Uniting Reconstruction Software for Native Use in GPI

Nicholas R. Zwart[1], Ashley G. Anderson III[1], Ryan K. Robison[1], Andrew Li[2], Mariya Doneva[2,3], Frank Ong[2], Martin Uecker[2], Michael Lustig[2], and James G. Pipe[1]

[1]Imaging Research, Barrow Neurological Institute, Phoenix, AZ, United States, [2]Electrical Engineering, University of California, Berkeley, CA, United States, [3]Philips Research, Hamburg, Germany

Synopsis

This work proposes the use of the software development platform called GPI (Graphical Programming Interface) as a tool for resourcing other work for integration and comparison. The GPI software structure is designed to facilitate the encapsulation of outside libraries and provides a plug-in model to isolate package dependencies. The library featured in this work is the Berkley Advanced Reconstruction Toolkit (BART) which provides, multi-platform compatible, compressed sensing and parallel imaging algorithms.

Introduction

The development of magnetic resonance imaging techniques employs algorithms from different disciplines that are implemented for the simulation, reconstruction and acquisition of MR data. In order to continue to improve and refine these techniques it is often necessary to compare and integrate multiple techniques on a single platform. Reproducing these techniques for comparison can be hindered by implementing ideas from theory or incorporating another's code. This work proposes the use of the software development platform called GPI (Graphical Programming Interface) [1] as a tool for resourcing other work for integration and comparison. GPI has been shown to be a versatile prototyping platform for developing a wide range of reconstruction techniques [2–7]. The GPI software structure is designed to facilitate the encapsulation of outside libraries and provide a plug-in model to isolate package dependencies. GPI also provides an environment for visualizing the algorithm flow and inspecting data, which are beneficial to communicating reconstruction algorithms. The library featured in this encapsulation model is the Berkley Advanced Reconstruction Toolkit (BART) which provides multi-platform compatible compressed sensing and parallel imaging algorithms [8].

Methods

The BART provides a group of C libraries that can be compiled into a suite of command-line utilities. For the purposes of encapsulating the BART functionality, a simple communication interface was developed to reduce the code overhead for transferring data between GPI and the BART command-line interface. The code for this example can be found at **https://github.com/nckz/bart/tree/master/gpi**.

Figure 1 shows a code snippet with the interface of a BART node that produces trajectory coordinates. The code imports three classes that assist in wrapping executables: *IFilePath*, *OFilePath* and *Command*. The two file-path classes manage the creation and deletion of temporary files and the file format required by the executable. The *Command* class executes the final command-line string as a separate process. The BART node library was used to implement example ESPIRiT reconstructions (**http://mikgroup.github.io/bart/examples.html**), demonstrate the ease of integration with existing spiral reconstructions, and show the step by step flow of a quintessential compressed sensing reconstruction.

Results & Discussion

Several example GPI networks use the encapsulation method to show GPI as a scripting analog, as an interface for merging algorithms (old and new), and as a teaching tool to facilitate the exchange of complex reconstructions.

ESPIRiT Examples

The encapsulation tools, allowed the BART to be wrapped into a GPI node library in short order. This in-turn enabled the development of a GUI for each of the BART functions. An example GUI for the *Traj* node is shown in figure 2. Figure 3 shows the GPI representation of the ESPIRiT examples. The core GPI nodes were able to replace some of the data manipulation methods provided by the BART, which allowed the assembly of the example networks with key BART modules.

Spiral Examples

The BART nodes were used in conjunction with the spiral design [9] and gridding reconstruction methods to produce coil sensitivity maps via the ESPIRiT technique. Figure 4 shows the resulting reconstruction for a simulated data set (using the BART simulation tool) for the gridding reconstructed coil maps. These maps were then used in the parallel imaging and

compressed sensing (PICS) reconstruction from the BART. Some minor data manipulations such as transposing and scaling were necessary to correctly communicate data between the BART and core GPI nodes, which was handled with existing nodes.

Compressed Sensing Guide

Figure 5 shows a step by step breakdown of an iterative compressed sensing reconstruction. The data are transformed into a sparse domain, a threshold is applied, and then the data are transformed back into k-space where consistency with the sampled data is enforced. The graphical representation provides an instructional view of the algorithm and allows the user to easily probe result at any point.

Conclusion

The examples presented in this work show how GPI can be used to encapsulate code from other libraries. In the process of developing these tools, the authors have been able to communicate algorithms and full reconstructions using GPI as a common platform. The binary encapsulated BART nodes can be used as an initial step towards generating a python interface for the BART source code (for more efficient native-GPI use of BART in the future). This approach provides a simple testing ground for the python interface development and can reuse the existing GUI that has been generated for each BART module. The GPI encapsulated node library makes use of the BART's multi-platform capabilities and can be used as a 'BART plug-in' to the GPI environment.

Acknowledgements

No acknowledgement found.

References

[1] NR Zwart and JG Pipe. "Graphical programming interface: a development environment for MRI methods". In: Magnetic Resonance in Medicine (2014). url: **http://gpilab.com**.

[2] JG Pipe et al. "Revised motion estimation algorithm for PROPELLER MRI". In: Magnetic Resonance in Medicine 72.2 (2014), pp. 430– 437.

[3] Y Chang et al. "The effects of SENSE on PROPELLER imaging". In: Magnetic Resonance in Medicine (2014).

[4] D Wang et al. "Analytical three-point Dixon method: With applications for spiral water–fat imaging". In: Magnetic Resonance in Medicine (2015).

[5] Z Li et al. "Arterial spin labeled perfusion imaging using three-dimensional turbo spin echo with a distributed spiral-in/out trajectory". In: Magnetic Resonance in Medicine (2015).

[6] M Schar et al. "Dixon water-fat separation in PROPELLER MRI acquired with two interleaved echoes". In: Magnetic Resonance in Medicine (2015).

[7] M Schar et al. "Two repetition time saturation transfer (TwiST) with spill-over correction to measure creatine kinase reaction rates in human hearts". In: Journal of Cardiovascular Magnetic Resonance 17.1 (2015), pp. 1–11.

[8] M Uecker et al. "Berkeley Advanced Reconstruction Toolbox". In: Proceedings of the Joint Annual Meeting of ISMRM-ESMRMB, Toronto, Canada. 2015. url: **https://github.com/mikgroup/bart**.

[9] JG Pipe and NR Zwart. "Spiral trajectory design: A flexible numerical algorithm and base analytical equations". In: Magnetic Resonance in Medicine 71.1 (2014), pp. 278–285.

Figures

Figure 1: A code snippet from the GPI-node *Traj* which wraps the BART command-line utility *traj*. This demonstrates the interface required to read data in from an external command-line utility.

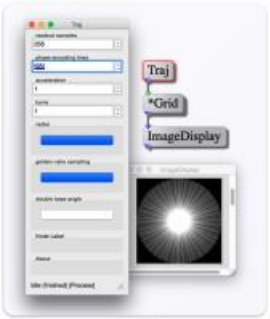

Figure 2: The *Traj* node menu and gridded coordinate display.

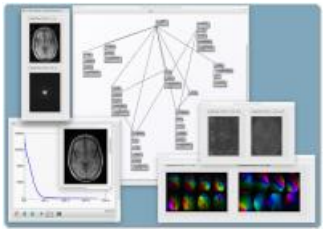

Figure 3: Examples of the ESPIRiT reconstruction in comparison with other BART-implemented techniques. This example mirrors the examples from **http: //mikgroup.github.io/bart/examples.html**.

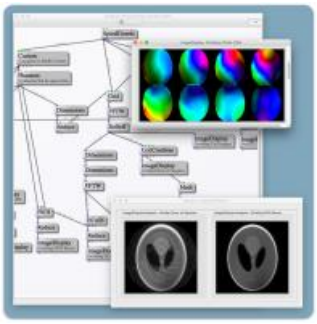

Figure 4: A GPI network using BART nodes to perform Parallel Imaging and Compressed Sensing (PICS) on simulated spiral data that is reconstructed using gridding and nuFFT. The images shown are gridding without (left) and with (right) PICS reconstruction.

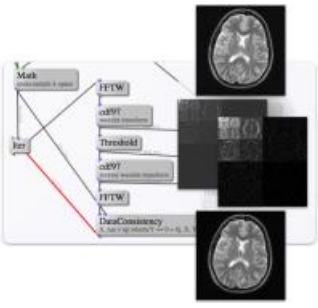

Figure 5: A simplified compressed sensing example network showing the basic iteration loop.