# Monalisa: An Open Source, Documented, User-Friendly MATLAB Toolbox for Magnetic Resonance Imaging Reconstruction

Mauro Leidi[1,2,3,4*†], Yiwei Jia[1,2,3†], Dominik Helbing[1,2†], Jaime Barranco[1,2,5,6†], Berk Can Açikgöz[3,7,8†], Eva Peper[7,8†], Jean-Baptiste Ledoux[5,6†], Jessica A.M. Bastiaansen[7,8†], Benedetta Franceschiello[1,2,6†], Bastien Milani[1,2†]

[1]Institute of Systems Engineering, School of Engineering, HES-SO Valais-Wallis, Rue de l'industrie 23, Sion, Switzerland.
[2]The Sense Innovation and Research Center, Lausanne and Sion, Switzerland.
[3]Graduate School for Cellular and Biomedical Sciences, University of Bern, Bern, Switzerland.
[4]Department of Data Analysis, Ghent University, Ghent, Belgium.
[5]CIBM Center for Biomedical Imaging, Lausanne, Switzerland.
[6]Department of Diagnostic and Interventional Radiology, Lausanne University Hospital (CHUV) and University of Lausanne (UNIL), Lausanne, Switzerland.
[7]Department of Diagnostic, Interventional and Pediatric Radiology (DIPR), Swiss Institute for Translational and Entrepreneurial Medicine, Bern, Switzerland.
[8]Translational Imaging Center (TIC), University of Bern, Bern, Switzerland.

*Corresponding author(s). E-mail(s): mauro.leidi@hevs.ch;
Contributing authors: yiwei.jia@hevs.ch; dominik.helbing@epfl.ch; jaime.barrancohernandez@hevs.ch; berk.acikgoez@students.unibe.ch; evaspeper@gmail.com; jean-baptiste.ledoux@chuv.ch; jessica.bastiaansen@unibe.ch; benedetta.franceschiello@hevs.ch; bastien.milani@hevs.ch;
†All authors contributed equally to this work.

**Abstract**

Magnetic Resonance Imaging (MRI) is a pivotal tool in modern diagnostics and research, yet image reconstruction is often used by clinicians and researchers as a black box. In this work, we introduce Monalisa, an open-source, user-friendly MATLAB framework designed to simplify the MRI reconstruction process. Monalisa decomposes the reconstruction pipeline into clear modular steps, including raw data reading with flexible file-type abstraction, trajectory computation, density compensation, advanced coil sensitivity mapping, and tailored binning strategies through its "mitosius" preprocessing stage. The framework supports a suite of reconstruction methods, including iterative-SENSE (also named CG-SENSE), GeneRalized Autocalibrating Partial Parallel Acquisition (GRAPPA) reconstructions, and Compressed Sensing (CS) supporting both spatial and temporal regularization using $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$ techniques, accommodating both Cartesian and non-Cartesian acquisitions. Benchmark experiments comparing Monalisa with established frameworks such as Berkeley Advanced Reconstruction Toolbox (BART) on simulated 2D radial acquisitions demonstrate competitive performance, with trade-offs observed between Structural Similarity Index (SSIM) and $\boldsymbol{l_2}$ error metrics. In some benchmark reconstructions, Monalisa produced images with fewer visible artifacts than BART, although these differences were not always reflected in SSIM or l2 distance metrics. By providing comprehensive documentation and modular design, Monalisa serves not only as a powerful tool for research and clinical imaging but also as an educational platform to facilitate innovation in MRI reconstruction.

**Keywords:** magnetic resonance imaging, image reconstruction, compressed sensing, iterative reconstruction, non-Cartesian imaging, dynamic MRI, open-source software

## Acronyms

*ADMM* Alternating Direction Method of Multipliers. 18, 19

*BART* Berkeley Advanced Reconstruction Toolbox. 1, 3, 4, 18–22

*CGD* Conjugate Gradient Descent. 16, 18, 19
*CS* Compressed Sensing. 1, 3, 4, 16, 22

*DL* Deep Learning. 3

*FFT* Fast Fourier Transform. 22
*FOV* Field of View. 6, 8, 18

*GRAPPA* GeneRalized Autocalibrating Partial Parallel Acquisition. 1, 3, 8, 14, 16, 22, 23

*HPC* High-Performance Computing. 13, 14

*MIP* Maximum Intensity Projection. 10, 11
*MRI* Magnetic Resonance Imaging. 1–5, 7, 9, 13, 18, 22, 27

*PI* Parallel Imaging. 3

*RMS* Root Mean Square. 10, 11
*ROI* Region of Interest. 8, 10, 11, 14, 19

*SENSE* Sensitivity encoding. 3, 8, 15, 16, 22
*SIGMA* Sensitivity Iterative Gradient-descent Mapping Algorithm. 8–12
*SNR* Signal-To-Noise Ratio. 8, 12
*SSIM* Structural Similarity Index. 1, 19–22

## 1 Introduction

MRI is a powerful, non-invasive, and versatile technique, considered a pillar in modern medical practice. It provides measurements that reflect properties of tissues, organs, and their pathologies [1], or changes in oxygen level in neural tissue, linked to neural activity, known as functional MRI [2, 3, 4]. MRI enables the investigation of a wide variety of physiological phenomena, based on the physics of nuclear magnetic resonance [1]: the properties of the nuclei are measured in the spatial-frequency domain, called k-space. Images can be created with contrasts reflecting tissue properties (determined by proton density, T1 and T2 relaxation times of the nuclei) or tissue susceptibility variations inducing magnetic field inhomogeneities. MRI measurements can also reflect biomechanical properties, electrical currents, oxygen levels, among other applications [1]. MRI reconstruction transforms measured k-space data into images that can be used for diagnostic or research purposes.

Acquisition and reconstruction strategies have evolved significantly to address the challenge of long acquisition times and motion sensitivity. A key approach to acceleration is partial sampling, which reduces the number of acquired data points but introduces aliasing artifacts that require correction during reconstruction [5]. While full Cartesian sampling offers uniform k-space coverage, it is often impractical at high spatial resolution due to long scan times and its sensitivity to motion [6]. Partial Cartesian sampling has been proposed [5], but this approach generates coherent aliasing artifacts that are difficult to correct. Alternatively, non-Cartesian schemes, such as radial acquisition schemes, introduce incoherent aliasing, which can be effectively mitigated using advanced reconstruction algorithms [7, 5]. Additionally, non-Cartesian sampling schemes—such as 2D golden-angle radial [8, 9, 10], 3D spiral phyllotaxis [11], and to some extent stack-of-stars [12] improve robustness to motion by repeatedly sampling the k-space center [13, 14, 15]. Both partial and non-Cartesian sampling come at the expense of more complex and slower reconstruction techniques. While these schemes are used and enable high-quality image

reconstructions, the theoretical framework underlying these methods is less fully established than for Cartesian sampling. In particular, the lack of a general sampling theorem does not allow to rigorously define key concepts such as a fully sampled k-space and effective spatial resolution. Moreover, the reconstruction of non-Cartesian or motion-resolved datasets is often not straightforward and can require substantial implementation effort. These challenges possibly limit broader adoption among researchers. K-space undersampling remains a challenge in MRI, especially when imaging moving organs such as the heart, lungs, and eyes. To address these challenges, several innovative computational techniques have been developed, each leveraging distinct aspects of the acquisition process:

- **Accelerated Parallel Imaging (PI):** SMASH [16] uses the spatial information provided by specific arrangement of surface coils to replace part of the phase encoding typically generated by magnetic field gradients, enabling partially parallel image acquisition. Techniques such as Sensitivity encoding (SENSE) [17], GRAPPA [18], and iterative-SENSE [19] exploit coil sensitivity profiles either explicitly (SENSE, iterative-SENSE), or implicitly (GRAPPA), to both shorten acquisition times and reduce undersampling artifacts. By sharing the spatial information provided by multiple receiver coils in order to fill in the missing data, these methods can reconstruct images from incomplete k-space data.
- **CS:** This approach is based on the premise that the image to be reconstructed is sparse in a certain domain (i.e., a vector basis) [20], and that the measurements are acquired in a randomized manner, leading to incoherent artifacts that resemble random noise. In contrast to PI, where information is shared among different coils, CS allows to share information between incompletely measured datasets and some prior knowledge. This allows to complete the missing data and enable high-quality reconstructions from reduced partially sampled datasets, thereby shortening scan times [21, 22, 5, 23, 24]. For example, a strong prior knowledge in CS reconstructions is that temporal neighboring frames in a cinematic image (CINE) must be very similar. In that case, CS can efficiently be used to share information between different frames in order to complete the unacquired information. However, integration of CS into clinical workflows is still limited by its high computational demands and prolonged reconstruction times [25].
- **Deep Learning (DL):** Recent advances in DL have transformed MRI reconstruction by learning mappings between undersampled k-space and fully reconstructed images. The strategy of DL reconstruction is to share information between the incompletely acquired data and the data acquired from other patients in order to complete the missing data. These "stronger together" methods have achieved significant improvements in both reconstruction speed and image quality, demonstrating notable potential for clinical applications [26, 27]. Most physics-based DL reconstruction methods developed so far rely on supervised learning, which requires high-quality datasets consisting of paired undersampled k-space data and corresponding ground truth images. These ground truth images are typically obtained by fully sampling and reconstructing the data. However, for high-dimensional imaging (4D and beyond), this is impractical, as acquiring even a single fully sampled MRI sequence can take several hours. As a result, training datasets for such high-dimensional data must be reconstructed using conventional non-DL methods. Recently, fully self-supervised approaches have emerged as a promising alternative [28], enabling networks to be trained with loss functions that do not rely on any reconstructed images, and demonstrating encouraging performance.

While the theoretical framework for reconstruction is rooted in well-established mathematical principles, the practical implementation of these processes often remains opaque. On the one hand, this is justified, as MRI remains a clinical tool used in everyday diagnostic processes. On the other hand, the transition from raw data to images at the MRI console is often perceived by imaging researchers as a "black box", with proprietary closed-source algorithms limiting accessibility. Recently, remarkable efforts have been made to open the "black box": several open source initiatives have emerged [29, 30, 31, 32, 33, 34, 35, 36, 37, 38], democratizing access to MRI reconstruction capabilities and fostering innovation across research and clinical settings. Among these, we highlight **BART**, a versatile toolkit supporting PI, non-Cartesian reconstruction, and machine learning integrations. BART provides an extensive suite of tools for advanced MRI reconstruction workflows and is widely adopted in the research community [29]. **MRIReco.jl**[30] is another open source toolkit written in Julia, reusing existing functionality from other efficient Julia packages and optimized for MRI-related operations. MRIReco.jl's modularity allows for efficient MRI reconstruction workflows. Reflecting this broader trend, vendors have begun to engage in similar efforts; for instance, Siemens Healthineers recently launched **OpenRecon** [39], a vendor-supported interface allowing integration directly at the scanner console of external software. Open-source reconstruction toolboxes still face limitations in usability and learnability. The inherent complexity of image reconstruction and the need for computationally efficient implementations force

the adoption of low-level programming languages and modular frameworks, making the code difficult to digest. Additionally, the rapid introduction of new features to BART often outpaces documentation updates, potentially making it difficult for novice developers seeking to learn or contribute. In this article, we aim to introduce **Monalisa**, a comprehensive, open source toolbox for MRI reconstruction. Monalisa supports both Cartesian and non-Cartesian sampling, it supports advanced techniques such as CS and iterative reconstructions. Monalisa is specifically designed to address the challenges of non-Cartesian and motion-resolved reconstruction workflows, with the goal of simplifying their implementation and making them more accessible. Monalisa is MATLAB-based, easy to install, well documented, making it a good educational tool for learning and teaching MRI reconstruction.

# 2 Methods

## 2.1 Notations and Definitions

To facilitate understanding, we introduce the following notation and definitions used throughout this work:

- **r**: is a vector whose components represent the spatial coordinates in the spatial domain. The spatial domain is the physical space, where the imaged object is located. These coordinates are defined with respect to the center of one voxel next to the center of the image, which serves as the origin for the reconstruction.
- **k**: is a vector whose components represent the coordinates of a spatial frequency in the k-space. These coordinates are defined with respect to the center k-space, which expresses spatial frequency of 0 (no oscillation in space). The reference axes in the spatial frequency domain are named $k_x$, $k_y$ and $k_z$.
- **Frame**: a frame represents a single snapshot of the physical space—whether in 2D or 3D—much like a photograph. However, instead of containing conventional pixel intensities, each frame is the list of complex values assigned to each voxel-center of a Cartesian grid. In Monalisa, these frames are stored as arrays.
- $\nu$ *or* **frDim**: the spatial dimension of a frame ($frDim = 2$ for 2D frames, $frDim = 3$ for 3D frames).
- **nFr**: the number of frames of a given image. For example, if the image is a sequence of 10 different temporal states, $nFr$ would be 10. This is important since the definition of an image can differ from common usage: a single image can contain multiple frames, making it more akin to a video in everyday language.
- **Image**: an image is a ordered group of $nFr$ complex-valued frames each of dimension $frDim$. In Monalisa, we store images using a cell array of size $nFr$, each cell containing a distinct frame.
- **frSize**: the frame size (commonly referred to as matrix size in the MRI community) defines the number of discrete samples along each dimension ($frDim$) within a given frame. For example, if we have $nFr = 5$ frames, each with $frDim = 3$ and $frSize = [120, 120, 120]$, it means that we are reconstructing five 3D complex images, each with $120 \times 120 \times 120$ voxels.
- **x**: any possible complex-valued anatomical image in the spatial domain.
- $\mathbf{x}^{true}$: the physical underlying complex-valued image. The ideal output if the measurements and reconstruction processes were perfect, assuming that the physical model is exact.
- $\mathbf{x}^{targ}$: the target complex-valued image. Since obtaining $x^{true}$ is not feasible for reasons explained in subsubsection 2.2.4, the reconstruction instead aims to recover an image $x^{targ}$ that closely resembles $x^{true}$ but is not identical to it.
- $\mathbf{x}^{\#}$: the complex-valued reconstructed image in the spatial domain. This is simply the output of the reconstruction algorithm. A successful reconstruction yields $x^{\#}$ as close as possible to $x^{targ}$.
- $\mathbf{x}_l$: the image obtained via the receiver coil number $l$, also named coil image number $l$, in the spatial domain for a given candidate image $x$.
- **y**: any possible measured data in k-space. We define $nSamp$ as the total number of measured complex values. The elements of $y$ are referred to as measurements or samples. The total number of k-space points along the acquisition trajectory is denoted as $nPt$. The number of coils or channels is represented by $nCh$. Hence, $nSamp$ is equal to $nCh$ times $nPt$. The sampled points are composed by readout lines, each containing $N$ single trajectory points. In Monalisa, we store raw data using arrays of size $[nCh, N, nLiness]$ or a permutation of it.
- **Y**: the data space, a complex vector space that contains all possible vectors y.
- $\mathbf{y}_0$: the raw data acquired by the coils during the MRI experiment. As part of the reconstruction preparation, $y_0$ can be modified by redistributing its samples among different bins and filtering out

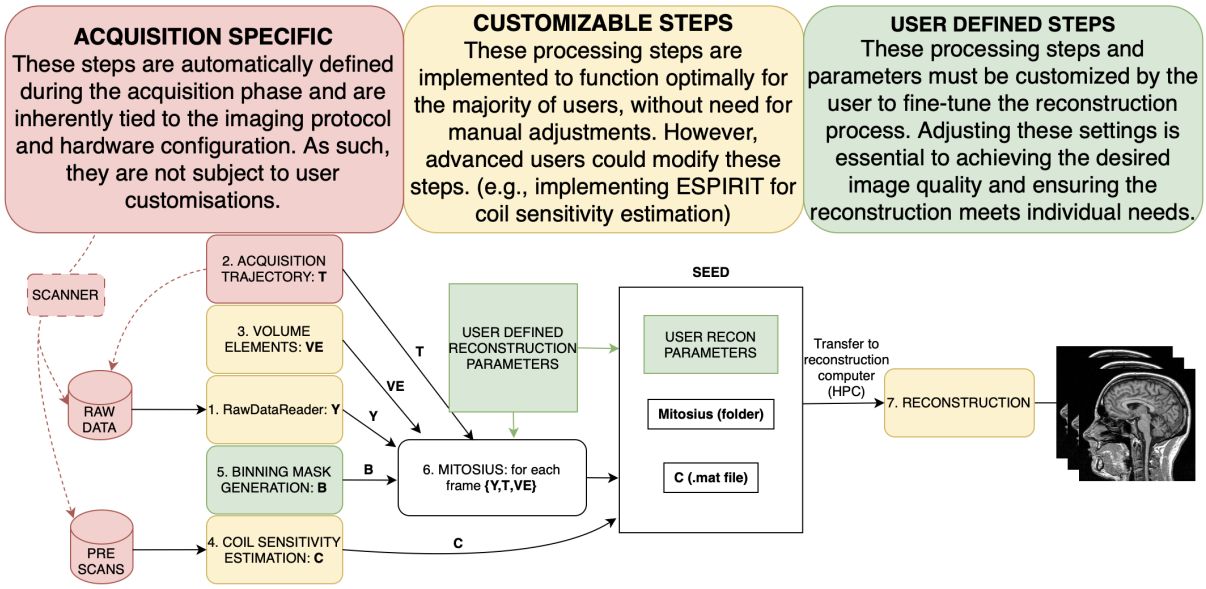



**Fig. 1**: Simplified schematic of the pipeline in Monalisa. Information from the acquisition raw data files is processed through multiple stages, together with additional information, such as the trajectory, volume elements, and user-defined binning and reconstruction parameters. Colored blocks indicate the degree of user control: red denotes acquisition-specific steps that are fixed and cannot be modified after acquisition; yellow highlights configurable steps that are rarely changed but can be customized or improved; green represents user-defined steps that must be adapted for each reconstruction. Initially, the workflow defines binning masks alongside coil sensitivity estimation and volume element computation. This is followed by the mitosius preparation, and finally, image reconstruction. Refer to Table 1 for a description of the reconstruction parameters.

certain readouts. Even after preparation, the data remains labeled as $y_0$ and is referred to as the reconstruction data.

- $\mathbf{C}_l$: : The (relative) coil sensitivity map for the $l$-th receiver coil. Its size at the time of estimation is $C_frSize$, while its dimension is $frDim$. At reconstruction time, $C_l$ is interpolated so that its size becomes equal to $frSize$.
- $\mathbf{C}$: The coil sensitivity map. In MRI theory $C$ is often defined as the matrix resulting from vertical concatenation of each single $C_l$. In Monalisa implementation, $C$ is estimated as an array of size $[C_frSize, nCh]$. Just before reconstruction, $C$ is interpolated to a single precision array of size $[frSize, nCh]$.
- $\mathbf{C}_l^{nrel}$: The non-relative coil sensitivity map of the $l$-th receiver coil. This is usually referred to as coil sensitivity in the literature. For a given image $x$, the associated coil images $x_l$ is given by $C_l^{nrel} \cdot x$.
- $\mathbf{t}$: The k-space complete acquisition trajectory. The size of the trajectory is $[frDim, nPt]$. As part of the reconstruction preparation, $t$ can be modified by redistributing its samples among different bins and filtering out certain readouts. Even after preparation, the trajectory is still referred to as $t$ and is referred to as the reconstruction trajectory.
- $\mathbf{B}_1, \ldots, \mathbf{B}_{nB}$: The user-defined binning masks. $nB$ is the number of binning masks, which is 1 or 2 for the reconstructions currently implemented in Monalisa. The size of binning mask number $b$ is $[nBin_b, nLines]$, where $nBin_b$ is the number of bins in binning mask number $b$.
- **nBins**: The total number of bins, or groups of data from which single frames are reconstructed. Since binning masks are combined using a Cartesian product, $nBins = nBin_1 \cdot \ldots \cdot nBin_{nB} = nFr$ .
- **Chain**: A one dimensional array of cells (i.e. a Matlab one-dimensional cell-array). Every cell can contain a multi-dimensional arrays, such as a frame, a trajectory or a raw-data array.
- **Sheet**: A two dimensional array of cells.

## 2.2 Monalisa workflow

Figure 1 illustrates the pipeline of Monalisa. Reconstructing raw data is a multi-step process combining trajectory definition, estimation of volume elements, coil sensitivity estimation, binning, mitosius preparation, and reconstruction. In this section, we describe all the steps that are necessary to transform raw k-space data into the desired reconstructed images.

### 2.2.1 Reading Raw Data

Prior to any reconstruction or analysis, k-space raw data must be parsed into MATLAB objects. Monalisa achieves raw data-type abstraction using the RawDataReader[1] interface class, an abstract base class that defines two key methods:

- **readMetaData**: this method is responsible for parsing header information and acquisition parameters from the raw data file. This includes extracting details such as the number of coils, acquisition lines, echo counts, and Field of View (FOV). Additionally, it computes other metrics, such as determining when steady state is reached by analyzing signal statistics.
- **readRawData**: this second method handles the extraction of the actual k-space data from the raw file. This method not only reads the data but also reshapes it into the expected MATLAB array format (typically organizing dimensions such as coils, readout points, number of acquisition lines). Moreover, depending on the provided flags, it can optionally filter out unwanted samples (e.g., non-steady-state lines)

Together, these two methods encapsulate the specifics of the data format, allowing the remainder of the Monalisa framework to interact with a uniform MATLAB object interface regardless of the underlying raw data file structure.

Monalisa implements two subclasses that override these methods: one for Siemens raw data files[2] and another for the ISMRM raw data format[3][40]. For the implementation of our Siemens rawDataReader, we rely on mapVBVD[4] originally developed by Philipp Ehses. Users working with a different file format must either convert their data to the ISMRM raw data format, or implement a new subclass that inherits from the base class.

### 2.2.2 Trajectory

The raw data does not contain information about k-space trajectories, it solely contain nSamp k-space complex-valued measurements. Therefore it is crucial for the reconstruction process to know the precise spatial k-space location at which each measurement was acquired. In Monalisa, the trajectory is represented as a double-precision array **t** of size [frDim, nPt]; hence, each column encodes the k-space coordinates for one measurement, and each row encodes the k-space coordinates along one dimension. Our convention is that every acquisition trajectory must be defined in physical units derived from the scanner's true input FOV[5]. In practice, this means that the k-space coordinates are expressed in physical units (e.g., $\mathrm{mm}^{-1}$), ensuring that the step size in each direction is directly linked to the intended FOV. For instance, if your acquisition is designed for a FOV of [200, 300] mm, the step size along the $k_x$ axis must be 1/200 (in $\mathrm{mm}^{-1}$) and along the $k_y$ axis 1/300 (in $\mathrm{mm}^{-1}$). Similarly, for a radial trajectory targeting a FOV of [360, 360] mm, the spacing between consecutive points along each radial spoke must be 1/360 (in $\mathrm{mm}^{-1}$). If you have already calculated your trajectory following a different convention, apply the appropriate scaling factors to convert the coordinates, following the guidelines in our documentation[6].

Monalisa provides a selector function, `t = bmTraj(mriAcquisitionNode)`, which offers several predefined k-space trajectories, including one that extracts trajectory coordinates directly from a PulSeq .seq file [41]. However, in practice, the exact sampling points are often implementation-dependent, even when the same nominal trajectory is used, different sequence implementations can yield slightly different coordinate sets. As a result, it is essential to verify that the trajectory being used during reconstruction precisely matches the one intended during acquisition.

[1] See implementation here
[2] See implementation here
[3] See implementation here
[4] mapVBVD. As stated in our license, these functions are not part of the toolbox and are located in the subfolder /monalisa/third_part/twix_for_monalisa/.
[5] Be aware that in Siemens scanners, if you input a given FOV in the readout direction, the true FOV will be twice the input value entered.
[6] See: How to use custom trajectory

### 2.2.3 Estimation of Volume Elements

Volume elements serve as a weighting mechanism that is essential for non-Cartesian MRI. The following qualitative explanation is intended to help the reader understand why these volume elements are useful. Imagine we want to compute the integral of a (well-behaved) integrable function $f$ over a 3D domain (e.g., 3D k-space):

$$\int_{\mathbb{R}^3} f(\mathbf{k})\, d^3\mathbf{k}.$$

In practice, on a computer, we must discretize this integral. If we know $f$ at a set of locations $\{\mathbf{k}_p\}_{p=1}^N$, we can approximate the integral as:

$$\int_{\mathbb{R}^3} f(\mathbf{k})\, d^3\mathbf{k} \approx \sum_{p=1}^{N} \Delta k_p\, f(\mathbf{k}_p),$$

where $\Delta k_p$ is the volume element associated with $\mathbf{k}_p$. Now, consider that these sampling points $\{\mathbf{k}_p\}$ belong to a trajectory $\mathbf{t}$. If $\mathbf{t}$ is Cartesian, all points are equidistant and all the volume elements $\Delta k_p$ are identical. In this case, we have:

$$\int_{\mathbb{R}^3} f(\mathbf{k})\, d^3\mathbf{k} \approx \Delta k \sum_{p=1}^{N} f(\mathbf{k}_p),$$

so that the influence of the volume elements reduces to a constant factor $\Delta k$, which in fact can be ignored in MRI reconstructions after normalization. In contrast, for a non-Cartesian trajectory $\mathbf{t}$, the volume element $\Delta k_p$ varies with the local point density. For example, radial trajectories oversample the k-space center, and for this reason the $\Delta k_p$ associated with those central points are lower. This non-uniformity requires each sample's contribution to be weighted appropriately by its individual volume element. Failing to account for this variation can lead to artifacts in the reconstructed image and may compromise the convergence of iterative reconstruction algorithms. Intuitively, given a finite list of sampling points (i.e., the trajectory $\mathbf{t}$), the volume element assigned to each point represents the volume "occupied" by that point relative to its neighbors. The local density is defined as the number of sampling points per unit volume, typically averaged over a small neighborhood (for example, within a spherical region). From this perspective, the local density at a given point is essentially the inverse of its associated volume element:

$$\text{Density at a point} \propto \frac{1}{\Delta k_p}$$

That is the reason why, historically, this concept is referred to as "density compensation" in MRI, originating from the transition from uniform trajectories, where the density is constant and can be neglected, to non-uniform trajectories. Although non-uniform density was once viewed as a problem to be "compensated", it is in fact the general case, with uniform sampling being a special scenario. We leave some references explaining the exact reason why the volume elements appear in different types of reconstruction for any interested reader (gridded reconstructions: [42], iterative reconstructions: [43]).

In Monalisa, volume elements are typically estimated as the volumes of the Voronoi cells associated with the trajectory points, where each Voronoi cell is defined as the region of space closer to a given point than to any other[44]. It is important to note that these volume elements depend on the binning (see subsubsection 2.2.5)—since the density may change depending on which measurements are selected for a given frame—and are stored in a double-precision array **ve** of size $[1, nPt]$. No additional rescaling is applied, so the volume elements represent the absolute volume of the Voronoi cells.

In Monalisa, the function implemented to compute the volume elements of any reconstruction trajectory is `bmVolumeElement`. It is invoked using the following syntax:

```
ve = bmVolumeElement(trajectory, type_of_trajectory, optional_arguments)
```

This function acts as a selector based on the specified trajectory type. Users are encouraged to visit our documentation[7] to determine which trajectory type best suits their specific case. For any radial trajectory, we suggest using Voronoi based methods. Regardless of the adopted method, the user must define an upper bound `ve_max` for the volume elements, in order to avoid convergence issues in iterative

[7] See: Volume Element Computation

reconstructions. As a rule of thumb, you can use `ve_max` to be a small multiple of the volume of a cell of the Cartesian k-space grid used for regridding (between 3 and 10). As an example,

$$\text{for 2D frames}: \texttt{ve_max} = 10 \cdot \texttt{dKu_x} \cdot \texttt{dKu_y}$$
$$\text{for 3D frames}: \texttt{ve_max} = 10 \cdot \texttt{dKu_x} \cdot \texttt{dKu_y} \cdot \texttt{dKu_z}$$

### 2.2.4 Coil Sensitivity Map Estimation

Parallel imaging employs arrays of many small receiver coils placed in close proximity to the tissues of interest. This close placement enhances the signal intensity detected by the coils, resulting in a high Signal-To-Noise Ratio (SNR) in the final image. However, this high SNR comes at a cost, which can be understood as follows. During acquisition, each region of tissue containing resonating nuclei acts as a source of electromagnetic waves that propagate through space, inducing a time-varying voltage in each coil according to the law of induction. The voltage measured by a given coil is the sum of contributions from all emitting sources, with each source's contribution depending on its spatial location with respect to each part of the coil. This spatial dependence can be characterized by a complex-valued function, referred to as "coil sensitivity", which describes how a given tissue region influences the voltage induced in a particular coil. Since coils for parallel imaging are small and very close to tissues, their spatial dependence is high, which means that the coil sensitivity of each individual proximity coil is spatially inhomogeneous. Moreover, the coil sensitivity of different coils is very different. Conceptually, we can say that each coil "sees" a very different image. The cost of using many surface coils resides therefore in the reconstruction challenge of combining these coil images into a single high-SNR image, a relatively complex process that requires accurate estimation of coil sensitivities. Coil sensitivity estimation can be either explicit or implicit, depending on the reconstruction method. One of the most widely used approach in the clinic is implicit estimation as part of GRAPPA reconstructions, but all iterative reconstructions used in research need an explicit estimation, which often involves acquiring an additional calibration scan—a low-resolution, full-field-of-view image, used to compute coil sensitivities. This section presents *Sensitivity Iterative Gradient-descent Mapping Algorithm (SIGMA)*, a method for explicitly estimating coil sensitivities. Although this method was previously evaluated on patient data [45], the theoretical foundations and implementation details were not disclosed. In this work, we present a comprehensive development of both the underlying formalism and its practical implementation. *SIGMA* builds upon key ideas from an early proposal by Pruessmann [17] and shares some concepts with JSENSE [46].

We begin by detailing the coil sensitivity estimation method proposed in [17], as it closely resembles our approach. Given the ground truth image $x^{true}$, the relationship between its coil images number $l$, $x_l^{true}$, and the the coil sensitivity number $l$, $C_l^{nrel}$, is a point-wise multiplication at each spatial location:

$$x_l^{true}(\mathbf{r}) \coloneqq C_l^{nrel}(\mathbf{r}) \cdot x^{true}(\mathbf{r}),$$

where $C_l^{nrel}(\mathbf{r})$ represents the coil sensitivity of coil $l$ evaluated in position $\mathbf{r}$ and is a complex number. A higher magnitude of $C_l^{nrel}(\mathbf{r})$ indicates a greater sensitivity of coil $l$ at location $\mathbf{r}$.

In [17], the first step in coil sensitivity estimation involves acquiring and reconstructing full-FOV images for each surface coil, denoted as $x_l^{\text{true}}$, as well as an image from the body coil, denoted as $x_{\text{body}}^{\text{true}}$. To perform an estimation of $C_l^{nrel}$, in [17] $x_l^{true}$ is then divided by $x_{body}^{true}$. While not stated in the study, this method implicitly assumes that the sensitivity of the body coil is constant over space (homogeneous) in the Region of Interest (ROI). In fact, if that assumption is true, the body-coil image is given by

$$x_{body}^{true}(\mathbf{r}) = C_{body}^{nrel} \cdot x^{true}(\mathbf{r}).$$

Note that here $C_{body}^{nrel}$ is independent of $r$. It holds then

$$\frac{x_l^{true}(\mathbf{r})}{x_{body}^{true}(\mathbf{r})} = \frac{C_l^{nrel}(\mathbf{r}) \cdot x^{true}(\mathbf{r})}{C_{body}^{nrel} \cdot x^{true}(\mathbf{r})} = s \cdot C_l^{nrel}(\mathbf{r}).$$

so that $C_l^{nrel}(\mathbf{r})$ can be estimated by $x_l^{true}(\mathbf{r})$ and $x_{body}^{true}(\mathbf{r})$ up to a scalar factor $s$. As a second step, to prevent the noise present in the images from altering the sensitivity estimation, [17] applies smoothing via polynomial fitting on a binary mask applied to the previously computed $\frac{x_l^{true}(\mathbf{r})}{x_{body}^{true}(\mathbf{r})}$. To create this binary mask, a threshold is first manually computed to split noise from signal-containing region, where

voxels with value smaller than the threshold are considered noise. Additionally, residual voxels that do have a sparse neighborhood are also considered noise. In a final step, an extrapolation zone is computed using a region growing around the mask. All voxels outside the mask and the extrapolation zone are assumed to be zero-valued, and polynomial fitting yields the final sensitivity estimates. We interpret this extrapolation zone as a heuristic designed to extend the sensitivity estimation slightly beyond the mask, to avoid abrupt changes in sensitivity at the edges. This helps reduce the oscillations often observed near discontinuities, particularly in polynomial fitting methods, which might lead to numerical inaccuracy or instability [47].

Our method SIGMA differs from the method of [17] primarily in four ways:

- We take advantage of the presence of two pairs of body-coils, instead of only using one. In conventional MRI scanners, there are usually $nBody = 2$ body coils.
- We achieve smoothing by a local voxel averaging, meaning that the value of a single voxels are replaced with the average of their value and those of all the adjacent voxels, instead of polynomial fitting.
- We extend the coil sensitivity estimate in regions without signal by solving a Laplace boundary value problem, instead of using the extrapolation zone method.
- We estimate the relative coil sensitivity maps, instead of non-relative coil sensitivity maps. By "relative" or "biased" coil sensitivity of channel number $l$ we mean the coil sensitivity defined by

$$C_\ell := \frac{C_\ell^{nrel}(\mathbf{r})}{C_{bias}(\mathbf{r})}$$

where $C_{bias}(\mathbf{r})$ is any complex valued function of space that makes $C_\ell$ estimable in practice. This choice of estimating a relative coil sensitivity instead of the true non-relative coil sensitivity allows us to relax the assumption done in [17], which was to consider $C_{body}^{nrel}$ homogeneous in space. We will choose $C_{bias}(\mathbf{r})$ proportional to the complex phase of one of the two body-coil sensitivities. Independently from our choice of $C_{bias}(\mathbf{r})$, the reconstructed image using the maps $C_\ell$ as coil sensitivities is the image $x^{targ}$ given by

$$x^{targ}(\mathbf{r}) := C_{bias}(\mathbf{r}) \cdot x^{true}(\mathbf{r})$$

This follows from the signal equation [8]

$$\begin{aligned} y_\ell(\mathbf{k}) &= \int_{\mathbb{R}^3} C_\ell^{nrel}(\mathbf{r})\, x^{true}(\mathbf{r})\, e^{-2\pi i\, \mathbf{k}\cdot\mathbf{r}}\, d\mathbf{r} = \int_{\mathbb{R}^3} \frac{C_\ell^{nrel}(\mathbf{r})}{C_{bias}(\mathbf{r})} \cdot C_{bias}(\mathbf{r}) \cdot x^{true}(\mathbf{r})\, e^{-2\pi i\, \mathbf{k}\cdot\mathbf{r}} \cdot d\mathbf{r} \\ &= \int_{\mathbb{R}^3} C_\ell(\mathbf{r}) \cdot x^{targ}(\mathbf{r})\, e^{-2\pi i\, \mathbf{k}\cdot\mathbf{r}}\, d\mathbf{r} \end{aligned}$$

We are now ready to present the formalism and implementation of SIGMA. In order to estimate the maps $C_\ell$ in a region where resonating nuclei are located (i.e. any signal-emitting spatial location during the MR experiment), we need the following definitions:

- the coil image number $l$ of $x^{targ}$

$$x_l^{targ}(\mathbf{r}) := C_l(\mathbf{r}) \cdot x^{targ}(\mathbf{r}) = \frac{C_\ell^{nrel}(\mathbf{r})}{C_{bias}(\mathbf{r})} \cdot C_{bias}(\mathbf{r}) \cdot\ x^{true}(\mathbf{r}) = x_l^{true}(\mathbf{r})$$

so that $x_l^{targ}$ and $x_l^{true}$ coincide, in contrast to $x^{targ}$ and $x^{true}$.
- The root-mean-squared body-coil image:

$$\mathrm{RMS}_{\mathrm{body}}(\mathbf{r}) := \sqrt{\frac{1}{nBody} \cdot \sum_{i=1}^{nBody} |x_{body,i}^{true}(\mathbf{r})|^2}$$

- The reference coil sensitivity:

$$C_{ref}(\mathbf{r}) := \frac{|x_{body,1}^{true}(\mathbf{r})|}{\mathrm{RMS}_{\mathrm{body}}(\mathbf{r})}$$

- The reference image:

$$x_{ref}(\mathbf{r}) := \frac{x^{true}_{body,1}(\mathbf{r})}{C_{ref}(\mathbf{r})}$$

These three quantities can be estimated by only using coil images. We now present the following claim:

**Claim 1.** *If it holds the approximation*

$$\mathrm{RMS}_{\mathrm{body}}(\mathbf{r}) \approx s_2 \cdot |x^{true}|$$

*for some constant $s_2$, and if we define*

$$C_{bias}(\mathbf{r}) := s_2 \cdot exp\left(i\phi\left(C^{nrel}_{body,1}\right)\right)$$

*using $\phi(z)$ as the phase of the complex number $z$, and that scalar value $s_2$, then it holds the approximation*

$$C_\ell(\mathbf{r}) \approx \frac{x^{true}_l}{x_{ref}}$$

*Proof.* From the definitions and the assumption in **claim 1**, it follows that

$$x_{ref}(\mathbf{r}) \;=\; \frac{x^{true}_{\mathrm{body},1}(\mathbf{r})}{C_{\mathrm{ref}}(\mathbf{r})} = \frac{\mathrm{RMS}_{\mathrm{body}}(\mathbf{r}) \cdot x^{true}_{body,1}(\mathbf{r})}{|x^{true}_{body,1}(\mathbf{r})|} = \mathrm{RMS}_{\mathrm{body}}(\mathbf{r}) \cdot \exp\left(i\phi(C^{nrel}_{body,1}(\mathbf{r}))\right) \cdot \frac{x^{true}(\mathbf{r})}{|x^{true}(\mathbf{r})|}$$

where $\phi(z)$ is the phase of the complex number $z$. Therefore

$$\frac{x^{true}_l(\mathbf{r})}{x_{\mathrm{ref}}(\mathbf{r})} = \frac{C^{nrel}_l(\mathbf{r}) \cdot x^{true}(\mathbf{r})}{\mathrm{RMS}_{\mathrm{body}}(\mathbf{r}) \exp\left(i\phi(C^{nrel}_{body,1}(\mathbf{r}))\right)} \cdot \frac{|x^{true}(\mathbf{r})|}{x^{true}(\mathbf{r})} \approx \frac{C^{nrel}_l(\mathbf{r})}{s_2 \cdot \exp\left(i\phi(C^{nrel}_{body,1}(\mathbf{r}))\right)} = \frac{C^{nrel}_l(\mathbf{r})}{C_{bias}(\mathbf{r})} = C_\ell(\mathbf{r})$$

□

Since $x^{true}_l$ is easy to estimate, we can by **claim 1**, under the assumption that the Root Mean Square (RMS) of the body coil is proportional to the magnitude of $x^{true}$, estimate $C_\ell$ in all locations where the resonating nuclei generate a signal significantly larger than the noise. Our assumption is weaker than assuming the existence of a single spatially homogeneous body coil, since we are only assuming we can estimate the magnitude of such a homogeneous coil, which in our case is a virtual coil that is obtained by combining both body coils.

We can now formulate our SIGMA method as the following steps:

- We estimate a binary mask that indicates which spatial voxels contain anatomical regions. Only the voxels that are within this estimated area will be used to estimate the coil sensitivity. This is justified, as the image outside the anatomical region should in theory be zero. The binary mask is generated through a thresholding process applied to both the Maximum Intensity Projection (MIP) [48] and RMS [49] images. Both MIP and RMS are methods to combine multiple surface coil images into a single representation. MIP takes the maximum intensity value across multiple-coil images, emphasizing bright structures and enhancing contrast-noise ratio [50]. RMS uses a sum of squares combination, averaging out noise fluctuations [51].
  We implemented both manual and automated mask extraction. In manual mode, the user first visually adjusts a threshold on the MIP image, denoted as *mipthrs*, and on the RMS image denoted as *rmsthrs*. Additionally, a bounding box around the ROI is manually adjusted. In automatic mode, a threshold detection algorithm is implemented by fitting a bimodal Gaussian distribution to the histogram of pixel intensities for both images (RMS and MIP). The two Gaussian components are interpreted as representing the noise and signal distributions, respectively. The threshold is then set at the intensity value where the probability of the pixel belonging to the noise distribution equals the probability of it belonging to the signal distribution. Any pixel with value lower than the threshold is excluded from the mask. The automatic bounding box extraction works by first reshaping the 3D image and computing summed projections along each dimension. These projections are then normalized and binarized using Otsu's method [52], allowing to isolate the largest connected component in each 2D projection. The bounding boxes for these components are determined and then merged—with added padding—to generate the final bounding box around the region of interest.

For manual and automatic modes, the mask is then created by including only those voxels where the MIP value exceeds *mipthrs*, the RMS value exceeds *rmsthrs*, and that lie within the bounding box.

- We estimate the maps $\mathrm{RMS}_{\mathrm{body}}$, $C_{ref}$ and $x_{ref}$ following the previous definition. We then estimate each relative coil sensitivity $C_\ell$ by **Claim 1** using the associated assumption. This leads to a first estimate of the coil sensitivities inside the mask.
- In order to mitigate the effect of noise on the estimation of each $C_\ell$, we average the value of each voxel with the values of its neighbors, without considering voxels outside the mask to avoid issues at the boundary of the mask. This process achieves smoothing of the estimated values within the mask. For clarity we denote the smooth coil sensitivities as:

$$\begin{cases} sC_\ell = 0 & \text{where } mask = 0, \\ sC_\ell = LocalSmoothing(\frac{x_l^{true}(\mathbf{r})}{x_{\mathrm{ref}}(\mathbf{r})}) & \text{where } mask = 1, \end{cases}$$

- Finally, to avoid abrupt changes in estimated coil sensitivity at the edges of the ROI, we extend the estimation outside the mask solving, via the Jacobi method [53], the Laplace boundary value problem. We denote the extended smooth coil sensitivity as $esC_\ell$, obtained by solving two independent Laplace equations for the real and imaginary components:

$$\begin{cases} \Delta\,\mathrm{Re}(esC_\ell) = 0 & \text{where } mask = 0, \\ \mathrm{Re}(esC_\ell) = \mathrm{Re}(sC_\ell) & \text{where } mask = 1, \end{cases} \quad \begin{cases} \Delta\,\mathrm{Im}(esC_\ell) = 0 & \text{where } mask = 0, \\ \mathrm{Im}(esC_\ell) = \mathrm{Im}(sC_\ell) & \text{where } mask = 1. \end{cases}$$

  Here, $\Delta$ is the discrete Laplacian operator, and the smooth sensitivity values $sC_\ell$ inside the mask serve as Dirichlet boundary conditions.

Monalisa offers two implementation approaches for *SIGMA*: (1) using body coil calibration images and (2) using calibration raw data from which the corresponding images can also be reconstructed. If pre-scan raw data are unavailable, the coil sensitivity estimation concludes here. Otherwise, inspired by [46], we further refine the coil sensitivity maps, $C$, using a heuristic alternating gradient descent approach. This refinement alternates between optimizing the reconstructed image $\mathbf{x}$ and the coil sensitivities $C$, progressively improving the estimated sensitivity maps.

The alternating gradient descent method heuristically minimizes the following objective function:

$$\mathcal{L}(C, \mathbf{x}) = \|\mathcal{F}\ DIAG(\mathbf{x})\ C - \mathbf{y}\|_{Y,2}^2,$$

where $\mathcal{F}$ is the spatial Fourier transform, and $DIAG(\mathbf{x})$ is a diagonal matrix containing the values of $\mathbf{x}$ replicated $nCh$ times as follows:

$$DIAG(\mathbf{x}) = \underbrace{\begin{bmatrix} diag(\mathbf{x}) & & \\ & \ddots & \\ & & diag(\mathbf{x}) \end{bmatrix}}_{nCh\ Blocks}$$

with each block $diag(\mathbf{x})$ being the diagonal matrix listing the values of $\mathbf{x}$. Function $\mathcal{L}$ ensures that the forward model—comprising the reconstructed image, coil sensitivity maps, and Fourier transform—matches the acquired $k$-space data $\mathbf{y}$. The optimization alternates between the following steps:

1. **Keep $C$ constant and optimize $\mathbf{x}$:**

$$\mathbf{x}^{(k+1)} = \underset{x}{\operatorname{argmin}}\ \mathcal{L}(C^{(k)}, \mathbf{x})$$

   This step refines the reconstructed image using the current coil sensitivity maps.
2. **Keep $\mathbf{x}$ constant and optimize $C$:**

$$C^{(k+1)} = \underset{C}{\operatorname{argmin}}\ \mathcal{L}(C, \mathbf{x}^{(k+1)})$$

   This step updates the coil sensitivity maps based on the current reconstructed image.

To prevent divergence of the two optimized variables (e.g., $x \to x^{\#}/\alpha$ and $C \to \alpha C^{\#}$ for some positive scalar $\alpha \to +\infty$), the body coil sensitivity is concatenated with the surface coil sensitivities as an additional channel and is held constant during the optimization process, ensuring that only the coil sensitivities of the surface coils are updated, while the body coil serves as a fixed reference. The concatenated body coil acts as a reference, stabilizing the scale and preventing divergence. To ensure an equilibrated weighting of each surface coil sensitivity and the body coil sensitivity in the iterative refinement, a normalization step, acting on all the images, is performed at the start of the estimation process.

In Monalisa, several functions are provided to estimate coil sensitivity maps, and our documentation offers detailed guidance on how to use these tools[8]. Among these, the mlComputeCoilSensitivity function offers a particularly simple interface for estimating coil sensitivity maps:

```
C = mlComputeCoilSensitivity(BCreader, HCreader, optional_arguments)
```

The sensitivity estimation produce a MATLAB array of single-precision values with dimensions $[C_frSize, nCh]$. Here, $nCh$ represents the number of receiver-coils, each coil yielding its own sensitivity map, and $C_frSize$ is the frame size of the sensitivity maps during the estimation of $C$, which we usually choose to be much smaller than the image frame size. By estimating $C$ at a lower spatial resolution, we reduce the adaptation of the estimated sensitivities to noise in the pre-scan data. This strategy is effective because larger voxels (resulting from lower spatial resolution) contain more signal, thereby increasing the SNR. A common choice is $C_frSize = 64$ for robust estimation. This low resolution approach also shortens the computation time for the coil sensitivity estimation. At reconstruction time, the coil sensitivity will be interpolated to the size $[frSize, nCh]$, in order to match the image frame size.

For completeness, we provide the pseudocode of our coil sensitivity estimation algorithm as follows:

**Algorithm 1** *SIGMA* Coil Sensitivity Estimation

**Require:** Input images: `ImagesBody`, `ImageSurface`; `mode` (*manual/automatic*); flag
**Ensure:** Final coil sensitivity maps $C$

**// Step 1: Normalize Input Images**
(ImagesBody, ImageSurface) ← `normalize_images(ImagesBody, ImageSurface)`

**// Step 2: Compute ROI Mask**
ROImask ← `compute_mask(ImagesBody, mode)`

**// Step 3: Compute RMS Image for Body Coils**
rms_body ← `compute_rms(ImagesBody)`

**// Step 4: Compute Initial Reference Sensitivity Map**
$C_{\text{ref}} \leftarrow \frac{|\text{ImagesBody}(:,:,1)|}{\text{rms_body}}$

**// Step 5: Smooth and Extend $C_{\text{ref}}$ using Laplace Boundary Equation Solver**
$C_{\text{ref}}$ ← `laplace_solver(pseudo_diffusion(`$C_{\text{ref}}$, ROImask`))`

**// Step 6: Compute Initial Surface Coil Sensitivity Maps**
$C \leftarrow \frac{\text{ImagesSurface}}{\text{ImagesBody}(:,:,1)}$

**// Step 7: Smooth and Extend $C$ using Laplacian Solver**
$C$ ← `laplace_solver(pseudo_diffusion(`$C$, ROImask`))`

**// Step 8: Iterative Refinement (if raw calibration data is provided)**
**if** `raw_calibration_data_available` **then**
  $C$ ← `iterative_refine(`$C$, `raw_calibration_data`)
**end if**

**return** $C$

[8] See: How to compute the coil sensitivity

### 2.2.5 Binning: Flexible Readout Partitioning

Binning is a crucial operation in data processing for MRI of moving organs, enabling the partitioning of readouts into multiple groups (or bins). These groups define which readouts contribute to the reconstruction of individual image frames. In addition to partitioning, binning often involves excluding certain readouts that do not meet specific criteria, such as non-steady-state measurements at the very beginning of the acquisition. For example, while acquiring raw MRI data of the eye with an external eye tracker device, we can use an eye tracker to exclude the k-space points that are affected by motion, or to group in different bins readouts acquired while the eyes was staring at different positions. Monalisa requires the user to define each of the binning masks $B_1, \ldots, B_{nB}$ as a logical array, where **nB** is the number of binning masks. Here, $\mathbf{B_b}$ denotes the binning mask number $b$ and is of size [$\mathbf{nBin_b}$, **nLines**] where:

- $\mathbf{nBin_b}$ is the number of partitions expressed in binning mask number $b$,
- **nLines** is the total number of lines in the acquisition trajectory, which do not need to be straight lines (e.g. spiral lines). Note that we mask entire k-space lines rather than individual points, as all points on a given line are acquired in a very small time interval.
- $\mathbf{B_b(i,j)}$ is a Boolean value (true or false), indicating whether the $j$-th line contributes to the $i$-th bin of binning mask number $b$.

Each binning mask usually encodes for a different non-spatial dimension. For this reason, if multiple binning masks are used, they are combined using a Cartesian product. As an illustrative example, consider the reconstruction of dynamic CINE images [54] using both cardiac and respiratory gating. Suppose we want to resolve six cardiac phases and eight respiratory phases. This setup effectively maps each readout in real time into two distinct non-spatial dimensions: one corresponding to its position within the cardiac cycle, and the other to its position within the respiratory cycle. In this case, two separate binning masks are required—one for the cardiac pseudo-time and one for the respiratory pseudo-time—so we have $nB = 2$. Let us denote the cardiac binning mask as $B_1$ and the respiratory binning mask as $B_2$. Then, $nBin_1 = 6$ and $nBin_2 = 8$, meaning that $B_1$ partitions the data into six cardiac bins, and $B_2$ partitions it into eight respiratory bins. By combining the two masks, the total number of reconstructed frames will be $nBins = nBin_1 \cdot nBin_2 = 48$, as each unique combination of cardiac and respiratory phase defines a single image frame.

It follows that the total number of resulting bins $nBins = nBin_1 \cdot \ldots \cdot nBin_{nB} = nFr$, because for each bin a corresponding frame will be reconstructed. Since the binning generation is experiment-specific, our documentation provides a few examples to guide the user in implementing their own binning masks[9].

### 2.2.6 The Seed and the Mitosius: A Standardized Data Structure to Store the Result of the Reconstruction Preparation

In Monalisa, reconstructions are performed in two main sequential steps: a preparation step, followed by the reconstruction itself. Users are free to design the preparation step however they wish, as long as its output includes all the arrays and parameters required to run the reconstruction function.

Nevertheless, this section gives an overview of how to use the standardized data structure defined for Monalisa, that we highly recommend. This structure consists of two folders: the "seed" and the "mitosius", as illustrated in Figure 1. The seed folder, whose name can be freely chosen by the user, serves as the main reconstruction data container. It contains all the necessary inputs to start the reconstruction process and all the resulting outputs are written within it. Typically, the seed is the folder transferred to an High-Performance Computing (HPC) system to perform high-dimensional reconstructions. The mitosius is a subfolder of the seed and its name cannot be changed, it is part of the standard. The mitosius folder organizes the data $y$, trajectory $t$ and volume elements $ve$ for each bin in a separate subfolder, referred to as a "cell". For single-frame reconstructions, the mitosius folder contains only one cell. For multi-frame reconstructions, the mitosius folder contains $nFr$ subfolders, each containing $y$, $t$, and $ve$ of a single bin.

The estimated coil sensitivity maps are not stored within mitosius. Instead, they must be placed in a subfolder named "C" within the seed. In addition, the binning masks are not required to include all the acquired data. When this occurs, the mitosius reduces the amount of data copied to the reconstruction algorithm. This is an advantage when working with HPC, since the data transfer can be time-consuming.

[9] See: How to bin data

Creating the seed and mitosius completes the preparation step. To create the seed, create a new folder with the name of your choice. Next, create the mitosius within the seed, to do that ensure first that you have access to:

- **Acquired raw data y**: It can be simply read from Siemens raw data or ISMRMRD files using Monalisa's readers as previously described in subsubsection 2.2.1. It contains the totality of the acquired raw data.
- **Acquisition Trajectory t**: Computed as previously described. If you did not compute it yet, you are now at the right step to compute it. It is the raw trajectory that was used during the acquisition. It therefore contains the totality of the trajectory points.
- **Coil sensitivity map C**: Computed as previously described in subsubsection 2.2.4.
- **Binning Masks $\mathbf{B_1}, \ldots, \mathbf{B_{nB}}$**: Computed as previously described in subsubsection 2.2.5. $nB$ is the number of binning masks.

For single-frame reconstructions, volume elements $ve$ for $t$ are computed as described in subsubsection 2.2.3. In case of multi-frame reconstruction, the user can run the "mitosis" function before evaluating the volume elements. This function builds the different data bins and trajectory bins according to the binning mask(s) given as input. The mitosis function is invoked using the following syntax:

$$[\text{y, t}] = \text{bmMitosis(y, t, } B_1, \ldots, B_{nB});$$

This function updates $y$ and $t$, converting them into MATLAB cell arrays in which each cell corresponds to a specific bin. Conceptually, it partitions the original data arrays into smaller units, that are named cells (this is the origin of the name mitosis). The mitosis function transforms the complete **acquired data** into the **reconstruction data**, and the complete **acquisition trajectory** into the **reconstruction trajectory**.

Once the mitosis is performed, volume elements for each cell are computed as described in subsubsection 2.2.3 using $t$ and stored in a corresponding cell-array $ve$. Note that each binning mask must be of size $[nBin_b, nLines]$ where $nBin_b$ is the number of bins in the binning mask number $b$ and $nLines$ is the number of total acquired lines. After the mitosis function, $y$ and $t$ are cell arrays of size $[nBin_1, ..., nBin_{nB}]$, where each cell contains an array with dimensions determined by the contents of the corresponding bin. The mitosis function supports any number of binning mask. However, Monalisa's reconstructions currently support only one-dimensional cell-arrays (that we call "chains", $nB = 1$) and two-dimensional cell-arrays (that we call "sheets", $nB = 2$).

At this point, it is recommended to perform a rescaling of the raw data $y$ by a positive value that we will call $v_{scale}$. To choose $v_{scale}$, we suggest that the user perform a preliminary gridded (zero-padded) reconstruction with the function "bmMathilda" if the data are non-Cartesian (or "Nasha_partialCartesian" for Cartesian data), display the magnitude image, draw a ROI, compute the average magnitude of the image in that ROI, and set $v_{scale}$ to be equal to such value. This operation ensures that the reconstructed image will have an average value close to 1 in the ROI selected during the rescaling process. We observe that if $y$ contains multiple cells, rescaling implies dividing the content of each cell by the same value $v_{scale}$. Additional information and instructions are available in our documentation[10].

Finally, permute each array $y\{i\}$ so that its size is $[nPt\{i\}, nCh]$, where $nPt\{i\}$ is the number of trajectory points selected by bin number $i$. By running the function bmCreateMitosius[11], the mitosius folder with its content will be saved on the disk: in each subfolder of the mitosius, you will find the arrays $y$, $t$ and $ve$.

This terminates the preparation step. You are now ready to reconstruct your images. If you are working with an HPC cluster, the only folder you need to transfer is the seed, which contains the mitosius, your coil sensitivity estimates and other reconstruction parameters that you can load just before the call of the reconstruction function in the script running on the HPC.

### 2.2.7 Reconstruction Methods

In this section, we provide a brief overview of the reconstruction methods available in Monalisa by presenting the corresponding reconstruction problems for each (except for GRAPPA reconstructions).

[10] See: Normalizing data
[11] Source code of bmCreateMitosius

Monalisa primarily supports non-Cartesian reconstructions, which rely on data gridding to handle non-uniform sampling of k-space. However, for many of these methods, an equivalent reconstruction is also implemented for Cartesian acquisitions, solving the same optimization problem without requiring a gridding step. These Cartesian versions can be identified by function names ending with the suffix "_partialCartesian". An exception to this naming pattern is the non-Cartesian method `bmMathilda`, whose Cartesian counterpart is called `bmNasha_partialCartesian`. Despite the suffix, methods ending in "_partialCartesian" are not restricted to undersampled data; they can also be applied to fully sampled Cartesian acquisitions.

In general, a reconstruction algorithm aims to find an image estimate $x^{\#}$ that best explains the measured k-space data $y_0$ given a forward model $A$, while also incorporating any available prior knowledge, if provided. This model $A$ is the composition of the (Cartesian or non-Cartesian) spatial Fourier transform $F$ and the coil sensitivities matrix $C$ (see subsubsection 2.2.4):

$$A = FC$$

This model maps any candidate image $x$ to the modeled data $Ax$. For non-regularized reconstructions, the goal is simply to minimize the distance between the measured data $y_0$ and the modeled data $Ax$, where the distance is usually defined in terms of $l_2$-norm [55]:

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2} \|Ax - y_0\|_{Y,2}^2.$$

Regularization is a way to incorporate prior knowledge. It is done by adding a regularization term $R(x)$ to the objective function [55]. This term is balanced with the data fidelity term by a weighting factor $\lambda$:

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2} \|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2} R(x).$$

A well-designed regularization function $R$ formalizes our prior knowledge by assigning low values to any image exhibiting the desired property. By including $R(x)$ in the objective function, the reconstruction then favors only those candidates $x$ that conform to what we already know about the true image, among all candidates for which the modeled data $Ax$ is close to $y_0$. For example, $R$ may be a measure of the simplicity of $x$, in which case the associated prior knowledge is that the solution $x^{\#}$ we are looking for is sparse in a given vector basis. The theory of compressed sensing [20, 56, 57] specifies some conditions under which $R$ can be written as a $l_1$-norm. Monalisa's reconstruction methods are the following:

- **A Zero-Padded, Gridded, Non-Iterative Reconstruction (`bmMathilda`)** for non-Cartesian sampling. It performs an approximative-inverse Fourier transform for each coil and, if the coil sensitivity matrix $C$ is provided, combines the resulting coil images with the pseudo-inverse of the coil sensitivity matrix. This approximative-inverse is a linear map that we will write $F^{\sim 1}$ so that

  $$FF^{\sim 1}y \approx y$$

  for any raw data vector $y$. The implementation of $F^{\sim 1}$ is the composition of:

  1. A gridding step, which is a discrete convolution finding an approximation of $y_0$ on the positions of a Cartesian grid [58, 42]. The value of any position on the Cartesian grid that is not sufficiently close to $t$ is set to 0 (zero-padding),
  2. A conventional inverse discrete Fourier transform (DFT),
  3. A deapodisation to cancel the filter resulting from the gridding operation.

  Writing $F^{\sim 1}y$ the list of coil images, multiplying it by the coil sensitivity pseudo-inverse leads the solution of

  $$x^{\#} = \underset{x}{\operatorname{argmin}} \|Cx - F^{\sim 1}y\|_2^2$$

  It is the result returned by Mathilda. It is by construction an approximate solution of the least-squares problem solved by Sensa (see hereafter) but it does not share the information between coils in order to fill the missing unsampled information, as it is done in SENSE and iterative-SENSE reconstructions. The Cartesian counterpart of Mathilda is Nasha_partialCartesian (implemented in `bmNasha_partialCartesian`).

- **A 2D and a 3D GRAPPA implementation (`bcaNeith2` and `bcaNeith3`)** for uniformly partially sampled Cartesian trajectories. Like Mathilda, it is a non-iterative single-frame reconstruction. However, it does not work directly with coil sensitivity maps. By its design, GRAPPA needs a calibration scan that consists of a few fully sampled lines acquired in the neighborhood of the k-space center. These few lines implicitly carry out information about the coil sensitivity maps and they are used to build some GRAPPA kernels, which are then convoluted with the measured data to complete the non-sampled part of k-space. Since this technique does not require an explicit estimation of the coil sensitivities, we say that it is using coil sensitivities implicitly. We refer the reader to the original publication for additional information [18].

- **An Iterative SENSE implementation (`bmSensa`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2.$$

  Sensa is our implementation of iterative-SENSE [19] using sensitivity encoding (SENSE) and iterative Conjugate Gradient Descent (CGD). This method is useful for both Cartesian and non-Cartesian data and can reconstruct images from incompletely sampled data by sharing information between different channels. However, it can struggle with heavily undersampled data because it operates frame-by-frame without sharing information between frames, making it less suitable for dynamic imaging. Despite this limitation, iterative-SENSE plays a fundamental role in the theoretical framework of image reconstruction.

- **A least-square reconstruction with $l_2$ spatial regularization (`bmSleva`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2}\|x\|_{X,2}^2.$$

  Sleva is a single-frame reconstruction method. It uses $l_2$-regularization in order to make the solution unique. As $\lambda$ approaches zero, the optimization problem associated with Sleva returns the solution to the least-squares problem with smallest $l_2$-norm. Unlike the $l_1$-norm, the $l_2$-norm regularization does not enforce sparsity in any form. Sleva is therefore not a CS reconstruction.

- **A least-square reconstruction with $l_1$ spatial regularization (`bmSteva`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2}\|\nabla_r x\|_{X^\nu,1}.$$

  Steva is a single-frame reconstruction method. It is a CS method with spatial regularization by discrete anisotropic total variation. The $l_1$-norm enforces sparsity in the spatial gradient of the image written $\nabla_r x$, which means that it encourages many coefficients of $\nabla_r x$ to be exactly zero. This helps preserve sharp edges and fine details in spatially piecewise-constant images.

- **A least-square reconstruction with $l_2$ temporal regularization, with or without motion compensation (`bmSensitivaMorphosia_chain`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2}\|\widetilde{D}_t^b\ x\|_{X,2}^2$$

  If deformation matrices are provided to the reconstruction function, then $\widetilde{D}_t^b\ x$ denotes the list of (backward) motion-compensated temporal differences, and it follows that the regularization is the sum of the squared $l_2$-norms of the motion-compensated finite differences between temporal neighbors. Otherwise, $\widetilde{D}_t^b\ x$ corresponds to the (backward) finite-difference time derivative of $x$, and the the regularization is the sum of the squared $l_2$-norms of the finite differences between temporal neighbors. This reconstruction strategy is specifically designed for dynamic imaging applications.
- **A least-square reconstruction with $l_1$ temporal regularization, with or without motion compensation (`bmTevaMorphosia_chain`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2}\|\widetilde{D}_t^b\ x\|_{X,1}$$

  Again, if deformation matrices are provided to the reconstruction function, then $\widetilde{D}_t^b\ x$ denotes the list of motion-compensated temporal differences. It follows that the $l_1$-regularization enforces sparsity

in the motion-compensated (backward) differences between temporal neighboring frames. Otherwise, $\widetilde{D}_t^b\ x$ corresponds to the finite-difference time derivative of $x$, and the the $l_1$-regularization enforces sparsity in the (backward) differences between temporal adjacent frames.

- **A least-square reconstruction with $l_2$ backward and forward temporal regularization, with or without motion compensation (`bmSensitivaDuoMorphosia_chain`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2}\|\widetilde{D}_t^{b,f}\ x\|_{X,2}^2$$

The SensitivaDuoMorphosia_chain method extends SensitivaMorphosia_chain by using both forward and backward finite-difference temporal derivatives.

- **A least-square reconstruction with $l_1$ backward and forward temporal regularization, with or without motion compensation (`bmTevaDuoMorphosia_chain`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2}\|\widetilde{D}_t^{b,f}\ x\|_{X,1}^2$$

The TevaDuoMorphosia_chain method extends TevaMorphosia_chain by using both forward and backward finite-difference temporal derivatives.

- **A least-square reconstruction with $l_2$ temporal regularization, with or without motion compensation, with 2 temporal dimensions (`bmSensitivaMorphosia_sheet`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2}\|\widetilde{D}_c^b\ x\|_{X,2}^2 + \frac{\lambda}{2}\|\widetilde{D}_s^b\ x\|_{X,2}^2$$

Here, the subscripts $c$ and $s$ correspond to the parameters of the two temporal dimensions. Note that by "temporal" we mean here any non-spatial dimension that can be interpreted as a time or a pseudo-time. Typically, $c$ represents the cardiac time, while $s$ denotes the respiratory time, meaning the time with respect to the start of the last cardiac or respiratory cycle respectively. The linear map $\widetilde{D}_t^b$ for $t \in \{c, s\}$ have the same meaning as above for each pseudo-time: if deformation matrices are provided to the reconstruction function, then $\widetilde{D}_t^b\ x$ denotes the list of motion-compensated temporal differences between temporal neighbors. Otherwise, $\widetilde{D}_t^b\ x$ corresponds to the finite-difference pseudo-time derivative of $x$

- **A least-square reconstruction with $l_1$ temporal regularization, with or without motion compensation, with 2 temporal dimensions (`bmTevaMorphosia_sheet`)** solving

$$x^{\#} \in \underset{x}{\operatorname{argmin}} \frac{1}{2}\|Ax - y\|_{Y,2}^2 + \frac{\lambda}{2}\|\widetilde{D}_c^b\ x\|_{X,1} + \frac{\lambda}{2}\|\widetilde{D}_s^b\ x\|_{X,1}$$

It is the $l_1$ version of SensitivaMorphosia_sheet. Again, the linear maps $\widetilde{D}_t^b$ for $t \in \{c, s\}$ have the same meaning as above for each (pseudo) time. .

For all reconstruction methods, the input arguments must include, among others, the content of the mitosius (y, t, ve) and the coil sensitivity matrix $C$. Additional reconstruction parameters are described in Table 1.

The output of the reconstruction process consists of one image (single-frame or multi-frame) in the spatial domain.

In some cases, providing **deformation matrices** (written $T_u$ and $T_{ut}$ for their transpose) to the reconstruction function improves the effect of a temporal (or pseudo-temporal) regularization in the presence of spatial misalignment between frames. Deformation matrices are used to compensate for motion. This can be particularly useful in applications like cardiac, kidney, liver, or lung imaging, where large motion occurs between frames. Note that the non-spatial dimension can be different from actual physical time. It can represent any list of motion states.

Each reconstruction method requires adjustment of the number of iterations and regularization parameters among others. The choice of these parameters depends on the degree of undersampling, the noise level, and the desired temporal resolution among other factors. In practice, a series of tests may be necessary to optimize these settings for a given dataset.

| Parameter | Description |
|---|---|
| **Coil sensitivity C** | Coil sensitivity maps as described in subsubsection 2.2.4. |
| **N_u** | The size of the Cartesian grid used for regridding in k-space. It is independent from the frame size, but it is often chosen to match it. Reconstruction time depends on this parameter. Double-precision array with dimensions:<br>• $[\mathrm{N}_x, \mathrm{N}_y]$ for 2D spatial acquisitions<br>• $[\mathrm{N}_x, \mathrm{N}_y, \mathrm{N}_z]$ for 3D spatial acquisitions |
| **dK_u** | Step size of the grid used for regridding in k-space. Single-precision array with dimensions:<br>• $[\mathrm{dK}_x, \mathrm{dK}_y]$ for 2D spatial acquisitions<br>• $[\mathrm{dK}_x, \mathrm{dK}_y, \mathrm{dK}_z]$ for 3D spatial acquisitions |
| **frSize** | Size of the reconstructed images in pixels. Double-precision array with dimensions:<br>• $[\mathrm{frN}_x, \mathrm{frN}_y]$ for 2D spatial acquisitions<br>• $[\mathrm{frN}_x, \mathrm{frN}_y, \mathrm{frN}_z]$ for 3D spatial acquisitions |
| **Reconstruction FOV** | The reconstruction FOV, that can be different from the acquisition FOV for non Cartesian acquisitions (the acquisition FOV can be defined starting from the distance between two consecutive points along one line within the acquisition sampling pattern). |
| **ve_max** | Maximum volume elements value, an upper bound used to limit ve and avoid convergence issues. Single-precision scalar. |
| **nIter** | Number of iterations for the outer-loop of any iterative reconstruction. Double-precision scalar. |
| **nCGD** | Number of iterations for the CGD inner-loop to solve the least-square sub-problem of the ADMM algorithm. Double-precision scalar. |
| **x0** | Initial image for iterative reconstruction. Typically computed using a gridded reconstruction. Complex-valued, single-precision array. |
| **delta** ($\delta$) **or lambda** ($\lambda$) | Regularization parameter. Single-precision value, which can be:<br>• A scalar<br>• A list of two scalars $[\delta_{\min}, \delta_{\max}]$ where $\delta_{\min} < \delta_{\max}$<br>• A vector of length nIter |
| **rho** ($\rho$) | Convergence parameter of the Alternating Direction Method of Multipliers (ADMM) algorithm. Single-precision scalar. |
| **Gu** | The gridding (sparse) matrix used for forward gridding in iterative non-Cartesian reconstructions. |
| **Gut** | The transposed matrix of Gu used for backward (not inverse) gridding in iterative non-Cartesian reconstructions. |
| **Tu** | The deformation (sparse) matrix used for forward deformation in iterative temporally regularized reconstructions. |
| **Tut** | The transposed matrix of Tu used for backward (not inverse) deformation in iterative temporally regularized reconstructions. |

**Table 1**: Parameter descriptions for reconstructions in Monalisa.

## 2.3 Comparison with existing reconstruction frameworks

To evaluate the performance of Monalisa compared to existing reconstruction frameworks, we conducted a controlled experiment[12]s comparing Monalisa and BART [29]. The comparison was performed on three different 2D images, including a numerical Shepp–Logan brain phantom [59], a brain slice, and a cardiac slice (see Figure 2a, ground truth column). The raw MRI measurements were synthetically generated by simulating a 2D radial acquisition with 30 lines of 512 points each. The trajectory was constructed by rotating each line sequentially by an angle of $\frac{2\pi}{30}$. The number of lines was deliberately chosen to induce undersampling artifacts in the gridded reconstruction, thereby highlighting the advantages of regularized reconstruction methods. Monalisa and BART were tested using iterative reconstructions with $l_1$ and $l_2$ regularization terms. The same coil sensitivity maps and volume elements were used for both frameworks. Thus, the goal is to compare two implementations of the same reconstruction algorithm,

[12]Benchmarking code and data are available at: https://github.com/MattechLab/comparisonMonalisaBart/ .

using identical inputs, in order to demonstrate that our implementation quality is on par with that of a widely established toolbox. Since regularization can influence the magnitude of the reconstructed images, we applied post-reconstruction image rescaling to match the mean intensity of the ground truth image within a manually predefined elliptical ROI.

To ensure fair evaluation, for both frameworks, regularization parameters were optimized via a grid search, and the regularization values yielding the highest SSIM were selected for each reconstruction independently. Reconstructions were performed using 140 iterations, a number which guarantees convergence for both frameworks. The number 140 was empirically determined by trying several incremental values and checking the convergence. For both frameworks, $l_1$-regularized reconstruction was solved using ADMM, while $l_2$-regularized reconstruction was solved using CGD.

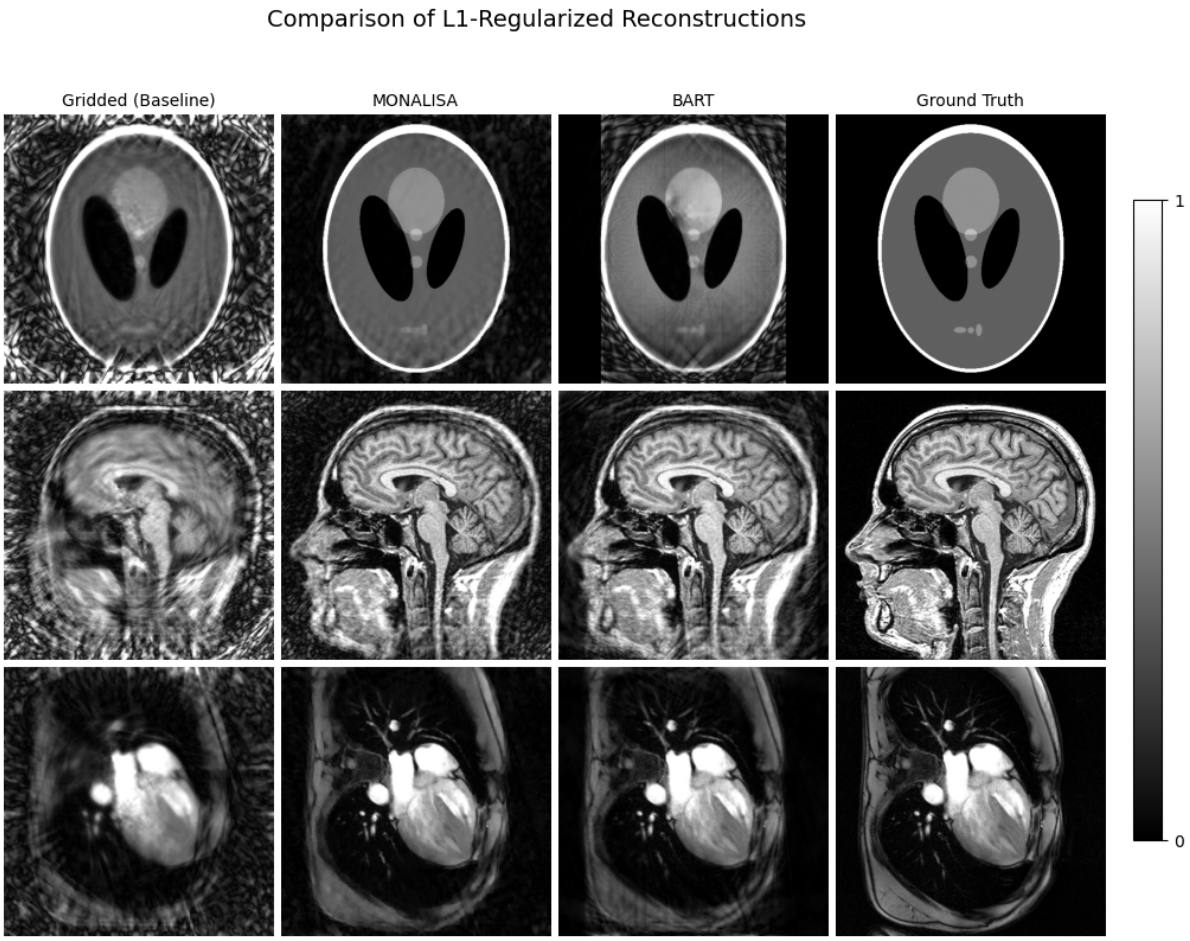
Reconstructions were evaluated using both SSIM [60] and $l_2$ distance with respect to the associated ground truth images. $l_2$ distance computes the sum of squared differences between corresponding pixel intensities in two images. It is a straightforward, pixel-wise error measure of the reconstruction that treats every pixel equally. SSIM, on the other hand, is designed to mimic the human visual system by considering perceptual phenomena. It evaluates image quality based on three key components: luminance, contrast, and structure.

# 3 Results

Images reconstructed using $l_1$ and $l_2$ regularization are shown in Figure 2a and Figure 3a, respectively. Corresponding quantitative metrics are reported in Figure 2b and Figure 3b. Both the visual results and the metrics indicate that regularized iterative reconstructions outperform Monalisa's gridded reconstruction across both frameworks.

When comparing iterative reconstructions between frameworks, for both the numerical phantom and the brain images the reported metrics reveal a trade-off: when one framework achieves a higher SSIM score, the other tends to have a lower $l_2$ distance, indicating that neither framework consistently produces superior image reconstructions. Visually, for the brain and phantom images, BART reconstructions appear more affected by artifacts.

For the cardiac image, quantitative metrics favor Monalisa, although visual inspection shows comparable image quality between frameworks.

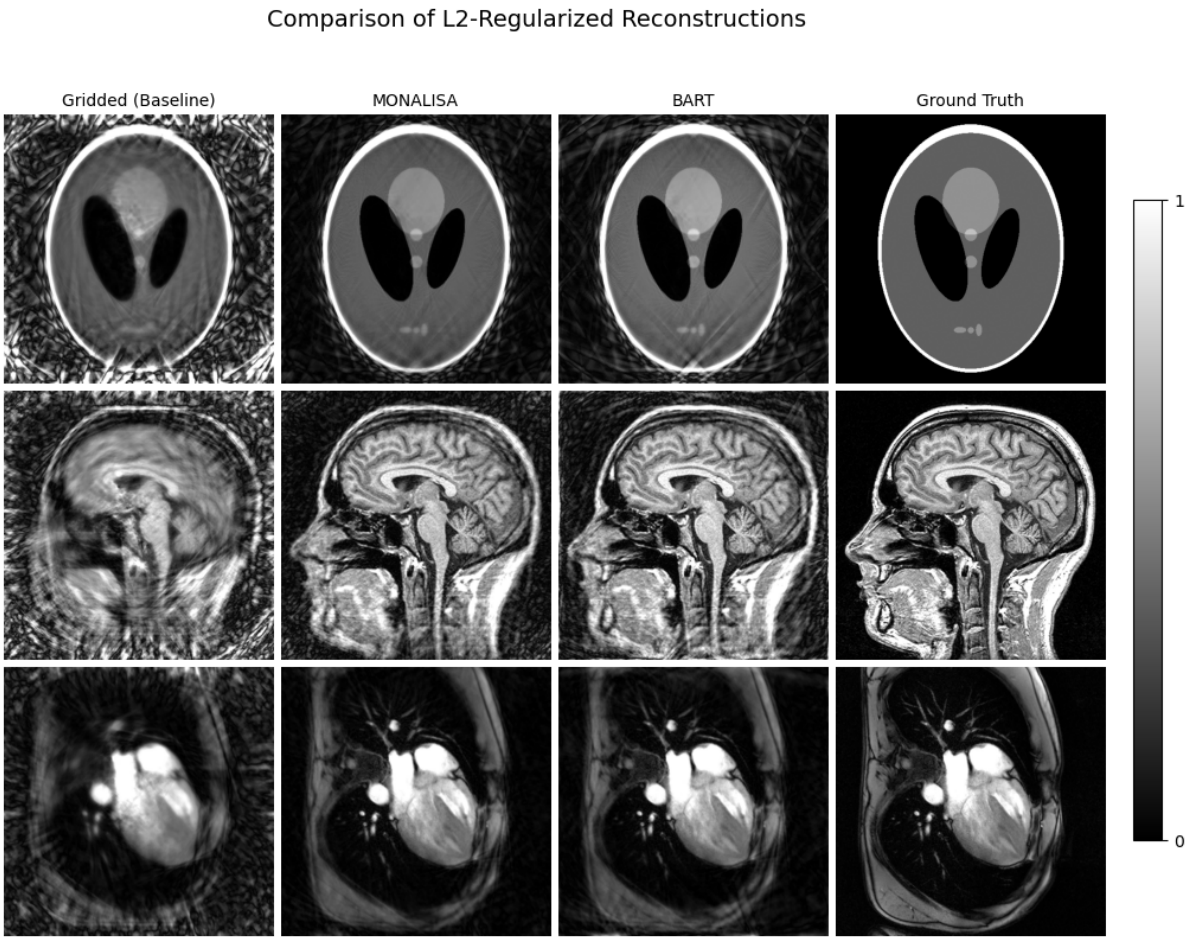



(a) Visual comparison of gridded reconstruction, Monalisa, and BART $l_1$ regularized iterative reconstructions. To enhance contrast and make background artifacts easier to visualize, the value representing white in each row is set to a fraction (between 30% and 70%) of the image's maximum value.

| Metric | Gridded Recon | Monalisa $l_1$-Reg | BART $l_1$-Reg |
|---|---|---|---|
| **Phantom** | | | |
| **SSIM** | 0.3745 | 0.5867 | **0.6883** |
| **$l_2$-Distance** | 93.9247 | **29.3729** | 52.0827 |
| **Brain** | | | |
| **SSIM** | 0.2368 | 0.5332 | **0.5645** |
| **$l_2$-Distance** | 28329.9280 | **13523.8670** | 13531.1620 |
| **Cardiac** | | | |
| **SSIM** | 0.5141 | **0.8316** | 0.8080 |
| **$l_2$-Distance** | 63.3103 | **26.3936** | 31.9331 |

(b) Quantitative comparison showing metrics for each method. We defined optimality as the minimizing difference between the reconstructed images and the ground-truth. Best values are highlighted in **bold**.

**Fig. 2**: Comparison of $l_1$-regularized reconstruction results using visual and quantitative metrics.




(a) Visual comparison of gridded reconstruction, Monalisa, and BART $l_2$ regularized iterative reconstructions. To enhance contrast and make background artifacts easier to visualize, the value representing white in each row is set to a fraction (between 30% and 70%) of the image's maximum value.

| Metric | Gridded Recon | Monalisa $l_2$-Reg | BART $l_2$-Reg |
|---|---|---|---|
| **Phantom** | | | |
| **SSIM** | 0.3745 | **0.5049** | 0.4670 |
| $l_2$**-Distance** | 93.9247 | 47.4290 | **47.0134** |
| **Brain** | | | |
| **SSIM** | 0.2368 | 0.5370 | **0.5567** |
| $l_2$**-Distance** | 28329.9280 | **13309.8610** | 15796.3090 |
| **Cardiac** | | | |
| **SSIM** | 0.5141 | **0.8350** | 0.7946 |
| $l_2$**-Distance** | 63.3103 | **29.7165** | 32.6275 |

(b) Comparison of $l_2$-regularized reconstructions between BART and Monalisa, including Gridded Reconstructions as a baseline. We defined optimality as the minimizing difference between the reconstructed images and the ground-truth. The best values are highlighted in **bold**.

**Fig. 3**: Comparison of $l_2$-regularized reconstruction results using visual and quantitative metrics.

# 4 Discussion

In this study, we have presented Monalisa, a comprehensive and open access MRI reconstruction framework. In this work, we detail both the software design choices behind Monalisa, and the steps required to transform raw k-space data into high-quality images. By modularizing the reconstruction workflow, composed of the following steps: data reading, trajectory scaling, volume elements computation, coil sensitivity estimation, binning, and mitosius preparation, Monalisa enables users to implement and experiment with a variety of reconstruction algorithms, including GRAPPA, iterative SENSE, CS with a variety of spatial and temporal regularization.

Our benchmarking against BART on simulated undersampled 2D radial data reveals that Monalisa achieves comparable performance. The observed differences between SSIM and $l_2$ distance can likely be attributed to variations in algorithmic implementation.

A key strength of Monalisa lies in its moderate complexity, thanks to its reliance on a high-level programming language and its extensive documentation, which make the framework significantly simpler to understand compared to other existing reconstruction toolkits. This accessibility makes Monalisa an ideal platform for teaching and learning MRI reconstruction, while also offering researchers an agile environment to test and develop new reconstruction strategies. Additionally, our framework directly integrates the recent trends of open source acquisition frameworks, supporting ISMRMRD raw data format [40], and trajectory generation with PulSeq [41]. In doing so, Monalisa makes a significant contribution to the MRI community by demystifying the reconstruction "black box" and promoting at the same time transparency and innovation.

However, Monalisa's current implementation does not support specialized hardware acceleration (e.g., GPUs), which could enhance computational efficiency, especially for large datasets. Nonetheless, this can be easily achieved by replacing the Fast Fourier Transform (FFT) operations with a CUDA-based alternative, which is called CUFFT (CUDA-based FFT implementation), or another GPU-based FFT implementation, as successfully demonstrated in previous work [61, 62]. Additionally, because Monalisa is built on MATLAB—a licensed environment—it may present accessibility challenges, although most research labs in this domain typically have access to it. Future work will focus on implementing GPU support and exploring more accessible development platforms to further improve both performance and accessibility.

Check out our documentation at: `https://mattechlab.github.io/monalisa/`

Check out our source code at: `https://github.com/MattechLab/monalisa/`

# Declarations/ Acknowledgments


- **Funding:** This work was supported by the following institutions and organizations:
  - Swiss National Science Foundation (SNSF) (Grants: #220433 #229214 to Prof. Franceschiello).
  - Swiss National Science Foundation (SNSF) (Grant: #320030B_201292 to Prof. Stuber)
  - Swiss National Science Foundation (SNSF) (Grant: #320030B_194296 to Prof. Bastiaansen)
  - The Sense Innovation and Research Center(Grant: KiCk fMRI, PI: Prof. Franceschiello with Dr. Juliane Schneider).

  We sincerely thank all contributors and organizations whose support has made this project possible.

- **Conflict of interest/Competing interests:** Authors have no conflicts of interest to declare.
- **Data availability:** The data for the benchmarking Monalisa against BART is open source, accessible at: `https://github.com/MattechLab/comparisonMonalisaBart/`
- **Code availability:** Monalisa complete source code is currently open source, accessible at: `https://github.com/MattechLab/monalisa/`. The code for the benchmarking Monalisa against BART is open source, accessible at: `https://github.com/MattechLab/comparisonMonalisaBart/`
- **Conflict of interest statement:** The authors declare no conflict of interest.
- **Author Contributions (CRediT Taxonomy)**
  - **Mauro Leidi**: *Conceptualization; Software; Methodology; Validation; Writing – Original Draft; Writing – Review & Editing; Visualization.*
    Set up the documentation infrastructure using Sphinx-Matlabdomain and configured continuous integration pipelines for automated documentation deployment and for unit testing. Authored major parts of the documentation (Raw Data Reading, Volume Elements, Quickstart, Installation, Binning, Acquisition, Tutorial 1). Developed raw data loader classes, Dockerfile, and compilation scripts. Conducted benchmarking against BART. Served as lead writer of the article.

- **Yiwei Jia**: *Software; Writing – Review & Editing; Validation.*
  Authored the *mitosius* documentation section, contributed example scripts, maintained code with code checkers, and assisted with the Dockerfile and setup.
- **Dominik Helbing**: *Software; Writing – Review & Editing.*
  Contributed in-code documentation, authored the coil sensitivity section, and optimized coil sensitivity estimation.
- **Jaime Barranco**: *Software; Data Curation; Writing – Review & Editing.*
  Developed an automated API for the documentation website, implemented CI for spell checking, and supported GitHub repository setup.
- **Berk Can Açikgöz**: *Software.*
  Implemented the 2D and 3D GRAPPA reconstruction modules of the Monalisa toolbox.
- **Eva Peper**: *Investigation; Resources.*
  Carried out cardiac data acquisition for the study.
- **Jean-Baptiste Ledoux**: *Investigation; Resources.*
  Responsible for the acquisitions used in the development of the coil sensitivity estimation method, contributed to the conception of acquisition protocols, provided information on scanner hardware and interface.
- **Jessica Bastiaansen**: *Investigation; Resources.*
  Responsible for acquiring the cardiac data used in the article.
- **Benedetta Franceschiello**: *Supervision; Project Coordination; Conceptualization; Funding Acquisition; Writing – Original Draft; Writing – Review & Editing.*
  Lab leader and Principal Investigator. Provided mentorship, proposed novel applications and benchmarks, managed meetings, secured legal and financial support, and ensured the project's success. The main writer of the introduction of this article.
- **Bastien Milani**: *Software; Formal analysis; Methodology; Supervision; Conceptualization; Writing – Review & Editing.*
  Original implementer of Monalisa and main contributor. Supervisor of the present article. Contributed to the conceptual development of the project. The main writer of the reconstruction section both of this article and of the documantation.

# References


[1] Donald B Plewes and Walter Kucharczyk. Physics of mri: a primer. *Journal of magnetic resonance imaging*, 35(5):1038–1054, 2012.

[2] Seiji Ogawa, Tso-Ming Lee, Asha S. Nayak, and Paul Glynn. Oxygenation-sensitive contrast in magnetic resonance image of rodent brain at high magnetic fields. *Magnetic Resonance in Medicine*, 14(1):68–78, apr 1990.

[3] S. Ogawa, D. W. Tank, R. Menon, J. M. Ellermann, S. G. Kim, H. Merkle, and K. Ugurbil. Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging. *Proceedings of the National Academy of Sciences*, 89(13):5951–5955, jul 1992.

[4] Peter A Bandettini, Eric C Wong, R Scott Hinks, Ronald S Tikofsky, and James S Hyde. Time course epi of human brain function during task activation. *Magnetic resonance in medicine*, 25(2):390–397, 1992.

[5] Michael Lustig, David Donoho, and John M Pauly. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[6] Kai Tobias Block, Hersh Chandarana, Girish Fatterpekar, Mari Hagiwara, Sarah Milla, Thomas Mulholland, Mary Bruno, Christian Geppert, and Daniel K Sodickson. Improving the robustness of clinical t1-weighted mri using radial vibe. *Magnetom Flash*, 5:6–11, 2013.

[7] Chi-Ming Tsai and Dwight G Nishimura. Reduced aliasing artifacts using variable-density k-space sampling trajectories. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 43(3):452–458, 2000.

[8] Paul C Lauterbur. Image formation by induced local interactions: examples employing nuclear magnetic resonance. *nature*, 242(5394):190–191, 1973.

[9] Stefanie Winkelmann, Tobias Schaeffter, Thomas Koehler, Holger Eggers, and Olaf Doessel. An optimal radial profile order based on the golden ratio for time-resolved mri. *IEEE transactions on medical imaging*, 26(1):68–76, 2006.

[10] Rachel W Chan, Elizabeth A Ramsay, Charles H Cunningham, and Donald B Plewes. Temporal stability of adaptive 3d radial mri using multidimensional golden means. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 61(2):354–363, 2009.

[11] Davide Piccini, Arne Littmann, Sonia Nielles-Vallespin, and Michael O Zenge. Spiral phyllotaxis: the natural way to construct a 3d radial trajectory in mri. *Magnetic resonance in medicine*, 66(4):1049–1056, 2011.

[12] Kai Tobias Block, Hersh Chandarana, Sarah Milla, Mary Bruno, Tom Mulholland, Girish Fatterpekar, Mari Hagiwara, Robert Grimm, Christian Geppert, Berthold Kiefer, et al. Towards routine clinical use of radial stack-of-stars 3d gradient-echo sequences for reducing motion sensitivity. *Investigative Magnetic Resonance Imaging*, 18(2):87–106, 2014.

[13] Maxim Zaitsev, Julian Maclaren, and Michael Herbst. Motion artifacts in MRI: A complex problem with many partial solutions. *Journal of Magnetic Resonance Imaging*, 42(4):887–901, oct 2015.

[14] Hersh Chandarana, Tobias K Block, Andrew B Rosenkrantz, Ruth P Lim, Danny Kim, David J Mossa, James S Babb, Berthold Kiefer, and Vivian S Lee. Free-breathing radial 3d fat-suppressed t1-weighted gradient echo sequence: a viable alternative for contrast-enhanced liver imaging in patients unable to suspend respiration. *Investigative radiology*, 46(10):648–653, 2011.

[15] Gary H Glover and John M Pauly. Projection reconstruction techniques for reduction of motion effects in mri. *Magnetic resonance in medicine*, 28(2):275–289, 1992.

[16] Daniel K Sodickson and Warren J Manning. Simultaneous acquisition of spatial harmonics (smash): fast imaging with radiofrequency coil arrays. *Magnetic resonance in medicine*, 38(4):591–603, 1997.

[17] Klaas P Pruessmann, Markus Weiger, Markus B Scheidegger, and Peter Boesiger. Sense: sensitivity encoding for fast mri. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 42(5):952–962, 1999.

[18] Mark A Griswold, Peter M Jakob, Robin M Heidemann, Mathias Nittka, Vladimir Jellus, Jianmin Wang, Berthold Kiefer, and Axel Haase. Generalized autocalibrating partially parallel acquisitions (grappa). *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 47(6):1202–1210, 2002.

[19] K. P. Pruessmann, M. Weiger, P. Börnert, and P. Boesiger. Advances in sensitivity encoding with arbitrary k-space trajectories. 46(4):638–651.

[20] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306,

2006.
[21] Michael Lustig, D.L. Donoho, J.M. Santos, and J.M. Pauly. Compressed Sensing MRI. *IEEE Signal Processing Magazine*, 25(2):72–82, mar 2008.
[22] Alice C Yang, Madison Kretzler, Sonja Sudarski, Vikas Gulani, and Nicole Seiberlich. Sparse reconstruction techniques in magnetic resonance imaging: methods, applications, and challenges to clinical adoption. *Investigative radiology*, 51(6):349–364, 2016.
[23] Tolga Çukur, Michael Lustig, Emine U Saritas, and Dwight G Nishimura. Signal compensation and compressed sensing for magnetization-prepared mr angiography. *IEEE transactions on medical imaging*, 30(5):1017–1027, 2011.
[24] Shreyas S Vasanawala, MJ Murphy, Marcus T Alley, Peng Lai, Kurt Keutzer, John M Pauly, and Michael Lustig. Practical parallel imaging compressed sensing mri: Summary of two years of experience in accelerating body mri of pediatric patients. In *2011 ieee international symposium on biomedical imaging: From nano to macro*, pages 1039–1043. IEEE, 2011.
[25] Oren N Jaspan, Roman Fleysher, and Michael L Lipton. Compressed sensing mri: a review of the clinical literature. *The British journal of radiology*, 88(1056):20150487, 2015.
[26] Florian Knoll, Kerstin Hammernik, Chi Zhang, Steen Moeller, Thomas Pock, Daniel K Sodickson, and Mehmet Akcakaya. Deep learning methods for parallel magnetic resonance image reconstruction. *arXiv preprint arXiv:1904.01112*, 2019.
[27] Javier Montalt-Tordera, Vivek Muthurangu, Andreas Hauptmann, and Jennifer Anne Steeden. Machine learning in magnetic resonance imaging: image reconstruction. *Physica Medica*, 83:79–87, 2021.
[28] Burhaneddin Yaman, Seyed Amir Hossein Hosseini, Steen Moeller, Jutta Ellermann, Kâmil Uğurbil, and Mehmet Akçakaya. Self-supervised physics-based deep learning mri reconstruction without fully-sampled data. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 921–925. IEEE, 2020.
[29] Martin Uecker, Frank Ong, Jonathan I Tamir, Dara Bahri, Patrick Virtue, Joseph Y Cheng, Tao Zhang, and Michael Lustig. Berkeley advanced reconstruction toolbox. In *Proc. Intl. Soc. Mag. Reson. Med*, volume 23, page 9, 2015.
[30] Tobias Knopp and Mirco Grosser. Mrireco. jl: An mri reconstruction framework written in julia. *Magnetic resonance in medicine*, 86(3):1633–1646, 2021.
[31] Michael Schacht Hansen and Thomas Sangild Sørensen. Gadgetron: an open source framework for medical image reconstruction. *Magnetic resonance in medicine*, 69(6):1768–1776, 2013.
[32] Daniel Gallichan, José P Marques, and Rolf Gruetter. Retrospective correction of involuntary microscopic head movement using highly accelerated fat image navigators (3d fatnavs) at 7t. *Magnetic resonance in medicine*, 75(3):1030–1039, 2016.
[33] Frank Ong and Michael Lustig. Sigpy: a python package for high performance iterative reconstruction. In *Proceedings of the ISMRM 27th Annual Meeting, Montreal, Quebec, Canada*, volume 4819, 2019.
[34] Matthew J Muckley, Ruben Stern, Tullie Murrell, and Florian Knoll. Torchkbnufft: A high-level, hardware-agnostic non-uniform fast fourier transform. In *ISMRM Workshop on Data Sampling & Image Reconstruction*, volume 22, 2020.
[35] Loubna El Gueddari, Chaithya Giliyar Radhakrishna, Zaccharie Ramzi, Samuel Farrens, Sophie Starck, Antoine Grigis, Jean-Luc Starck, and Philippe Ciuciu. Pysap-mri: A python package for mr image reconstruction. In *ISMRM workshop on Data Sampling and Image Reconstruction*, 2020.
[36] Florian Knoll, Andreas Schwarzl, Clemens Diwoky, and Daniel K Sodickson. gpunufft-an open source gpu library for 3d regridding with direct matlab interface. In *Proceedings of the 22nd annual meeting of ISMRM, Milan, Italy*, volume 4297, 2014.
[37] Jakob Assländer, Martijn A Cloos, Florian Knoll, Daniel K Sodickson, Jürgen Hennig, and Riccardo Lattanzi. Low rank alternating direction method of multipliers reconstruction for mr fingerprinting. *Magnetic resonance in medicine*, 79(1):83–96, 2018.
[38] Efrat Shimron, Andrew G Webb, and Haim Azhari. Core-deblur: parallel mri reconstruction by deblurring using compressed sensing. *Magnetic Resonance Imaging*, 72:25–33, 2020.
[39] Siemens Healthineers. Magnetom cima.x fit – upgrade to make the difference. https://www.siemens-healthineers.com/de-ch/magnetic-resonance-imaging/options-and-upgrades/upgrades/magnetom-cima-x-fit, n.d. Accessed: February 18, 2025.
[40] Souheil J Inati, Joseph D Naegele, Nicholas R Zwart, Vinai Roopchansingh, Martin J Lizak, David C Hansen, Chia-Ying Liu, David Atkinson, Peter Kellman, Sebastian Kozerke, et al. Ismrm raw data format: A proposed standard for mri raw datasets. *Magnetic resonance in medicine*, 77(1):411–421,

2017.
[41] Kelvin J Layton, Stefan Kroboth, Feng Jia, Sebastian Littin, Huijun Yu, Jochen Leupold, Jon-Fredrik Nielsen, Tony Stöcker, and Maxim Zaitsev. Pulseq: a rapid and hardware-independent pulse sequence prototyping framework. *Magnetic resonance in medicine*, 77(4):1544–1552, 2017.
[42] Jeffrey A Fessler. On nufft-based gridding for non-cartesian mri. *Journal of magnetic resonance*, 188(2):191–195, 2007.
[43] Bastien Milani and Bistro de l'Atlantic. A mathematical language for mri reconstructions. 2023.
[44] Georgy Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*, (133):97–178, 1907.
[45] Bastien Milani, Christopher Roy, Jean-Baptiste Ledoux, David C Rotzinger, Ambra Masi, Renaud Troxler, Salim Si-mohamed, Jerome Yerly, Ludovica Romanin, Tobias Rutz, et al. Improving 3d-cine ttv-regularized whole-heart mri reconstruction. *MedRxiv*, pages 2024–02, 2024.
[46] Leslie Ying and Jinhua Sheng. Joint image reconstruction and sensitivity estimation in sense (jsense). *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 57(6):1196–1202, 2007.
[47] Gioele Janett. Discontinuities in numerical radiative transfer. *Astronomy & Astrophysics*, 622:A162, 2019.
[48] J.W. Wallis, T.R. Miller, C.A. Lerner, and E.C. Kleerup. Three-dimensional display in nuclear medicine. *IEEE Transactions on Medical Imaging*, 8(4):297–230, 1989.
[49] Peter B Roemer, William A Edelstein, Cecil E Hayes, Steven P Souza, and Otward M Mueller. The nmr phased array. *Magnetic resonance in medicine*, 16(2):192–225, 1990.
[50] Sandy Napel, Geoffrey D Rubin, and R Brooke Jeffrey. Sts-mip: a new reconstruction technique for ct of the chest. *Journal of computer assisted tomography*, 17(5):832–838, 1993.
[51] Santiago Aja-Fernández, Carlos Alberola-López, and Carl-Fredrik Westin. Noise and signal estimation in magnitude mri and rician distributed images: a lmmse approach. *IEEE transactions on image processing*, 17(8):1383–1398, 2008.
[52] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.
[53] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
[54] Andrew C Larson, Richard D White, Gerhard Laub, Elliot R McVeigh, Debiao Li, and Orlando P Simonetti. Self-gated cardiac cine mri. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 51(1):93–102, 2004.
[55] B.P. Sutton, D.C. Noll, and J.A. Fessler. Fast, iterative image reconstruction for mri in the presence of field inhomogeneities. *IEEE Transactions on Medical Imaging*, 22(2):178–188, 2003.
[56] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.
[57] Fadil Santosa and William W Symes. Linear inversion of band-limited reflection seismograms. *SIAM journal on scientific and statistical computing*, 7(4):1307–1330, 1986.
[58] Jeffrey A Fessler and Bradley P Sutton. Nonuniform fast fourier transforms using min-max interpolation. *IEEE transactions on signal processing*, 51(2):560–574, 2003.
[59] Lawrence A Shepp and Benjamin F Logan. The fourier reconstruction of a head section. *IEEE Transactions on nuclear science*, 21(3):21–43, 1974.
[60] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
[61] Ludovica Romanin, Bastien Milani, Christopher W Roy, Jérôme Yerly, Aurélien Bustin, Salim Si-Mohamed, Milan Prsa, Tobias Rutz, Estelle Tenisch, Juerg Schwitter, et al. Similarity-driven motion-resolved reconstruction for ferumoxytol-enhanced whole-heart mri in congenital heart disease. *PLoS One*, 19(6):e0304612, 2024.
[62] Ludovica Romanin, Milan Prsa, Christopher W Roy, Xavier Sieber, Jérôme Yerly, Bastien Milani, Tobias Rutz, Salim Si-Mohamed, Estelle Tenisch, Davide Piccini, et al. Exploring the limits of scan time reduction for ferumoxytol-enhanced whole-heart angiography in congenital heart disease patients. *Journal of Cardiovascular Magnetic Resonance*, page 101854, 2025.

# Supplementary Material

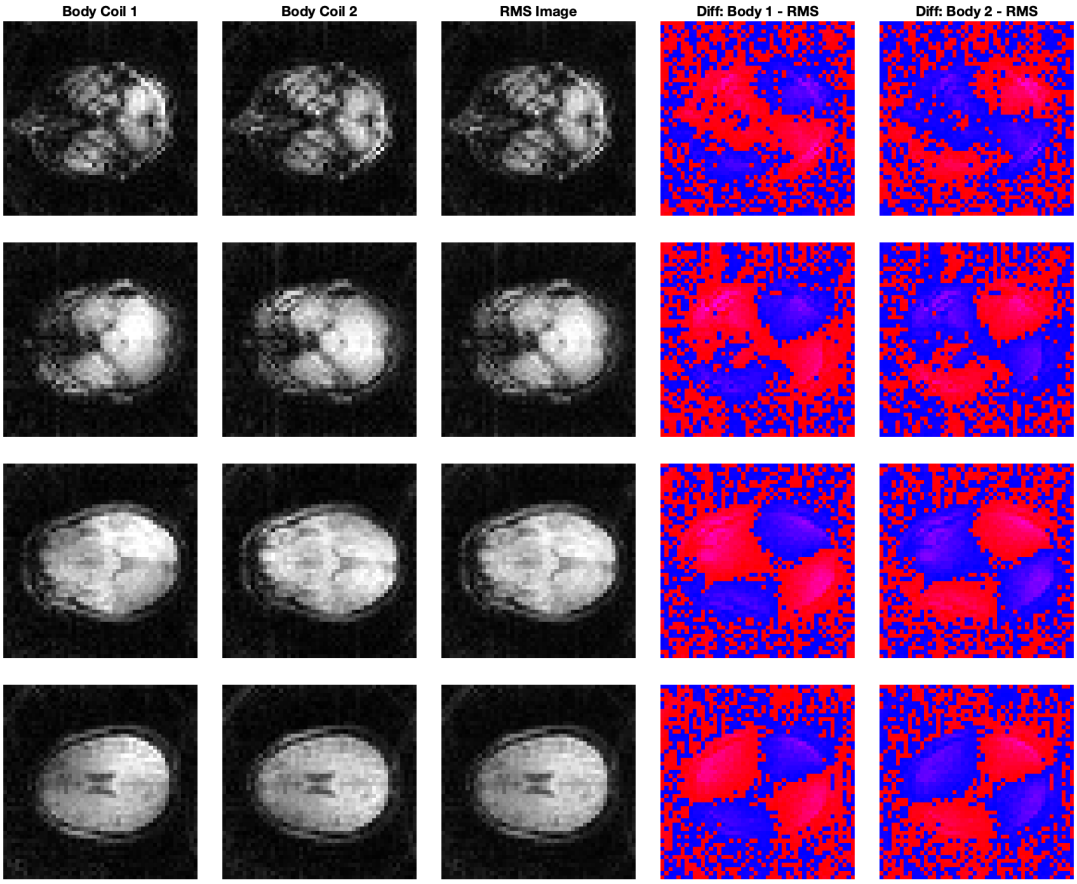



**Fig. 1**: Comparison of individual body coil images and the corresponding RMS image for a single brain calibration scan. Results presented in this image are consistent across all calibration scans. The last two columns reveal significant inhomogeneities in the body coil images relative to the RMS, highlighting the orientation of the body coil in the MRI scanner and their resulting sensitivity variations.

In Figure 1, we present a representative example illustrating the inherent inhomogeneity of single body coil images. The last two columns distinctly show that both body coil images exhibit substantial deviations from the RMS image, which serves as a reference for a more uniform image. If the assumption of [17] was valid—namely, that the body coils are homogeneous—no variation would be visible. These discrepancies indicate non-uniform sensitivity profiles of the body coils, strongly influenced by their orientation and position within the scanner. This observation underscores the limitations of relying on individual body coils for accurate sensitivity estimation. Constructing a virtual reference coil as a combination of measurements from both body coils offers a more robust solution. In facts, assuming that the inhomogeneities in each body-coil sensitivity are independent and random, averaging their contributions yields to an estimate with reduced variance. This is consistent with the central limit theorem, resulting in a more robust sensitivity map than that obtained from a single body coil.