Interventional Device Tracking and Imaging Using RTHawk, an Extensible Real-Time System

Ethan Brodsky, Orhan Unal, and Walter Block

Radiology and Medical Physics, University of Wisconsin, Madison, WI, USA

Introduction

We have previously demonstrated using an interleaved 3DPR acquisition to simultaneously track an endovascular interventional device and image the surrounding tissue (Fig. 1), using a real-time system that involved a custom-written pulse sequence program, reconstruction program, and visualization [1]. This system was complicated, specialized to that particular acquisition sequence, and difficult to modify for new applications, making it undesirable for use as a base for a comprehensive interventional system.

RTHawk (HeartVista, Inc., Palo Alto, CA) is an extensible software architecture that permits a variety of pulse sequences, acquisition trajectories, and reconstruction techniques to be easily developed and interleaved at run-time [2]. It consists of a stub pulse sequence program that can carry out a variety of different acquisitions using RF and gradient waveforms specified in MATLAB. Interleaving of multiple sequences is coordinated via bidirectional communication between the scanner host and user client. The reconstruction, user interface, and visualization are written in JavaScript and can use both predefined and user-developed blocks (implemented as Qt objects in C++). Reconstruction algorithms are similarly described as a pipeline of processing blocks and typically require only a small amount of simple programming to achieve real-time operation.

We describe here a reimplementation of our earlier simultaneous tracking and imaging system within the RTHawk framework.

Materials and Methods

We have reimplemented our previous 3DPR-based tracking and imaging algorithm in RTHawk, as well as a simple tracking and imaging sequence that interleaves three orthogonal projections with a conventional 2D Cartesian acquisition that follows the device tip. Both these techniques are implemented using a single PSD and reconstruction system, with the only changes being in the specified readout trajectory, reconstruction pipeline, and custom processing blocks (Fig. 2).

Results and Discussion

A generic pulse sequence included in RTHawk implements a variety of contrast mechanisms and allows the use of arbitrary readout gradients generated using a MATLAB script.

Reconstruction is implemented using a scripting language to define a data processing pipeline using a combination of predefined and user-developed blocks, a modular approach that greatly simplifies adapting the system to new applications.

Initial results suggest that interventional sequences can easily adapted to work within this framework. RTHawk abstracts away much of the complexity of pulse sequence and reconstruction programming, allowing simple intuitive representations of the acquisition trajectory and reconstruction pipeline. It becomes trivial to interleave sequences, and visualization of final and intermediate results is also greatly simplified.

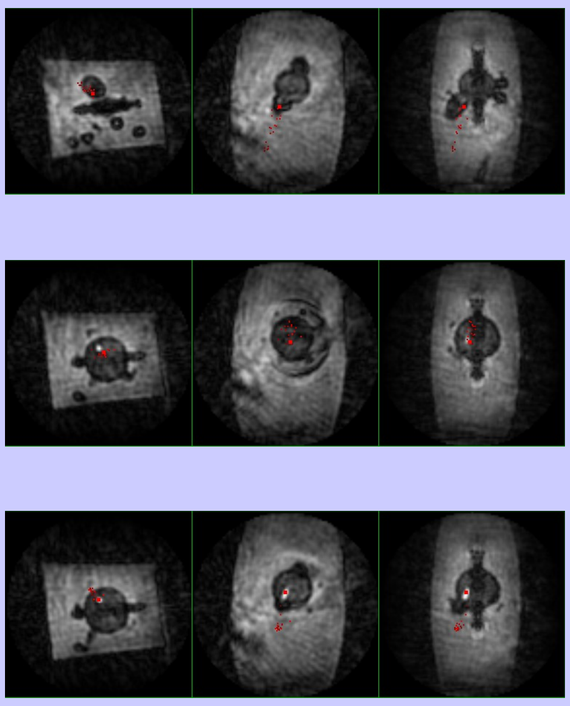
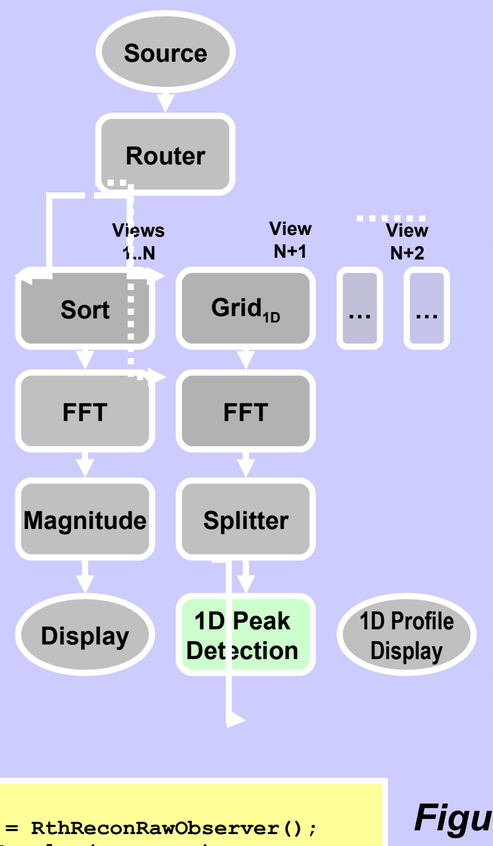


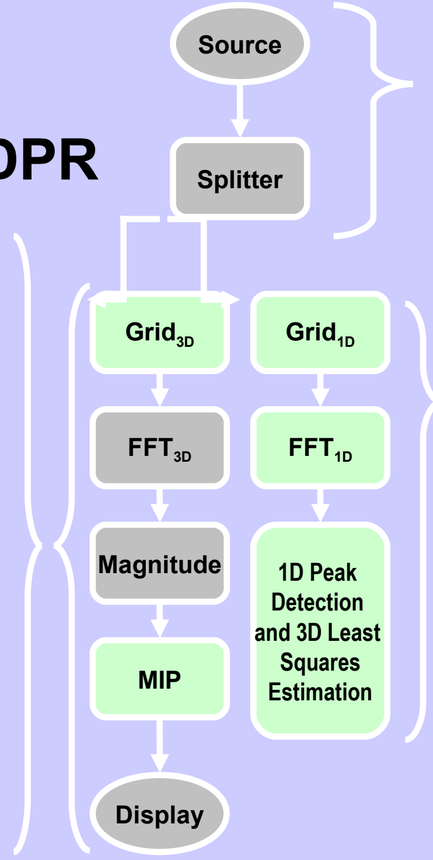
***Figure 1**: The previous real-time system permitted simultaneous tracking and imaging. Here, an endovascular device is moved through a flow phantom. The 3D volume is continuously acquired (using an external receiver coil) and updated every 4 s and the tip position is updated (using signal from the tip tracking coil [3]) every 1/8 s. Slices aligned with the device tip along each of the orthogonal axes are automatically selected and displayed. Small injections of dilute MR contrast made through the device are visible in the second and third images.*

Interleaved Tracking and 2D Cartesian Imaging




```
// Gridding
var sort_n = RthReconRawToImageSort();
sort_n.setPhaseEncodes(1);
sort_n.setSamples(256);
sort_n.setXSize(256);
sort_n.setYSize(1);
sort_n.setInput(router.output(n));

// FFT
var fft_n = RthReconImageFFT();
fft_n.setInput(sort_n.output());

// magnitude
var absImage_n = RthReconImageAbs();
absImage_n.setInput(fft_n.output());

var split_n = RthReconImageSplitter();
Split_n.setInput(absImage_n.output(0));

qt.script.importExtension("rth.peakFinder");
var peakFinder_n = RthReconImagePeakFinder();
peakFinder_n.setAxis(n);
peakFinder_n.setInput(split_n.output(0));

var plot_n = RthReconImagePlot();
plot_n.setInput(split_n.output(1));
```

Simultaneous Imaging and Tracking using 3DPR




```
// source
var observer = RthReconRawObserver();
observer.setSamples(acq_xres);
observer.setSequenceId(0);
observer.setInitialView(1);
observer.setFinalView(subframe_nproj);

// split data for imaging and tracking
var splitter = RthReconImageSplitter();
splitter.setInput(observer.output(0));
```

```
// Gridding
var grid3D = RthReconRawToImageGrid3D();
grid3D.setKSpace(ksp_file, grid_size, res);
grid3D.setInput(splitter.output(1));

// FFT
var fft = RthReconImageFFT();
fft.setInput(grid3D.output());

// magnitude
var magImage = RthReconImageAbs();
magImage.setInput(fft.output());

// MIP
qt.script.importExtension("rth.MIP");
var mipImage = RthReconImageMIP();
mipImage.setInput(magImage.output());
mipImage.setAxis(3); // z

// display
var image = RthReconImageToRthDisplayImage();
image.setInput(mipImage.output());
image.newImage.connect(RTHawk.newImage);
```

```
qt.script.importExtension("rth.gridX");
var gridX = RthReconRawToImageGridX();
trackX.setIinput(splitter.output(0));

qt.script.importExtension("rth.fftX");
var fftX = RthReconImageFFTX();
fftX.setInput(trackX.output());

qt.script.importExtension("rth.track3DPR");
var track3DPR = RthReconImageTrack3DPR();
track3DPR.setKSpace(ksp_file);
track3DPR.setInput(fftX.output());
```

***Figure 2**: The reconstruction process is specified using a JavaScript-like description of a pipeline that can use a combination of predefined blocks (gray) and user-defined blocks (green). Sophisticated processing of large data sets can be implemented with very little programming effort – a previous real-time system implementing the same algorithms with conventional techniques required extensive PSD programming and several thousand lines of custom reconstruction code.*

References

[1] Brodsky *et al*., MRM 56:247 (’06)
[2] Santos *et al*., Proc IEEE Eng Med Biol 2:1048 (’04)
[3] Unal *et al*. Proc MRA Club 18:4.2 (’06)

This work was supported by NCI R01-CA116380-04 and ARRA MSN128450.