4675

MRI-NUFFT: An open source Python package to make non-Cartesian MR Imaging easier

Pierre-Antoine Comby[1], Guillaume Daval-Frérot[2], Chaithya GR[3], Alexandre Vignaud[3], and Philippe Ciuciu[3,4]
[1]CEA/Neurospin, Gif-sur-Yvette, France, [2]Chipiron, Paris, France, [3]CEA/Neurospin, Gif-Sur-Yvette, France, [4]Inria/MIND, Gif-sur-Yvette, France

Synopsis

Keywords: Software Tools, Software Tools, NUFFT, Non-Cartesian

Motivation: Non-Cartesian imaging remains complicated to use for MRI due to the high computational cost of the Non-Uniform Fourier Transform for image reconstruction.

Goal(s): To provide a uniform interface for reconstructing magnetic resonance images from non-Cartesian k-space data and a collection of non-Cartesian sampling trajectories

Approach: We propose an open-source Python package (https://github.com/mind-inria/mri-nufft/) providing a standard interface to existing NUFFT libraries, with extended models for multi-coil imaging and static-field (B0) inhomogeneities correction.

Results: MRI-NUFFT can generate sampling trajectories, compliant with hardware constraints, as well as simple forward/adjoint operations and density compensation for use in advanced image reconstruction scenarios.

Impact: With MRI-NUFFT, non-Cartesian MRI trajectories and reconstruction algorithms become accessible, efficient, and affordable to everyone for research and education purposes.

Introduction

Non-Cartesian imaging techniques are not yet widely used in Magnetic Resonance Imaging (MRI) even though their high acceleration capabilities in k-space data acquisition make it possible to reduce scan time significantly[1,2]. Additionally, non-Cartesian k-space trajectories are prone to off-resonance effects due to B0 field inhomogeneities[3].

Further, the adoption of non-Cartesian imaging has been hindered by the computational complexity of the Non-Uniform Fast Fourier Transform (NUFFT) and the iterative procedure involved in Compressed Sensing (CS) techniques[4].

To address this challenge, we developed MRI-NUFFT, a Python package that simplifies non-Cartesian image reconstruction techniques in MRI. This package provides a uniform interface for performing the NUFFT operation on non-Cartesian MRI data including correction for static off-resonance effects. Additionally, it offers a comprehensive collection of 2D and 3D non-Cartesian k-space trajectories.

Material and Methods

Collection of non-Cartesian Trajectories

Built on our previous work [6], we provide non-Cartesian trajectory generation, inspection, and display tools for both 2D (Figure 2) and 3D (Figure 3). The generated trajectories can then be discretized and input in specific FLASH GRE sequence

Capabilities of MRI-NUFFT

Multiple NUFFT libraries are available in the community using CPU or GPU computing. MRI-NUFFT supports a majority of them, as reported in Figure 1. However, only a few natively support the extended Fourier model of MRI acquisition: parallel imaging with coil sensitivities S_ℓ)[7] and static-field inhomogeneity modeling ($\tilde{\mathcal{F}}_\Omega$)[8].

$$y_{i,\ell} = \int_{\mathbb{R}^d} S_\ell(\boldsymbol{u}) x(\boldsymbol{u}) e^{-2i\pi \boldsymbol{u}\cdot\boldsymbol{\nu_i} + \Delta\omega(\boldsymbol{u})t_i} d\boldsymbol{u} + n_{i,\ell}$$

which is modeled as the following linear system:

$$\tilde{\boldsymbol{y}} = \begin{bmatrix} \tilde{\mathcal{F}}_\Omega S_1 \\ \vdots \\ \tilde{\mathcal{F}}_\Omega S_L \end{bmatrix} \boldsymbol{x} + \boldsymbol{n} = \tilde{\boldsymbol{\mathcal{F}}}_\Omega \boldsymbol{S} \otimes \boldsymbol{x} + \boldsymbol{n}$$

A weighted sum of Non-Uniform FFT approximates the forward operator. The adjoint/Type-1 NUFFT consists of spreading non-uniform points to an upsampled k-space grid and using the IFFT and apodization correction to get an image. The forward/Type-2 performs these steps in reverse order [10,11]

MRI-NUFFT provides the user with a unifying interface to do those operations and optimizes the call to raw NUFFT operators whenever possible. Supported features include:

- 2D and 3D imaging
- Single and multi-coil, with or without sensitivity map estimation
- Extra batch dimension (for multi-echo or dynamic imaging)
- Optimized forward (image to k-space), adjoint (k-space to image), and gradient descent steps.
- Operational density compensation mechanism using an iterative scheme[9] or an estimate based on the Voronoi diagram.
- Modeling and correction of static-field inhomogeneities.

On top of the various backends, we also provide a 2.5D-NUFFT implementation using the FFT over the last dimension for stacked 2D non-Cartesian sampling patterns.

MRI-NUFFT is also compatible with the PySAP-MRI[10] reconstruction toolbox to perform CS-based image reconstruction.

Benchmarking NUFFT for MRI

The unified interfaces for the NUFFT and the trajectory generation tools make real-use-case benchmarks possible. We compared the forward, adjoint, and data-consistency operation regarding computation time and memory on a large-scale multi-coil 3D-MRI acquisition of matrix size 384x384x208 at 0.6mm isotropic resolution. Script for running the benchmark is available in the MRI-NUFFT repository (https://github.com/mind-inria/mri-nufft/)

Raw Computation performances were assessed for the 4 most used NUFFT libraries, parametrized with an aliasing error of 1e-3 and grid oversampling factor 2 (Figure 4), used in a single or multi-coil setup with sensitivities maps. We considered the complete runtime for each step (with copies to/from GPU or filesystem (for BART)) and the peak memory usage.

Image quality comparisons were run over four in-out trajectories at an acceleration factor of 20 (compared to 3D-Cartesian) or approximately 3,994 shots of 10,240 samples (dwell-time 2µs). Parameters were chosen to maximize coverage under gradient and slew rate constraints (40mT/m and 110T/m/ms, respectively): Stack-of-spirals (208 stacks of 19 shots, 7 revolutions), FLORET[12] (6 revolutions), Seiffert spirals[13] (6 revolutions).

Results

We report time performance in Figure 4. BART is outperformed by its competitors, due to the need for dumping results to the filesystem. GpuNUFFT benefits from asynchronous memory transfers (notably in multi-coil settings), but its API requires data on CPU

Memory. Cufinufft can work with on-device data, and we leverage this aspect in the gradient computation.

For Image quality comparison, we showcase the impact of trajectory choice on CS reconstruction[10]. We used gpuNUFFT and compared the trajectories using SSIM. The upsampling factor and precision (ε) parameters have a limited impact on the image quality compared to the trajectory choice but are relevant for the total runtime and memory footprint of the algorithm.

Conclusion

With its user-friendly and standardized interface, a wide choice of k-space trajectories, and the actual support for various NUFFT implementations, MRI-NUFFT is a valuable tool for researchers and practitioners seeking to explore and implement non-Cartesian imaging techniques in MRI. The package simplifies the reconstruction process and promotes the development of novel non-Cartesian acquisition schemes, potentially leading to improved image quality, reduced scan times, and enhanced clinical applications.

Acknowledgements

No acknowledgement found.

References

[1] K. L. Wright, J. I. Hamilton, M. A. Griswold, V. Gulani, and N. Seiberlich, “Non-Cartesian parallel imaging reconstruction,” J. Magn. Reson. Imaging, vol. 40, no. 5, pp. 1022–1040, 2014, doi: 10.1002/jmri.24521.

[2] G. R. Chaithya, P. Weiss, G. Daval-Frérot, A. Massire, A. Vignaud, and P. Ciuciu, “Optimizing Full 3D SPARKLING Trajectories for High-Resolution Magnetic Resonance Imaging,” IEEE Trans. Med. Imaging, vol. 41, no. 8, pp. 2105–2117, Aug. 2022, doi: 10.1109/TMI.2022.3157269.

[3] Z. Amor et al., “B0 field distortions monitoring and correction for 3D non-Cartesian fMRI acquisitions using a field camera: Application to 3D-SPARKLING at 7T,” presented at the Joint Annual Meeting ISMRM-ESMRMB & ISMRT 31st Annual Meeting, May 2022. Accessed: May 08, 2023. [Online]. Available: https://hal.science/hal-03901242

[4] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, “Compressed Sensing MRI,” IEEE Signal Process. Mag., vol. 25, no. 2, pp. 72–82, Mar. 2008, doi: 10/cgzmm6.

[6] O. Maier et al., “CG-SENSE revisited: Results from the first ISMRM reproducibility challenge,” Magn. Reson. Med., vol. 85, no. 4, pp. 1821–1839, 2021, doi: 10.1002/mrm.28569.

[7] B. P. Sutton, D. C. Noll, and J. A. Fessler, “Fast, iterative image reconstruction for MRI in the presence of field inhomogeneities,” IEEE Trans. Med. Imaging, vol. 22, no. 2, pp. 178–188, Feb. 2003, doi: 10.1109/TMI.2002.808360.

[8] J. G. Pipe and P. Menon, “Sampling density compensation in MRI: Rationale and an iterative numerical solution,” Magn. Reson. Med., vol. 41, no. 1, pp. 179–186, 1999, doi: 10.1002/(SICI)1522-2594(199901)41:1<179::AID-MRM25>3.0.CO;2-V.

[9] P. J. Beatty, D. G. Nishimura, and J. M. Pauly, “Rapid gridding reconstruction with a minimal oversampling ratio,” IEEE Transactions on Medical Imaging, vol. 24, no. 6, pp. 799–808, Jun. 2005, doi: 10.1109/TMI.2005.848376.

[10] S. Farrens et al., “PySAP: Python sparse data analysis package for multidisciplinary image processing,” Astronomy and Computing, vol. 32, p. 100402, 2020, doi: 10/ghw8zm.

[11] A. H. Barnett, “Aliasing error of the exp$(\beta \sqrt{1-z^2})$ kernel in the nonuniform fast Fourier transform,” arXiv:2001.09405 [cs, math], Oct. 2020, Accessed: Mar. 04, 2022. [Online]. Available: http://arxiv.org/abs/2001.09405

[12] R. K. Robison, A. G. Anderson III, and J. G. Pipe, “Three-dimensional ultrashort echo-time imaging using a FLORET trajectory,” Magnetic Resonance in Medicine, vol. 78, no. 3, pp. 1038–1049, 2017, doi: 10.1002/mrm.26500.

[13]T. Speidel, P. Metze, and V. Rasche, “Efficient 3D Low-Discrepancy k -Space Sampling Using Highly Adaptable Seiffert Spirals,” IEEE Transactions on Medical Imaging, vol. 38, no. 8, pp. 1833–1840, Aug. 2019, doi: 10.1109/TMI.2018.2888695.

Figures

Backend	Hardware Support	Batch computation	Density Estimation	MRI-NUFFT Interface
cufinufft	GPU (CUDA)	✅	✅ (3)	cupy/torch
finufft	CPU	✅	TBA	numpy
gpunufft	GPU	✅	✅	numpy
tensorflow-nufft	GPU (CUDA)	❌	✅	tensorflow
pynufft-cpu	CPU	❌	❌	numpy
sigpy	CPU	✅	TBA	numpy
bart	CPU	✅	❌ (4)	numpy
pynfft (1)	CPU	❌	❌	numpy
stacked (2)	CPU/GPU	✅	N/A	numpy

Figure 1: Available backends in MRI-NUFFT. Specificities are detailed in MRI-NUFFT documentation.

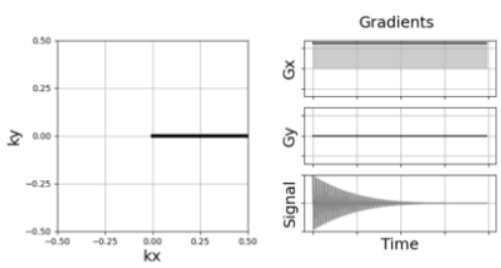

Figure 2 (animated): Examples of normalized 2D trajectories (left) and corresponding gradient waveforms (right) available and customizable in the library (Radial, Spiral, Cones, Sinusoids, Rings, Rosette, Waves, Lissajous).

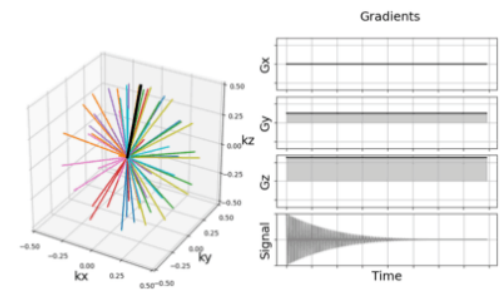

Figure 3 (animated): Examples of normalized 3D trajectories (left) and corresponding gradient waveforms (right) available and customizable in the library (3D Cones, FLORET, Seiffert Spirals/Yarnballs, Concentric Shells, Wave-CAIPI).

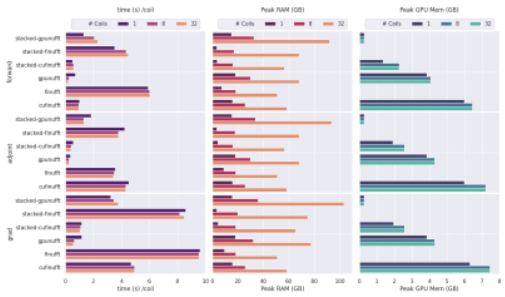

Figure 4: Benchmark of classical NUFFT operation for full 3D Acquisition.

Sigpy and BART were not tested as they are an order of magnitude slower. gpuNUFFT outperforms its competitors, benefitting from optimized memory transfers. Cufinufft is expected to benefit from the same optimization in a near future.

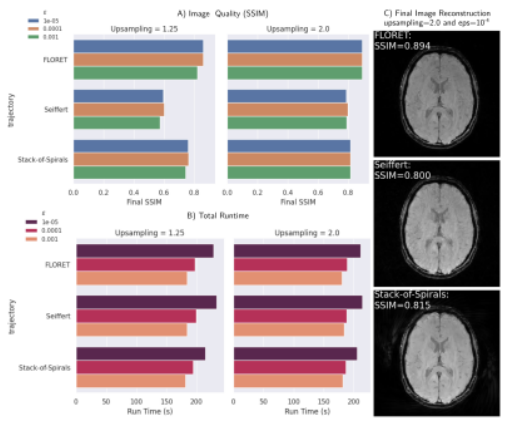

Figure 5: Image Quality benchmark for trajectories.

We Use gpuNUFFT as the Fourier Operator, in a single coil setting, and report the impact of trajectory and NUFFT parameters on the image quality (A) as well as for the total runtime (B). The FLORET trajectory performs best in term of SSIM (C). The precision (ε) and upsampling factor parameters have little impact on the image quality but participate respectively to the total runtime and GPU memory footprint.