Schmidt J, Scheibe P, Carreto Fidalgo D, Marek A, Trampel R, Weiskopf N. Fast and scalable joint-LORAKS reconstruction and data-driven sampling optimisation of high-dimensional MRI datasets using a GPU-accelerated and learning-free differentiable framework: PyLORAKS. *Aperture Neuro*. 2026;6. doi:10.52294/001c.156499

Original Research Articles

Fast and scalable joint-LORAKS reconstruction and data-driven sampling optimisation of high-dimensional MRI datasets using a GPU-accelerated and learning-free differentiable framework: PyLORAKS

Jochen Schmidt, M.Sc.[1,2][a], Patrick Scheibe, Ph.D.[1], David Carreto Fidalgo, Ph.D.[1,3], Andreas Marek, Ph.D.[3], Robert Trampel, Ph.D.[1], Nikolaus Weiskopf, Prof. Dr.[1,4,5]

[1] Department of Neurophysics, Max Planck Institute for Human Cognitive and Brain Sciences, [2] International Max Planck Research School on Neuroscience of Communication: Function, Structure and Plasticity, [3] Max Planck Computing and Data Facility, [4] Felix Bloch Institute for Solid State Physics, Faculty of Physics and Earth System Sciences, Leipzig University, [5] Wellcome Centre for Human Neuroimaging, Institute of Neurology, University College London

Keywords: GPU-accelerated reconstruction framework, LORAKS, image reconstruction, MRI reconstruction, joint-reconstruction, low-rank matrix completion, low-rank reconstruction, GPU computing, high-resolution MRI, PyTorch

https://doi.org/10.52294/001c.156499

Aperture Neuro

Vol. 6, 2026

Magnetic resonance imaging (MRI) benefits significantly from parallel imaging and low-rank matrix completion approaches to reconstruct accelerated multidimensional acquisitions. As one such algorithm, low-rank matrix modelling of local k-space neighbourhoods (LORAKS) provides image reconstruction by exploiting sparsity in undersampled multichannel data acquisitions. Besides using spatial and multichannel redundancies, joint-reconstruction of multi-contrast data provides higher-quality reconstruction and enables higher acceleration factors. However, the computational demands of LORAKS limit its applicability for joint-reconstruction of modern multichannel multi-contrast high-resolution datasets, but methods to speed up the acquisition and reconstruction process are highly desired.

In this work, we present PyLORAKS, a graphics processing unit (GPU) accelerated implementation of LORAKS built using the PyTorch framework. The framework employs efficient parallelisation, GPU compute strategies, and randomised singular value decomposition methods for increased computational efficiency in dealing with huge LORAKS matrix sizes encountered in modern high-resolution MRI. The computation speedup enabled parameter optimisation via Bayesian methods, while automated differentiable tracing of the LORAKS formulation was used for data-driven subsampling optimisation. In benchmark experiments, reduction in memory overhead was achieved, paired with approximately 5–6× speedup over the previous (MATLAB) LORAKS implementations using CPU computation, and a ~30-fold improvement using PyLORAKS on GPUs. While ensuring similar reconstruction performance, the computational efficiency allows for reconstruction of larger data matrices previously posing challenges to memory or computation time demands. Automated parameter optimisation of LORAKS parameters λ and r yielded better results than manual selection, avoiding inter-contrast leakage artefacts. Additionally, the highest reconstruction quality was achieved by using the maximal number of available contrasts for joint-reconstruction, exemplarily resulting in an increased structural similarity (SSIM) from 0.9468 to 0.9688 when doubling the number of echoes in joint-reconstruction of a 4-fold undersampled multi-echo acquisition. Furthermore, complementary sampling across echoes was found beneficial as reconstruction of the same sampling per echo achieved an SSIM of 0.955, whereas interleaving phase encode lines per echo increased the SSIM to 0.972 for the same

[a] Corresponding Author:
Jochen Schmidt
jochen.schmidt@tuta.io

acquisition and otherwise optimal parameter combinations. We further demonstrate superior performance of complementary sampling using an optimal sampling pattern obtained by backpropagation through the PyLORAKS algorithm.

Overall, PyLORAKS enables fast and efficient reconstruction of multichannel multi-contrast high-resolution MRI data. It enables parameter and data-driven reconstruction and acquisition optimisation and provides a platform for physics-informed or artificial intelligence-augmented development. The framework is openly available, including containerised environments for large-scale deployment.

INTRODUCTION

Magnetic resonance imaging (MRI) is an indispensable tool in medical diagnostics, offering detailed images of the human brain's internal structures. However, the acquisition of high-quality MR images is often constrained by the trade-off between scan time and image resolution. High-resolution imaging in particular requires large k-space acquisition matrices, which in turn depend on the collection of vast data.

Parallel Imaging (PI) techniques like generalised autocalibrating partially parallel acquisitions (GRAPPA)[1] and sensitivity encoding (SENSE)[2] are routinely used in multichannel receive coil setups to decrease scan time by undersampling k-space acquisition. Promising reconstruction techniques have been demonstrated relying on low-rank matrix completion, like low-rank matrix modelling of local k-space neighbourhoods (LORAKS).[3] LORAKS exploits the low-rank properties of specific operator matrices built from small neighbourhoods around each k-space point to facilitate reconstruction. The neighbourhoods can be extended by combining available channels to reconstruct multichannel data.[4]

Many MRI acquisitions depend on sequence variations to produce different contrasts, such as multi-echo temporal signal weighting[5-8] or diffusion gradient weighting.[9,10] Algorithms to handle reconstruction jointly across contrasts have already been demonstrated.[11,12] In particular, joint-echo reconstruction using a Joint-LORAKS (J-LORAKS) framework has been shown to outperform GRAPPA-based variants in image reconstruction quality.[13]

Despite its effectiveness and improvements to the original algorithm,[14] such as AC-LORAKS using an autocalibration (AC) region to speed up computations,[15] the computational demands of J-LORAKS pose significant challenges. Commonly, a singular value decomposition (SVD) operation is used to enforce low-rank structure. SVD scales quadratically in time and linearly in memory computation complexity.[16,17]

The structured matrices used in LORAKS scale linearly with the number of receive channels and contrasts (in modern scan settings: ≥ 32 receive channels, ≥ 2 contrasts), hindering reconstruction optimisation by high computational time and restricting high-resolution reconstruction by infeasible memory requirements. For example, we carried out the joint-reconstruction of an MRI scan with an acquisition matrix of 280×280 in plane, 100 slices, 32 receive channels, and three echoes on a modern compute server with the recent LORAKS software implementation.[18] The time consumed is > 3 d to process the whole dataset, while the requirement on the available system memory (RAM) is around 170 GB. Similarly, in comparison to other reconstruction techniques, LORAKS required the longer reconstruction times, without even including its computationally extensive phase constraint formalism or joint-reconstruction capabilities.[19]

Another challenge of LORAKS is the choice of the algorithm parameters (regularisation weighting λ and rank r), which are not known *a priori* and require careful balancing. Influence of the rank parameter r is commonly gained empirically by visual data assessment after reconstruction.[14,20,21] Especially for J-LORAKS, inherent high computational cost and slow computation speed hinder brute force methods or manual user selection. Automation of the parameter choices has been proposed using Stein's unbiased risk estimator (SURE),[22] in conjunction with coil sensitivity map estimation for a SENSE-based PI reconstruction.[23] Similarly, the LORAKS rank parameter has been subjected to SURE optimisation using compressed-sensing, total variation, and wavelet regularisation,[24] which further increases the computational needs upon reconstruction, with the time feasibility depending on the J-LORAKS reconstruction performance.

Multiple reconstruction frameworks and LORAKS extensions have been suggested to address its poor scaling and computational performance, utilising specific optimisation formulations or additional constraints.

A fast convolutional procedure (HICU) tailored to low-rank matrix completion introduced computational optimisations regarding gradient computation and precise step size estimation in its iterative solving.[25] Furthermore, randomised SVD (rSVD) was used to approximate low-rank deficiency,[25] offering improved stability and parallel computing.[26]

Despite speedup and memory reduction, HICU only extends the LORAKS compact support constraint, not using the additional phase-constraint, which was found to be superior in a number of applications.[27] Nevertheless, substituting the core matrix-decomposition with rSVD proves beneficial,[17,28] and recent variants like subspace-orbit rSVD[16] or randomised nullspace approximation (RANDNS)[29] may yield further efficiency gains for large matrices encountered in J-LORAKS.

Efficient parallelised computations, specifically accessing GPUs, were used in a plethora of variations either replacing or mimicking the regularisation term through deep-learning algorithms.[30-32] In this context the iterative and convolutional nature of LORAKS operators, AC-LORAKS in particular, allows viewing LORAKS itself as an autoregressive network. Thus, AC-LORAKS was implemented as a net-

work building block in an autoregressive ensemble approach for k-space interpolation.[33] Although implemented in PyTorch,[34] graphics processing unit (GPU)-specific details were not provided, computation times were not reported, and the achieved fidelity comes at the expense of computational overhead by adding multiple other ensemble blocks.

LORAKI replaced the explicit computationally demanding low-rank enforcement by a parametric recurring neural network (RNN).[35] The LORAKS prior thus is encoded in the RNN weights rather than using an explicit mathematical formulation. Furthermore, the RNN needs to be trained, which was reported with runtimes of 1 h for rather small slice dimensions (256×187) and 12 channels.[35] Usually, for inference, the computation time of the trained model is much faster and the process might be subject to similar optimisations shown for the original RAKI implementation,[36] but to the authors' knowledge, no such implementation was made available at the time of this study.

Our implementation uses PyTorch for parallel execution on GPUs, previously used to accelerate MRI reconstruction.[37-39] PyTorch's automatic differentiation engine[40] enables efficient gradient computation through the entire reconstruction pipeline. It allows treating the LORAKS reconstruction itself as a differentiable computational graph. Such a traceable LORAKS constraint can be combined with other regularisation terms,[41] or as a building block for physics-informed deep learning[42,43] or generative AI models[44] and allows novel optimisation approaches.

However, in our implementation, we intended to keep the inherent mathematical rigour of the low-rank mechanism building a learning-free framework, as the black-box character of deep-learning methods poses challenges in interpretability and evaluation of the models.[45]

In this paper, we present an enhanced LORAKS implementation designed to overcome computational limitations, enabling efficient joint-reconstruction of undersampled, multichannel, multi-contrast MRI images. We develop PyLORAKS, a Python-based LORAKS algorithm using PyTorch, enabling GPU utilisation. Further, we changed the operator construction in order to enable efficient linear memory access, and ensured PyLORAKS reconstruction similarity to, and benchmarked against the latest publicly available LORAKS MATLAB[46] version.[18]

To extend the LORAKS mechanism, we incorporated and tested recent rSVD methods and introduced an automatic backpropagation feature. Additionally, we introduce batching strategies for efficient leveraging of multichannel, multi-contrast data redundancy to enable joint-reconstruction of high-resolution data with feasible time and computational constraints.

Furthermore, we demonstrate a number of exemplary implications of our framework for MRI research. For a specific 2D multi-echo acquisition, we evaluate the choices of LORAKS parameters λ and r on joint-reconstruction to optimise the reconstruction and provide general recommendations for the parameter settings in PyLORAKS. Additionally, we demonstrate the ability for acquisition optimisation using our framework by computing an optimised sampling scheme and showing increased reconstruction quality, which is transferable to reduced scan time in high-resolution MRI acquisitions.

The steps involved in LORAKS reconstruction, together with modifications and additional features presented in this study, are sketched in Figure 1 for reference.

MATERIALS & METHODS

LORAKS can be regarded as a two-fold process: (1) building the LORAKS neighbourhood matrix (from k-space data $\boldsymbol{k}$) and (2) optimising a loss function to drive a low-rank matrix constraint and ensure data fidelity. The reconstruction of the 5D k-space data $\boldsymbol{k} \in \mathbb{C}^{N_x \times N_y \times N_z \times N_c \times N_e}$, consisting of three spatial dimensions ($N_{x,y,z}$), multichannel and multi-contrast data ($N_{c,e}$), with missing and zero-filled entries, becomes the following minimisation problem[3]:

$$\hat{\boldsymbol{k}} = \min_{\boldsymbol{k} \in \mathbb{C}^R} ||\boldsymbol{F}\boldsymbol{k} - \boldsymbol{d}||_2^2 + \lambda J_r(\boldsymbol{P}(\boldsymbol{k})) \tag{1}$$

Here, $\boldsymbol{F}$ is the sampling operator, which neglects missing entries and $\boldsymbol{d}$ is the actual acquired k-space samples. $J_r(\cdot)$ is a nonconvex regularisation penalty that enforces rank r on its argument. $\boldsymbol{P}(\cdot)$ is the LORAKS matrix operator, which constructs the LORAKS matrix from the k-space input by collecting neighbourhood patches around each spatial position in $\boldsymbol{k}$. LORAKS employs distinct types of neighbourhood matrix operators which impose different restrictions on k-space. For example, the C-operator requires limited support, and the S-operator imposes a smooth phase. We restrict the study to LORAKS reconstruction using the S-operator, since empirical findings suggest superior results across a range of applications.[27] However, all methods can be applied to the C-operator as well.[3] Without loss of generality, assuming $\boldsymbol{k}$ to be 2D slices of k-space data, we denote the matrix dimensions $\boldsymbol{S} = \boldsymbol{P}(\boldsymbol{k}) \in \mathbb{R}^{2m_x \times 2m_{ce}}$, where the factor of two arises from the S-operator's conjugate-symmetry construction.

The spatial dimension $m_x \sim N_x \times N_y$ includes all patches fully contained within the slice, where $N_{x,y}$ are the slice dimensions. The neighbourhood dimension m_{ce} depends on the square neighbourhood patch of size p_s. For the multichannel formulation, the channels are concatenated along this dimension, as are the contrasts for a joint-reconstruction setup.[13] Thus $m_{ce} = p_s^2 \times N_c \times N_e$, where $N_{c,e}$ are the numbers of channels and contrasts, respectively. In a typical clinical setting N_c is around 16 or 32 and N_e can be as high as 10 depending on the acquisition method. Thus, with a multi-echo acquisition of 5 echoes and 32 receive channels $m_{ce} = 4000$. We set $m_b = 2m_{ce}$ for convenience throughout this manuscript. The matrix can be seen in Figure 1, outlining the steps of the approach.

Several methods have been proposed in conjunction with least-square solvers[3] to find solutions to the minimisation problem in equation 1. Further computational improvements were made using the approximation[15]

$$J_r(\boldsymbol{P}(\boldsymbol{k})) \sim \|\boldsymbol{P}(\boldsymbol{k})\boldsymbol{V}\|_F^2, \tag{2}$$

that uses the Frobenius norm for an appropriate choice of the matrix $\boldsymbol{V}$. When the subsampled k-space contains autocalibration (AC) data, $\boldsymbol{V}$ can be extracted from the

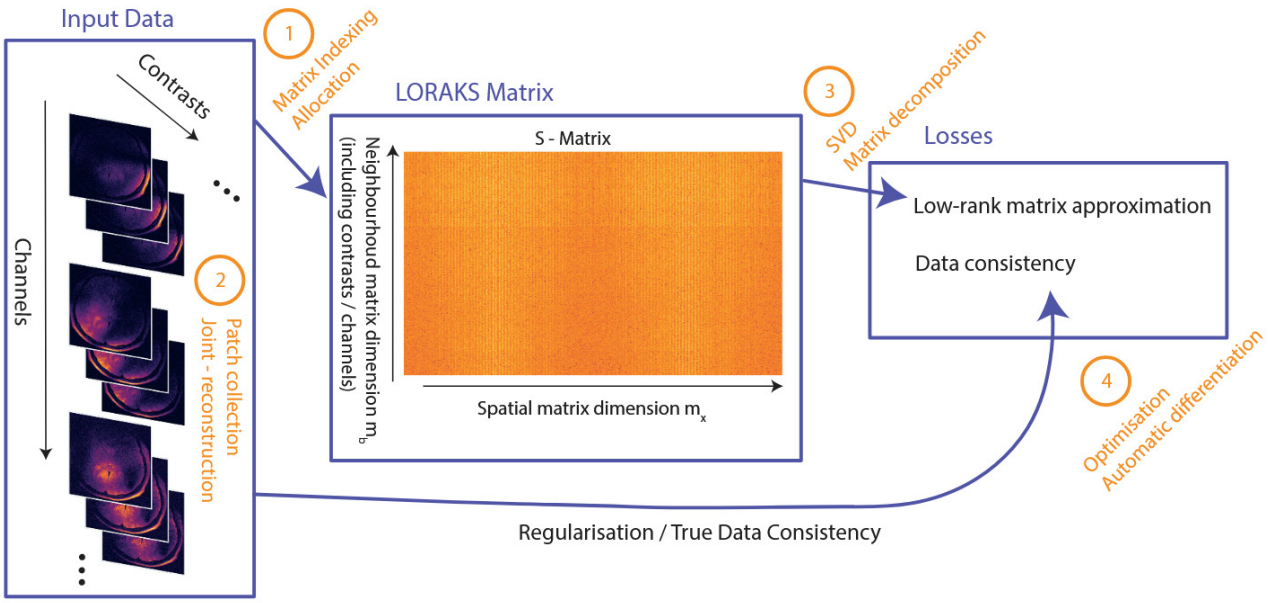


Figure 1. LORAKS reconstruction implementation steps.

The input data (left column) is processed per 2D-slice and arranged to combine multiple contrasts (2). Dependent on the projected memory size of the algorithm, the combined contrast and channel dimension is batched using a channel-correlation-based clustering (described in the data combination and batching subsection). The Casoratti-style LORAKS matrix can be built by indexing of the combined input data (1). The indexing was designed to allow linear memory usage and avoid loops for computational efficiency. The LORAKS matrix is used to enforce a low-rank constraint which is using a rank-revealing decompoisition like SVD. With increasing LORAKS matrix sizes, due to high-resolution input data or joint reconstruction of many channels and/or contrasts, this method is computationally costly. Huge matrix sizes can effectively be subjected to low-rank matrix completion using randomised methods (3). Finally, the used framework allowed the implementation of AI-style backpropagation which can be used for data-driven sampling optimisation (4).

nullspace of the LORAKS matrix built from the AC region $\boldsymbol{P}(\boldsymbol{k}_{\text{ac}})$.[15] The properties of the LORAKS matrix and the convolutional structure ingrained in the matrix norm in equation 2 allow for further computational gains by fast fourier transforms (FFT) to find the solution.[14]

We restrict this study to the fast AC-LORAKS reconstruction method, focusing on its practical applicability rather than providing an exhaustive comparison. The assumption of available AC data is well justified, since it underlies the majority of parallel imaging strategies, including widely used approaches such as SENSE[2] and GRAPPA,[1] for which a number of autocalibration methods like FLEET[47,48] exist. However, original loss-solving methods, LORAKS matrix operators (C and S - matrix) and calibrationless P-LORAKS have been included in PyLORAKS. Our motivation for PyLORAKS was its use for sampling optimisation and sequence development in our research, requiring low computation time to enable fast development cycles. Similar results can be achieved using PyLORAKS with other algorithmic variations as the improvements presented are not specific to the particular algorithmic choice.

DATA & HARDWARE

For evaluation of the algorithm and its features, we used different virtual data and *in vivo* MRI high-resolution data scans.

VIRTUAL DATA

RANDOM MATRICES

To simulate different S-matrix sizes in our experiments testing randomised matrix decomposition methods, we used `torch.randn()` to create random matrices $\tilde{\boldsymbol{M}} \in \mathbb{R}^{m_x \times m_b}$. We set $m_x = 86016 = 2 \times 224 \times 192$ as representative spatial LORAKS S-matrix axis for an image slice. The LORAKS neighbourhood dimension (representing concatenated channels and/or contrast) was varied from $m_b \in \{2 \times 200, \ldots, 2 \times 15000\}$, using 20 equally spaced values.

SHEPP-LOGAN

For memory and speed benchmarks of the PyLORAKS framework, we used a Shepp-Logan virtual data phantom[49] to simulate one slice of imaging data.

To vary the neighbourhood dimension m_b of the LORAKS matrix, we simulated different receive channel coil-sensitivities by placing N_c 2D Gaussian windows at random points within the imaging slice with a standard deviation of $\sigma \in \left[\sqrt{N_{x,y}/8}, \sqrt{N_{x,y}/3}\right]$, where $N_{x,y}$ is the respective image dimension. The number of receive channels was set to be $N_c \in \{4, 8, 12, \ldots, 36\}$.

Additionally, we assigned randomised decay rates to the different intensity classes in the Shepp-Logan phantom to generate a Shepp-Logan decay rate map R, which was used to simulate additional N_e relaxation contrasts. Thus, decay images were created multiplying the phantom by the factor $e^{(-N_e R(x,y))}$, varying $N_e \in \{2, 3, 4, 5, 6\}$.

Random Gaussian noise was added to the k-space data of each contrast and channel, respectively, using the `torch.randn()` function setting `torch.manual_seed(0)`.

The neighbourhood patch size was chosen to be $p_s = 5$ in PyLORAKS and a radius of $R = 3$ in Matlab for an equal number of patch elements. For the neighbourhood dimension defined by $m_b = 2m_{ce} = 2p_s^2 N_e N_c$, we get $m_{ce} \in \{400, \dots, 10800\}$ with varying N_c and N_e. The spatial dimension of the LORAKS matrix was varied with from $N_{x,y} = 100$ to $N_{x,y} = 280$, increasing by 10 at each step, and, therefore, $m_x \in \{10000, \dots, 78400\}$.

MRI DATA

For data-driven optimisations regarding the joint-reconstruction data combination and parameter and sampling optimisations, we used high-resolution *in vivo* data.

ACQUISITION

All data were acquired on a Magnetom Terra 7 T MRI scanner (Siemens Healthineers, Erlangen, Germany). We used an 8Tx/32Rx channel phased array RF head coil (Nova Medical, Wilmington, MA, USA). Note that the number of transmit (Tx) channels is irrelevant for the reconstruction considerations in this study. The transmit mode was set to Siemens "True Form" to produce a circularly polarized (CP) B_1^+ transmit field. For *in vivo* data, three participants (two females ages 20 y and 24 y, one male age 37 y) were scanned. The study was approved by the Ethics Commission at the Faculty of Medicine of Leipzig University and written informed consent was obtained.

We acquired a 2D multi-echo spin-echo (MESE) sequence designed using PyPulseq.[50,51] PyPulseq delivered a well-defined raw data stream, and the particular sequence was chosen for convenience to demonstrate multi-echo reconstruction. Additionally, PyPulseq enabled fast manipulation of sampling-scheme routines within the sequence.

We used an echo-train length of 8, an echo-spacing of 7.73 ms and a T_R of 4.5 s. Fully sampled data was acquired in 17m 15s with an FOV of $212\text{mm} \times 165\text{mm} \times 28\text{mm}$ and a matrix size of $304 \times 237 \times 28$.

SAMPLING

Retrospective undersampling was achieved using different masking of k-space data and zero-filling. The fully sampled dataset served as a reference. We used the following sampling patterns, which can be appreciated in Figure 12:

1. Pseudo-randomised sampling of points from a uniform distribution outside a central spherical AC region, without replacement.
2. Pseudo-randomised sampling of lines along one dimension from a uniform distribution outside the AC region, without replacement.
3. GRAPPA[1] sampling, i.e. skipping of lines outside the AC region.
4. Interleaved skipping of lines outside the AC region with respect to the echo dimension, a GRAPPA-style pattern with a more homogenous coverage across echoes.[52]

Only the GRAPPA method does not provide complementary sampling across echoes. The other methods capture different k-space lines across the various echoes in coherent or randomised fashion, as it was found beneficial in joint-reconstruction settings.[13,52]

For each pattern we created an AC region with 36 central lines. We used an acceleration factor of $a = 3$ in the data combination experiments. In our parameter optimisation experiments, acceleration factors $a \in \{2, 3, \dots 6\}$ were evaluated. For the sampling optimisation experiments we used a factor of $a = 5$, such that, due to the data size, each echo was sampled using 76 lines.

EVALUATION METRICS

In all comparison or optimisation experiments involving the *in vivo* data, we calculated evaluation metrics with respect to the fully sampled data reference. We used normalised mean squared error (NMSE) η_{nmse},[53] peak signal-to-noise ratio (PSNR) η_{psnr},[54] and the structural similarity coefficient (SSIM) η_{ssim}.[54]

COMPUTE SYSTEM

We used a dedicated Linux compute server with a 72-core Xeon Platinum 2.4 GHz CPU (4800 bMIPS) with 1 TB RAM and an NVIDIA l40S GPU with 46 GB RAM in all computational experiments.

Note, computation speed is dependent on used floating point precision (we used 32-bit precision) and the compute system, as well as underlying library implementations and backends. Thus, the presented results might not be directly transferable to other systems or software versions.

ALGORITHM INNOVATIONS & TECHNICAL IMPLEMENTATION

We used PyTorch[34] (version 2.4.1) and Python (version 3.10) to implement the PyLORAKS MRI reconstruction framework.[55]

We will refer to the original algorithm[3,14] implemented in Matlab[46] as M-LORAKS. Compared to M-LORAKS, a number of improvements were implemented, which will be outlined in the following sections. Additionally, the reconstruction result was tested against M-LORAKS to ensure comparable results.

EFFICIENT COMPUTATION AND GPUS

In modern computing multiple factors lead to an immense increase in efficiency for vectorised or parallel computations.[56-58] Hence, whenever possible, iterative or sequential loops should be avoided in favour of batched or vectorised operations, which are available in most modern scientific computing libraries. Additionally, using PyTorch, data and most computations can be transferred to GPUs (graphical processor units) for efficient and fast parallel computations. We used CUDA[59] (version 12.4) and associated libraries (libcublas, libcufft, libcusolver, libcusparse)

including vectorised functions for all presented matrix decomposition methods and other linear algebra operations.

LINEAR INDEXING FOR LORAKS MATRIX OPERATORS

M-LORAKS collects the LORAKS neighbourhoods of each k-space position within a circular neighbourhood by sequential iteration through all spatial k-space dimensions. This approach splits an indexing operation, i.e. the neighbourhood collection and rearrangement into a matrix form, into two steps. (A) the circular design necessitates a masking operation, and (B) the iterative collection uses a memory pre-allocation, which effectively is creating a copy of the input.

In our implementation, we used a linear indexing strategy to construct the neighbourhood matrix efficiently without excessive memory overhead, treating multidimensional array indices as offsets in the array's flat (linear) memory space. Our implementation leverages this by first computing the linear offsets for a rectangular single patch and then broadcasting those offsets across all valid patch positions.

Thus, LORAKS matrix operators can be implemented as a single indexing operation into the image-tensor, yielding the desired neighbourhood matrix of shape (number of patches) $\times$ (patch size). Vectorised counterparts are available to implement the back-projection equally efficiently. We create such linear index tensors once for the specific matrix operator using the mathematical formalisms detailed in Haldar.[3] A single copy of these indices is moved to the GPU device, together with a k-space reconstruction candidate and the measured k-space data (in condensed form, i.e. only sampled points), yielding a simple indexing operation on the GPU to create the LORAKS matrix operator.

Effective reconstruction has been demonstrated with M-LORAKS use of a circular radius of 3, which limits the matrix size.[14,18] Hence, we used $p_s = 5$ in the PyLORAKS framework to ensure similar neighbourhood sizes with rectangular patches. No effect on reconstruction performance was observable with the neighbourhood shape change. An example for the testing and validation is provided in the supplementary material.

JOINT-RECONSTRUCTION

LORAKS reliably reconstructs multichannel data,[4] if available using AC data for computational speedup (AC-LORAKS).[15] Beyond coil redundancy, joint reconstruction across contrasts further increases data fidelity.[13,52] We implemented PyLORAKS to use joint reconstruction capability by default and support contrast - and slice - dependent sampling masks, opposed to the original M-LORAKS implementation, where sampling was assumed to be equal across the input. Using the indexing scheme above, channels/contrasts were concatenated via reshaping (`torch.Tensor.view()`) without the creation of memory copies. In practice, iterative loops and masking were effectively avoided, the neighbourhood sampling is defined by the linear indexing method outlined above, while the channel and contrast combination is set using broadcasting and reshaping of the indexed data.

COMPUTATIONAL SPEED AND MEMORY BENCHMARK

The tests were set up on the same machine (see data & hardware section) using the PyLORAKS framework with CPU and GPU processing and `AC_LORAKS` (version 2.1)[14,18] as M-LORAKS in Matlab[46] aligning the algorithmic details (using the FFT approximation, alg=4, in M-LORAKS). We used a convergence tolerance of 1e-6 and maximum number of iterations of 5 in both cases to ensure speed benchmarks are not biased by early termination. Algorithmic performance was ensured to be similar to M-LORAKS in various test scenarios (up to numerical offsets expected with different software and due to the patch shape change). An exemplary test is provided in the supplementary material.

The timing was collected using Python `timeit` module and Matlab native functionality `tic - toc` with two warmup runs, and the results of three consecutive timed runs were averaged. Memory was measured using PyTorch functionality and the NVIDIA CUDA framework[59] in the case of GPU computations.

CPU memory allocation tracking was implemented using a subprocess tracking for both M-LORAKS and PyLORAKS CPU computations based on Linux `/proc` - reads and a child process collection to extract the combined virtual memory high-water mark (VmHWM) value across all subprocesses.

Virtual data was created using a Shepp-Logan phantom (see data & hardware section). The details of the subsampling and acquisition did not influence the optimisation process and performance benchmark, as those are only dependent on data sizes and the settings of the weighing factor $\lambda = 0.0$ and the rank, which was set to $r = \lfloor \max\{15, m_b/10\} \rfloor$.

To keep the implementations as comparable as possible, the `torch.linalg.eigh` function was used to extract $\boldsymbol{V}$ similar to the M-LORAKS code. Hence, the benchmarks did not include possible speedup by different rSVD variants (see fast SVD methods subsection) but showcased implementation improvements and GPU capabilities. We adapted the M-LORAKS code to allow for joint-reconstruction by concatenating channel and contrast dimensions and allowing non-uniform sampling schemes throughout the contrast dimension.

LOSS ESTIMATION AND AUTOMATIC DIFFERENTIATION

We implemented the minimisation of equation 1, subject to the approximation given in equation 2, using a nullspace estimated from the LORAKS matrix built from AC data $\boldsymbol{k}_{ac}$. Here, the LORAKS constraint is approximated by projection of the operator matrix against $\boldsymbol{V}$, which is obtained by computing the nullspace of the LORAKS operator matrix using only AC data as outlined in [15]. The nullspace is regarded to be identical to the eigenvectors associated with the $m_b - r$ lowest eigenvalues. To compute $\boldsymbol{V}$, the operator matrix was computed on the GPU device as outlined above (linear in-

dexing subsection) using AC-data, before proceeding in two different ways:

1. We implemented a gradient descent solver relying on PyTorch's automatic differentiation function[40,60] for backpropagating the gradients through the loss function. A k-space candidate (copy of the zero-filled input, and PyTorch gradients attached) was initialised on the GPU device, and direct minimisation of the loss function in equation 2 using gradient descent was used. For the descent we used a learning rate (from 0.1 to 0.001) decreasing linearly across the iteration steps.
2. We mirrored the default algorithm of M-LORAKS subject to the available performance optimisations.[3,14] That is, a fast approximation of equation 1 using FFTs was used and the backpropagation feature was not employed. This effectively reduced the optimisation to a least-squares problem for which an effective conjugate gradient descent (CGD) solver was used.

We present details on the advantages of both implementations.

DATA COMBINATION AND BATCHING

Additional to achieving computational performance increases by careful allocation and implementation with PyTorch, PyLORAKS memory demands need to be reduced. Especially limited GPU memory restricts feasible use in high-resolution joint-reconstruction. We favoured joint-contrast over multichannel reconstruction, since complementary sampling patterns allow for more homogenous k-space coverage.[52] I.e. we preferred batching the data in channel dimension when using joint-reconstruction if the available (GPU) memory was limited. That is, data was combined using all available contrasts and subsets of the available channels in batches.

Note, this is different from channel compression approaches, as all individual dimensions are reconstructed. We present the framework in this way for two reasons: (1) we use individual channel data in subsequent processing steps in our research, (2) adding coil compression can be regarded as a simple pre-processing step not impeding the results shown but rather facilitating further computational speedup. Coil compression was shown to be effective in conjunction with LORAKS reconstruction,[3,13,14,33,41] and a respective module is included in the PyLORAKS framework.

The batched channel dimension was maximised to meet memory limitations, such that fast GPU computation performance was possible, but the data batches need to be reconstructed sequentially.

This is profitable, as data post-processing is possible after reconstruction without the necessity to correct data alternation by the respective channel compression method. As such, dedicated magnitude or channel combination methods, or denoising methods like LCPCA[61,62] or NORDIC[63] can be applied to the reconstructed complex-valued data. However, the expense is that bigger data sizes need to be accommodated compared to using channel-compression algorithms.

To maximise the leveraged data redundancy, we implemented channel-batching based on coil-sensitivity correlations. The undersampled AC data, composed of N_{ac} k-spaces points across N_c channels, $k_c \in \mathbb{C}^{N_{ac} \times N_c}$ was used to calculate a distance using the channel correlation coefficients matrix $\boldsymbol{R}$:

$$\boldsymbol{R}_{ij} = \frac{\boldsymbol{C}_{ij}}{\sqrt{C_{ii} C_{jj}}}, \text{ with } \boldsymbol{C} = \mathrm{Cov}\left(\boldsymbol{k}_c^H, \boldsymbol{k}_c\right) \tag{3}$$

$$\tilde{\Sigma}_c = 1 - \boldsymbol{R} \tag{4}$$

Based on this, clusters were calculated to maximise mutual channel correlations within each batch using a KMeans clustering algorithm[64,65] and a pre-calculated cluster size n_{bc} depending on memory considerations. Thus, we obtained $n_b = N_c / n_{bc}$ clusters (ensuring integer division) $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_{n_b}\}$, using:

$$\mathcal{C} = \min_{\hat{\mathcal{C}}} \sum_{i=1}^{n_b} \sum_{j=1}^{n_{bc}} \left\| \boldsymbol{v}_j - \boldsymbol{\mu}_i \right\|^2, \tag{5}$$

where $\boldsymbol{v}_j \in \mathbb{R}^{N_c}$ is the j-th row of $\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{c}}$ and, $\boldsymbol{\mu}_i$ is the centroid of cluster i:

$$\boldsymbol{\mu}_i = \frac{1}{|\hat{\mathcal{C}}_i|} \sum_{j}^{n_{bc}} v_j \tag{6}$$

To evaluate the batching influence, we reconstructed retrospectively undersampled *in vivo* data (see data & hardware section), varying the used channel-batching method. The data was jointly reconstructed per batch across the channel-batches and the number of available contrasts. This procedure was done per-slice, as the channel correlations are dependent on the coil sensitivity profiles, which will vary across the extent of the input volume.

The data $\boldsymbol{Q} \in \mathbb{C}^{N_x \times N_y \times N_s \times N_c \times N_e}$ was reconstructed slice wise for the $N_s = 15$ slices. For each slice we combined batches of $n_{bc} = 4$ channels and all echoes, purposely choosing a low cluster size to evaluate the influence of the clustering. Hence, the input data contained of the matrices $\boldsymbol{Q}_i \in \mathbb{C}^{N_b \times N_x \times N_y \times N_n}$, where $N_b = N_c/4$ and $N_n = 4\, N_e$.

The algorithm was set up using a rank parameter $r = 120$ and the neighbourhood patch side-length was $p_s = 5$. We proceeded with two different reconstruction strategies, which differed only in the preparation of the data batches, while all other reconstruction parameters remained constant. That is, (a) the channel dimension was split in chunks of size 4 to form $\boldsymbol{Q}_i$ from the input data stream, and (b) the input data was subject to the channel-correlation-based batching with cluster size 4. This led to different channel combinations per slice.

The aim of this experiment was to evaluate the influence of mutual information across a batch. We hypothesised that using channels with a higher cross-correlation would yield increased low-rank matrix properties and thus possibly increased LORAKS performance, which would also support the use of coil compression techniques.

DATA SCALING AND FAST SVD METHODS

In addition to batching or compression strategies, other implementation details can be changed to accomodate increasing matrix sizes inherent in high-resolution joint-reconstruction MRI settings. Specifically, the computational

performance of the matrix decomposition at the core of the low-rank enforcement is poor if SVD is employed. Since SVD is a heavily used technique in many data-driven sciences, performant variations have previously been suggested.[17] Besides using the built-in `torch.svd_lowrank` method (LR-SVD)[26] which mirrors computational gains described in prior work,[25] we implemented matrix decomposition methods in PyLORAKS to increase computational performance. Since randomised variants of SVD do not specifically leverage or are dependent on the Hankel-like structure, and are suited for strong spectral features found in LORAKS matrices, reliable subspace capture is assumed in their deployment.[66]

The following variation of methods were implemented for benchmarking as they were found computationally beneficial compared to a standard SVD: a randomised SVD (rSVD) method[17] similar to LR-SVD, subspace-orbit randomised SVD (SOR-SVD),[16] and a randomised QLP (RAND-QLP) algorithm.[67]

All randomised matrix decomposition algorithms were previously found to benefit from using power iterations n_{iter} and oversampling o,[26] two parameters facilitating the randomised sampling and compression of the input data. In the process, a data matrix of reduced size is formed to which a QR decomposition is applied, resulting in the computational benefit. This matrix is of rank $m_r = r + o$, i.e. the complexity of randomised SVD variants scales with the target rank m_r, which is usually much smaller than the input matrix size ($\min(m_x, m_b)$) for joint-reconstruction of high-resolution input. We used empirically determined values $n_{\text{iter}} = 2$ and $o = 10$ for all randomised decomposition implementations, which have been found to increase estimation accuracy.[16]

The matrix $\boldsymbol{V}$ in equation 2 is estimated from the nullspace of the LORAKS operator matrix built from AC data ($\boldsymbol{M}_{AC}$), consequently minimising the use of matrix decomposition methods in AC-LORAKS to a single nullspace estimation. Increased computational efficiency previously was reached by using an eigenvalue decomposition on a Hermitian matrix $\boldsymbol{A} = \boldsymbol{Q}\,\mathrm{diag}(\boldsymbol{\Lambda})\boldsymbol{Q}^{\boldsymbol{H}} \in \mathbb{C}^{m_b \times m_b}$, $\Lambda \in \mathcal{R}^{m_b}$, built from $\boldsymbol{A} = \boldsymbol{M}_{ac}^{\boldsymbol{H}}\boldsymbol{M}_{ac}$, reducing the dimensionality to the shorter axis (usually m_b). The nullspace can be estimated using the trailing $m_b - r$ vectors $\boldsymbol{Q}[r:]$ of the eigenspace corresponding to the lowest eigenvalues. Thus, we included PyTorch *torch.linalg.eigh* function (EIGH) in the investigation of rank-revealing methods, as its Matlab counterpart was found to perform best in the M-LORAKS default implementation. An algorithm for randomised nullspace approximation was also tested (RAND-NS) estimating the trailing $m_b - r$ eigenvectors through randomised projections.[29]

First, we evaluated the speed and memory performance of the randomised matrix decomposition methods for our large matrix sizes. This was deemed necessary as the underlying computational libraries used for matrix algebra possibly switch to different backends dependent on the matrix input sizes.[68] Thus, randomised SVD variants might not perform close to their theoretically projected memory and time complexity. We used synthetic matrices (see data & hardware section) to benchmark the matrix decomposition. Theoretical predictions about the computational complexity scaling in time and memory are available.[16,17] For an instructive overview, Figure 5 shows the complexities reached in our settings.

Second, we compared the reconstruction performance of the PyLORAKS algorithm substituting the rank-revealing operation at its core. This reconstruction was done using retrospectively undersampled *in vivo* data (see data & hardware section). For this experiment only the methods LR-SVD, rSVD, and SOR-SVD were used, as faster computation times and lower memory footprints were found in the above benchmark.

We used these variants for the estimation of $\boldsymbol{V}$ in equation 2, and later approximation of the low-rank loss. For all randomised variants tested, we used a target rank $m_r = 4\,r$, with r being the rank parameter of the PyLORAKS algorithm. We ensured the target to be much smaller than the matrix size of the LORAKS matrix ($m_r \ll \min\{m_x, m_b\}$), as this defines the respective method's computational efficiency. We picked the trailing $n_{\text{eig}} = m_r - r$ eigenvectors to approximate the nullspace $\boldsymbol{V}$.

Both of these benchmark experiments were solely done using the GPU capability.

NEW ADVANCES AND POSSIBILITIES FOR MRI

Our target was a computationally efficient LORAKS implementation considering memory and time constraints. Thus, additionally to the presented methods, PyLORAKS can be used as drop in replacement of the current M-LORAKS implementation to offer large computational boosts. For example, the computationally demanding A-LORAKS-CS[24] or the OEDIPUS[69] framework can benefit from efficient and fast LORAKS computations. Furthermore, the capability of backpropagation through the algorithm makes PyLORAKS a suitable candidate to be used in physics-informed deep learning[42,43] or generative AI models,[44] separating our implementation from previous attempts to algorithm computation feasibility enhancement,[38,39] enabling new features.

To demonstrate the capabilities of PyLORAKS, we used it for a multi-echo 2D acquisition. We evaluated the choice of algorithm parameters r and λ on reconstruction quality, as well as for the design of an optimised sampling pattern. The presented methods generalise to arbitrary acquisition schemes.

PARAMETER SELECTION

Usually, the value of the algorithm parameters r and λ are manually chosen by the user, and its influence is assessed empirically after reconstruction.[14,20,21] In conjunction with J-LORAKS this procedure is impractical due to low computation speed and high memory demand.

The LORAKS algorithm enforces the LORAKS operator matrix to be of rank r. A low-rank parameter r encourages effective filling of missing data and denoising by reducing the variance of the available data based on the used LORAKS operator, but can introduce bias if too low rank enforcement is used.[3] While the denoising capabilities of LO-

RAKS have been demonstrated,[3,13,15] no head-to-head comparison for LORAKS denoising capabilities compared to dedicated denoising algorithms is available. However, recent studies suggest that for optimal control and results of image denoising, dedicated algorithms should be preferred.[21,63,70]

On the other hand, high rank favours greater data fidelity but may fail to exploit the low-rank prior, which comes at the cost of acquisition acceleration capability. Other effects on the algorithm are more intricate. For example, in AC-LORAKS a higher rank parameter reduces the size of $\boldsymbol{V}$ extracted from the nullspace of the LORAKS matrix used in equation 2, and thus reduces the computational cost *after* extraction of $\boldsymbol{V}$.

Similar effects can be seen with changes in the regularisation parameter λ. If the data signal-to-noise ratio is sufficient, λ can be set very low, which was found to be beneficial for interpolation of missing data without modifying acquired samples.[13] Alternatively, in the true data consistency formulation $\lambda = 0.0$ of the algorithm, the LORAKS constraint is calculated for missing data points only. This is preferable for computational reasons, as the computation iterations are only done over a reduced subset of the overall k-space.

Automation of the LORAKS parameter choices has been proposed using SURE and a soft-thresholding approach[71] in conjunction with compressed-sensing and total variation constraints,[24] at the cost of additional computational overhead. SURE allows calculating the mean squared error in estimating a true signal from measurements corrupted by additive Gaussian noise[22] using only acquired samples. Consequently, SURE has been used in the context of denoising algorithms,[72] and was extended to reconstruction algorithms when the constraint formalism resembles linear denoising operators,[73] for which analytic expressions have been proposed.[23] This is possible for the independent LORAKS constraint[24] ($J_r(\boldsymbol{P}(\boldsymbol{k}))$ in equation 1).

However, the full LORAKS implementation involving data consistency and constraint regularisation (via λ) is nonlinear, typically necessitating Monte Carlo or Hutchinson methods to estimate the operator divergence further adding computational complexity. Furthermore, finding more complex dependencies (e.g. LORAKS rank dependence on matrix size or sampling pattern) might not be possible using SURE, as e.g. divergence with respect to a sampling pattern is not reasonably defined. Nonetheless, we included a comparison with a SURE optimised rank r_{SURE} using a method similar to Ilicak, Saritas, and Çukur,[24] briefly described in the supplementary material. For this other parameters (complementary subsampling scheme, number of contrasts and $\lambda = 0$) were kept fixed and were evaluated by the method below.

It is common for MRI studies that the data sampling method and the basic data structure (dimensionality, FOV) remain similar between acquisitions. An optimisation of the algorithm parameters and evaluation of their dependencies can be tied to the specific acquisition and is a possible prior to reconstruction of individual data acquisitions. We evaluated the reconstruction performance with respect to the algorithm parameter settings of r and λ.

We used a fully sampled and retrospectively subsampled *in vivo* dataset obtained from the multi-echo spin-echo (MESE) acquisitions (see data & hardware section). The PyLORAKS joint-reconstruction was followed by a root-sum-of-squares (RSOS) channel combination to obtain magnitude data.

Since the autograd framework is not suited for mixed integer optimisation, we chose a Bayesian sampling optimisation method[74] within the `wandb` framework[75] (version 0.17.3). Thus, the presented parameter investigation is enabled by the computational gains of PyLORAKS. To vary the neighbourhood matrix size m_b of our input data, we varied the channel batch size n_{bc}. In our optimisation, the reconstruction time for one optimisation run, for a single slice of the input data ($304 \times 237 \times 32 \times 8$), and joint-reconstruction took ~ 5 s on average. It is varying with the choice of m_b.

We introduce an overall minimisation score to optimise the input parameters r and λ with respect to all of our metrics, scaling each of the individual metric values accordingly:

$$\min_{r,\lambda} \ \eta_{\mathrm{loss}} = -\eta_{\mathrm{ssim}} - 0.01\,\eta_{\mathrm{psnr}} + \eta_{\mathrm{nmse}} \tag{7}$$

The evaluation followed three distinct aims with respect to the algorithm input parameters r and λ:

1. Find correlations between the number of missing points and the parameters.
2. Find correlations between the used input matrix dimensions and the parameters.
3. Evaluate the influence of the undersampling pattern with respect to variation in the parameters.

This assessment is specific to the used sequence and hardware due to the available number of channels and contrasts in the joint-echo reconstruction, the individual contrast content, and SNR. Nevertheless, a number of general conclusions can be drawn from the optimisation procedure.

SAMPLING OPTIMISATION

PyTorch's autograd framework, described in the loss estimation and automatic differentiation section, allows direct gradient computation via backpropagation, with additional support for complex valued differentiation using Wirtinger calculus,[76,77] essentially tracking gradients in each computation step and applying the chain rule. Further information can be found in.[40] This feature was used to inform the sampling density of the input data using a single forward and backward pass of the reconstruction algorithm. We used fully sampled data, assuming true data consistency (i.e. only computing the LORAKS regularisation part in equation 1), to compute the gradient of the PyLORAKS loss function with respect to each sampled point.

We denote the loss function in Equation 1 with $\mathcal{L}(k)$ and compute the gradient with respect to the input by propagating through the loss objective in a single pass, obtaining PyTorch gradients attached to the k-space candidate:

$$g(x, y, c, e) = \nabla_k \mathcal{L}(k) \in \mathbb{C}^{N_x \times N_y \times N_c \times N_e} \tag{8}$$

Taking $k \sim k_{\text{true}}$ by considering a fully sampled k-space input, we note that $\nabla_k \mathcal{L}(k) \sim \mathcal{L}(k + \delta e_i) - \mathcal{L}(k)$ can be obtained from the Taylor expansion around k, using δe_i to indicate a small perturbation at point i. Thus, under the assumption of smoothness and linearity in local k-space subregions, the gradient amplitude can be regarded as a first order predictor of the change induced by corrupting (or removing) sample i. Hence, equation 8 serves as a local sensitivity measure of the reconstruction loss to each sample under the imposed low-rank constraint.
We interpret a larger $|g|$ as a sign of greater importance of the respective samples to the image reconstruction problem. Collapsing over channels yields:

$$s(x, y, e) = \sum_{c=1}^{N_c} \left| g(x, y, c, e) \right| \tag{9}$$

This way we can deduce a sampling density mask across the input slice. Since in our case, we were interested in subsampling along the phase encode dimension of a 2D sequence, we aggregated to obtain the phase encode density per index i_y and formed a probabilistic sampling density per echo:

$$d_e\left(i_y\right) = \sum_{k_x} s\left(k_x, i_y, e\right) \tag{10}$$

$$w_e\left(i_y\right) = \frac{d_e\left(i_y\right)}{\sum_{k_y} d_e\left(k_y\right)} \tag{11}$$

Complementary echo sampling has previously been shown to increase the reconstruction performance of LORAKS.[13, 52] Thus, we sampled a mask from w_e per echo using an adaptive sampling method[78] to ensure complementarity across echoes.

We applied this procedure to fully sampled *in vivo* MESE data from each participant (see data & hardware section) to compute one-pass gradients of PyLORAKS and derive an average $w_e(i_y)$. Averaging across subjects yields a nominal density for the MESE 2D sequence. Because the gradient field is contrast- and sequence-dependent, the resulting density should be viewed as a data-driven guideline that can be recomputed for other sequences or contrasts.

The obtained sampling pattern was used for retrospective undersampling of the data and compared to the four other 2D sampling strategies given in the previous section. We used 36 AC lines and an acceleration factor of 5 with respect to the coverage outside this region. Hence, due to the data size, each echo was sampled using 76 lines.

The undersampled data was subjected to PyLORAKS joint-reconstruction using the same parameters in all reconstruction runs. The channel batch size was set to $n_{bc} = 8$. LORAKS parameters where $\lambda = 0.0$, i.e. true data consistency, and $r = 150$.

The resulting reconstructions were Fourier transformed and channel-combined using an RSOS combination to obtain magnitude image data. We used the same metrics to assess the reconstruction quality: NMSE, PSNR, and SSIM.

We conducted an additional validation experiment. For each k-space point i in the fully sampled dataset, we removed the sample by inserting 0, and measured the degradation to the full k-space reconstruction, using η_{NMSE} (as well as η_{LOSS} defined in equation 7:

$$\Delta_i = \eta(k_{\text{fully sampled}}, k_{i,\text{dropped}}) \tag{12}$$

Effectively, we measured the k-space interpolation ability of LORAKS for each point i, and we take Δ as the ground truth for the location importance to compare against the density obtained via gradient propagation in Equation 11 by calculating Pearson and Spearman correlation coefficients.
This approach is computationally exhaustive, necessitating a full reconstruction run for each k-space point. Thus, we used some simplifications for the comparison: We adapt aggregating the readout direction and channels, as in the gradient-based optimisation method (Equation 9). Hence, we are calculating Equation 12 per phase encode, zero-filling each full line at a time.

The reconstruction was done using the same settings as above (channel batch size $n_{bc} = 8,\ \lambda = 0.0, r = 150$) and took $\sim$ 2 h per slice iterating through all phase encode lines per echo, except for lines in the AC region as this would yield erratic AC-LORAKS performance. For comparison, the single-pass computation using propagation of Equation 8 took $\sim$ 4 s per slice.

RESULTS

ALGORITHM INNOVATIONS AND TECHNICAL IMPLEMENTATION

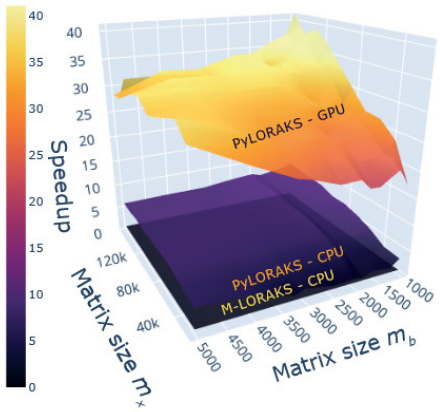
Different tests and validation runs demonstrated that the implementation of PyLORAKS achieved the same reconstruction fidelity as M-LORAKS when using equal algrotihm parameters. An exemplary comparison can be found in the supplementary material.

COMPUTATIONAL SPEED

The PyLORAKS implementation offers faster computation compared to M-LORAKS. The speed differences between PyLORAKS in CPU and GPU processing modes versus the (CPU only) M-LORAKS implementation can be found in Figure 2.

A great improvement was achieved using PyLORAKS compared to M-LORAKS irrespective of matrix sizes. Across all benchmarked input matrices we found an average speed increase of 5.75 using PyLORAKS and CPU computations. However, the highest speedups were observed when increasing spatial matrix size m_x and fairly small or moderate neighbourhood matrix dimension m_b. This is presumably due to the improvements in linear memory layout and parallelisation of computations. The performance of the actual matrix decomposition algorithm (`eigh` in this case) was not assumed to vary drastically between Matlab and Python implementations and scaled with the smaller matrix dimension, i.e. m_b.

The PyLORAKS GPU implementation resulted in a substantial acceleration of computations, yielding an average speed increase of 29.11 across all tested matrix sizes compared to M-LORAKS. Note that due to the high computational demands, LORAKS is commonly used on a per-slice computation iteration (in M-LORAKS and PyLORAKS). Hence, the per-slice time savings presented here addition-



Implement.	$m_x \times m_b$	Time [s] Mean ± Std.	Speedup
Mat. - CPU	96000 x 1600	6.26 ± 0.05	1.00
Py. - CPU	96000 x 1600	1.02 ± 0.01	6.16
Py. - GPU	96000 x 1600	**0.230 ± 0.002**	**27.5**
Mat. - CPU	96000 x 5000	34.96 ± 0.20	1.00
Py. - CPU	96000 x 5000	5.65 ± 0.04	6.19
Py. - GPU	96000 x 5000	**1.10 ± 0.01**	**31.79**
Mat. - CPU	156800 x 1600	10.11 ± 0.07	1.00
Py. - CPU	156800 x 1600	1.42 ± 0.01	7.14
Py. - GPU	156800 x 1600	**0.280 ± 0.002**	**35.85**
Mat. - CPU	156800 x 5000	50.62 ± 0.31	1.00
Py. - CPU	156800 x 5000	8.21 ± 0.07	6.16
Py. - GPU	156800 x 5000	**1.78 ± 0.01**	**28.4**

Figure 2. PyLORAKS performance speed benchmarking data.

Two m_x and m_b matrix sizes, respectively, were selected from the benchmark, and the computation times are provided in the table to the right. For each matrix size, the last column gives the fraction of M-LORAKS computation time over the respective PyLORAKS computation time, which is also depicted in the plot to the left.

ally scale with the number of slices to process, considering whole volume reconstruction. The resulting speed improvements mark a step towards real-time-imaging applicability of joint-reconstruction PyLORAKS, with the computational speed well below 1s for scan sizes encountered in high-resolution imaging (see Table 2).

Furthermore, the matrix sizes are given by $m_x \sim 2 \times N_x \times N_y$ and $m_b = 2 \times N_c \times N_e \times N_b$. The biggest matrix sizes given in Table 2 correspond roughly to a data size (without slice dimension) of $280 \times 280 \times 32 \times 3$, i.e. a scan using a 32-channel receive coil and three contrasts, or similar configurations if batching is used.

MEMORY

The memory allocation of the algorithm was tracked, including the data loading to RAM and the overhead created by the software (e.g. memory allocation for matrix decomposition operations). The resulting benchmark is visualised, and a subset of the numbers are given in Figure 3. The projected memory complexities match literature values.[16,17]

Averaged across all tested matrix sizes, we found PyLORAKS to be using only 21.2% of the memory compared to M-LORAKS for the same computation when using CPU processing. This is presumably due to increases in efficiency of memory allocations and linear indexing and the absence of any loops in the core functionality. The memory usage ratio compared to M-LORAKS dropped even more for big matrix sizes, to as low as 11% in some cases, as can be seen in Figure 3.

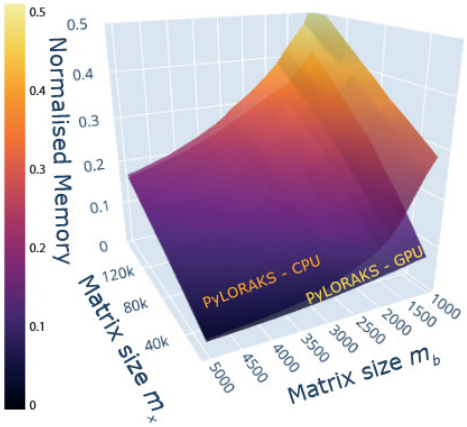
In both cases modern hardware might be able to efficiently use parallel CPU computing to process datasets. However, our memory tracking was set up to follow all child processes created by the algorithms. Thus, if hardware memory is limited, the processing might not be done in parallel but sequentially and is possible with smaller available total memory than the values presented here, at the cost of increased computation time.

For the PyLORAKS GPU computation mode an even smaller memory footprint was achieved, using only 16.4% compared to the M-LORAKS processing when averaged across all tested matrix sizes. Again, the memory usage ratio compared to M-LORAKS was lower for big matrix sizes, reaching below 11% in some cases, as can be seen in Figure 3. The drop compared to the PyLORAKS CPU mode is due to careful transfer of only necessary data to the GPU RAM, which on average saves around 1 GB of memory. However, it follows that PyLORAKS GPU mode allocates this amount of memory additionally on CPU memory, primarily for loading in the data.

Note, AC-LORAKS is based on an approximation of $\boldsymbol{V}$, which decreases in size with increasing rank. This way, the allocated memory is not only dependent on the input data size but also the algorithm parameters (r and λ).

DATA COMBINATION AND BATCHING

The reconstruction of the data using different channel combination methods in conjunction with batching along the channels can be seen in Figure 4. We compared the reconstruction to the fully sampled data after brain masking, and the derived metrics can be found in Table 1. The chan-



Implementation	Size ($m_x \times m_b$)	Mem. [GB]	Ratio
Matlab - CPU	96000 x 1600	9.25	1.00
Python - CPU	96000 x 1600	3.01	0.33
Python - GPU	96000 x 1600	**2.15**	**0.23**
Matlab - CPU	96000 x 5000	161.93	1.00
Python - CPU	96000 x 5000	18.05	0.11
Python - GPU	96000 x 5000	**17.08**	**0.11**
Matlab - CPU	115600 x 1600	10.56	1.00
Python - CPU	115600 x 1600	4.45	0.42
Python - GPU	115600 x 1600	**3.52**	**0.33**
Matlab - CPU	115600 x 5000	169.40	1.00
Python - CPU	115600 x 5000	28.99	0.17
Python - GPU	115600 x 5000	**27.89**	**0.16**

Figure 3. PyLORAKS performance memory benchmarking data.

Two m_x and m_b matrix sizes, respectively, were selected from the benchmark, and the profiled memory is provided in the table to the right. For each matrix size, the last column gives the fraction of PyLORAKS memory footprint over the respective memory used by M-LORAKS, which is also depicted for the plot to the left. Compared to M-LORAKS, the highest memory savings were be achieved with relatively small spatial dimensions m_x and large neighbourhood dimensions m_b. We randomly tested for variability in the estimates for a small subsample of the data but found no significant changes ($std < 0.01$).

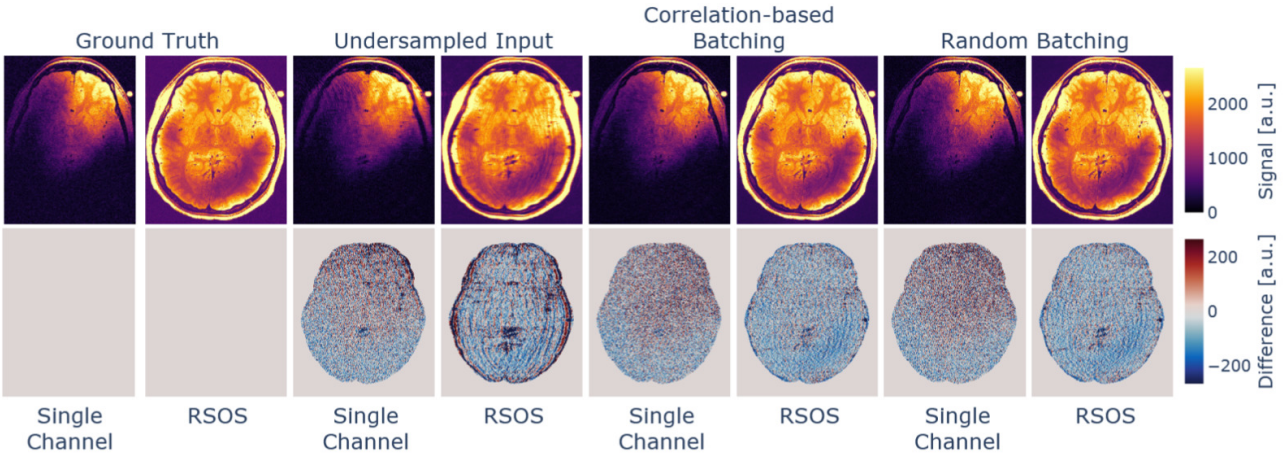


Figure 4. Reconstruction performance using different channel batching strategies for limited memory in joint-reconstruction.

In the first row, we show magnitude data for each reconstruction with one randomly selected channel and the RSOS combination of all channels per column respectively. As ground truth data, we used a fully sampled data set (first two columns), which was retrospectively undersampled (second and third columns). The bottom row shows the difference of the fourier transformed output to the ground truth data. Similar reconstruction results can be achieved for random batching of channels (two rightmost columns) and the proposed channel-correlation-based method (fifth and sixth columns), with incrementally reduced difference to ground truth visible for the latter.

nel-correlation-based batching resulted in increased performance metrics, while the difference in the reconstruction is hard to notice visually. This can be regarded as a minor performance boost but likely will not suffice, e.g., to be traded for higher acceleration in the acquisition process.

DATA SCALING AND FAST SVD METHODS

For optimisation of the loss function in 1, the rank-revealing decomposition (SVD or eigenvalue decomposition) can be substituted by a randomised variant more effective for big matrix sizes. The performance of each method was tested separately, processing the decomposition on simple input matrices. The speedup of the respective method was

Table 1. Comparison metrics of different channel batching strategies for joint-reconstruction.

Batching Method	PSNR	NMSE	SSIM
Under-sampled data	33.254	0.027	0.9249
Random channel batching	39.135	0.0696	0.9713
Channel correlation clustering	**39.209**	**0.0688**	**0.9724**

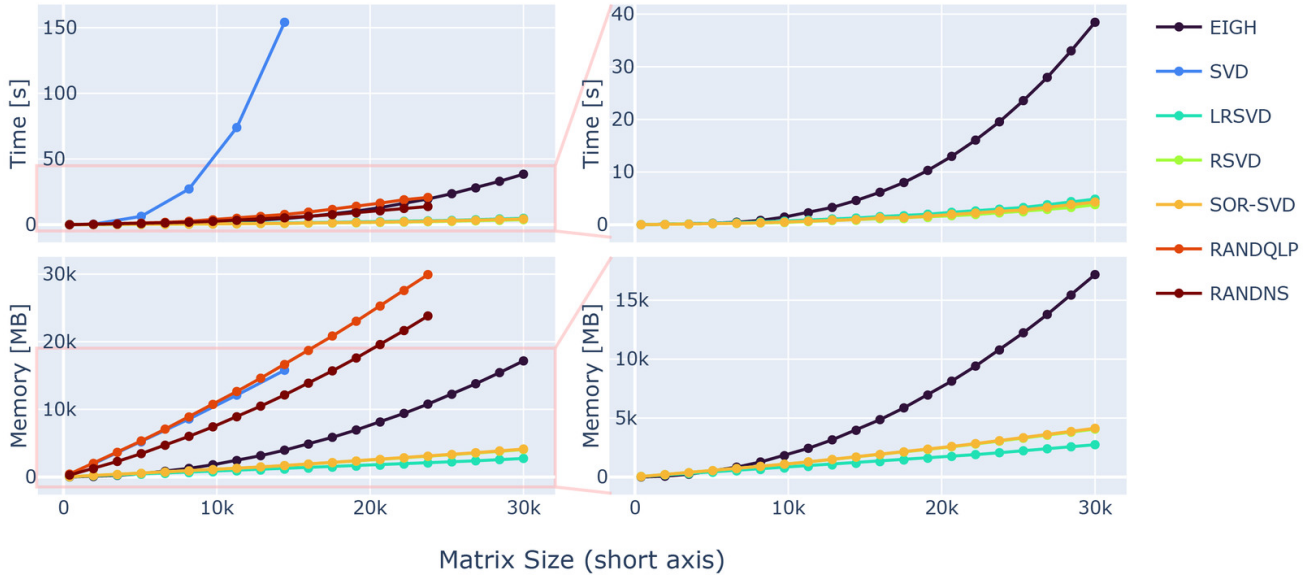


Figure 5. SVD variant benchmark.

Computation time (top row) and maximum allocated memory (bottom row) are given for different rank-revealing matrix decompositions using random input matrices with various sizes. The standard SVD algorithm performs poorly with respect to computation time, showing a $O(m_b) \sim m_b^3$ complexity with respect to the input. Additionally, the memory consumption scales linearly with the input size. While other variants showed much increased computation times, the RANDQLP and RANDNS algorithms also show comparable memory consumption. The inset on the left column focused on variants with increased computation times and much lower memory consumption (EIGH, rSVD, SOR-SVD), zoomed in on the right column. For increasing matrix dimensions, it is clear that the EIGH algorithm is outperformed by the randomised variants.

substantial and can be appreciated in Figure 5. For growing matrix sizes, an SVD scales linearly in Memory usage and quadratically in computation time.

The memory overhead of randomised versions is dependent on the target rank m_r and an oversampling parameter o, making memory saving possible, as the memory allocation overhead of randomised SVD variants is dependent on the chosen target rank. Hence, memory allocation of the respective randomised methods is possible in a controlled manner, e.g. tailored to the available hardware.

While the EIGH does offer considerable performance increase ($\sim 17\times$ faster) compared to the standard SVD, it falls short compared to other randomised rank-revealing decomposition alternatives (see 5, right column). While the RAND-NS and RAND-QLP methods offer considerable speedup, close to what can be achieved with the EIGH method, their demand on memory is similarly high as standard SVD methods.

Further speed increases are possible by using the proposed rSVD,[17] LR-SVD[26] or SOR-SVD[16] methods, offering a $\sim$ 4-fold speed increase for the biggest matrix dimension used in the test comparison to EIGH. The computational efficiency is projected to be even greater with higher-dimensional input. Thus, the use of randomised SVD variants is preferable, especially with increasing matrix sizes encountered in high-resolution MRI reconstructions.

Similarly, EIGH, rSVD, LR-SVD and SOR-SVD are all preferable in terms of memory consumption. Where again the former shows deficits, with more than 3-fold more memory demand for big input matrices. The memory consumption agrees well with complexity estimates given in literature.[16,17]

We compared the reconstruction performance of the SVD variants with favorable memory and time demands, which is shown in Figure 6, by substituting the matrix decomposition operation within the algorithm with the respective method. All tested randomised SVD variants showed similar reconstruction performance to the EIGH (default in M-LORAKS) method, the used metrics are shown in Table 2. While the LR-SVD method appears to be optimal with respect to the NMSE and PSNR, the original EIGH method achieves the highest SSIM score. However, rSVD and SOR-SVD were able to achieve $\sim 5-7\%$ quicker computations for the dataset used.

A substitution of the rank-revealing operation when using LORAKS reconstruction for high-dimensional input data was beneficial computationally and resulted in comparable image quality as the standard LORAKS reconstruction implementation. The computational benefit of using

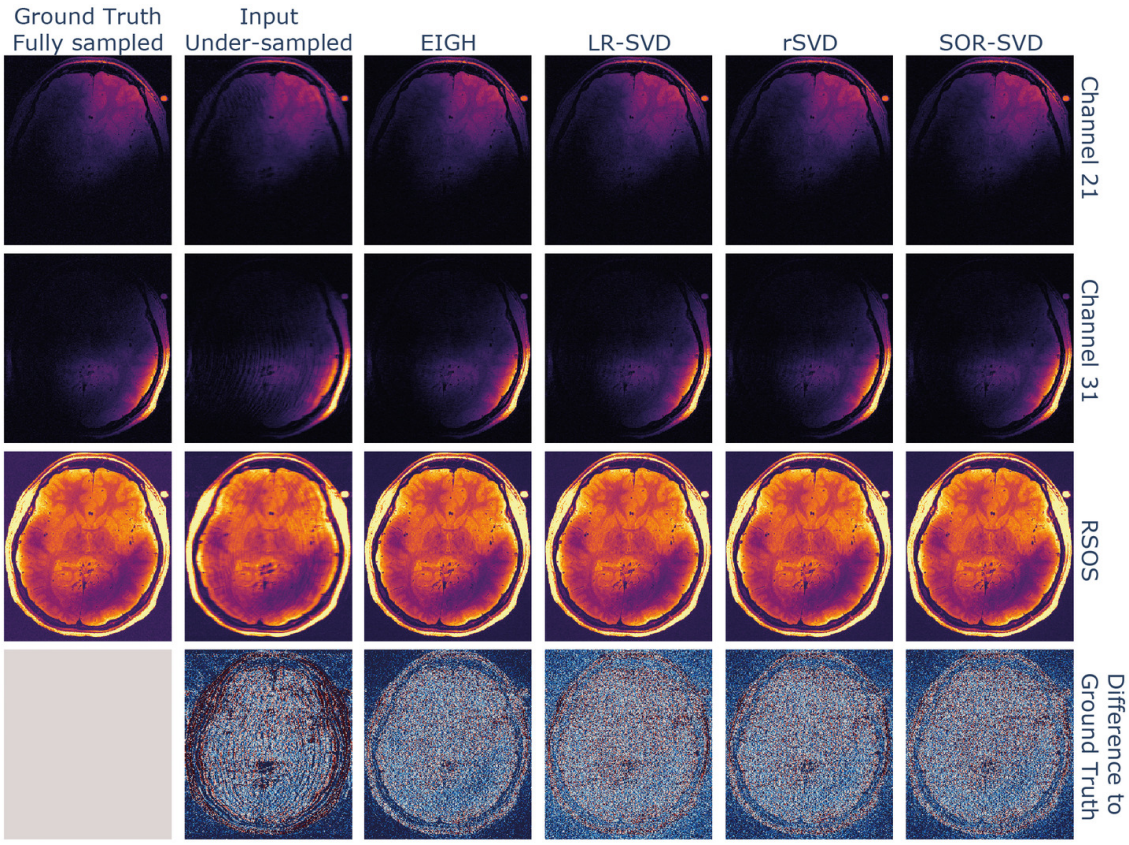


Figure 6. SVD variant reconstruction performance.

A fully sampled dataset was used as ground truth (first column). Data is shown for two randomly selected channels (first two rows) and the RSOS channel combination (third row). The difference of the channel combined magnitude image to the ground truth is given in the bottom row. Retrospective under-sampling (second column) was used and subject to PyLORAKS reconstruction using different rank-revealing decompositions (respective following columns). The result appears similar in all reconstructions and close to the ground truth. The LR-SVD (4^{th} column) shows the lowest error.

Table 2. SVD variant performance for the joint-reconstruction of MESE data using PyLORAKS' GPU capabilities.

Matrix decomposition method	PSNR	NMSE	SSIM	Computation Time [s]
EIGH	38.6960	0.0068	**0.9713**	4.2
LR-SVD	**39.6522**	**0.0055**	0.9693	4.9
rSVD	39.1998	0.0061	0.9704	**3.9**
SOR-SVD	39.2340	0.0060	0.9685	4.0

randomised SVD variants is dependent on the number of nullspace eigenvectors used for the nullspace approximation (here $n_{\text{eig}} = 3r$), as well as the matrix sizes in general. Presumably, even higher speedup is possible given the randomised SVD benchmarks in Figure 5 when bigger matrix sizes are involved or with optimising the dimensionality used for nullspace approximation.

NEW ADVANCES AND POSSIBILITIES FOR MRI

The results of the acquisition optimisation for our MESE sequence are presented in the following, using different features of the PyLORAKS framework, such as its backpropagation capabilities.

PARAMETER SELECTION

We optimised the reconstruction algorithm parameters for the highest quality reconstruction of our data compared to a fully sampled dataset. We obtained parameter trendlines for r, λ and the used sampling patterns for a 3-fold acceleration, which can be seen in Figure 7. Less heterogeneous k-space coverage across the sampling pattern for joint contrasts, like in GRAPPA, reduced the resulting reconstruction quality. Using random-line sampling or interleaved skipping of outer k-space lines across different contrasts yielded the lowest loss values, which is in line with previous findings of performance increases with complementary echo sampling.[13,52]

We found optimality in all metrics using the true data consistency formulation ($\lambda = 0.0$) over the regularised version, updating only missing data points in the reconstruc-

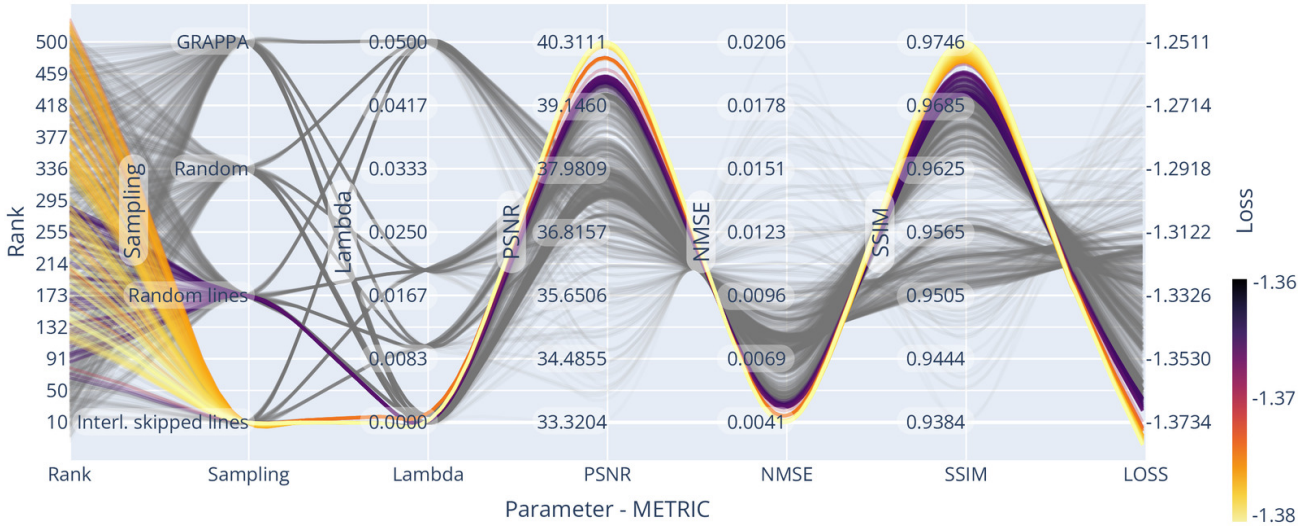


Figure 7. Parameter optimisation trendlines.

The three varied input parameters are given on the left (Rank, Sampling, and Lambda) and their influence is plotted against the three evaluation metrics PSNR, NMSE and SSIM, and a combined overall loss score given in equation 7. Randomised lines and interleaved skipping of outer lines per echo resulted in higher scores compared to grappa or random sampling. For all sampling methods, the true data consistency formulation ($\lambda = 0.0$)) provides optimality. The rank parameter values vary between 120 and 200 and, dependent on the used sampling scheme, are optimal at 143 for the interleaved line sampling.

tion. Whenever noise suppression and handling of imperfect data was not one of the main aims in the reconstruction or could be achieved with dedicated processing steps, the true data consistency formulation offers advantages.

For example, for all considered sampling patterns except GRAPPA, setting $\lambda = 0.0$ resulted in the highest SSIM and PSNR and the lowest NMSE. On average the NMSE increased $\geq 10\%$ for $\lambda = 0.001$ compared to $\lambda = 0.0$ and much higher values for higher settings of λ. Similarly, SSIM decreased $1 - 2\%$ for $\lambda = 0.001$ compared to $\lambda = 0.0$, and again even lower values were found for higher settings of λ. For the GRAPPA sampling, low values of $\lambda = 1e - 3$ resulted in loss minimisation and a small advantage over the true data consistency setting, which corresponds to other findings[13] for GRAPPA acquisitions.

However, GRAPPA sampling was found to underperform compared to other sampling methods, possibly due to the lack of complementary sampling across echoes. For example, the NMSE was $\eta_{\text{nmse}} = 0.0077$ and SSIM was $\eta_{\text{ssim}} = 0.955$ for GRAPPA sampling ($\lambda = 1e - 3$), compared to $\eta_{\text{nmse}} = 0.0049$ and $\eta_{\text{ssim}} = 0.972$ for echo - interleaved skipped line sampling ($\lambda = 0.0$).

The Bayesian optimisation framework was arriving at (or approximated) the optimum already after few iterations (~ 15). Thus finding a close to optimal rank parameter for a single slice was possible in ~ 75 s, totalling ~ 35 m for the full volume. We included trendlines from a more exhaustive search for visualisation purposes in Figure 7, Figure 9 and Figure 10 to spot trends more easily.

Optimal rank parameters for all subsampling schemes were in similar ranges ($120 - 200$). The optimal loss for joint-reconstruction of our input (slice dimensions $304 \times 237 \times 32 \times 8$ with batched channels) was found using echo - interleaved skipping of lines $\lambda = 0.0$ and the rank $r = 143$. For comparison, we used the SURE optimisation approach (see the supplementary material) on AC data of the same input, yielding an optimal rank value of $r_{\text{SURE}} = 141$.

If the rank parameter is selected to be too low, the increase in the loss is accompanied by intensity variations in the reconstructed images not immediately evident if no ground truth for comparison is available. If further processing involves physical modelling of the data, as e.g., in our case the estimation of quantitative R_2 values, this will degrade the robustness of the estimates. This influence is depicted in Figure 8 and is presumably caused by mixing of contrasts, as for MESE data, late echoes show a higher dynamic range between white matter and ventricles compared to early echoes, which is visible in the data. The effect might be more subtle and might not disappear the same way as e.g. aliasing artefacts with increased rank settings.

All evaluated metrics, however, approach an asymptotic value below the respective optimum with increasing rank parameter. Thus, using a higher rank parameter does not affect the reconstructed image in the same way. With increasing rank parameter the reconstructed image appears to be more granular, but intensity variation artefacts were absent, as also visible in Figure 8. Thus, we recommend using rather high rank parameter settings.

We evaluated the influence of rank, matrix size, and acceleration on reconstruction quality, using the previously found optimal settings regarding the subsampling pattern and $\lambda = 0.0$. The trendlines can be seen in Figure 9. The joint-reconstruction of as many channels and echoes as possible (i.e. high m_b) improves reconstruction quality.

The extracted optimal rank using this setting (fixed sampling and using true data consistency) is shown with respect to the number of missing k-space points, i.e. the acceleration, in Figure 10. The optimal rank is dependent on the matrix size m_b but shows little influence on the used acceleration. Assuming the information found in local k-

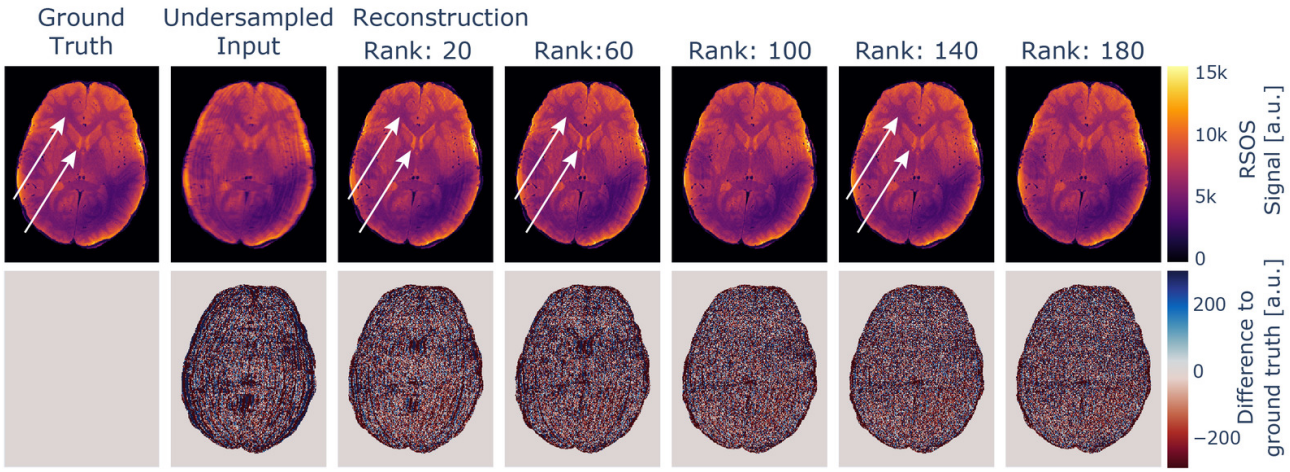


Figure 8. Influence of rank parameter on joint-reconstruction.

RSOS channel combinations are shown in the upper row, with the fully sampled data as a reference (first column), and undersampled input shown in the second column. The second row provides the difference to the fully sampled ground truth data. Using a small rank parameter (e.g., third and fourth column) has intricate effects. Residual aliasing is visible, reducing in severity with increasing rank parameter. Additionally, a brighter, delineated ventricle is visible (white arrow) together with slightly reduced intensity, for example, in parts of white matter (white arrow). This is in line with contrast mixing across echo times, considering transverse decay in MESE data causes high signal contributions from CSF or water contributions and lower contributions elsewhere.

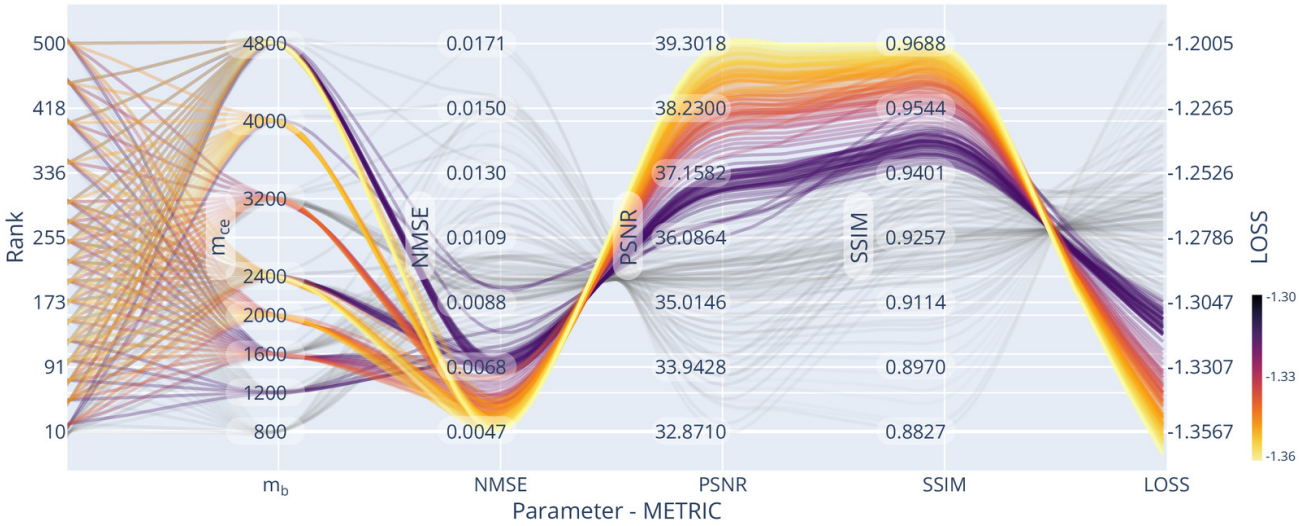


Figure 9. Parameter optimisation trendlines for varying the algorithm rank parameter and input data matrix size (Rank and m_b).

The influence on the three evaluation metrics PSNR, NMSE, and SSIM is plotted and optimisation with respect to a combined loss given in Equation 7. Optimal rank varies with input matrix size. The minimisation of loss correlates with an increase in matrix size, i.e., joint-reconstruction with the maximum number of input channels and contrasts results in increased scores in the evaluated metrics.

space neighbourhoods is limited, we expect asymptote of the optimal rank parameter.

Fitting an exemplary asymptotic function modelling the trend, e.g.,

$$f(x) = a(1 - e^{-x/b}), \tag{13}$$

to the mean value across used accelerations, the rank parameter would thus approach an optimum at around 214 for big input matrices, which is close to the default settings used in the original multichannel M-LORAKS algorithm.[18] If a fully sampled dataset or subset is available or can be acquired for piloting a particular sequence acquisition scheme, the PyLORAKS framework includes the presented optimisation procedures for finding suitable algorithm settings using Bayesian parameter optimisations.[74]

SAMPLING OPTIMISATION

The automated gradient computation through the PyLORAKS algorithm was used for estimation of the sampling densities w_e, which are shown in Figure 11.

An additional validation experiment was used to find Δ_i given in Equation 12 using ablation of each respective phase encode line. Δ_i was computed using η_{nmse} and η_{loss} from Equation 7. These measures of sampling importance were correlated to w_e using Pearson and Spearman corre-

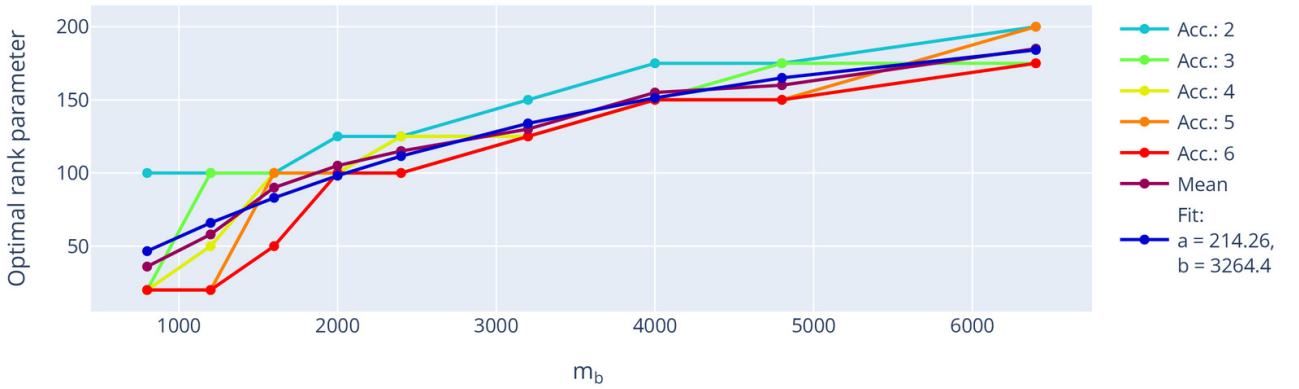


Figure 10. The optimal rank is shown against the short matrix size m_b of the LORAKS matrix.

The spatial dimension m_x was kept fixed, as was the sampling scheme (echo interleaved skipping of lines) and the $\lambda = 0.0$ parameter. For this setting the optimal rank parameter was obtained by Bayesian optimisation sampling. This optimal rank setting slightly varies with the used acceleration, i.e., the amount of missing data in the matrix, where generally higher rank parameter achieved reconstruction optimality with lower acceleration factors. However, the general trend shows asymptotic behaviour of the optimal rank driven primarily with the matrix size m_b rather independent of the acceleration used, approaching a value around 200 for big matrix sizes.

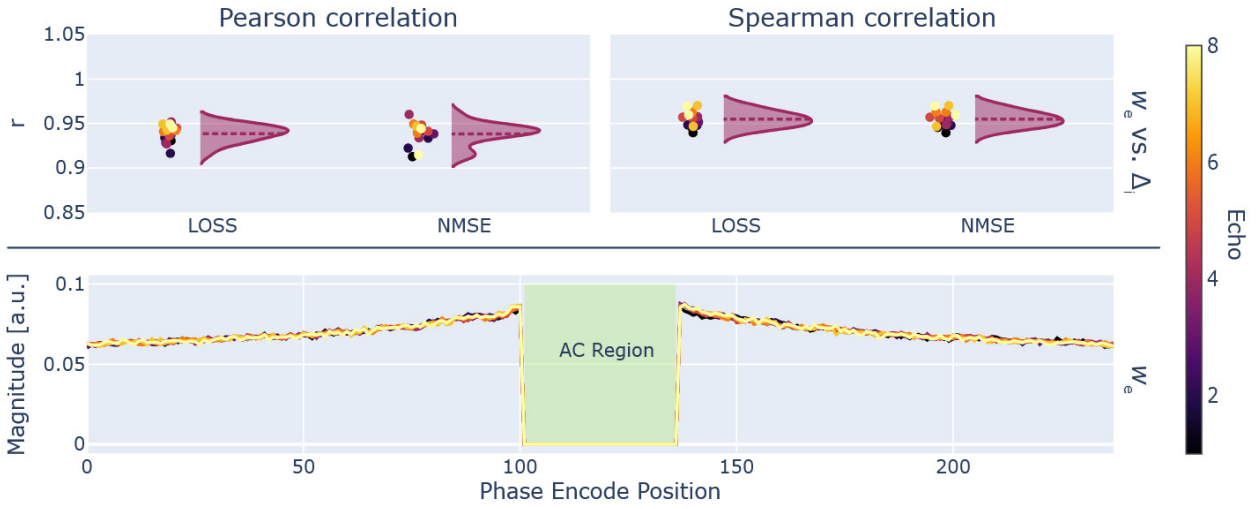


Figure 11. Results of validation experiments.

The first row shows the results of the validation experiments, after extracting Δ_i per phase encode, $\eta_n mse$ and $\eta_l oss$ were calculated, yielding a sensitivity measure of reconstruction fidelity reduction dependent on phase encode position i. Computation of the Person correlation coefficient (top left) and Spearman correlation coefficient (top right) between Δ_i and w_e is shown yielding high correlation between individual echo sensitivity measures. The sampling density per echo $w_e(i_y)$ (Equation 11) is obtained (bottom row), deduced after aggregation and normalisation of one-pass back-propagated gradients. Excluding the AC region, a similar pattern for all echoes is visible with an increase in sampling importance towards the k-space centre.

lation coefficients. The individual data per echo is given in Figure 11. When using the average, across all available echoes, the following coefficients were computed: Pearson correlation $r = 0.986, p = 0.0126$ and Spearman correlation $r = 0.990, p = 0.0171$ for $\Delta_i(\eta_{\text{nmse}})$ versus w_e, and Pearson correlation $r = 0.981, p = 0.0126$ and Spearman correlation $r = 0.991, p = 0.0172$ for correlating $\Delta_i(\eta_{\text{loss}})$ versus w_e. However, a single gradient backpropagation pass through PyLORAKS was achieved in $1/1500$ of the computation time.

Using the obtained w_e and an adaptive sampling method[78] we created a sampling pattern matching the input dimensions, AC line, and acceleration constraints. The resulting sampling pattern, alongside the ones used for comparison, can be seen in Figure 12, showing sampling variation across echoes for the phase encode direction.

The obtained sampling pattern was tested against the other 2D joint-echo sampling strategies. The resulting reconstructions are exemplified in Figure 13. We evaluated the reconstruction performance, and the respective evaluation metrics are given in Table 3.

Consequently, the optimised subsampling strategy ensured optimal reconstruction metrics, close to the previously found optimum for the interleaved line skipping method. Noticeably, the latter provided similar overall scores, but the reconstructed images showed residual aliasing artefacts. The obtained sampling density function was similar for all used *in vivo* datasets.

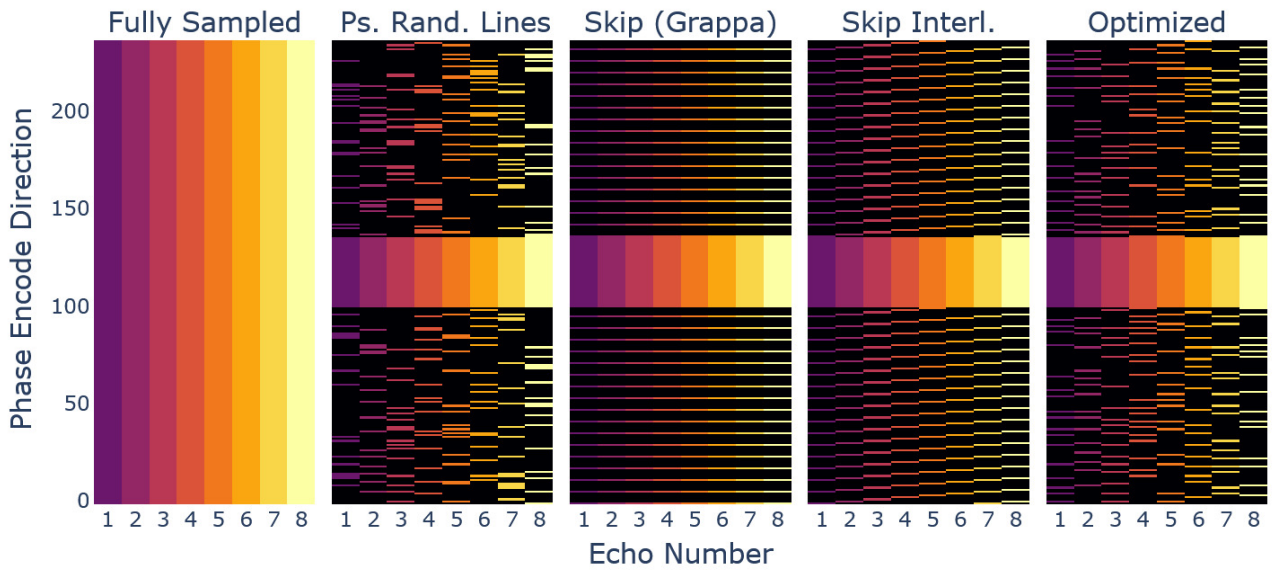


Figure 12. Tested sampling patterns to test optimal reconstruction results.

A fully sampled sampling is given in the left column. Columns 2-4 show the sampling patterns described in the sampling subsection, and the right column depicts the sampling obtained by the proposed optimisation approach. The colours are used only for easier visualisation of the individual echo samplings, black lines indicate zero-filling.

The strategy allows for data-driven optimisation of the sampling scheme for a given MRI acquisition, which in turn can facilitate enhanced reconstruction results. Alternatively, the gained reconstruction capability can be traded for increased acceleration or resolution. Furthermore, PyLORAKS reconstruction might be considered as a building block for physics-informed AI modelling, due to its capabilities for gradient propagation.[79]

DISCUSSION

In this study, we introduce PyLORAKS to address the computational demands of low-rank-based joint reconstruction for state-of-the-art MRI datasets. By incorporating efficient parallel CPU and GPU computations, PyLORAKS serves as a fast tool for reconstruction of high-resolution MRI data. Beyond speed, PyLORAKS enables scaling to high-dimensional data by replacing classical matrix decompositions with randomised variants, which will be required for future increases in data size.

We demonstrate the capabilities of our framework by presenting data-driven acquisition optimisations and showing a subsampling optimisation that improves reconstruction quality for multi-echo acquisitions, enabling higher acceleration factors. Furthermore, we present a Bayesian parameter optimisation strategy for evaluating optimal algorithmic parameters that results in increased reconstruction fidelity. We observed improved performance with true-data consistency ($\lambda = 0$) and note inter-contrast artefacts when the rank parameter was set too low, relative to the LORAKS matrix size. The resulting extensible framework opens up potential, specifically through the implementation of backpropagation. This enables future integration with acquisition and reconstruction optimisation or deep learning-based data modelling.

HARDWARE LIMITS AND SPEED

Despite substantial speed increases, PyLORAKS (like LORAKS in general) benefits from modern hardware. The largest gains were achieved by using GPUs. Recent high-memory GPU nodes may be unavailable or too costly for certain research settings. Additionally, benchmarks will depend on the CPU/GPU architecture and may vary across different hardware systems.

However, we present data-combination and batching strategies to enable joint reconstruction under tighter hardware constraints. Furthermore, the openly available framework includes deployment guides for HPC/cluster or cloud use, although computational speed gains may be offset by data-transfer overhead.

In any case, PyLORAKS accelerates computations for CPU units as well. Emerging sparsity-exploiting frameworks for efficient memory handling and computation[80] may further reduce computational demands in the future.

FAST SVD METHODS

Randomised SVD algorithms reduce memory and time for processing of large LORAKS matrices and computational demands are tailored by target rank and oversampling. Thus, explicit control of the memory footprint with respect to hardware limits is possible, but a dedicated optimisation of the influence on the reconstruction was beyond the scope of this work.

For example, introducing randomised SVD methods will be much more efficient for P-LORAKS as the LORAKS constraint is enforced with a low-rank approximation of the LORAKS matrix, allowing for big data compression within the matrix decomposition method since $r \ll m_b$. On the other hand, with the AC-LORAKS framework presented here, the rank enforcement is done by approximating the

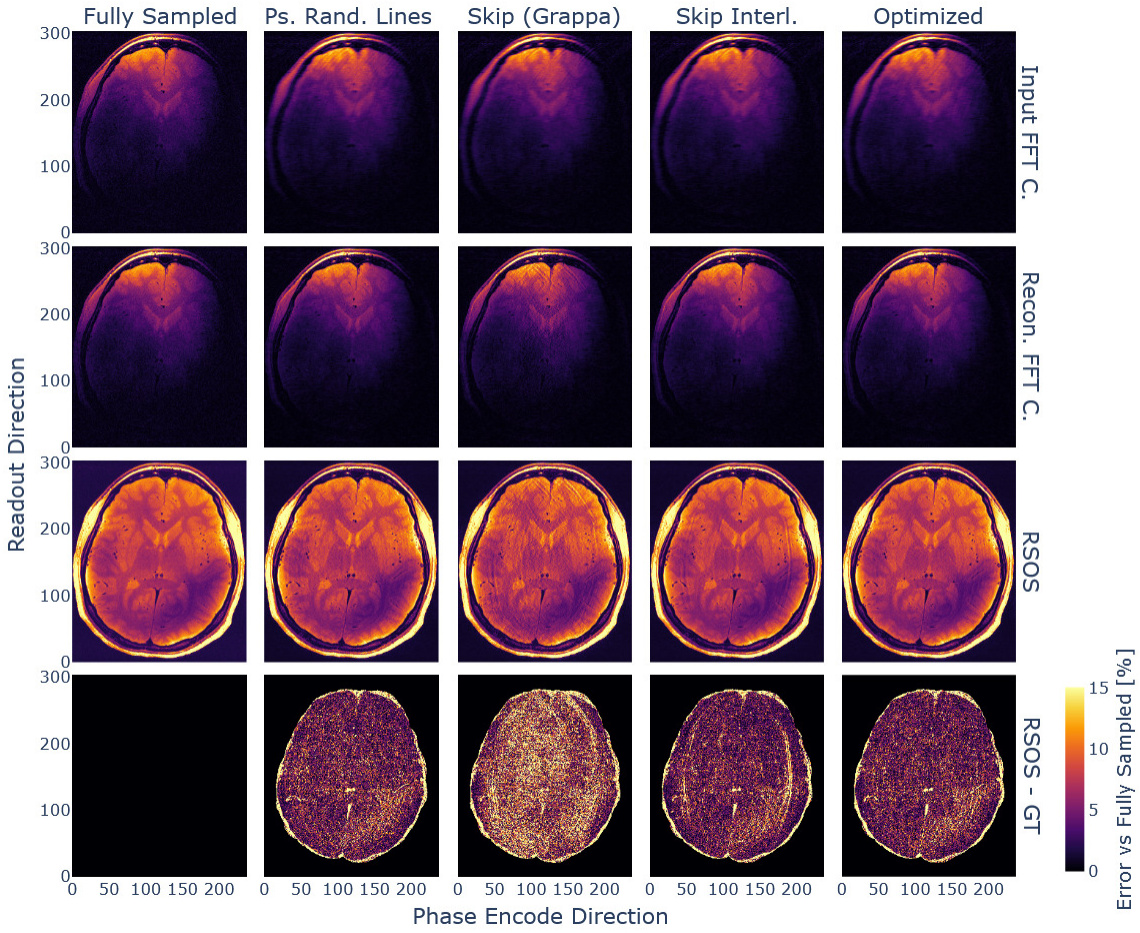


Figure 13. Reconstruction performance with respective input undersampling pattern.

The pattern detailed in Figure 12 was reconstructed using the same algorithm settings for PyLORAKS, and compared to the fully sampled reference given in the left column. For each input a simple FFT to image space shows the influence of the sampling pattern on aliasing artefacts in the first row for a randomly selected channel and the first echo. The second row displays the respective reconstruction after using PyLORAKS for the same channel and the first echo. We used an RSOS channel combination to obtain magnitude images of the first echo provided in the third row, of which the difference to the reference RSOS was calculated and is displayed in the fourth row.

Table 3. Resulting metrics from comparing the reconstruction of different subsampling patterns with the fully sampled input.

Subsampling Method	PSNR	NMSE	SSIM
Pseudo-Random Lines	36.207	0.0186	0.9396
Skipped Lines (GRAPPA)	33.782	0.0325	0.8869
Interleaved Skipped Lines	36.373	0.0179	0.9411
Optimized	**36.539**	**0.0172**	**0.9445**

nullspace of the LORAKS matrix formed by AC data. This nullspace is of dimensionality $m_b - r$ and an effective compression using randomised SVD variants has little computational benefit for small r. Therefore, we decided to use a different approximation of the nullspace by computing the leading $4r$ eigenvectors of the nullspace by the randomised SVD method. Further increase in computational efficiency might be possible by further reduction of this dimensionality, and effective memory control is possible by adjusting to the available hardware.

SCALABILITY AND 3D ACQUISITIONS

Given the computational efficiency improvements achieved with PyLORAKS, we demonstrated effective joint-reconstruction of modern high-resolution data. However, we limited the study to slice-wise reconstruction most suited for 2D sequences. While 3D sequences can be reconstructed in this fashion (e.g. by formation of a hybrid space with the Fourier-transformed readout direction taken as the iterative dimension), 3D subsampling might benefit from extending the LORAKS neighbourhood to a 3D volume as previously suggested.[3,18] However, the scaling implications are beyond what has been investigated within the scope

of this study. Not only does the neighbourhood size contribute with the power of its dimensionality to the LORAKS matrix size $m_b \propto p_s^{n_{\text{dim}}}$, also the larger matrix dimension $m_x \propto \prod N_{x,y,z}$ scales linearly with an added dimension. This counterbalances the improvements presented in our study and might be prohibited by memory limits in a joint-reconstruction setting and high-resolution data. We suggest using the PyLORAKS framework in iterative fashion across 2D slices to achieve the presented advantages. However, further exploitation of the use of randomised matrix decomposition methods enable higher dimensional data processing, as well as future hardware advancement possibly allows for joint-LORAKS 3D reconstruction of high-resolution data using PyLORAKS.

NOISE

One aspect of joint-echo reconstruction and LORAKS reconstruction in general is its effect on imaging noise not studied in the presented research. LORAKS has previously been shown to have denoising effects, which are additionally related to specific image structures.[3] The resulting spatially dependent noise reduction or amplification possibly extends across contrasts in a joint-reconstruction and, to the authors' knowledge, has not yet been thoroughly investigated. Due to the nature of randomised SVD methods, using random sampling of the LORAKS matrix to facilitate data compression, additional effects on imaging noise are likely. In our presented PyLORAKS framework focus was placed on reconstruction of individual channel and contrast data respectively, allowing for a more straight forward investigation of noise influence compared to e.g. channel compression approaches, but is left to future research efforts.

EVALUATION OF ALGORITHM PARAMETERS

The presented methods for data-driven evaluation of algorithm parameters revealed joint-reconstruction pitfalls that are applicable to other data, such as the mixing of contrast information when using too aggressive rank settings. Results such as the optimality of true data consistency or specific rank parameters may not generalise to all acquisition modalities, but the presented methods can be used to tailor the algorithm parameter settings to the specific acquisition method. Furthermore, if fully sampled data can not be acquired for reference, optimisations with respect to the parameter choice are possible using a target reconstruction. The applicability of our results to other imaging data remains to be demonstrated. Nonetheless, the usage of PyLORAKS as a drop-in replacement within reconstruction optimisations like A-LORAKS-CS[24] to increase computational performance opens up other pathways to data-driven parameter selection beyond the presented method. However, presenting generalisable optimal LORAKS settings remains a challenging task.

AUTOMATIC DIFFERENTIATION AND BACKPROPAGATION

We demonstrate automatic gradient computation for backpropagation through the PyLORAKS algorithm, enabling input or condition optimisation. However, great efforts were previously invested into enhancing the LORAKS framework.[18] The least-squares solver mechanism is a fast and efficient method, especially in conjunction with CGD approaches, and thus has been proven to be quicker than the gradient descent used with backpropagation. This is mainly due to the update rate being unknown *a priori*, which is the case for most machine-learning approaches using the same PyTorch mechanism. Thus, similar loss optimisation efforts in conjunction with backpropagation might be used, for example, adopting stochastic optimisation like Adam.[81] Additionally, computation of an optimal or adaptive step size is part of recent AI - research[82-84] and emerging results might be applicable to the PyLORAKS framework.

SUBSAMPLING OPTIMISATIONS

The automatic differentiation feature was used for end-to-end optimisation of the sampling pattern of a multi-echo acquisition, which provided an increase in reconstruction quality transferable to increased acquisition acceleration. Due to the stochastic sampling approach, the optimisation for other acceleration factors is implemented quickly. We found the obtained sampling densities to be similar across participants, and thus to be valid across equal acquisitions. However, due to the limited number of subjects, generalisation to other acquisitions or changes in acquisition parameters (such as T_E or FOV) warranting re-computation need to be further analysed in future research. In principle, the automatic differentiation framework can be used to compute a deterministic mask with additional improvements beyond the scope of this work. Furthermore, the processing relied on the availability of fully sampled data for at least parts of the FOV. To circumvent this for cases where this is not available or hard to obtain, a target reconstruction can be used instead. In any case, PyLORAKS can be suitable for integration in existing sampling optimisation frameworks like OEDIPUS[69] and thus allows for a plethora of sampling optimisation options.

ASSESSMENT

We used a number of metrics for assessment of the reconstruction quality and the optimisation procedures. Each metric (NMSE, PSNR, SSIM) has specific advantages and limitations; we consider SSIM most suitable for perceptual image quality assessment. Nonetheless, the used images have a high resolution, and reduction of image content into one quantitative metric has its drawbacks. For example, the second-best sampling pattern (based on metrics) showed notable aliasing artefacts compared to others. Thus, visual inspection of the data is still necessary, should always be carried out, and future improvements in image evaluation quantification are needed.

CONCLUSION

We present PyLORAKS, a fast and computationally efficient implementation of the LORAKS reconstruction approach, particularly for large MRI datasets. We incorporated joint-

contrast reconstruction and optimised code to achieve high computational efficiency and a speed increase of around 5–6-fold compared to the original MATLAB implementation. Additionally, GPU parallelisation further accelerates single-slice joint reconstruction by up to 30-fold. The estimated increase in computation time scales with the number of slices, drastically reducing reconstruction time of whole volume multichannel multi-contrast data. We leverage these computational gains to enable Bayesian sampling-based optimisation of the algorithm's main parameters for regularisation and low-rank completion (λ and r), which previously were chosen manually after visual inspection. We found a performance increase with the use of the LORAKS true-data consistency formulation ($\lambda = 0$) and demonstrated bleeding artefacts between contrasts if the rank parameter is chosen too low compared to the LORAKS matrix size. Furthermore, we introduced automatic back-propagation, enabling data-driven optimisations and physics-informed AI-modelling. We demonstrated its capabilities in the design of an optimised subsampling scheme, which can be used to increase acquisition acceleration. PyLORAKS is openly available, including containerised environments for deployment large-scale compute clusters.

DATA AND CODE AVAILABILITY

The software used in this study is available on GitHub at https://github.com/schmidt-jo/PyMRItools.[55] This includes the LORAKS reconstruction framework for the AC and P-LORAKS algorithms, the S and C LORAKS matrix formulations, and the different solvers using CGD and automatic gradient descent mentioned in this paper. Additionally, a collection of tools and scripts to demonstrate the experiments used is included, and a guide for setting up containerised environments for high-performance compute cluster utilisation is provided.

The code repository contains phantom data that allows reproduction of the experiments. The presented subject data contains sensitive patient information and is subject to European GDPR regulations. It is thus only available upon request and with a formal data sharing agreement.

AUTHOR CONTRIBUTIONS

Jochen Schmidt: conceptualisation, data curation, formal analysis, investigation, methodology, project administration, software, visualisation, writing—original draft. **Patrick Scheibe**: conceptualisation, data curation, methodology, software, writing–review and editing. **David Carreto Fidalgo**: data curation, software. **Andreas Marek**: data curation, software, resources. **Robert Trampel**: conceptualisation, supervision, writing—review and editing. **Nikolaus Weiskopf**: conceptualisation, funding acquisition, resources, supervision, writing—review and editing.

FUNDING SOURCES

This project has received funding from the German Federal Ministry of Education and Research (BMBF) under support code 01ED2210. Furthermore, this work was supported by the Max Planck Society for the Advancement of Science, Germany.

CONFLICTS OF INTEREST

The Max Planck Institute for Human Cognitive and Brain Sciences and Wellcome Centre for Human Neuroimaging have institutional research agreements with Siemens Healthcare. Nikolaus Weiskopf holds a patent on acquisition of MRI data during spoiler gradients (US 10,401,453 B2). Nikolaus Weiskopf was a speaker at an event organised by Siemens Healthcare and was reimbursed for the travel expenses.

ACKNOWLEDGMENTS

First and foremost, we are deeply indebted to Justin P. Haldar for our discussions and the exchange about our ideas and encountered problems, to which he always responded with great patience and remarkable detail. His expertise has been invaluable, and this work would not have been possible without his guidance.

We also appreciate the support given by the Max Planck Computing and Data Facility for their help in implementing the software for AMD architectures and running our experiments at scale.

We thank Valerij Kiselev for his input on random matrix theory. We are grateful to Barbara Dymerska for the exchange and her shared insights into the user experience of LORAKS. Given the early stage of PyLORAKS, we are especially thankful to Tiago José Timóteo Fernandes for the extensive testing, which highlighted the importance of hardware constraint estimation.

Lastly, we extend our special thanks to all the MTAs at the Max Planck Institute for Human Cognition and Brain Sciences involved in scanning participants.

Submitted: October 10, 2025 CDT. Accepted: January 26, 2026 CDT. Published: February 25, 2026 CDT.

REFERENCES

1. Griswold MA, Jakob PM, Heidemann RM, et al. Generalized autocalibrating partially parallel acquisitions (GRAPPA). *Magnetic Resonance in Medicine*. 2002;47(6):1202-1210. doi:10.1002/mrm.10171

2. Pruessmann KP, Weiger M, Scheidegger MB, Boesiger P. SENSE: Sensitivity encoding for fast MRI. *Magnetic Resonance in Medicine*. 1999;42(5):952-962.

3. Haldar JP. Low-rank modeling of local k-space neighborhoods (LORAKS) for constrained MRI. *IEEE Transactions on Medical Imaging*. 2014;33(3):668-681. doi:10.1109/tmi.2013.2293974

4. Haldar JP, Zhuo J. P-LORAKS: Low-rank modeling of local k-space neighborhoods with parallel imaging data. *Magnetic Resonance in Medicine*. 2015;75(4):1499-1514. doi:10.1002/mrm.25717

5. Weiskopf N, Suckling J, Williams G, et al. Quantitative multi-parameter mapping of R1, PD*, MT, and R2* at 3T: A multi-center validation. *Frontiers in Neuroscience*. 2013;7. doi:10.3389/fnins.2013.00095

6. Haacke EM, Chen Y, Utriainen D, et al. STrategically acquired gradient echo (STAGE) imaging, part III: Technical advances and clinical applications of a rapid multi-contrast multi-parametric brain imaging method. *Magnetic Resonance Imaging*. 2020;65:15-26. doi:10.1016/j.mri.2019.09.006

7. Marques JP, Kober T, Krueger G, Zwaag W van der, Van de Moortele PF, Gruetter R. MP2RAGE, a self bias-field corrected sequence for improved segmentation and T1-mapping at high field. *NeuroImage*. 2010;49(2):1271-1281. doi:10.1016/j.neuroimage.2009.10.002

8. Schmidt J, Radunsky D, Scheibe P, et al. High resolution quantitative T2 mapping of the human brain at 7 T using a multi-echo spin-echo sequence and dictionary-based modeling. *Imaging Neuroscience*. Published online 2025. doi:10.1162/imag.a.81

9. Le Bihan D. Diffusion MRI: What water tells us about the brain. *EMBO Molecular Medicine*. 2014;6(5):569-573. doi:10.1002/emmm.201404055

10. Baliyan V, Das CJ, Sharma R, Gupta AK. Diffusion weighted imaging: Technique and applications. *World Journal of Radiology*. 2016;8(9):785. doi:10.4329/wjr.v8.i9.785

11. Huang F, Akao J, Vijayakumar S, Duensing GR, Limkeman M. K-t GRAPPA: A k-space implementation for dynamic MRI with high reduction factor. *Magnetic Resonance in Medicine*. 2005;54(5):1172-1184. doi:10.1002/mrm.20641

12. Liu W, Zhao X, Ma Y, Tang X, Gao J. DWI using navigated interleaved multishot EPI with realigned GRAPPA reconstruction. *Magnetic Resonance in Medicine*. 2015;75(1):280-286. doi:10.1002/mrm.25586

13. Bilgic B, Kim TH, Liao C, et al. Improving parallel imaging by jointly reconstructing multi-contrast data. *Magnetic Resonance in Medicine*. 2018;80(2):619-632. doi:10.1002/mrm.27076

14. Kim TH, Justin H. LORAKS software version 2.0: Faster implementation and enhanced capabilities. *Technical Report USC-SIPI-443*. Published online 2018. doi:10.1002/mrm.20641

15. Haldar JP. Autocalibrated loraks for fast constrained MRI reconstruction. In: *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*. IEEE; 2015:910-913. doi:10.1109/isbi.2015.7164018

16. Kaloorazi MF, Lamare RC de. Subspace-orbit randomized decomposition for low-rank matrix approximations. *IEEE Transactions on Signal Processing*. 2018;66(16):4409-4424. doi:10.1109/tsp.2018.2853137

17. Weiss S, Proudler IK, Barbarino G, Pestana J, McWhirter JG. On properties and structure of the analytic singular value decomposition. *IEEE Transactions on Signal Processing*. 2024;72:2260-2275. doi:10.1109/tsp.2024.3387726

18. Kim TH, Haldar J. LORAKS 2.1: Implementation and examples. https://mr.usc.edu/download/LORAKS2/

19. Josephs O, Dymerska B, Graedel NN, Balbastre Y, Corbin N, Callaghan MF. MORSE: Multiple orthogonal reference sensitivity encoding. doi:10.48550/ARXIV.2510.09098

20. Lobos RA, Hoge WS, Javed A, et al. Robust autocalibrated structured low-rank EPI ghost correction. *Magnetic Resonance in Medicine*. 2020;85(6):3403-3419. doi:10.1002/mrm.28638

21. Dong Y, Ye X, Li C, Osch MJP van, Börnert P. Navigator-free multi-shot diffusion MRI via non-local low-rank reconstruction. *Magnetic Resonance in Medicine*. Published online May 2025. doi:10.1002/mrm.30554

22. Stein CM. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*. 1981;9(6). doi:10.1214/aos/1176345632

23. Iyer S, Ong F, Setsompop K, Doneva M, Lustig M. SURE-based automatic parameter selection for ESPIRiT calibration. *Magnetic Resonance in Medicine*. 2020;84(6):3423-3437. doi:10.1002/mrm.28386

24. Ilicak E, Saritas EU, Çukur T. Automated parameter selection for accelerated MRI reconstruction via low-rank modeling of local k-space neighborhoods. *Zeitschrift für Medizinische Physik*. 2023;33(2):203-219. doi:10.1016/j.zemedi.2022.02.002

25. Zhao S, Potter LC, Ahmad R. High-dimensional fast convolutional framework (HICU) for calibrationless MRI. *Magnetic Resonance in Medicine*. 2021;86(3):1212-1225. doi:10.1002/mrm.28721

26. Halko N, Martinsson PG, Tropp JA. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. Published online 2009. doi:10.48550/ARXIV.0909.4061

27. Kim TH, Setsompop K, Haldar JP. LORAKS makes better SENSE: Phase-constrained partial fourier SENSE reconstruction without phase calibration. *Magnetic Resonance in Medicine*. 2016;77(3):1021-1035. doi:10.1002/mrm.26182

28. Abdelnaby M, Moussa MR. A benchmarking study of random projections and principal components for dimensionality reduction strategies in single cell analysis. Published online February 2025. doi:10.1101/2025.02.04.636499

29. Park T, Nakatsukasa Y. A fast randomized algorithm for computing an approximate null space. *BIT Numerical Mathematics*. 2023;63(2). doi:10.1007/s10543-023-00979-7

30. Hammernik K, Klatzer T, Kobler E, et al. Learning a variational network for reconstruction of accelerated MRI data. *Magnetic Resonance in Medicine*. 2017;79(6):3055-3071. doi:10.1002/mrm.26977

31. Heckel R, Jacob M, Chaudhari A, Perlman O, Shimron E. Deep learning for accelerated and robust MRI reconstruction. *Magnetic Resonance Materials in Physics, Biology and Medicine*. 2024;37(3):335-368. doi:10.1007/s10334-024-01173-8

32. Cao C, Cui ZX, Wang Y, et al. High-frequency space diffusion model for accelerated MRI. *IEEE Transactions on Medical Imaging*. 2024;43(5):1853-1865. doi:10.1109/tmi.2024.3351702

33. Kim TH, Haldar JP. Learning how to interpolate fourier data with unknown autoregressive structure: An ensemble-based approach. In: *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE; 2019:1471-1475. doi:10.1109/ieeeconf44664.2019.9048755

34. Paszke A, Gross S, Massa F, et al. PyTorch: An imperative style, high-performance deep learning library. GitHub Repository. doi:10.48550/ARXIV.1912.01703

35. Kim TH, Garg P, Haldar JP. LORAKI: Autocalibrated recurrent neural networks for autoregressive MRI reconstruction in k-space. doi:10.48550/ARXIV.1904.09390

36. Zhang C, Hosseini SAH, Weingärtner S, Uğurbil K, Moeller S, Akçakaya M. Optimized fast GPU implementation of robust artificial-neural-networks for k-space interpolation (RAKI) reconstruction. Bagci U, ed. *PLOS ONE*. 2019;14(10):e0223315. doi:10.1371/journal.pone.0223315

37. Stone SS, Haldar JP, Tsao SC, Hwu W m. W, Sutton BP, Liang ZP. Accelerating advanced MRI reconstructions on GPUs. *Journal of Parallel and Distributed Computing*. 2008;68(10):1307-1318. doi:10.1016/j.jpdc.2008.05.013

38. Gai J, Obeid N, Holtrop JL, et al. More IMPATIENT: A gridding-accelerated toeplitz-based strategy for non-cartesian high-resolution 3D MRI on GPUs. *Journal of Parallel and Distributed Computing*. 2013;73(5):686-697. doi:10.1016/j.jpdc.2013.01.001

39. Murphy M, Alley M, Demmel J, Keutzer K, Vasanawala S, Lustig M. Fast l1-SPIRiT compressed sensing parallel imaging MRI: Scalable parallel implementation and clinically feasible runtime. *IEEE Transactions on Medical Imaging*. 2012;31(6):1250-1262. doi:10.1109/tmi.2012.2188039

40. Paszke A, Gross S, Chintala S, et al. Automatic differentiation in PyTorch. In: *NIPS-w.* ; 2017.

41. Kim TH, Bilgic B, Polak D, Setsompop K, Haldar JP. Wave-LORAKS: Combining wave encoding with structured low-rank matrix modeling for more highly accelerated 3D imaging. *Magnetic Resonance in Medicine*. 2018;81(3):1620-1633. doi:10.1002/mrm.27511

42. Slavkova KP, DiCarlo JC, Wadhwa V, et al. An untrained deep learning method for reconstructing dynamic MR images from accelerated model-based data. *Magnetic Resonance in Medicine*. 2022;89(4):1617-1633. doi:10.1002/mrm.29547

43. Jacobs L, Mandija S, Liu H, Berg CAT van den, Sbrizzi A, Maspero M. Generalizable synthetic MRI with physics-informed convolutional networks. Published online 2023. doi:10.48550/ARXIV.2305.12570

44. Wang Z, Yu X, Wang C, et al. One for multiple: Physics-informed synthetic data boosts generalizable deep learning for fast MRI reconstruction. doi:10.48550/ARXIV.2307.13220

45. Hofmann SM, Goltermann O, Scherf N, et al. The utility of explainable AI for MRI analysis: Relating model predictions to neuroimaging features of the aging brain. *Imaging Neuroscience*. 2025;3. doi:10.1162/imag_a_00497

46. The MathWorks Inc. MATLAB. Published online 2022. https://www.mathworks.com

47. Polimeni JR, Bhat H, Witzel T, et al. Reducing sensitivity losses due to respiration and motion in accelerated echo planar imaging by reordering the autocalibration data acquisition. *Magnetic Resonance in Medicine*. 2015;75(2):665-679. doi:10.1002/mrm.25628

48. Seifert AC, Xu J. Impact of autocalibration method on accelerated EPI of the cervical spinal cord at 7 t. *Magnetic Resonance in Medicine*. 2022;88(6):2583-2591. doi:10.1002/mrm.29415

49. Shepp LA, Logan BF. The fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science*. 1974;21(3):21-43. doi:10.1109/tns.1974.6499235

50. Layton KJ, Kroboth S, Jia F, et al. Pulseq: A rapid and hardware-independent pulse sequence prototyping framework: Rapid hardware-independent pulse sequence prototyping. *Magnetic Resonance in Medicine*. 2016;77(4):1544-1552. doi:10.1002/mrm.26235

51. Ravi K, Geethanath S, Vaughan J. PyPulseq: A python package for MRI pulse sequence design. *Journal of Open Source Software*. 2019;4(42):1725. doi:10.21105/joss.01725

52. Schmidt J, Radunsky D, Scheibe P, Ben-Eliezer N, Trampel R, Weiskop N. Fast quantitative T2 mapping at ultra-high field using sparse sampling and dictionay-based reconstruction. In: *Proc. Intl. Soc. Mag. Reson. Med. 31, 1786.* ; 2023.

53. Zbontar J, Knoll F, Sriram A, et al. FastMRI: An open dataset and benchmarks for accelerated MRI. doi:10.48550/ARXIV.1811.08839

54. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*. 2004;13(4):600-612. doi:10.1109/tip.2003.819861

55. Schmidt J, Scheibe P. PyMRITools: git. https://github.com/schmidt-jo/PyMRItools

56. Hennessy JL, Patterson DA. *Computer Architecture, Fifth Edition: A Quantitative Approach*. 5th ed. Morgan Kaufmann Publishers Inc.; 2011.

57. Wulf W, McKee S. Hitting the memory wall: Implications of the obvious. *Computer Architecture News*. 1996;23.

58. *Intel 64 and IA-32 Architectures Optimization Reference Manual*. Intel Corporation; Intel Corporation; 2019.

59. NVIDIA, Vingelmann P, Fitzek FHP. CUDA, release: 12.4. 2020. https://developer.nvidia.com/cuda-toolkit

60. Luo T, Noll DC, Fessler JA, Nielsen JF. Joint design of RF and gradient waveforms via auto-differentiation for 3D tailored excitation in MRI. *IEEE Transactions on Medical Imaging*. 2021;40(12):3305-3314. doi:10.1109/tmi.2021.3083104

61. Bazin PL, Alkemade A, Zwaag W van der, Caan M, Mulder M, Forstmann BU. Denoising high-field multi-dimensional MRI with local complex PCA. *Frontiers in Neuroscience*. 2019;13. doi:10.3389/fnins.2019.01066

62. Does MD, Olesen JL, Harkins KD, et al. Evaluation of principal component analysis image denoising on multi-exponential MRI relaxometry. *Magnetic Resonance in Medicine*. 2019;81(6):3503-3514. doi:10.1002/mrm.27658

63. Moeller S, Pisharady PK, Ramanna S, et al. NOise Reduction with DIstribution Corrected (NORDIC) PCA in dMRI with complex-valued parameter-free locally low-rank processing. *NeuroImage*. 2021;226:117539. doi:10.1016/j.neuroimage.2020.117539

64. Agarwal PK, Mustafa NH. K -means projective clustering. In: *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. SIGMOD/PODS04. ACM; 2004:155-165. doi:10.1145/1055558.1055581

65. Falkner JK. Torch KMeans. https://github.com/jokofa/torch_kmeans

66. Minster R, Saibaba AK, Kar J, Chakrabortty A. Efficient algorithms for eigensystem realization using randomized SVD. *SIAM Journal on Matrix Analysis and Applications*. 2021;42(2):1045-1072. doi:10.1137/20m1327616

67. Kaloorazi MF, Liu K, Chen J, Lamare RC de. An efficient randomized QLP algorithm for approximating the singular value decomposition. doi:10.48550/ARXIV.2110.01011

68. PyTorch Contributors. SVD is slow on GPU vs CPU for skinny matrices. 2019. Accessed December 9, 2025. https://github.com/pytorch/pytorch/issues/41306

69. Haldar JP, Kim D. OEDIPUS: An experiment design framework for sparsity-constrained MRI. *IEEE Transactions on Medical Imaging*. 2019;38(7):1545-1558. doi:10.1109/tmi.2019.2896180

70. Faes LK, Lage-Castellanos A, Valente G, et al. Evaluating the effect of denoising submillimeter auditory fMRI data with NORDIC. *Imaging Neuroscience*. 2024;2:1-18. doi:10.1162/imag_a_00270

71. Candes EJ, Sing-Long CA, Trzasko JD. Unbiased risk estimates for singular value thresholding and spectral estimators. *IEEE Transactions on Signal Processing*. 2013;61(19):4643-4657. doi:10.1109/tsp.2013.2270464

72. Ramani S, Blu T, Unser M. Monte-carlo sure: A black-box optimization of regularization parameters for general denoising algorithms. *IEEE Transactions on Image Processing*. 2008;17(9):1540-1554. doi:10.1109/tip.2008.2001404

73. Jin KH, Lee D, Ye JC. A general framework for compressed sensing and parallel MRI using annihilating filter based low-rank hankel matrix. *IEEE Transactions on Computational Imaging*. 2016;2(4):480-495. doi:10.1109/tci.2016.2601296

74. Dewancker I, McCourt M, Clark S, Hayes P, Johnson A, Ke G. Evaluation system for a bayesian optimization service. doi:10.48550/ARXIV.1605.06170

75. Biewald L. Experiment tracking with weights and biases. 2020. https://www.wandb.com/

76. Kreutz-Delgado K. The complex gradient operator and the CR-calculus. doi:10.48550/ARXIV.0906.4835

77. Notes D. Autograd mechanics. Accessed July 18, 2025. https://docs.pytorch.org/docs/stable/notes/autograd.html#autograd-for-complex-numbers

78. Takahashi A, Kumagai Y, Aoki H, Tamura R, Oba F. Adaptive sampling methods via machine learning for materials screening. *Science and Technology of Advanced Materials: Methods*. 2022;2(1):55-66. doi:10.1080/27660400.2022.2039573

79. Wu Y, Sicard B, Gadsden SA. Physics-informed machine learning: A comprehensive review on applications in anomaly detection and condition monitoring. *Expert Systems with Applications*. 2024;255:124678. doi:10.1016/j.eswa.2024.124678

80. Ivanov A, Dryden N, Ben-Nun T, Ashkboos S, Hoefler T. STen: Productive and efficient sparsity in PyTorch. *CoRR*. 2023;abs/2304.07613. doi:10.48550/ARXIV.2304.07613

81. Kingma DP, Ba J. Adam: A method for stochastic optimization. doi:10.48550/ARXIV.1412.6980

82. Hu X, Wen S, Lam HK. Dynamic random distribution learning rate for neural networks training. *Applied Soft Computing*. 2022;124:109058. doi:10.1016/j.asoc.2022.109058

83. Dharanalakota V, Raikar AA, Ghosh PK. Improving neural network training using dynamic learning rate schedule for PINNs and image classification. *Machine Learning with Applications*. 2025;21:100697. doi:10.1016/j.mlwa.2025.100697

84. Jin Y, Zhou T, Zhao L, et al. AutoLRS: Automatic learning-rate schedule by bayesian optimization on the fly. doi:10.48550/ARXIV.2105.10762

SUPPLEMENTARY MATERIALS

Supplementary Material

Download: https://apertureneuro.org/article/156499-fast-and-scalable-joint-loraks-reconstruction-and-data-driven-sampling-optimisation-of-high-dimensional-mri-datasets-using-a-gpu-accelerated-and-learn/attachment/330090.pdf