Veldmann Marten (Orcid ID: 0000-0003-2444-9649)
Ehses Philipp (Orcid ID: 0000-0002-5839-6525)
Chow Kelvin (Orcid ID: 0000-0003-0698-1746)
Stoecker Tony (Orcid ID: 0000-0002-8946-9141)

Open-Source MR Imaging and Reconstruction Workflow

Marten Veldmann,[1] Philipp Ehses,[1] Kelvin Chow,[2] Jon-Fredrik Nielsen,[3] Maxim Zaitsev,[4] and Tony Stöcker[1,5]

[1] MR Physics, German Center for Neurodegenerative Diseases (DZNE), Bonn, Germany.
[2] MR R&D Collaborations, Siemens Medical Solutions USA Inc., Chicago, IL, United States.
[3] Department of Biomedical Engineering, University of Michigan, Ann Arbor, MI, United States.
[4] Division of Medical Physics, Department of Radiology, Faculty of Medicine, Medical Center – University of Freiburg, Freiburg, Germany.
[5] Department of Physics & Astronomy, University of Bonn, Bonn, Germany.

Total word count: 4910

Corresponding author:
Tony Stöcker, Ph.D.
German Center for Neurodegenerative Diseases (DZNE), MR Physics
Venusberg-Campus 1, Gebäude 99
53127 Bonn, Germany E-Mail: tony.stoecker@dzne.de
Phone: +49-228-43302-860

This is the author manuscript accepted for publication and has undergone full peer review but has not been through the copyediting, typesetting, pagination and proofreading process, which may lead to differences between this version and the Version of Record. Please cite this article as doi: 10.1002/mrm.29384

Abstract

Purpose

This work presents an end-to-end open-source MR imaging workflow. It is highly flexible in rapid prototyping across the whole imaging process and integrates vendor-independent openly available tools. The whole workflow can be shared and executed on different MR platforms. It is also integrated in the JEMRIS simulation framework, which makes it possible to generate simulated data from the same sequence that runs on the MRI scanner using the same pipeline for image reconstruction.

Methods

MRI sequences can be designed in Python or JEMRIS using the Pulseq framework, allowing simplified integration of new sequence design tools. During the sequence design process, acquisition metadata required for reconstruction is stored in the MRD format. Data acquisition is possible on MRI scanners supported by Pulseq and in simulations through JEMRIS. An image reconstruction and postprocessing pipeline was implemented into a Python server that allows real-time processing of data as it's being acquired. The BART toolbox is integrated into this framework for image reconstruction. The reconstruction pipeline supports online integration through a vendor-dependent interface.

Results

The flexibility of the workflow is demonstrated with different examples, containing 3D parallel imaging with CAIPIRINHA acceleration, spiral imaging and B0 mapping. All sequences, data and the corresponding processing pipelines are publicly available.

Conclusion

The proposed workflow is highly flexible and allows integration of advanced tools at all stages of the imaging process. All parts of this workflow are open-source, simplifying collaboration across different MR platforms or sites and improving reproducibility of results.

key words

MR imaging workflow; open-source; sequence development; image reconstruction; simulation

Introduction

Open Science and open-source software tools are of increasing importance in today's MR research as the number of available open-source software has constantly grown over the years. The ISMRM website MR-Hub (1) and the website of the Open Source Imaging Initiative (2) currently list over 40 MRI related open-source tools. Many of these tools are actively developed and contain state of the art algorithms for MR imaging. Open-source imaging software and hardware readily helps many researchers to collaborate and improve their own research, as well as reproduce outcomes of published literature. Recently, results from the first ISMRM reproducibility challenge targeting MR image reconstruction were published (3). In this challenge, reconstruction results from CG-SENSE implementations of different submissions were compared. It concluded that small variations in implementation details or input parameters can lead to significant differences in images and that access to the original source code and data is indispensable for a reliable reproduction of research results. Well-maintained online resources consolidating open-source tools, such as opensourceimaging.org (2), are expected to play an increasingly important role in promoting reproducibility and sustainability of MR imaging studies.

Open-source software tools are available for many parts of the MR imaging process including sequence development, data acquisition, image reconstruction and image postprocessing or analysis. In the case of development or modification of MRI sequences, the exact time course of RF pulses and gradients may be of high importance for reproducibility. However, publications typically do not contain the detailed fine-grained timing information of new sequences, but only the general idea and high-level features. Sequence source code itself may also not be shareable for intellectual property or contractual reasons and hardware and software versions may be incompatible.
The Pulseq (4) and TOPPE (5) file formats provide an open-source description of a complete pulse sequence's timing and waveforms defined in one file, which can be executed on scanners running different software versions and also from different vendors.

On the other end of the imaging pipeline, reconstruction of MR images is increasingly dependent on parameter choices, as algorithms get more complicated and allow more tuning parameters. The results may depend on the specific implementation of a reconstruction algorithm, making comparisons between studies difficult. Novel reconstruction algorithms are often executed offline, due to the difficulty of integrating them into the existing vendors' reconstruction frameworks. The Gadgetron (6) project addresses this problem, by using an extensible image reconstruction framework based on streaming data pipelines, that can be integrated into the existing reconstruction environment of the MRI scanner.

The diversity in input raw data such as format, ordering, and preprocessing further complicates the development of a generalizable pipeline. Therefore, widely used open-source data formats, such as the ISMRM raw data (MRD, originally ISMRMRD) (7) for MR raw data or the NIfTI format (8) for image data, are crucial for standardizing data structures and sharing algorithms efficiently.

These openly available tools contribute towards improving reproducibility of published research results. However, there is currently no open-source workflow covering all aspects

of the MR imaging process from sequence design to image reconstruction. For example, results may only be partly reproducible if specific raw datasets are needed to reproduce the results of an image reconstruction algorithm, as the sequence may not be made available. The proposed workflow aims at combining different tools to form an open-source end-to-end imaging pipeline, which is completely shareable and can easily be extended by new tools. The pipeline covers MRI sequence development, data acquisition, image reconstruction and postprocessing of images. MRI sequence development and data acquisition is based on the Pulseq framework, while the MRD format is used for storage of MR raw- and metadata. Acquired raw data are processed by a Python based server. The data can be streamed to the reconstruction server either offline or online, where the latter requires a vendor dependent streaming interface. Reconstruction is done with the Berkeley Advanced Reconstruction (BART) toolbox (9).

The pipeline was also integrated into the JEMRIS (10,11) simulation framework, by adding an interface to the image reconstruction pipeline to the framework. As a result, sequences designed and simulated in JEMRIS can be executed without modification on the MRI scanner, and simulated and acquired data can be reconstructed with the same pipeline.

The whole workflow is based on openly available tools, with the exception of the interfaces for on-scanner sequence execution and data streaming for online reconstruction. These interfaces are vendor-dependent and thus not entirely open-source. However, the streaming interface is optional as the image reconstruction pipeline can also be executed offline. The Pulseq interpreter sequence is also shared in the source code form within the respective vendor communities.

Methods

The open-source imaging workflow contains sequence design, data acquisition, image reconstruction and optional postprocessing of images. An overview of the whole pipeline is shown in Figure 1.

Sequence Development

Sequence design can be done with the Pulseq framework, using either the official MATLAB toolbox (12), the Python implementation PyPulseq (13) or JEMRIS. All tools generate Pulseq sequence files, which contain the complete timing for RF pulses, gradients and ADC sampling points. In the present paper, PyPulseq and JEMRIS are used, as neither depends on commercial software.

The Pulseq format currently has no support for transferring metadata and k-space information to the MRI scanner, it only contains the sequence timing. The vendor's raw data files that originate from the Pulseq sequence execution therefore contain only the acquired data with no information on k-space sampling. These raw data files do contain a header section, but only with dummy values. For this reason, an additional MRD "metadata" file is created together with the Pulseq sequence. This file contains all relevant information about the measurement and is merged with the raw data before image reconstruction.

For identification of the files, the MD5 hash of the Pulseq sequence file is calculated and appended to both the sequence and the metadata file as a signature.

PyPulseq

The PyPulseq toolbox implements the functionalities of the official Pulseq MATLAB toolbox in Python. It provides common RF pulses and gradient waveforms, as well as example sequences. Arbitrary gradient and RF waveforms are also possible allowing for high flexibility. The additional metadata file is created with the Python implementation of the MRD format (7).

First, all elements of a sequence containing RF pulses, gradient waveforms and ADCs are defined. Sequence parameters such as the FOV, resolution, number of slices and the k-space trajectory type are added to the MRD header as illustrated in Figure 2 (left). The timing of the sequence is represented as a gapless concatenation of time slices termed "blocks" in Pulseq. Each block may define a single RF, ADC or gradient pulse event per gradient axis, whereas each of these events may be delayed by an arbitrary period of time. The duration of each block is defined by the duration of the longest event within that block or an optional additional delay object that can be used to increase the duration of the block.

For each ADC/readout event present in the sequence, acquisition parameters are added to the MRD metadata file, containing k-space counters, flags and optionally the k-space trajectory. Further sequence-specific information for reconstruction and postprocessing (e.g., b-values for diffusion sequences) can be added by using user defined parameters and arrays. Auxiliary information, such as the sequence name or the FOV can be added to the Pulseq

file. Knowledge about the FOV is useful for a correct visualization of the acquisition volume at the scanner.

JEMRIS

JEMRIS provides a graphical user interface for sequence development, where the sequence output are XML files that can also be edited directly. JEMRIS also provides common RF and gradient shapes, as well as the ability to import user-defined shapes via HDF5 files. Sequences can be simulated directly in JEMRIS or exported to the Pulseq format for scanner execution. The export of the JEMRIS sequence XML file to Pulseq is done automatically based on the provided XML file. For this work, the Pulseq file export was extended to support rotations of gradient waveforms with a given rotation angle. This simplifies sequence development for rotationally symmetrical k-space trajectories such as radial or spiral trajectories. Additionally, a new time-optimized spiral gradient (14) was implemented in JEMRIS.

An MRD metadata file is automatically created during export to the Pulseq format. Header information is taken directly from the parameter module in JEMRIS. K-space positions are defined by the k-space trajectory, which is calculate for each ADC event in the sequence. However, as JEMRIS does not distinguish between different types of loops or ADCs, different k-space acquisitions cannot be separated easily. Therefore, the "Loop Type" and "ADC Type" parameters were added to each pulse module, which is shown in Figure 2 (right). The "Loop Type" classifies loops to distinguish, if different lines in k-space (e.g., phase encoding and partition loop) or different images (e.g., slice, contrast, set or average loop) are acquired. The "ADC Type" defines the ADC sampling as an imaging ADC or some sort of calibration ADC (noise, parallel imaging calibration, phase calibration).

Data acquisition and simulation

Raw data in the presented workflow can originate either from a real acquisition on an MRI scanner or from a simulation with the JEMRIS framework. For the latter, the sequence must be designed in JEMRIS, as there is currently no efficient import of Pulseq sequences in JEMRIS.

In a real experiment, the sequence file is exported to the MRI scanner and selected in the scanner GUI for execution as shown in Figure 1. Pulseq sequences are run with a vendor-specific interpreter sequence, which supports integrated field-of-view positioning. Currently, sequence interpreters for Siemens, GE and table-top MRI scanners are available. During sequence execution, both the sequence name and the MD5 signature are saved in the raw data header in order to identify the correct metadata file in the reconstruction.

Simulation of a sequence in JEMRIS is executed either from the command line or in the GUI by providing the sequence XML file, the digital phantom and its MRI-relevant parameters, and optionally receive and transmit coil sensitivities. An MRD file is generated after simulation (Figure 1), containing both the MRI signal and the metadata, as well as receive coil sensitivities for multi-coil simulations.

Image reconstruction & Postprocessing

The image reconstruction is initiated by streaming the raw data to a Python server running inside a Docker container. Data processing scripts are selected by a configuration string, sent to the server together with the raw data. An overview of the pipeline is given in Figure 3.

MRI scanner data

Raw data from the scanner are converted to the MRD format and streamed to the reconstruction server by a client using a format initially developed by the Gadgetron framework and extended for other workflows. This is done either online with a vendor-dependent interface or offline with a converter and a Python-based client. Converters from the most common vendor data formats to MRD are provided by the MRD project (https://github.com/ismrmrd). The prototype Siemens Framework for Image Reconstruction Environments (FIRE) (15) was used as the vendor interface for online reconstruction in this work. This interface allows real-time streaming of acquired data, which can be selected in the scanner GUI prior to execution of the sequence. The online pipeline is configured with an XML file, that is linked to the Pulseq interpreter sequence (15), similar to the configuration used by the Gadgetron project (6).

Prior to image reconstruction, the MRD metadata file is transferred to the reconstruction server. The header and k-space information from the metadata file are automatically merged with the corresponding raw data in the reconstruction pipeline. Optional trajectory correction with the gradient impulse response function (GIRF) (16) can be performed by supplying gradient shapes instead of k-space trajectories. This requires knowledge of the scanner specific GIRF as well as additional information for aligning the trajectory with the ADC readout samples.

Image reconstruction is triggered, when all data for a complete image is collected, for example by a metadata flag identifying the last acquisition in a slice. The pipeline contains processing steps for sorting the data, noise pre-whitening with noise scans and parallel imaging calibration using reference k-space data. Prescan data is separated from imaging data, by reading the corresponding metadata flags. Calculation of coil sensitivity maps is done with the ESPIRiT algorithm (17), implemented in the BART toolbox. Other reconstruction steps, such as k-space filtering and application of phase navigator data can be integrated into the existing pipeline.

Fully sampled Cartesian data are reconstructed with a simple FFT in Python, while undersampled and non-Cartesian data are processed with the BART toolbox, using its parallel imaging and NUFFT implementations. If sensitivity maps were calculated, the parallel imaging with compressed sensing reconstruction ("pics") implemented in BART is executed. Online reconstructed data are streamed back to the MRI scanner in real time and can be viewed in the scanner console GUI while the acquisition is still ongoing. Images, that were reconstructed in offline mode are stored in the MRD image format.

JEMRIS simulation data

For simulated data, the reconstruction pipeline can either be started from the JEMRIS GUI or from the command line. In the first step, the MRD data are streamed to the server application (Figure 3). Simulated data are processed with the same pipeline as data from the MRI scanner that were acquired with a JEMRIS sequence. The MRD file created after simulation already contains both metadata and imaging data. In the case of multiple simulated receiver coils, the coil sensitivities that were used during the simulation are directly passed to the pipeline. If no additional reference data for parallel imaging calibration are acquired in the simulation, these coil sensitivities are also used in the reconstruction.

Currently, it is not possible to define Cartesian k-space sampling for JEMRIS sequences in the metadata or to detect Cartesian sampling during the reconstruction. Therefore, the image reconstruction treats all simulated data as non-Cartesian and thus requires the k-space trajectory, even if all data points lie on a Cartesian grid. Reconstruction is done either with BART's NUFFT or with its parallel imaging reconstruction implementation. After reconstruction, images are saved to an HDF5 file and displayed in the JEMRIS GUI, if the pipeline was started from the GUI.

Experiments

Different imaging sequences were created with the JEMRIS GUI as well as with PyPulseq to demonstrate the flexibility of the presented workflow. Experimental data were mainly acquired on a 7T scanner in Bonn (Siemens MAGNETOM 7T Plus, Siemens Healthineers, Erlangen, Germany), while one example sequence was additionally executed on several 3T scanners as described below.

The first example sequence designed with JEMRIS contains a 3D GRE Cartesian readout. Signal excitation was achieved by a non-selective block excitation pulse with a duration chosen to achieve water excitation (suppressing fat signal) at 7T ($d = 1.02$ ms). The acquisition was accelerated by a factor of $R = 4$ in the first phase encoding direction, with and without a CAIPIRINHA (18) shift of $\delta = 1$. FLASH-based low-resolution reference scans were acquired prior to the measurement in order to obtain coil sensitivity maps. The FOV was 210 x 210 x 160 mm^3 at 1 mm isotropic resolution. The measurement was repeated with four different variations:

1. TE = 5 ms, TR = 10 ms, with RF spoiling
2. TE = 5 ms, TR = 10 ms, with RF spoiling, fat-selective sinc-pulse (1 kHz bandwidth) instead of water excitation
3. TE = 25 ms, TR = 30 ms, no RF spoiling
4. TE = 25 ms, TR = 30 ms, no RF spoiling, no CAIPIRINHA shift

As a non-Cartesian example, a 2D spiral sequence with a time-optimized (14) k-space trajectory was created using both JEMRIS (without fat saturation pulse) and PyPulseq (with fat saturation pulse). One slice with a slice thickness of 1 mm and a FOV of 220 x 220 mm^2 at 1 mm isotropic resolution was acquired. The PyPulseq version of this sequence was additionally executed at two 3T scanners in Bonn (3T Skyra) and Freiburg (3T Prisma, both Siemens Healthineers, Erlangen, Germany) to demonstrate portability. It was successfully

executed also on a 3T Vida scanner (Siemens Healthineers, Erlangen, Germany) running on a different vendor's software version (results not shown in the present paper). Additionally, the sequence was converted to the TOPPE (5) file format using the "PulseGEq" converter provided by the TOPPE project (19). It was then executed on a 3T UHP scanner (General Electric Healthcare, Waukesha, WI, USA) to show the compatibility of the pipeline across two different vendors. At 3T, both the slice thickness (3 mm) and TR (200 ms) were increased for higher SNR and better contrast. Since the TOPPE format currently does not support ADC sampling intervals of different duration, coil sensitivity calibration was performed with the spiral data.

A slightly modified version of the same spiral sequence was simulated in JEMRIS demonstrating the influence of chemical shift and susceptibility in a sample. In the simulation, gradient spoiling was replaced with long TR spoiling, as correct simulation of gradient spoiling needs many simulated spins resulting in exceedingly long computation times (20).

A 2D Cartesian B0 mapping sequence was developed to show the postprocessing capabilities of the reconstruction pipeline and to allow for B0 correction of the spiral imaging data. One slice with 2 mm slice thickness and a FOV of 220 x 220 mm^2 at 2 mm isotropic resolution was acquired.

All images were reconstructed with the BART toolbox. Calculation of the B0 field map from raw GRE images was done with Python, using the scikit-image (21) and SciPy (22) libraries for phase unwrapping and filtering. The spiral data acquired with the PyPulseq sequence were reconstructed with a GIRF predicted trajectory. The PowerGrid toolbox (23) was used in the pipeline to achieve B0 correction of the spiral data with a time segmented reconstruction approach (24), using the B0 field map calculated before. Online reconstruction was performed exclusively on the 7T MRI scanner.

The source code, the sequence and metadata files created with PyPulseq and JEMRIS, as well as the raw data and reconstructed images can be found at https://github.com/mrphysics-bonn/python-ismrmrd-reco (Git hash 15df3aa). The filenames in the repository are linked to the figures of this paper as shown in Supporting Table S1. The repository also contains the reconstruction server with instructions on how to set up and use the pipeline (with and without GPU support). A Docker image of the reconstruction server can be found at https://hub.docker.com/repository/docker/mavel101/bart-reco-server. The new JEMRIS version with additional examples for the metadata file export and the reconstruction of simulated data are available in the JEMRIS GitHub repository (https://github.com/JEMRIS/jemris) and on the JEMRIS website (https://www.jemris.org/).

Results

Reconstructed images from the 3D GRE sequence designed with JEMRIS are displayed in Figure 4. Images with water excitation in the upper row show a typical T1 weighted contrast at short TE. Fat signal in the skull is suppressed, while it is the dominant signal in the fat excited images in the lower row. However, images acquired with fat excitation still show some residual water signal in the brain. Figure 5 shows images from the same 3D GRE sequence with a longer echo time with and without CAIPIRINHA shifts, demonstrating a T2* contrast. The CAIPIRINHA shift reduces artifacts, which are especially visible in the sagittal view, where stripe-shaped artifacts disappear.

In Figure 6 a) the magnitude GRE image from the B0 mapping sequence at the first echo time TE = 2.04 ms is shown. The phase difference map in Figure 6 b), which was calculated from both echoes, has no visible phase wraps in the brain. Figure 6 c) is the resultant B0 field map, that was smoothed with Gaussian ($\sigma = 0.5$ pix) and median filters (kernel size 2x2 pix).

Images acquired with the 2D spiral sequence are shown in Figure 7. The image a), acquired without fat-suppression, shows a stripe-shaped artifact at the periphery of the brain caused by folded fat signal. The overall blurring in this image is mainly due to B0 inhomogeneities. In image b) fat artifacts are removed due to the fat suppression pulse and the blurring is reduced significantly. In c) the same image with additional B0 correction using the map in c) has even less blurring and signal is recovered especially in the anterior part of the brain. The images acquired at all three 3T scanners in d)-f) show only minor artifacts in the frontal brain mostly caused by B0 inhomogeneities. Slight geometric distortions presumably due to gradient imperfections are visible in the posterior part of the brain.

Reconstructed images from one simulated slice acquired with a spiral sequence are shown in Figure 8. Simulating a clean digital phantom yields artifact free images. Adding the chemical shift of fat to the digital brain phantom results in stripe-shaped artifacts similar to the artifacts in Figure 6 d). Including magnetic susceptibility in simulations, that is causing B0 inhomogeneities, leads to the typical blurring artifact, well-known in spiral imaging. Both chemical shift and susceptibility differences lead to signal loss especially in the lower brain (upper row in Figure 8).

Discussion

Flexibility and Extensibility

The examples presented in this paper demonstrate the high flexibility of the proposed workflow regarding the sequence design methods and the applied reconstruction algorithms. Advanced imaging techniques such as parallel imaging with CAIPIRINHA or non-Cartesian sampling are integrated in the workflow. The workflow allows users to prototype new sequences and reconstruct the acquired data with vendor-independent tools. Existing code for sequence generation can easily be extended with additional sequence design tools, such as Sigpy (25) or the gradient optimization toolbox GrOpt (26).

Based on the example of a spiral sequence, we showed that sequence execution across different scanners and vendors is possible, using the same image reconstruction pipeline (with minor modifications). However, porting a sequence from one acquisition system to another requires adhering to any differences in hardware properties and safety limits that may exist. For example, in the presented spiral sequence, the gradient slewrate had to be slightly reduced from 7T to 3T scanners, due to peripheral nerve stimulation limits.

For the conversion of the spiral sequence to the GE-compatible format TOPPE, prescans for noise and coil sensitivity calibration had to be removed. These prescans can be acquired with separate sequences, but this does require manual integration of the calibration data into the spiral reconstruction. Small timing changes were needed to fit the requirements of the TOPPE format with only minimal effect on the acquired data for this particular sequence. As TOPPE is a relatively young file format under active development, future improvements regarding the compatibility of Pulseq and TOPPE are expected.

Furthermore, the workflow allows for comparison of data from the JEMRIS MRI simulator with an actual acquisition at the MRI scanner. This is useful for testing sequences before running them on a real MRI scanner or to investigate the influence of specific physical properties (e.g., presence of fat) on data acquisition. However, simulating the exact same sequence, that is running on the MRI scanner sometimes is not feasible, as some physical effects might not be included in the simulated model or require excessively long computation times.

The available reconstruction pipelines for both Pulseq and JEMRIS data can reconstruct images from many different MR imaging sequences and can be used as a starting point for more elaborate reconstructions or postprocessing techniques. Additional sequence-specific meta information such as inversion times or b-values can be transferred and accessed in the reconstruction pipeline by adding them as user defined parameters or arrays to the MRD metadata file. The BART toolbox already provides much functionality for preprocessing and calibration of data, as well as advanced image reconstruction algorithms. However, integration of new reconstruction or postprocessing tools into the existing pipeline is also possible.

Online integration of the reconstruction pipeline simplifies testing of novel sequences, that require non-standard reconstruction techniques such as non-Cartesian sequences. It also allows using reconstructed images from novel sequences for calibration such as B0 or B1-

shimming. Since reconstruction scripts can be dynamically embedded into the Docker container without rebuilding, reconstruction scripts can be changed and tested during a scanning session.

Openness and Reproducibility

All file formats used in the workflow are open-source, including the Pulseq sequence file, MRD metadata file and JEMRIS XML files. Source code of the reconstruction pipelines and sequences developed in Python can be made openly available, as no proprietary code is used. Reconstruction pipelines can be shared and deployed via Docker images, which require no additional modifications of the system, as all dependencies are already installed inside the container.

In summary, the presented workflow allows sharing the whole imaging workflow by providing the sequence file, metadata file and the reconstruction pipeline. In this way, it is potentially possible to reproduce data acquisition and reconstruction with the same parameters at MRI scanners from different vendors, with different software versions, and at different sites. Sharing the source code of both sequences and reconstruction can simplify collaborations between different sites. For sites already using the Pulseq framework, integration of the proposed workflow into existing pipelines would not require much effort.

In the present paper, we demonstrated the portability of the workflow, by acquiring images with the same sequence at three different 3T scanners located at two different sites using the same image reconstruction pipeline.

Inline reconstruction directly on the vendor's interface significantly improves the workflow by providing real-time feedback during experiments and improves the user experience by automating the reconstruction. However, inline integration is optional and all reconstruction pipelines can also be run offline if a vendor-dependent interface is unavailable or a fully open-source pipeline is desired. Therefore, no proprietary software is required for the post-acquisition part of the workflow, as converters to the MRD file format exist for all common vendor raw data formats.

Performance of the pipeline

If sequence parameters are changed, both sequence and metadata files must be recreated and transferred to the scanner and the reconstruction server. This procedure can be automated, depending on the local scanner setup. However, for long sequences, metadata files can get quite large due to redundant trajectory information stored for each readout. The computation times for creating metadata files increases with file size, which might limit rapid testing of different protocols as well as manual parameter optimization at the scanner for long sequences. However, recreation of the metadata file for online reconstruction is only necessary if reconstruction-specific parameters are changed.

The performance of the reconstruction pipeline depends on many factors including configuration of reconstruction parameters and possible preprocessing steps. Performance optimization is especially important, when the pipeline is to be executed online. The example 2D spiral reconstruction required about 10s of computation, while the 2D Cartesian recon-

struction finished in under a second. The much larger accelerated 3D Cartesian dataset required about 15 min of computation (16 core CPU, NVIDIA A6000 GPU). Reconstruction times for the simulated data are negligible compared to the simulation times.

For complex reconstructions and large datasets, the total reconstruction time mainly depends on the time for the coil sensitivity calibration and the reconstruction with BART. Optimization of reconstruction parameters or the usage of coil compression hold potential for future performance improvements. For large datasets, reading and merging the metadata takes a significant amount of time. In future development, metadata could be stored in a less redundant way or transferred directly to the scanner at sequence runtime via the Pulseq format to accelerate the merging process.

Limitations

In the current implementation of the reconstruction pipeline, calibration data must be acquired within the same sequence as the imaging data. Separately acquired prescans for coil sensitivity calibration or field correction must be integrated manually into the reconstruction, requiring modification of the reconstruction code. This is unfavorable, if the user wants to reconstruct multiple datasets using the same calibration from a single prescan. Future implementation of linking calibration to imaging data would increase usability of the pipeline.

The automatic metadata and sequence file creation from JEMRIS simplifies the development process as no programming is necessary. However, it is currently not possible to add arbitrary user-defined sequence-specific information to the metadata file. Further extension of JEMRIS to include such information may be the focus of future work. Data acquisition by simulation in JEMRIS is only possible for sequences designed in JEMRIS. Conversion of Pulseq sequence files to the JEMRIS XML format is possible (27), however, the high-level loop structure of a sequence cannot be recovered from Pulseq files, leading to excessively long computation times in simulations.

Setting up the whole pipeline and extending it for one's own experiments might require some time and experience with Pulseq, the MRD file format and the processing of streamed data. However, several examples for sequences and reconstruction scripts are available in the GitHub repository, that can be used or modified for one's own purposes.

Online reconstruction of acquired data requires a vendor dependent interface and is only feasible, if the reconstruction time is not excessively long. Time consuming reconstruction routines e.g., for non-Cartesian 3D acquisitions may therefore have to be performed offline depending on the computational power of the reconstruction computer.

Conclusion

The demonstrated end-to-end open-source sequence programming and image reconstruction workflow allows for rapid prototyping and testing of MRI sequences. By using the Pulseq framework, a flexible MRD-based metadata file and streamed reconstruction pipelines, the whole imaging workflow becomes highly extensible. The workflow enables comparison of data from different MRI scanners and from MRI simulations in JEMRIS using the same pipeline for image reconstruction. The (online) image reconstruction pipeline is versatile as it is not restricted to particular types of MRI sequences and can be extended in various ways with one's own code or using available open-source tools. As all software in the proposed workflow is open-source, both sequence code and image reconstruction pipelines are vendor-independent and can be shared freely, facilitating greater reproducibility of MRI experiments.

List of figure captions

FIG. 1: Overview of the whole workflow with data acquisition at an MRI scanner (light blue) or in JEMRIS simulations (light green). Pulseq sequence and MRD metadata files are created with either PyPulseq or JEMRIS. The sequence file is executed at the scanner using a vendor-specific interpreter. Raw data are sent to the reconstruction server via the FIRE interface and the metadata from the MRD file are merged. Images are reconstructed with BART and are sent back to the scanner via FIRE. In an offline reconstruction, the FIRE interface is replaced by an MRD converter and a Python-based client. Acquired data from JEMRIS simulations is merged with the metadata inside JEMRIS and saved in the MRD format. The same reconstruction pipeline as for data from an MRI scanner data is executed.

FIG. 2: Left: Sequence development and metadata file creation with PyPulseq. The metadata file is initialized and a header is created from global sequence parameters (full header function not shown). The sequence object is created, and event blocks are added. At the same time, readout information such as k-space flags, counters and the k-space trajectory are added to the metadata file. Right: Dump of the sequence tree from a sequence developed with the JEMRIS simulation framework. The metadata header is generated from the global parameters in JEMRIS. Acquisition-specific k-space information is generated from the new JEMRIS "Loop Type" and "ADC Type" parameters and added to the metadata together with the k-space trajectory. Green color indicates new features.

FIG. 3: Detailed view of the reconstruction pipeline for raw data from an MRI scanner or the JEMRIS simulation framework. Raw data from the scanner are first converted to MRD. The data is streamed to the reconstruction server, where the reconstruction pipeline is started. The pipeline supports an optional correction of gradient imperfections with the gradient impulse response function (GIRF). Image reconstruction and optional calculation of coil sensitivity maps is done with the BART toolbox. Reconstructed images are displayed in the GUI of the scanner, the JEMRIS GUI or saved to a file.

FIG. 4: Reconstructed images from a T1 weighted 3D GRE sequence created with JEMRIS with a TE of 5 ms and 4x undersampling with a CAIPIRINHA shift. Water images were acquired with block pulses of 1.02 ms length suppressing fat signal (upper images), whereas fat excitation was achieved with fat-selective sinc-pulses (lower images).

FIG. 5: Images from the same 3D GRE sequence as in Figure 4, with a TE of 25 ms. Upper images were acquired with a CAIPIRINHA shift, while lower images were acquired without this shift. The red arrow indicates artifacts in images without CAIPIRINHA.

FIG. 6: Reconstructed images from a B0 mapping sequence. Image a) is the first magnitude image with TE = 2.04 ms, b) is the phase difference map of the two echoes and c) shows the corresponding filtered B0 field map.

FIG. 7: Reconstructed images from a 2D spiral GRE sequence acquired at 7T (a-c) and 3T (d-e) scanners from four different subjects. Image a) was acquired with a spiral sequence without fat suppression, while in b) fat suppression was added to the sequence and GIRF trajectory correction was done in the reconstruction. Image c) was reconstructed from the same raw data, but with an additional B0 correction using the field map shown in Figure 6

c). Images d)-f) were acquired at three different 3T scanners with fat suppression, but without GIRF correction in the reconstruction. Red arrows indicate artifacts from gradient imperfections and from offresonance due to chemical shift and magnetic susceptibility.

FIG. 8: Images reconstructed from data simulated with the JEMRIS simulation framework. A spiral sequence was simulated for two different slices either with a clean digital phantom, with additional chemical shift from fat or with susceptibility differences across the digital brain phantom. Artifacts from chemical shift and susceptibility are indicated by red arrows. The B0 maps on the right show both chemical shift (at approximately 1 kHz) and susceptibility-induced offresonance effects.

Sup. Table S1: Pulseq sequence, raw data, metadata and image filenames in the GitHub repository linked to the Figures in the paper. Additionally, the reconstruction scripts, the sequence source code for PyPulseq sequences and XML files for JEMRIS sequences are listed.

References

1. ISMRM Reproducible Research Study Group MR-Hub Website. Available at ismrm.github.io/mrhub/. Accessed 25 April 2022.

2. Open Source Imaging Initiative. Available at www.opensourceimaging.org. Accessed 01 February 2022.

3. Maier O, Baete SH, Fyrdahl A, et al. CG-SENSE revisited: Results from the first ISMRM reproducibility challenge. Magn. Reson. Med. 2021;85:1821–1839 doi: 10.1002/mrm.28569.

4. Layton KJ, Kroboth S, Jia F, et al. Pulseq: A rapid and hardware-independent pulse sequence prototyping framework. Magn. Reson. Med. 2017;77:1544–1552 doi: 10.1002/mrm.26235.

5. Nielsen J-F, Noll DC. TOPPE: A framework for rapid prototyping of MR pulse sequences. Magn. Reson. Med. 2018;79:3128–3134 doi: 10.1002/mrm.26990.

6. Hansen MS, Sørensen TS. Gadgetron: An open source framework for medical image reconstruction. Magn. Reson. Med. 2013;69:1768–1776 doi: 10.1002/mrm.24389.

7. Inati SJ, Naegele JD, Zwart NR, et al. ISMRM Raw data format: A proposed standard for MRI raw datasets. Magn. Reson. Med. 2017;77:411–421 doi: 10.1002/mrm.26089.

8. NIfTI neuroimaging informatics technology initiative. National Institute of Mental Health website. Available at nifti.nimh.nih.gov. Accessed 02 May 2022.

9. Uecker M, Tamir JI, Ong F, Lustig M. The BART Toolbox for Computational Magnetic Resonance Imaging. In: Proc. Intl. Soc. Mag. Res. Med. 24.; 2016. doi: 10.5281/zenodo.592960.

10. Stöcker T, Vahedipour K, Pflugfelder D, Shah NJ. High-Performance Computing MRI Simulations. Magn. Reson. Med. 2010;64:186–193 doi: 10.1002/mrm.22406.

11. JEMRIS Website. Available at www.jemris.org. Accessed 15 February 2022.

12. Pulseq Website. Available at pulseq.github.io. Accessed 08 February 2022.

13. Ravi K, Geethanath S, Vaughan J. PyPulseq: A Python Package for MRI Pulse Sequence Design. J. Open Source Softw. 2019;4:1725 doi: 10.21105/joss.01725.

14. Lustig M, Kim SJ, Pauly JM. A fast method for designing time-optimal gradient waveforms for arbitrary k-space trajectories. IEEE Trans. Med. Imaging 2008;27:866–873 doi: 10.1109/TMI.2008.922699.

15. Chow K, Kellman P, Xue H. Prototyping Image Reconstruction and Analysis with FIRE. In: Proc. From 24th annu. SCMR sci. sess.; 2021.

16. Vannesjo SJ, Graedel NN, Kasper L, et al. Image reconstruction using a gradient impulse response model for trajectory prediction. Magn. Reson. Med. 2016;76:45–58 doi: 10.1002/mrm.25841.

17. Uecker M, Lai P, Murphy MJ, et al. ESPIRiT - An eigenvalue approach to autocalibrating parallel MRI: Where SENSE meets GRAPPA. Magn. Reson. Med. 2014;71:990–1001 doi: 10.1002/mrm.24751.

18. Breuer FA, Blaimer M, Mueller MF, et al. Controlled aliasing in volumetric parallel imaging (2D CAIPIRINHA). Magn. Reson. Med. 2006;55:549–56 doi: 10.1002/mrm.20787.

19. TOPPE Website. Available at toppemri.github.io/. Accessed 02 May 2022.

20. Fortin A, Salmon S, Baruthio J, Delbany M, Durand E. Flow MRI simulation in complex 3D geometries: Application to the cerebral venous network. Magn. Reson. Med. 2018;80:1655–1665 doi: 10.1002/mrm.27114.

21. Van Der Walt S, Schönberger JL, Nunez-Iglesias J, et al. Scikit-image: Image processing in python. PeerJ 2014;2014:1–18 doi: 10.7717/peerj.453.

22. Virtanen P, Gommers R, Oliphant TE, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nat. Methods 2020;17:261–272 doi: 10.1038/s41592-019-0686-2.

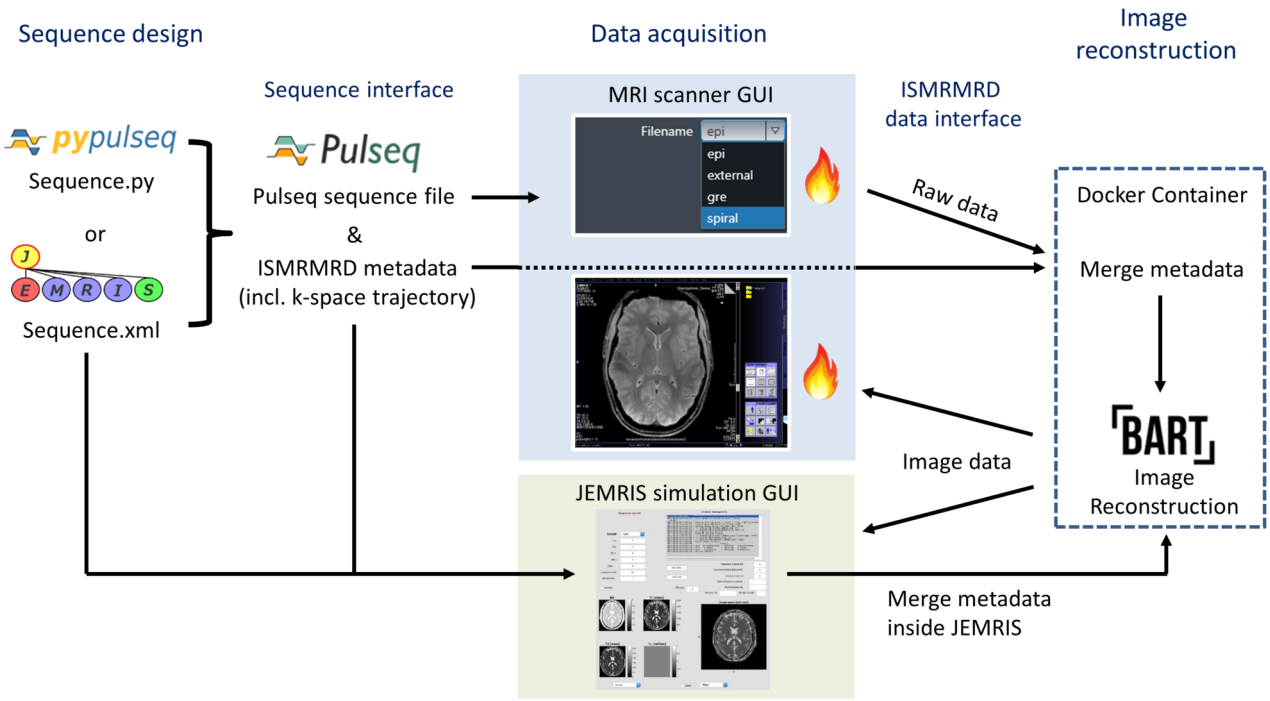
23. Cerjanic A, Holtrop JL, Ngo G-C, et al. PowerGrid: A open source library for accelerated iterative magnetic resonance image reconstruction. Proc. Intl. Soc. Mag. Res. Med. 24 2016:525.

24. Sutton BP, Noll DC, Fessler JA. Fast, iterative image reconstruction for MRI in the presence of field inhomogeneities. IEEE Trans. Med. Imaging 2003;22:178–188 doi: 10.1109/TMI.2002.808360.

25. Ong F, Lustig M. SigPy: A Python Package for High Performance Iterative Reconstruction. In: Proc. Intl. Soc. Mag. Res. Med. 27.; 2019.

26. Loecher M, Middione MJ, Ennis DB. A gradient optimization toolbox for general purpose time-optimal MRI gradient waveform design. Magn. Reson. Med. 2020;84:3234–3245 doi: 10.1002/mrm.28384.

27. Tong G, Geethanath S, Vaughan JTJ. Bridging Open Source Sequence Simulation and Acquisition with py2jemris. Proc. Intl. Soc. Mag. Res. Med. 29 2021.



MRM_29384_fig1.tif

PyPulseq

ISMRMRD Header:
- Sequence parameters
- Other metadata

```
# Create ISMRMRD file
prot = ismrmrd.Dataset(filename+'.h5')

# Create ISMRMRD header
seq_params = {'fov': fov, 'res': res, 'slice_thickness': slice_thickness,
              'dt': dwelltime, 'nitlv': nitlv }
hdr = create_hdr(prot, seq_params)
prot.write_xml_header(hdr.toXML('utf-8'))

# Make sequence for one 2D slice
seq = Sequence()
seq.set_definition('Name', filename)
seq.set_definition('FOV', [1e-3*fov, 1e-3*fov, 1e-3*slice_thickness])

for k in range(nitlv):
    # make spiral gradients with Pypulseq
    spiral_x_grad = make_arbitrary_grad(channel='x', waveform=spiral_x[k], system=system)
    spiral_y_grad = make_arbitrary_grad(channel='y', waveform=spiral_y[k], system=system)

    # add blocks to sequence
    seq.add_block(rf,gz)
    seq.add_block(gz_rew)
    seq.add_block(spiral_x_grad, spiral_y_grad, adc)

    # add acquisitions
    acq = ismrmrd.Acquisition()
    if (k == nitlv - 1):
        acq.setFlag(ismrmrd.ACQ_LAST_IN_SLICE)
    acq.idx.kspace_encode_step_1 = k
    acq.idx.slice = 0
    acq.resize(trajectory_dimensions = 2, number_of_samples=len(spiral_x_), active_channels=32)
    acq.traj[:] = np.stack([spiral_x_,spiral_y_]).T # save gradients for GIRF traj prediction
    prot.append_acquisition(acq)

# write sequence file
seq.write(filename+'.seq')
prot.close()
```

Make sequence & add blocks

ISMRMRD Acquisitions:
- K-space flags/counters
- K-space trajectory

JEMRIS

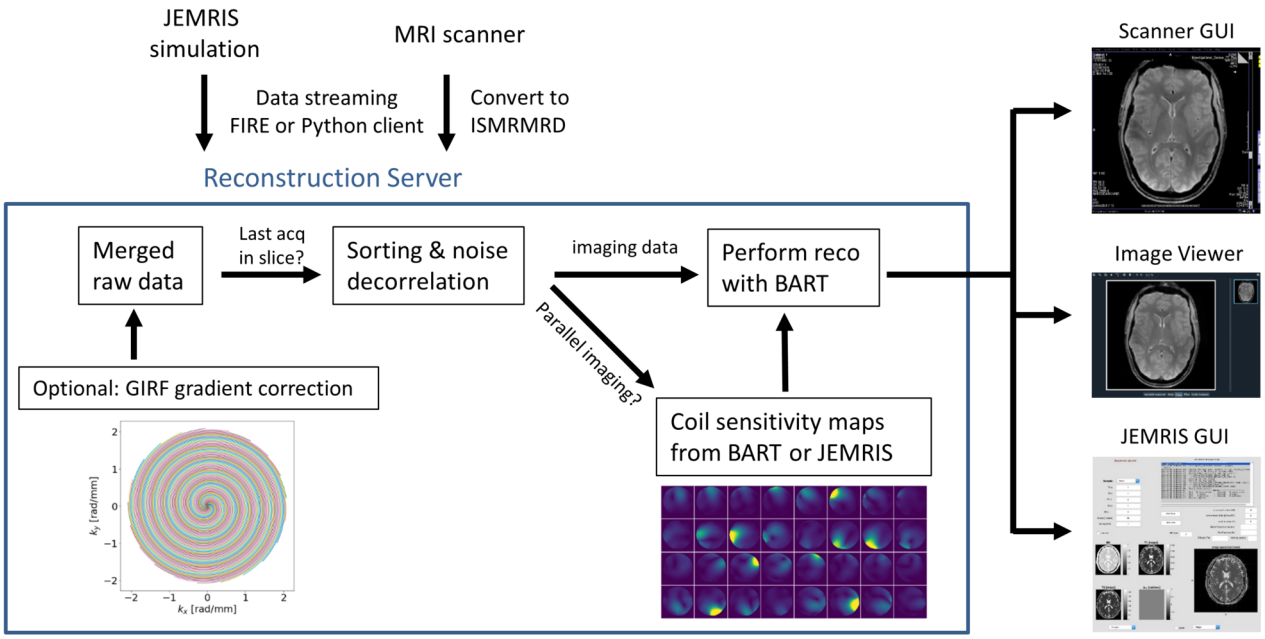
Loop type

ADC type

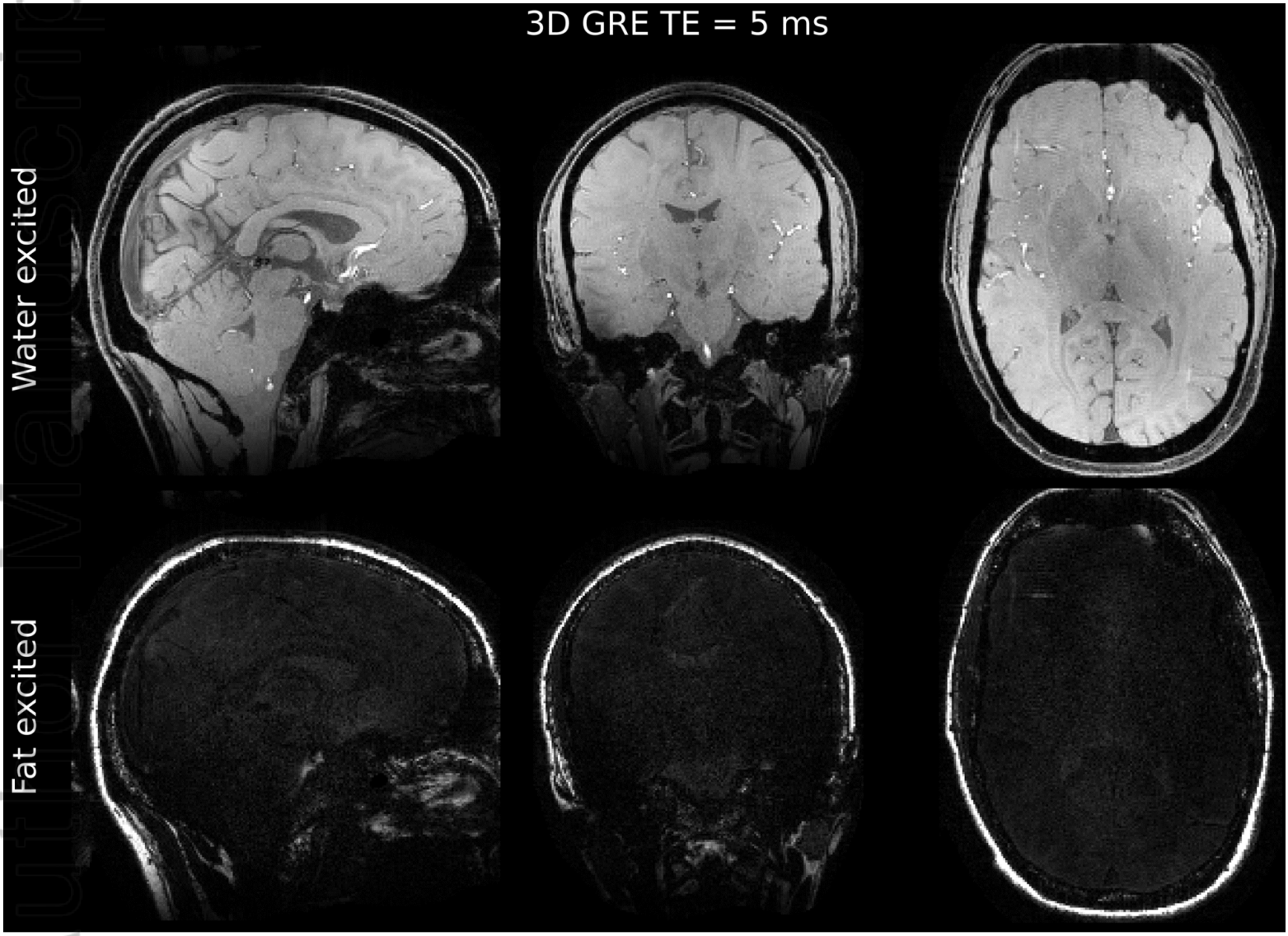
```
dump of sequence tree

                                  TYPE      NAME  duration      ADCs     TPOIs |  module specific
                                  ----- ------------------------------------- |  --------------
sequence-root-------->   CONCAT     Seq   120.000      6000      6210  |  Repetitions = 1, type SLI|PHA|PAR|SET|CON|AVG = 1|0|0|0|0|0
  |_ child 1   ------>   CONCAT    Intl   120.000      6000      6210  |  Repetitions = 15, type SLI|PHA|PAR|SET|CON|AVG = 0|1|0|0|0|0
    |_ child 1   --->     ATOM      A1     0.100         0         3  |
      |_ child 1   -->    PULSE      P1     0.100         0         3  |  Axis = RF  , (Flipangle,Phase,Bandwidth,Channel,Symmetry) = (
    |_ child 2   ---->    ATOM      A2     1.490       400       402  |  RotMtx(alpha,theta,phi) = (336,0,0)
      |_ child 1   -->    PULSE    SPox     1.490       400       402  |  Axis = GX  , ADC type ADC|IMG|ACS|PC|NOISE = 0|1|0|0|0, Area
      |_ child 2   -->    PULSE    SPoy     1.490         0         2  |  Axis = GY  , Area = 0.762885 , (Unit,grad_raster_time,tHward)
    |_ child 3   ---->    ATOM      A3     1.760         0         5  |  RotMtx(alpha,theta,phi) = (336,0,0)
      |_ child 1   -->    PULSE   Rephx     1.700         0         4  |  Axis = GX  , Area = -0.717537
      |_ child 2   -->    PULSE   Rephy     1.760         0         4  |  Axis = GY  , Area = -0.762885
    |_ child 4   ---->    ATOM      A4     1.000         0         2  |
      |_ child 1   -->    PULSE Spoiler     1.000         0         2  |  Axis = GX  , Area = 12.5664 , Amplitude = 12.5664
    |_ child 5   ---->    ATOM      TR     3.650         0         2  |  DelayType = B2E  , StartSeq = A1 , Delay = 8
      |_ child 1   -->    PULSE   eP TR     3.650         0         2  |  Axis = none
```

Pulseq Export in JEMRIS generates
1. Pulseq sequence file
2. Metadata file using:
   - Loop types: Slices, Phase Encoding, …
   - ADC types: Imaging, Calibration, …
   - Global Parameters: FOV, Matrix size
   - K-space trajectory

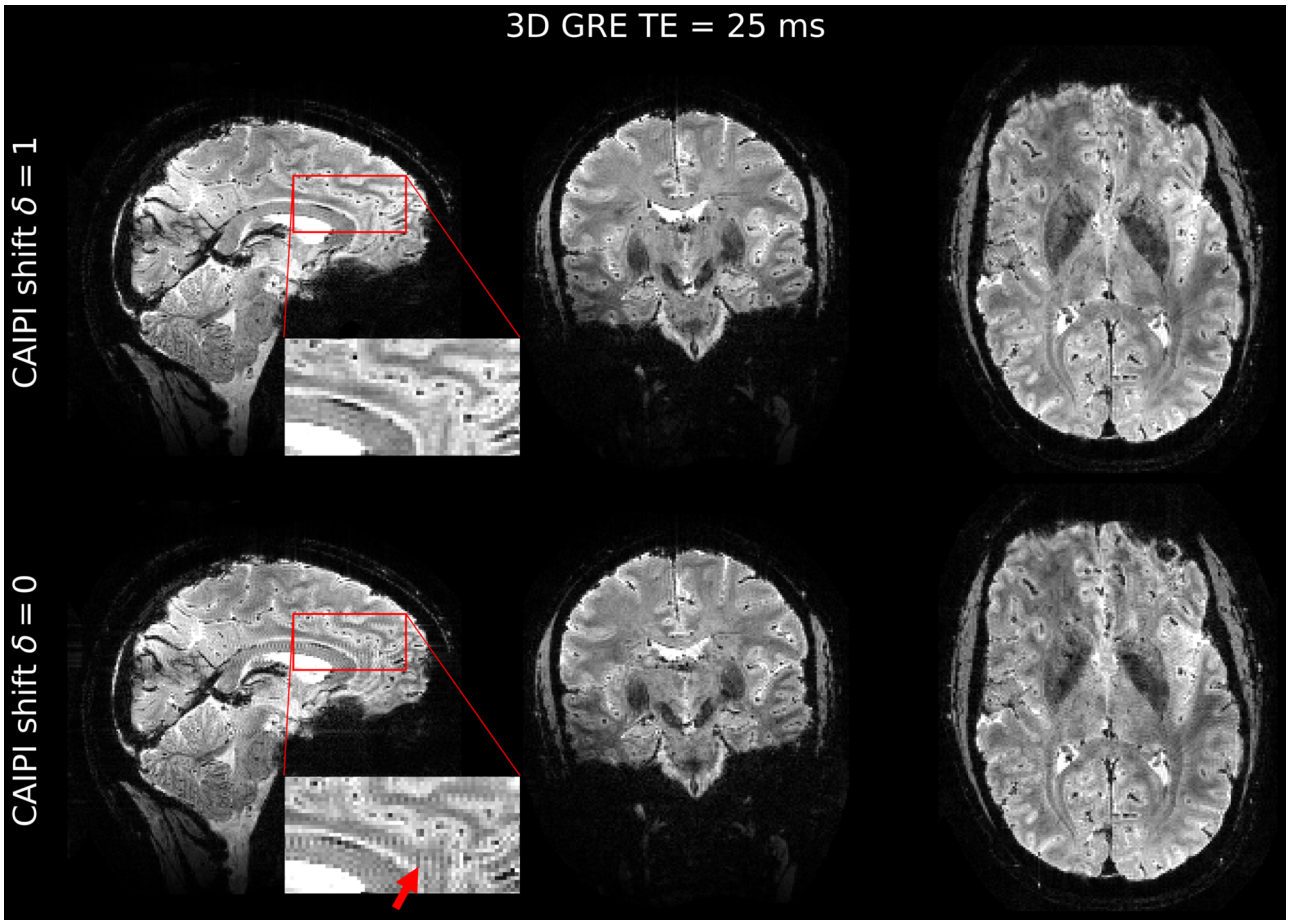
MRM_29384_fig2.tif



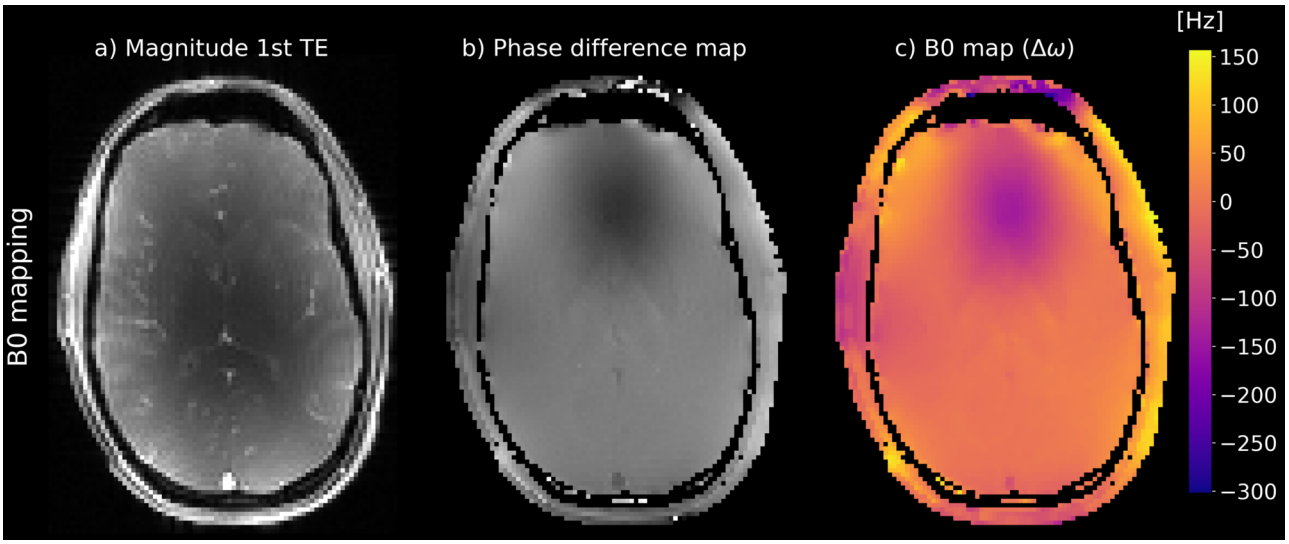
MRM_29384_fig3.tif



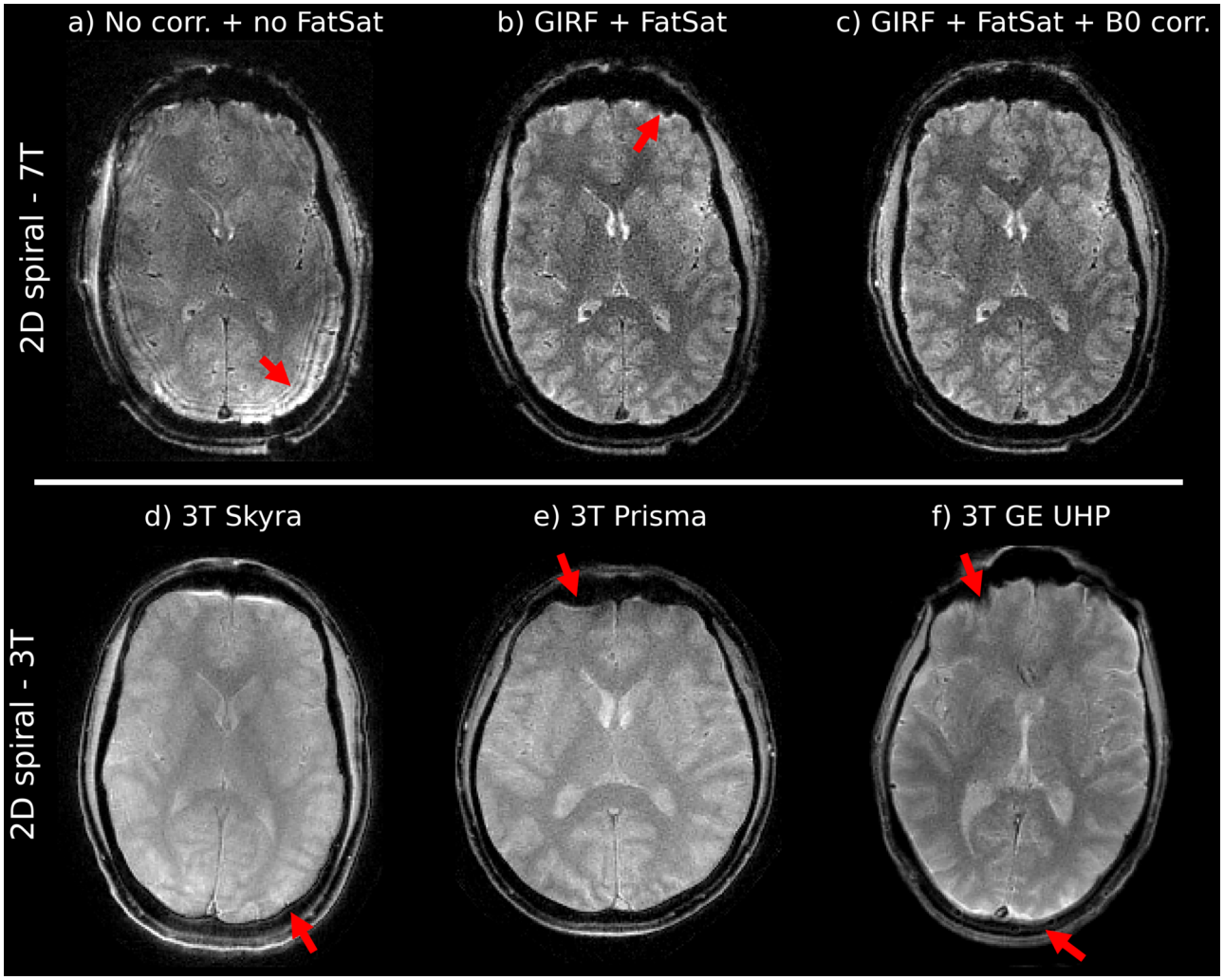
MRM_29384_fig4.tif



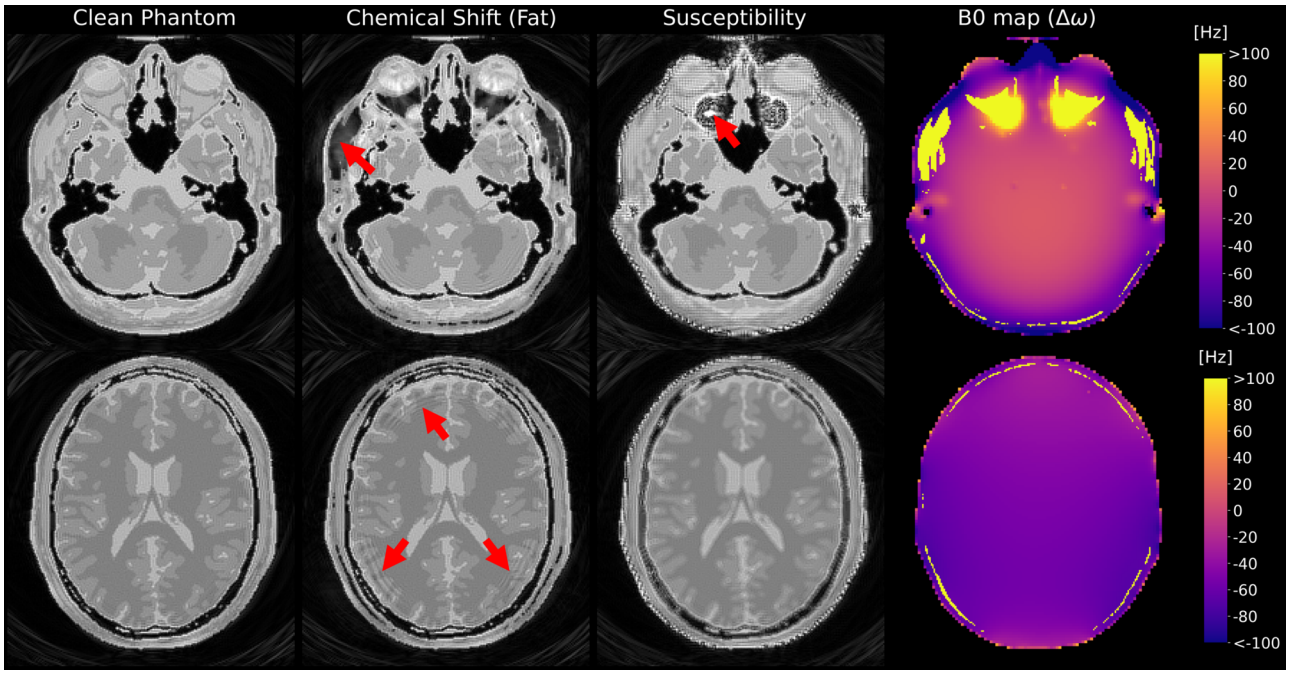
MRM_29384_fig5.tif



MRM_29384_fig6.tif



MRM_29384_fig7.tif



MRM_29384_fig8.tif