GR-MRI: A GnuRadio extension for MRI using software defined radios

Christopher J. Hasselwander[a,c], Zhipeng Cao[a,c], William A. Grissom[a,b,c]

[a] *Department of Biomedical Engineering, Vanderbilt University, Nashville, Tennessee, USA*
[b] *Department of Radiology and Radiological Sciences, Vanderbilt University, Nashville, Tennessee, USA*
[c] *Vanderbilt University Institute of Imaging Science, Nashville, Tennessee, USA*

Abstract

Hi.

Keywords: Software,

1. Introduction

Conventional commercial MR spectrometers are often limited in configurability, portability, scalability and cost. This has led several researchers to build lower-cost, smaller and more customizable architectures in-house [cite!]. One such approach is a software-defined radio (SDR) architecture [cite!], comprising high-speed A/D and D/A converters, an FPGA to perform high-bandwidth digital mixing and filtering, and a USB or Ethernet link to a PC for real-time data transfer. SDR?s have several strengths, including high-bandwidth direct RF signal synthesis and digitization with high bit depth, high configurability and low cost.

In this work we describe a SDR enabled MRI software package called gr-MRI that we paired with the commercially-available Ettus Research USRP1. The gr-MRI software package is offered as a free software package and aims to provide a cheap, flexible alternative to traditional MRI spectrometers. The package is built on top of GNU Radio, an open source and actively developed SDR interfacing signal processing software...

*Corresponding author address: Institute of Imaging Science, Vanderbilt University, 1161 21st Ave. South, MCN AA-1105, Nashville, TN 37232-2310

Email addresses: christopher.j.hasselwander@vanderbilt.edu (Christopher J. Hasselwander), zhipeng.cao@vanderbilt.edu (Zhipeng Cao), will.grissom@vanderbilt.edu (William A. Grissom)

2.2. Radio Synchronization

Due to the relatively long delays created by latency, radio synchronization can be an issue. The USRP1 uses a USB 3.0 interface that can have latency differences on the order of a few milliseconds, so it is important to synchronize the outputs of the two radios. To overcome this obstacle, the gr-MRI package uses a port on each radio to send a 10 sample, rectangle synchronization pulse every TR, which is looped back to the receive ports on the slaved USRP1. A function, called "sync()" `sync()` in any calibration or imaging sequence compares the delay between when each pulse is received and sets global master and slave delays to align the transmitted pulses.

2.3. GNU Radio (First) → start w/ higher level

GNU Radio uses an intuitive flow graph based interface that is used to set the TR, and map signals to the appropriate outputs and inputs. The flow graphs are manipulated using python, but lower level functions are written in C. GNU Radio was designed for continuous operation, so two custom blocks were created to enable sequencing, which are included in the gr-MRI package. The first is called Triggered Vector Source, and plays a specified waveform stored in a python vector whenever its input is high. There are settings in the block to control amplitude steps for the phase encoding gradient, and how many times to repeat each step to allow for averaging. The second custom block is called the Gated Vector Sink, which records the complex input to a python vector when a second input is high. Figure shows the GNU Radio flow graph for the gradient echo pulse sequence. The block on the far left is a signal generator that produces a square wave with period equal to the repetition time, or TR. That signal triggers the Triggered Vector Sources, each loaded with a waveform and settings that were defined in the gr-MRI script. The outputs from the Triggered Vector Sources are combined into complex signals to be sent out of specific ports of the USRP1s defined in the USRP Sink blocks on the far right. Delays before the USRP Sinks allow for synchronization of the two radios. Certain settings, such as the device serial addresses and sampling rates are defined in the GNU Radio flowgraphs.

The middle group of blocks defines the receive chain. The far left block is called the USRP Source and creates a stream of data received from the ADC. Because of the timing issues that USB latency can cause, the read window is a physical pulse that is sent out by the USRP1 and is looped back to a receive port. The pulse is sent at the same frequency as the Tx pulse. This pulse is used for two reasons. First, it defines the acquisition data that is sent back to the gr-MRI script. It is

2.5. Gradient Calibration

The gr-MRI package also has a built in automatic gradient calibration script which uses a spin echo sequence to define gradient strength in each direction. The user defines the direction to calibrate, and the object size in that dimension. The sequence then calculates the gradient strength based on standard gradients and the resultant bandwidth of the signal. Gradient strengths are saved in units of G/mm to a python pickle file named gcal.pkl and are used during imaging sequences to convert parameters of FOV and matrix size to gradient pulse output amplitude.

3. Hardware Overview

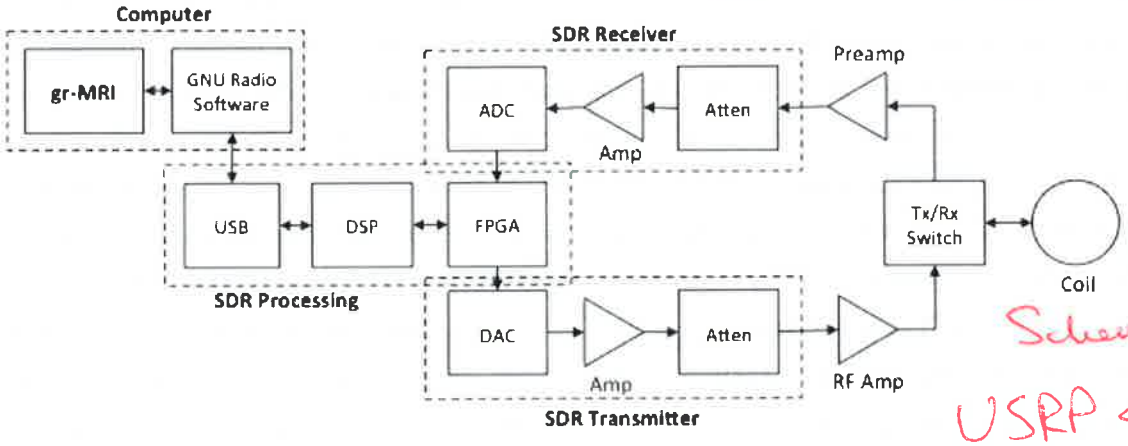

Figure 3: Flowgraph describing the signal pathway from gr-MRI to RF coil

The gr-MRI software package is compatible with any GNU Radio compatible software-defined radio, and a full list can be found at https://gnuradio.org/redmine/projects/gnuradio/wiki/Hardware. The work shown in this manuscript used two synced Ettus Research USRP1 radios. One radio (master) produces the RF signals and readout window, and the other (slave) produces the DC hardware control pulses and gradient pulses. Figure shows an overview of the signal pathway for the USRP1, however it is also the pathway for a general software-defined radio. The pulses and timing are defined in the gr-MRI software package and sent to GNU Radio, where the pulses are mapped to physical ports. The USRP1 has 4 transmit ports and 4 receive ports per motherboard. GNU Radio connects to the USRP1 via USB cable to download the pulse sequence to an Altera Cyclone FPGA

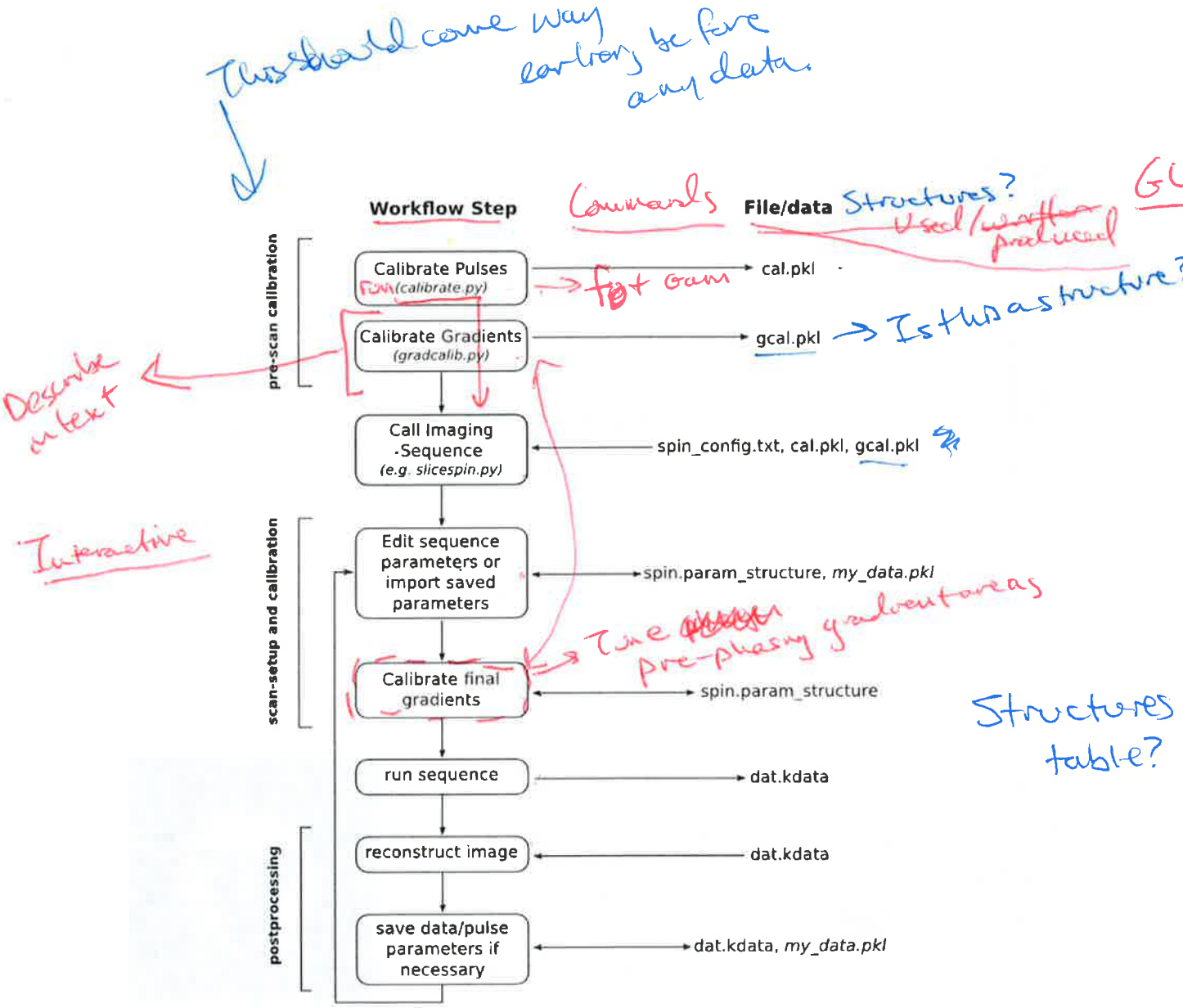

Figure 4: workflow caption

5. Results

All phantoms were a 15mm vial of copper sulfate doped water with a 9mm by 11mm 3-D printed Vanderbilt University Logo

Figure shows (A) The pulse sequence created by the stock gradient echo sequence, gradecho.py, and (B) the resulting image with 3 averages. The excitation pulse was a TBW 8 hamming windowed sinc pulse, played with a gradient in the z direction creating a 4mm slice. The echo time (TE) was

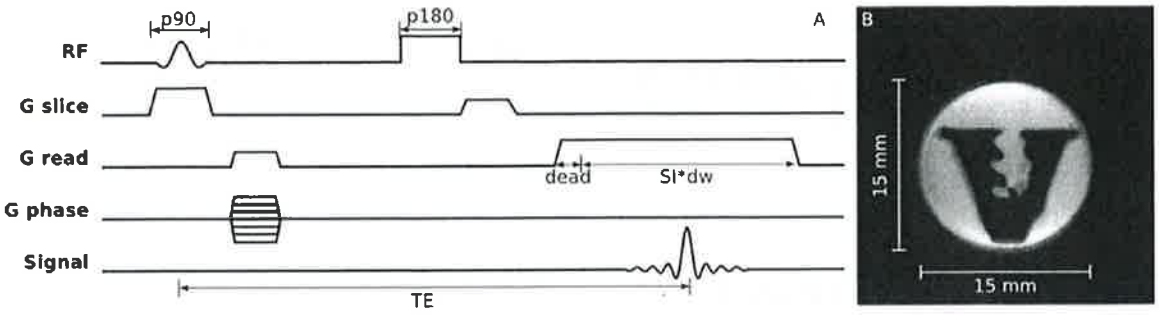

Figure 7: (A) Gradient echo pulse sequence created by gradecho.py in gr-MRI software package used to acquire (B)...

Would be good to show data for phase correction (i.e. phase variation + image w/ and w/o correction)

6. Discussion

7. Conclusion

We have introduced a flexible, powerful alternative to the traditional MRI spectrometer using software-defined radios, and the gr-MRI software package. It has shown the ability to acquire high resolution images using the three pre-built pulse sequences, and has the capacity to create more complex sequences and pulse shapes. As is, the software is compatible with any GNU Radio enabled software defined radio.

The full package, and instructions for use can be found at...

Acknowledgments

This work was supported by NIH R21 EB 018521.

References