TRON: Faster Non-Cartesian MRI Reconstruction through Prior Knowledge and Parallel Architectures

David S. Smith, Saikat Sengupta, and E. Brian Welch
Institute of Imaging Science, Vanderbilt University, Nashville, TN, USA

Purpose. In non-Cartesian MRI reconstruction, the acquired unequally spaced data are usually interpolated onto a Cartesian grid so that a fast Fourier transform can be used.[1,2] Interpolation is performed by scanning the unequal data, finding the nearest neighbors on the Cartesian grid, and adding the data with appropriate weights onto the Cartesian grid. This approach suffers from two problems:

(1) **Excess memory transfers**: Shuffling coordinates in addition to data around in memory roughly doubles the problem size. This is detrimental because data transfers are typically the computation bottleneck on high-end hardware.
(2) **Write conflicts**: When parallel threads try to write to the same point on the Cartesian grid at the same time, the writes must be serialized to prevent data corruption, reducing parallelism by $O(w^d)$, where w is the kernel width, and $d = 2$ or 3 for 2D or 3D.

Here we present **TRON** (for **TR**ajectory **O**ptimized **N**UFFT), a proof-of-concept reconstruction specialized for 2D radial MRI data that requires only the data (without coordinates), has no write conflicts, and is more than an order of magnitude faster than the best existing packages. As a bonus, separate computation of sample density compensation is not required.

Methods. TRON assumes a standard set of uncorrected radial coordinates for the data and grids from the uniform data perspective, i.e. one compute thread per uniform point. Reversing the gridding perspective is not new,[3] but directly computing (rather than searching) the nearest unequal points is. This is possible only because we know the data are radial. Algorithm G describes the gridding method in TRON.

To evaluate TRON, we reconstructed whole-body gradient echo data acquired on a Philips Achieva 3.0 T with a continuous golden angle radial trajectory. The radial acquisition dimensions were 8 channels $\times$ 512 radial samples $\times$ 20,271 spokes, reconstructed onto an image volume of $1024 \times 1024 \times 224$, and then cropped to $384 \times 384 \times 224$. Channels were combined with sum of squares.

Algorithm G (Gridding). This algorithm takes 2D radial data and interpolates it onto a Cartesian grid. Each *Cartesian* grid point is handled by a separate GPU thread.

G1. [Assign thread location.] Map GPU thread index to a unique (k_x, k_y) coordinate pair on the uniform grid.
G2. [Find contributing non-uniform points.] For each radial spoke, calculate radial band of twice the kernel width around (k_x, k_y). For golden angle, search angles for points within kernel radius within this band. For linear data, compute both angle and radius.
G3. [Interpolate.] For each contributing unequal point, compute the associated kernel weight. Add the weighted non-uniform datum to the Cartesian location. Keep a sum of the weights.
G4. [Sample density compensation.] Divide uniform datum by sum of weights, perfectly compensating for the sampling density. ▮

Results. We compared TRON to two ubiquitous non-uniform FFT packages in the MRI community: the image reconstruction toolbox[1] (IRT) and gpuNUFFT.[3] As far as we were aware at the time of writing, gpuNUFFT was the fastest open source NUFFT code available for MRI. The code timings were performed on a dual 8-core Intel Xeon E5-2665 workstation with dual Nvidia GeForce GTX TITAN X GPUs using CUDA 7.0.27 and MATLAB R2015b. An additional timing was run on our code using a Macbook Pro 11,3 (mid 2014) with an Nvidia GeForce GT 750M and CUDA 7.5.21. Only image production was timed, not file I/O, and all timings are best of three. Fig. 1 shows the timings, while Fig. 2 shows example reconstructions. For a fair comparison in Fig. 1, all codes used a 2$\times$ oversampling factor and a Kaiser-Bessel kernel of width 2.

As an aside on code complexity, TRON is only 907 lines long, while the minimal subset of IRT used here comprised 1838 lines, and gpuNUFFT included over 16,000 lines.

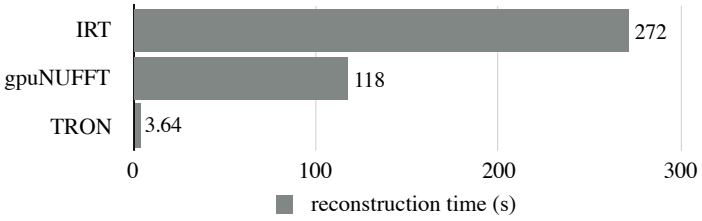

Figure 1—Time to reconstruct 224 slices and 8 channels on a 2$\times$ oversampled 1024 $\times$ 1024 grid with kernel width 2. TRON was over 30$\times$ faster than the nearest competitor.

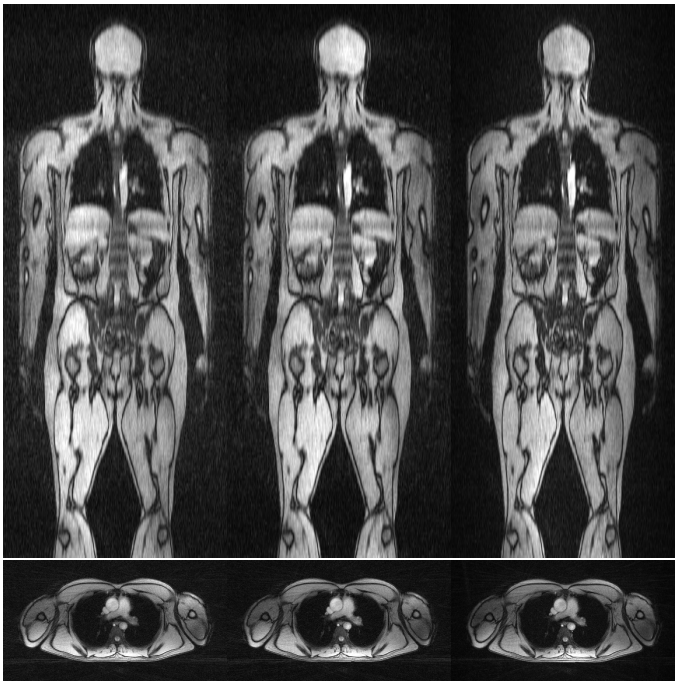

Figure 2—Example whole-body reconstructions: IRT (left), gpuNUFFT (middle), and TRON (right). In the rightmost image, we actually used a simple Gaussian kernel and no oversampling for an additional 2$\times$ speed boost and still produced a cleaner image. The image on the right was produced 74$\times$ faster than the image in the middle from gpuNUFFT.

Discussion. Our attempt to reduce memory transfers and eliminate write conflicts dramatically outperformed existing methods. The downside of this increased speed is decreased flexibility—TRON currently can handle 2D radial only. It is straightforward to adapt TRON to any trajectory, but some input parameters (number of channels, most notably) are hard-coded to enable parallel adaptive channel combination[5] at only a $\sim 2\times$ speed penalty, so separate executables would be required.

Conclusions. Customizing NUFFT algorithms for specific patterns of data can yield much higher performance with simpler code. TRON can be downloaded at `http://github.com/davidssmith/tron`.

References. [1] Fessler & Sutton, IEEE TSP, 2003, 51(2), 560. [2] Jackson et al. 1991, IEEE TMI, 10, 473. [3] Yang et al. 2013, MRI, 31, 313. [4] Knoll et al. 2014, Proc ISMRM 4297. [5] Walsh et al. 2000, MRM, 43, 682.

Acknowledgements. NCI K25 CA176219, NCATS UL1 TR000445