specific processing/analysis functionality. Alternatively, users can write their own XML-RPC server code and instruct the listener to broadcast to it. Finally, to simplify installation, we use 'pyinstaller' to package RT View into a stand-alone executable under Windows/Linux/OS X.

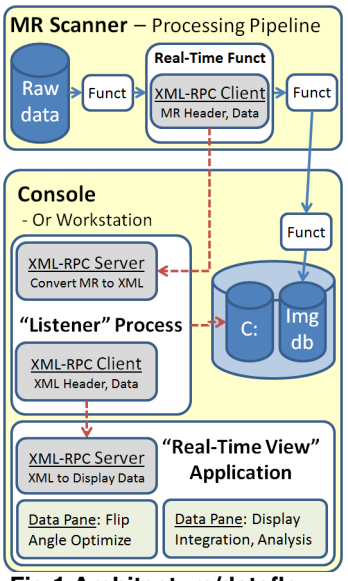

Fig.1 Architecture/dataflow

Results and Discussion: RTView platform was used to stream real-time multinuclear ^{31}P MR spectroscopy data off a 3T Trio (Siemens, Erlangen). The open source 'xmlrpc-c' library was used to create an XML-RPC client, compiled within the IcePrgSpectroscopy functor, that broadcast raw data and header info from within the ICE program on the Siemens MRIR computer. The xmlrpc-c library compiles under Visual C++ and Visual Studio making it compatible with most IDEA software versions. The client sends an initial XML message to the rtview-listener with pulse sequence name and acquisition parameters needed for processing. Subsequent XML messages to the rtview-listener bundle the MDH header and raw time data 'blob' for each scan processed by the ICE pipeline. The rtview-listener parses data blobs into XML and re-broadcasts them to the RTView application server.

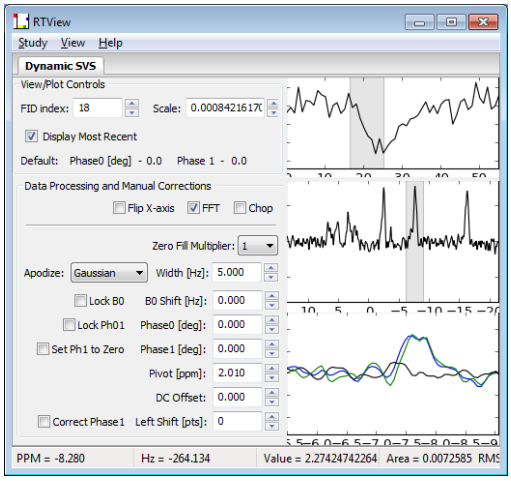

Fig.2 RTView GUI for dynamic ^{31}P phantom

We have so far created two analysis tabs for our research. The first accepts a series of data with a range of excitation flip angles from which the optimal power for a 'true 90' flip angle is derived. This is used for optimizing single voxel ^{1}H MRS acquisitions. A second analysis tab displays a series of MRS data as spectra plotted in time and allows the user to calculate areas under the curve to visualize dynamic changes in metabolite peaks. The RTView GUI is shown in Fig.2 for a phantom acquisition that simulates dynamic changes in ^{31}P ATP peak area changes upon delivery of a bolus of fructose to the liver.

The platform is easily reconfigured for MR image data by relocating the XML-RPC client functor within the appropriate ICE program to capture line scans, or slice or volume concatenations. Standard processing continues while a copy of the data is broadcast offline. Implementations on other MR scanners requires only the creation of client code for a given platform.

Acknowledgements: NIH funding - 1R01DK093568. Code available on request.