International Journal on Magnetic Particle Imaging
Vol 3, No 2, Article ID 1707002, 8 Pages

Infinite Science Publishing

Research Article

Low Latency Real-time Reconstruction for MPI Systems

Patrick Vogel[a,b,*] · Stefan Herz[b] · Thomas Kampf[a,c] · Martin A. Rückert[a] · Thorsten A. Bley[b] · Volker C. Behr[a]

[a] Department of Experimental Physics 5 (Biophysics), University of Würzburg, Würzburg, Germany
[b] Department of Diagnostic and Interventional Radiology, University Hospital Würzburg, Würzburg, Germany
[c] Department of Diagnostic and Interventional Neuroradiology, University Hospital Würzburg, Würzburg, Germany
[*] Corresponding author, email: patrick.vogel@physik.uni-wuerzburg.de

Received 25 November 2016; Accepted 9 July 2017; Published online 14 July 2017

Abstract

In the last decade research in the Magnetic Particle Imaging community focused mainly on hardware development as well as the optimization of reconstruction. These steps were essential for improving MPI technology and advance it to pre-clinical applications. However, another important part is the software package working behind MPI systems. This software has to be able to keep up with modern scanner features and should be able to support features such as adjustable scanning trajectories or multiple reconstruction methods. Especially a real-time reconstruction feature is important to use the full potential of the fast imaging capabilities of modern MPI scanners. This could be pivotal in applications such as guided vascular interventions. In this work, a software package is presented, which improves reconstructing MPI data with different reconstruction methods in real-time and with low latency.

I. Introduction

Magnetic Particle Imaging (MPI) is a fast growing technology with large leaps in progress in the last decade. Several hardware designs as well as adapted reconstruction methods have been demonstrated to improve the platforms and bring them into pre-clinical phase [1]. Due to different features of these hardware designs, their software packages for controlling the MPI scanner and post processing software to reconstruct images are unique and customized implementations. A first step towards a common consensus for all MPI-systems was the introduction of an open-source standard data format, which allows easy data sharing [2]. Another step was the decoupling of the reconstruction process from the scanner hardware, as proposed in [3]. This allows reconstructing pre-processed datasets (normalized datasets) acquired from the same type of MPI-scanner but with different acquisition settings with the same parameter set for the image reconstruction method.

A further step would be a common software platform, providing all necessary functions to run almost every MPI-system and perform a tailored image reconstruction to the specific application. There are several 'nice-to-have' features comprising easy-to-use graphical user interface (GUI) or high flexibility, which implicates addressing different scanner designs and reconstruction methods [1], as well as a customizable environment tailored for specific applications.

However, due to the fact, that MPI is a quite fast imaging technique providing up to 1840 images per second [4] and 46 volumes per second [5] one basic goal for such a software platform is to support real-time capability. This is an important feature for potential clinical applications

such as MPI guided percutaneous transluminal angiography (PTA). A first implementation for online reconstruction in 3D was demonstrated using a system matrix reconstruction [6]. However, the main challenge is reducing the latency time between pressing-the-button and image visualization (see Fig. 1). While the time lag between pressing-the-button until the scanner starts imaging is in the range of milliseconds, the scanning time depends on the sequence used for collecting data. Steps that are more problematic are data transfer from the analog digital converter (ADC) to the computer as well as the entire reconstruction process running on the PC. Due to the fact, that the data transfer between ADC and computer is often limited by the underlying protocol (USB, TCP/IP, etc.), the reconstruction process offers room for improvement. This means thinking of optimization the data processing to minimize the reconstruction pipeline times. One bottleneck in reconstruction methods using system matrices is the size of the processed data. System matrix optimization for real-time reconstruction by data size reduction and compression techniques was demonstrated for Fourier-based [7, 8] as well as image based [3] system matrices.

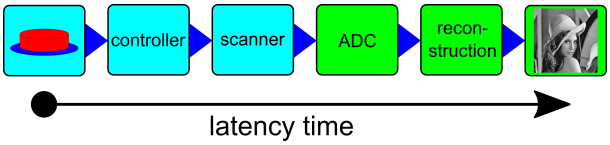

Figure 1: Sketch of the major steps starting from pressing-the-button to the final reconstructed image. The latency time is the sum of all time lags from the corresponding steps.

However, additional steps are necessary from the incoming data (ADC) to the final reconstructed image or volume. In this study an overview of all these steps is given and a description for optimizing some of these processes with respect to computation time is discussed.

II. Methods

As mentioned before, several types of MPI scanners (field-free point [1] or field-free line [9]) as well as appropriate reconstruction methods (system matrix [10], image-based [3], x-space [11]) have been published. Most systems use different trajectories and require different steps for the reconstruction process.

The flow chart in Fig. 2 shows the important steps to obtain a reconstructed image or volume covering most scanner types and reconstruction processes. Basically the process can be separated in pre-calculation step, raw data step, correction step, gridding step and reconstruction step.

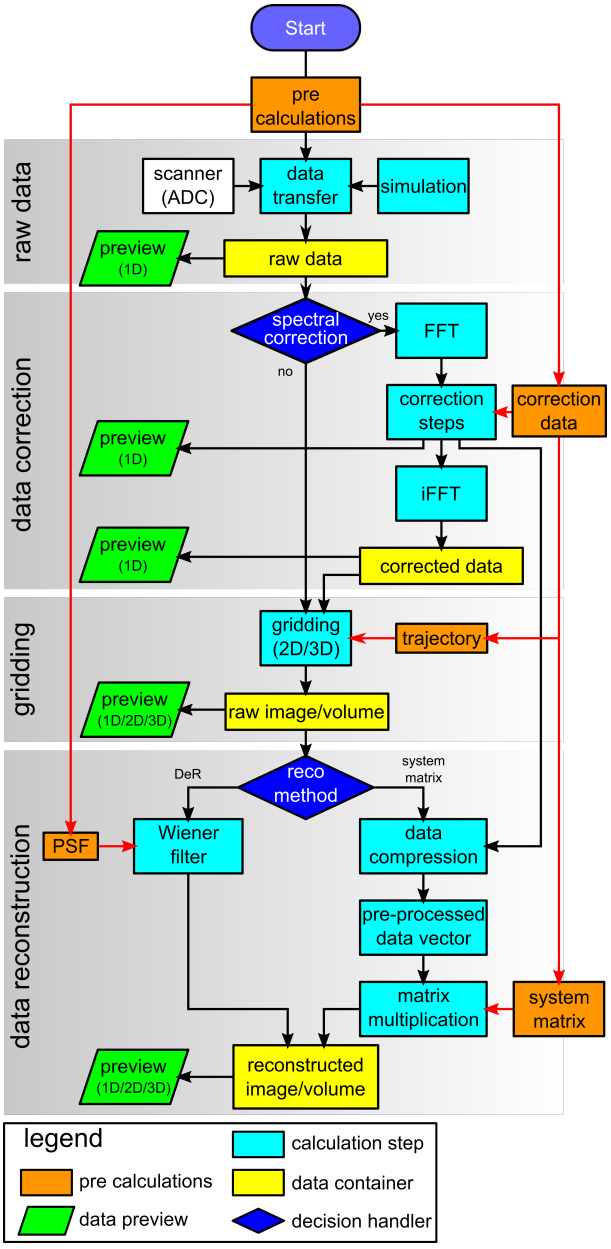

Figure 2: The flow chart shows the entire reconstruction process, which is separated into several sections: pre-calculation step, raw data step, correction step, gridding step, reconstruction step, and visualization step.

Pre-calculation step

Depending on the MPI system in a first step all static data can be pre-calculated, which includes pre-processing of the data sets required for background subtraction [12], correction of the receive chain (RCC), and relaxation correction [13]. Furthermore, the required point-spread-function (PSF) [14] for reconstruction using a deconvolution (deconvolution reconstruction - DeR) or the pseudo

inverse of the measured or simulated system matrix[1] M can be calculated in this prior step (see Sec. II.2).

Raw data step

The main routine starts with the raw data, which are acquired within the MPI scanner and transferred from the ADC. The data transfer from the ADC to the PC can be the first bottleneck in the entire reconstruction process. Depending on the transfer protocol and the size of the data set, the transfer times can vary. E.g. for a data set with $2{\cdot}10^6$ data points (16 bit integer) with a size of 3.9 MB sent via 1 GBit/s LAN, a theoretical throughput of about 31 data sets per second is possible. This gives a transfer time of about 32 ms per data set. For comparison, using USB2.0 (480 MBit/s) or USB3.0 (5 GBit/s) the transfer time is approx. 67 ms (15 data sets per second) and 6.5 ms (153 data sets per second) respectively.

It is also possible to generate raw data with a dedicated simulation tool [15]. These simulated data usually need no corrections that are necessary with respect of raw data from measurements due to distortions occurring from the receive chain. In this case the entire correction section can be skipped.

Correction step

In the next section, the raw data is edited and corrected. For that several steps have to be performed like background subtraction using a previously acquired data set, bandpass filtering, peak filtering based on the excitation frequencies, correction of receive chain distortions using a previously acquired data set, correction of relaxation effects (model based) [13], and calculations of derivatives of the signal [16]. Since all these steps can be performed in Fourier space, a fast Fourier transform is used [17].

The result is a corrected raw data set, which is comparable to a simulated data set. This is the reason why both simulated as well as experimental data can be used for further processing, either by directly processing the spectral data with Fourier-based methods (system matrix reconstruction - peak picking in Fourier space [10]) or by transforming data back in time domain with subsequent gridding for x-space based reconstructions (imaged-based reconstruction path) [3, 11].

Gridding step

Depending on their trajectory, frequencies, sampling rate (SR), trigger point (TP) and scanning method in the gridding section the raw image (raw volume) is generated [3, 16] using a pre-calculated path (see Sec. B).

[1] Basically, there are two ways available to generate a system matrix: it can be measured point-by-point or simulated using an appropriate model of the hardware and particle system. In addition, the values filling up the system matrix can be taken from Fourier space (peak picking) [10] or from the raw images [3] (see Sec. A).

This raw image (raw volume) is independent from the MPI scanner settings. That means that for scanners of the same type and same proportions but with different settings (frequencies, TP, SR) data from this point on can be treated identically [3].

Depending on the reconstruction path (imaged-based SM or x-space), this step can be used optionally.

Reconstruction step

For the final reconstruction step there are basically two methods available, the deconvolution reconstruction (DeR), which generates the final image (volume) using a Wiener filter and a suitable point spread function (PSF) [4, 16], and the system matrix (SM) approach. Depending on the SM type (Fourier- [10] or image-based [3]) the Fourier data or the raw image (raw volume) data are used for preparing the pre-processed data vector containing the specific information about the data suitable to the system matrix. In a final multiplication with the (pseudo) inverse system matrix, this vector is used for generating the final reconstructed image or volume.

Additionally, there is the possibility of omitting any of the reconstruction methods and preview directly the raw image/volume data, which is required for x-space reconstruction (peak-picking in Fourier space) [11, 14].

Visualization step

During the entire process there are several states providing the possibility of previewing the data. 1D preview for time signals as well as Fourier transformed signals and 2D/3D for previewing images or volumes respectively. For 3D preview a home-built visualization tool is available based on Open Graphics Library - OpenGL (Khronos Group) connected via inter process communication (IPC), which provides rapid data transfer between several independent processes sharing the same memory in RAM.

There are many steps necessary to get a reconstructed image, with some steps containing time consuming calculations like FFT/iFFT (using standard implementation), the relaxation correction (deconvolution with Wiener filter), 2D(3D)-deconvolution with the PSF or multiplication of large system matrices with the pre-processed data vector.

However, there are several possibilities available to circumvent potential bottlenecks. In the following, some important optimization steps are shown.

1. Optimization step - using highly optimized libraries

For n-dimensional (inverse) fast Fourier transforms (FFT/iFFT) there are optimized libraries available e.g. the 'fastest Fourier transform in the west' (FFTW) [17],

Intel® Math Kernel Library (MKL), or enormous Fast Fourier Transform (eFFT) [18]. These libraries provide a variety of algorithms and choose the preferable one for a given problem. The speed stems from the fact, that after the transform is broken into subtransforms of sufficiently small sizes, the library uses hard-coded unrolled FFTs for these small sizes. The calculation time for an FFT of a specific size is predictable and constant, because the required look-up tables can be pre-calculated in an earlier step.

In Fig. 3 (top) the results of a benchmark is shown using FFTW 3.3 library. The benchmark tool was compiled with Delphi 10.2 Tokyo (Embarcadero, USA). A data set with the size $1 \cdot 10^8$ data points (double) filled with random numbers is transformed using `fftw_plan_dft_r2c_1d` with flag `estimate`. With increasing number of threads used for the transform as well as with increasing cache size available, the wall time can be reduced. However, the wall time decreases not linearly with the amount of threads due to the memory-intensive calculation and the limited data throughput of the CPUs.

For the multiplication of large matrices as well as the singular value decomposition (SVD) calculation, which is necessary for generating the pseudo inverse of the system matrix, the linear algebra package (LAPACK) and basic linear algebra subprograms (BLAS) are available [19]. These libraries offer many routines to solve mathematical problems. They are written in Fortran and highly optimized with respect to speed.

For matrix multiplication the routine `DGEMM` is used, for SVD the routine `DGESDD`. In Fig. 3 (bottom) the result of a benchmark test running on an Intel i7 CPU (i7 4750HQ, 2-3.2 GHz, 6 MB cache) is shown. With increasing size of the matrix, the calculation time grows rapidly.

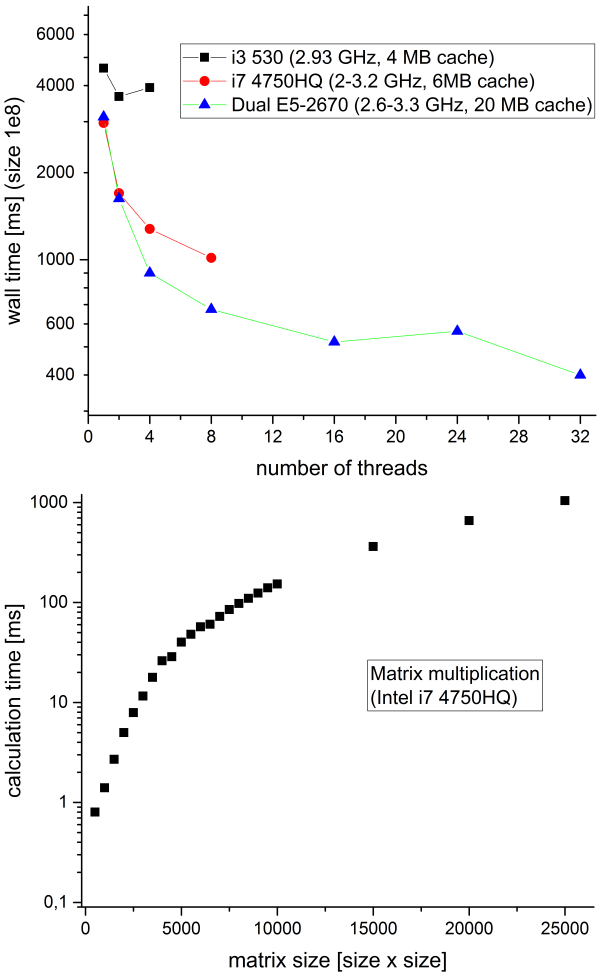

Figure 3: Top: Results of benchmarking the FFTW 3.3 library on different machines. Bottom: matrix multiplication benchmark (LAPACK) with increasing matrix size.

2. Optimization step - pre-calculation

Another optimization step is the possibility of pre-calculation static data like RCC data, relaxation correction data, PSF data, trajectory data, data required for deconvolution, and inverse system matrix data (based on SVD to modify truncation and weighting easily).

2.1. Pre-calculation of deconvolution parameters

A deconvolution process with a Wiener filter can be partially pre-calculated as described in the following. The raw data $y(t)$ is a convolution of the distribution $x(t)$ with the point-spread-function $h(t)$ superimposed by measurement noise $n(t)$.

$$y(t) = h(t) * x(t) + n(t) \tag{1}$$

The Wiener filter is an algorithm for deconvolving $y(t)$ to get $x(t)$, whereby the filter kernel $g(t)$ is calculated to estimate $x(t)$:

$$\hat{x}(t) = (g * y)(t). \tag{2}$$

In the Fourier space it is defined as

$$G(f) = \frac{H^*(f)S(f)}{|H(f)|^2 S(f) + N(f)}, \tag{3}$$

where $G(f)$ and $H(f)$ denote the Fourier transforms of $g(t)$ and $h(t)$, $S(f)$ the power spectral density of $x(t)$, $N(f)$ the mean power spectral density of the noise $n(t)$. $H^*(f)$ is the complex conjugated of $H(f)$. $S(f)$ and $N(f)$ can be approximated by a constant value, which allows to pre-calculate $G(f)$ and reduce the entire deconvolution step to a simple complex multiplication in the Fourier space.

2.2. Pre-calculation of pseudo inverse matrix

The pseudo inverse of a matrix M can be estimated using singular value decomposition (SVD) [3]. As a result, this

calculation gives three further matrices U, W, and V^T:

$$M = U W V^T, \tag{4}$$
$$M \in \mathbb{R}^{m\times n}, U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n},$$
$$W = \mathrm{diag}(k_1, \ldots, k_n) \in \mathbb{R}^{m\times n},$$

where the pseudo inverse $\widetilde{M}^{-1}$ is calculated by

$$\widetilde{M}^{-1} = V W^{-1} U^T. \tag{5}$$

The parameters k_i represent the singular values, which are stored in descending order. It is useful to pre-calculate the matrices and store them in the right form (see Eq. (5)), because of the time-consuming SVD [3]. It has to be emphasized that another pre-calculation step is useful, which calculates $\widetilde{M}^{-1}(k_{\mathrm{trunc}}, w_i)$ with the truncation level k_{trunc} and a Tikhonov filtering with weighting w_i [20]. Less truncation normally results in enhanced image quality at the cost of larger data for the pre-calculation but without cost for the real-time reconstruction afterwards.

2.3. Pre-calculating receive chain correction data

To correct the distortions of a receive chain of an MPI system, the transfer function has to be measured. The components in a receive chain can distort the signal in amplitude and phase over the entire bandwidth of the signal. The measured data can be pre-calculated in a prior step to reduce the calculation to a simple multiplication (amplitude) and addition (phase) in polar coordinates.

2.4. Pre-calculating of FFT calculation plans

Fast Fourier transform algorithms often use calculation plans, which only depend on the size of the data and the hardware (CPU, multithreading, cache, etc.). Creating these plans can require a lot of time depending on the accuracy. E.g. for FFTW library there are several flags available (`FFTW_ESTIMATE`, `FFTW_MEASURE`, and more [17]). To improve the speed of Fourier transformation, these plans should be pre-calculated.

3. Optimization step - calculations in Fourier space

All processes in the correction step, background subtraction, bandpass as well as peak filtering, receive chain correction, relaxation correction, and calculation of derivatives, can be performed in Fourier space. The advantage in contrast to processing in time domain is the possibility of reducing the amount of calculation steps in an easy way[2].

[2] It is also possible to perform all calculations in time domain, but it is less intuitive and requires algorithms that are more sophisticated (binning, finite-impulse filters, etc.).

Only the data within the range of the bandpass has to be calculated. This reduces the number of data points for a data set acquired in a TWMPI scanner driving in the slice-scanning mode (SSM) [16] from $2 \cdot 10^6$ data point (sampling rate 100 MS/s) down to 15500 (bandpass from 25 kHz to 800 kHz). This results in a total number of data points (symmetry as well as real and imaginary part) of $2 \cdot 2 \cdot 15500 = 62000$ data points, which have to be calculated. Thus a reduction of over 95 % can be reached.

Since data coming from an MPI system contains exclusively real data, after a Fourier transform only half of the spectrum need be used for further steps.

The remaining data points can be processed within the same for-loop, where the background subtraction, the correction of the receive chain (RCC) as well as the relaxation correction use pre-calculated data to reduce the calculation time.

III. Results

The implementation of all pre-calculation steps and the usage of optimized libraries end up in a reconstruction process, which is fast enough for real-time applications in 2D and 3D [21]. In Tab. 1 the calculation times of several steps are shown for different sizes of input raw data, different sizes of raw images, reconstruction methods (image-based deconvolution, image-based system matrix reconstruction) and reconstruction paths (peak-picking in Fourier space). For the image-based reconstruction path (deconvolution as well as system matrix reconstruction), a measured TWMPI data set is used [3]. For the peak-picking reconstruction path a simulated data set of a MPI scanner using Lissajous trajectory is used [5, 10]. The software is compiled with Delphi 10.2 Tokyo (Embarcadero, USA) and was running on a mobile i7 CPU (Intel i7 4750HQ, 2-3.2 GHz, 6 MB cache, 16 GB RAM, Win7 64 bit). The most expensive process is the gridding step especially for highly resolved raw images (500×500 px^2). The reason is the expensive interpolation step (see Fig. 2).

However, as mentioned in Fig. 1 the entire imaging process also contains the sequence as well as the data transfer times, which also can significantly contribute (see Sec. II,raw data step).

In Tab. 2 the results of a benchmark for a real experiment are given. For that, the reconstruction software directly runs on a digital oscilloscope (HDO8038, LeCroy, USA) used for the data acquisition (12 bit ADC).

The communication of the reconstruction software with the ADC is implemented over a local TCP/IP protocol. The transfer times are in the range of the theoretical transfer rates of a 1 Gbit LAN interface (see Sec. II, raw data step).

The measuring of the frame rates, especially the higher frame rates, was done by using an external ad-

Table 1: Overview of the computation times for several reconstruction steps depending on different sizes for the raw data, the raw images, reconstruction methods and reconstruction path. Since the acquisition time is 20 ms, the sampling rate is 10 MS/s and 100 MS/s for $2 \cdot 10^5$ and $2 \cdot 10^6$ respectively. The filter bandwidth is 25 kHz, ..., 800 kHz (15500 data points). The Fourier-based reconstruction path uses a specific peak-picking algorithm, which depends on the scanner architecture [3, 10][1)]. For the Fourier transform the FFTW library working on 4 threads is used. All results were averaged 10 times on an Intel i7 4750HQ (2-3.2 GHz, 6 MB cache). The size of the system matrix is 4150×2059 entries[2)] for imaged-based system matrix reconstruction and 4048×2025 for Fourier-based[3)]. For displaying the reconstructed image, a non-optimized algorithm is used[4)].

Reconstruction path	**Image-based**	**Image-based**	**Image-based**	**Image-based**	**Peak-picking (Fourier space)**
Data size	$2 \cdot 10^5$	$2 \cdot 10^5$	$2 \cdot 10^6$	$2 \cdot 10^6$	$2 \cdot 10^5$ / $2 \cdot 10^6$
Raw image size	(100×100 px^2)	(500×500 px^2)	(100×100 px^2)	(500×500 px^2)	–
FFT	~ 1 ms	~ 1 ms	~ 7.5 ms	~ 7.5 ms	~ 1 ms / ~ 7.5 ms
Background subtraction	~ 0.1 ms	~ 0.1 ms	~ 0.1 ms	~ 0.1 ms	~ 0.1 ms / ~ 0.1 ms
Filtering and RCC	~ 1.3 ms	~ 1.3 ms	~ 3.7 ms	~ 3.7 ms	–
Relaxation correction	~ 0.1 ms	~ 0.1 ms	~ 0.2 ms	~ 0.2 ms	–
Calculation of derivatives	~ 0.5 ms	~ 0.5 ms	~ 0.5 ms	~ 0.5 ms	–
Data preparation[1)]	–	–	–	–	~ 1 ms / ~ 1 ms
iFFT	~ 1 ms	~ 1 ms	~ 11 ms	~ 11 ms	–
Gridding	~ 5 ms	~ 27 ms	~ 11 ms	~ 37 ms	–
DeR/SM[2)]	~ 5.5 ms / ~ 13 ms	~ 9 ms / ~ 13 ms	~ 5.5 ms / ~ 13 ms	~ 9 ms / ~ 13 ms	–
SM[3)]	–	–	–	–	~ 13 ms / ~ 13 ms
Displaying[4)]	~ 0.5 ms / ~ 3.5 ms	~ 2 ms / ~ 3.5 ms	~ 0.5 ms / ~ 3.5 ms	~ 2 ms / ~ 3.5 ms	~ 3.5 ms / ~ 3.5 ms
$\sum$	**15.0 ms / 25.5 ms**	**42.0 ms / 47.5 ms**	**40.0 ms / 50.5 ms**	**71.0 ms / 76.5 ms**	**18.6 ms / 25.1 ms**

justable trigger and determining the point in time when the reconstruction software could not show the reconstructed images anymore.

IV. Discussion

In Tab. 1 an overview of the computation times for all steps is given. It can be seen that the most prominent values are those for the gridding process for 500×500 px^2 sized images. A closer look at the gridding process, which is described in [3] in more detail, shows, that for large raw images this step requires some time. The reason is the interpolation, which is necessary to fill up the empty spaces, arising from the gridding process (see Sec. B) [3]. The interpolation step can be speeded up by either using smaller raw images or by optimizing the algorithm and memory access procedures.

The calculation steps performed in Fourier space for computing background subtraction, filtering, receive chain correction, relaxation correction and calculation of derivatives, are the same with respect to the amount of data points, which have to be calculated, for different data sizes. In both cases, data size of $2 \cdot 10^5$ and $2 \cdot 10^6$, the resolution in Fourier space is the same due to adjusted sampling rates of 10 MS/s and 100 MS/s. However, the calculation times differ slightly (see Tab. 1). The reason is the data size, which have to completely fit in the CPU cache for efficient calculation.

The computation time for the system matrix approach, which contains in this state only a matrix multiplication, shows an acceptable value, but for higher resolved images, it grows almost quadratically (see Fig. 3, bottom). Here it is useful to optimize this step by using several small patches with suitable system matrices [3] and calculate them in a parallel processing procedure. However, this requires more RAM and an efficient merging process. This could be achieved using multiple CPUs.

It is also useful to think about outsourcing some calculation steps into a graphical processor unit (GPU), which allows fast calculations with a high parallelization capability [22].

The entire imaging process depends on the size of the data sets starting with the transfer times (~ 10 ms for

Table 2: Benchmark of the entire imaging process from press-the-button until the reconstructed image for different reconstruction paths and methods. The software directly works on the digital oscilloscope HDO8038 providing a full operating PC (Intel i5 4570S, 2.9-3.6 GHz, 6 MB cache, 8 GB RAM, Win7 64 bit).

Reconstruction path	**Image-based** ($\mathbf{100 \times 100\ px^2}$) / ($\mathbf{500 \times 500\ px^2}$)				**Peak-picking (Fourier space)**
Data size	$2 \cdot 10^5$	$2 \cdot 10^5$	$2 \cdot 10^6$	$2 \cdot 10^6$	$2 \cdot 10^5$ / $2 \cdot 10^6$
Data transfer time	~ 10 ms	~ 10 ms	~ 35 ms	~ 35 ms	~ 10 ms / ~ 35 ms
Reco type	DeR	SM	DeR	SM	SM
Frame rate	~ 31 / ~ 17	~ 31 / ~ 14	~ 7 / ~ 6	~ 7 / ~ 6	~ 31 / ~ 9

$2 \cdot 10^5$ and ~ 35 ms for $2 \cdot 10^6$) followed by Fourier transform. The correction step shows quite similar calculation times for the same filter settings. The most prominent time value arises from the gridding process, especially from the filling-up step. The gridding strongly depends on the size of the raw image, which is also noticeable in the deconvolution process (DeR).

In total, the reconstruction time values lay all below 100 ms. However, for small sizes of data sets and small raw images (see Tab. 1), the whole process of starting the sequence for the scanner and the data transfer dominate the time consumptions. The time lag between press start in the software and sequence start at the scanner is in the range of few milliseconds. The ADC provides the data essentially instantaneously after finishing the sequence (here 20 ms).

To overcome the bottleneck of data transfer, optimizations on the hardware can be applied to reduce the data. Using FPGAs (field-programmable gate arrays) allows implementing highly specific and quite fast data processing steps on specified ICs. E.g. by binning the time signal or preprocessing the Fourier transform the amount of data, which has to be sent to the PC can drastically be reduced.

Processing 3D data sets using the gridding path filling up a volume followed by a 3D deconvolution reconstruction using a pre calculated 3D PSF is possible way to get a reconstructed 3D data set. A 3D gridding with interpolation as well as a 3D deconvolution can be a quite time consuming procedure. In [21] an images-based system matrix reconstruction approach is demonstrated, which allows processing 3D data within the gridding path. However, for more complex trajectories it is recommended to use the way by peak-picking in Fourier space and system matrix reconstruction.

V. Conclusion

The implementation of the presented software package allows to reconstruct raw data from MPI systems in real-time with low latency. Depending on the raw data size, the raw image size and the reconstruction method, it is possible to reduce the entire reconstruction process to about 15 ms (data size $2 \cdot 10^5$ and image size $100 \times 100\ px^2$). Additional time will be required for data transfer from the ADC to the computer and also for running the sequence on the scanner (no parallelization). All together frame rates between 6 and 31 frames per second are possible, which should be sufficiently fast for pre-clinical application like MPI-guided percutaneous transluminal angioplasty (PTA).

Acknowledgements

This work was partially funded by the DFG (BE-5293/1-1).

A. System matrix generation

For generating the system matrix a delta-like sample is positioned on a grid in the field of view (FOV) to obtain a specific signal for every coordinate value [10] (see Fig. A.1). Several different methods are available for picking the required information: e.g. peak-picking for Fourier-based system matrices [5] or raw images for image-based system matrices [3]. However, in principle the data filling the system matrix have been prepared in the same way as the acquired data set, which has to be reconstructed.

B. Gridding process

As an example, in Fig. B.1 the gridding process for a 2D trajectory (slice-scanning mode TWMPI [3, 16]) is indicated. The path can be pre-calculated in a prior step depending on the size of the raw image (Fig. B.1 (a)), before the acquired data points are gridded point-by-point along them (Fig. B.1 (b)). Using linear interpolation, the empty space between adjacent data points can be filled-up to generate a smooth raw image (Fig. B.1 (c)).

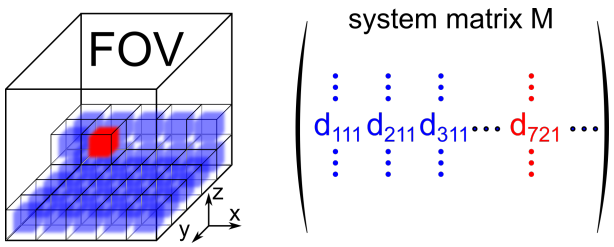

Figure A.1: Sketch of the system matrix generation. A data set is generated for every position on a grid in the FOV using a delta-like sample for measurement-based system matrices or using a simulation for model-based system matrices. The desired information of every data set is stored in the rows of the system matrix whereby the number of columns represents the amount of voxels in the FOV.

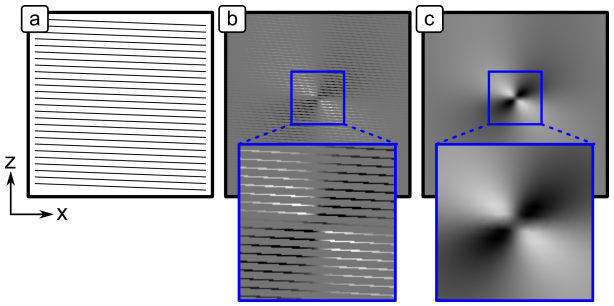

Figure B.1: Sketch of the gridding process for a 2D trajectory. (a) Shows the path of the trajectory, which can be precalculated. (b) Along the path, the data are gridded point-by-point. (c) After filling up the empty space by linear interpolation, the raw image is generated.

References

[1] N. Panagiotopoulos, R. L. Duschka, M. Ahlborg, G. Bringout, C. Debbeler, M. Graeser, C. Kaethner, K. Lüdtke-Buzug, H. Medimagh, J. Stelzner, T. M. Buzug, J. Barkhausen, F. M. Vogt, and J. Haegele. Magnetic particle imaging: current developments and future directions. *Int. Journ. Nanomed.*, 10:3097, 2015. doi:10.2147/IJN.S70488.

[2] T. Knopp, T. Viereck, G. Bringout, M. Ahlborg, J. Rahmer, and M. Hofmann. MDF: Magnetic Particle Imaging Data Format. *arXiv:1602.06072 [physics.med-ph]*, 2016.

[3] P. Vogel, T. Kampf, M. A. Rückert, and V. C. Behr. Flexible and Dynamic Patch Reconstruction for Traveling Wave Magnetic Particle Imaging. *Intern. J. Magnetic Particle Imaging*, 2(2):1611001, 2016. doi:10.18416/ijmpi.2016.1611001.

[4] P. Vogel, M. A. Rückert, P. Klauer, W. H. Kullmann, P. M. Jakob, and V. C. Behr. Superspeed Traveling Wave Magnetic Particle Imaging. *IEEE Trans. Magn.*, 51(2):6501603, 2015. doi:10.1109/TMAG.2014.2322897.

[5] J. Weizenecker, B. Gleich, J. Rahmer, H. Dahnke, and J. Borgert. Three-dimensional real-time in vivo magnetic particle imaging. *Phys. Med. Biol.*, 54(5):L1–L10, 2009.

[6] T. Knopp and M. Hofmann. Online reconstruction of 3D magnetic particle imaging data. *Phys. Med. Biol.*, 61(11):N257–N267, 2016. doi:10.1088/0031-9155/61/11/N257.

[7] J. Lampe, C. Bassoy, J. Rahmer, J. Weizenecker, H. Voss, B. Gleich, and J. Borgert. Fast reconstruction in magnetic particle imaging. *Phys. Med. Biol.*, 57(4):1113–1134, 2012. doi:10.1088/0031-9155/57/4/1113.

[8] A. Weber and T. Knopp. Reconstruction of the Magnetic Particle Imaging System Matrix Using Symmetries and Compressed Sensing. *Adv. Math. Phys.*, 2015:460496, 2015. doi:10.1155/2015/460496.

[9] J. Weizenecker, B. Gleich, and J. Borgert. Magnetic particle imaging using a field free line. *J. Phys. D: Appl. Phys.*, 41(10):105009, 2008. doi:10.1088/0022-3727/41/10/105009.

[10] J. Rahmer, J. Weizenecker, B. Gleich, and J. Borgert. Signal encoding in magnetic particle imaging: properties of the system function. *BMC Medical Imaging*, 9(4), 2009. doi:10.1186/1471-2342-9-4.

[11] P. W. Goodwill and S. M. Conolly. The x-Space Formulation of the Magnetic Particle Imaging process: One-Dimensional Signal, Resolution, Bandwidth, SNR, SAR, and Magnetostimulation. *IEEE Trans. Med. Imag.*, 29(11):1851–1859, 2010. doi:10.1109/TMI.2010.2052284.

[12] K. Them, M. G. Kaul, C. Jung, M. Hofmann, T. Mummert, F. Werner, and T. Knopp. Sensitivity Enhancement in Magnetic Particle Imaging by Background Subtraction. *IEEE Trans. Med. Imag.*, 35(3):893–900, 2016. doi:10.1109/TMI.2015.2501462.

[13] K. Bente, M. Weber, M. Graeser, T. F. Sattel, M. Erbe, and T. M. Buzug. Electronic field free line rotation and relaxation deconvolution in magnetic particle imaging. *IEEE Trans. Med. Imag.*, 34(2):644–651, 2015. doi:10.1109/TMI.2014.2364891.

[14] P. W. Goodwill and S. M. Conolly. Multidimensional X-Space Magnetic Particle Imaging. *IEEE Trans. Med. Imag.*, 30(9):1581–1590, 2011. doi:10.1109/TMI.2011.2125982.

[15] P. Vogel, M. A. Rückert, and V. C. Behr. 3D-GUI Simulation Environment for MPI. In *International Workshop on Magnetic Particle Imaging*, 2016.

[16] P. Vogel, M. A. Rückert, P. Klauer, W. H. Kullmann, P. M. Jakob, and V. C. Behr. First in vivo traveling wave magnetic particle imaging of a beating mouse heart. *Phys. Med. Biol.*, 61(18):6620–6634, 2016. doi:10.1088/0031-9155/61/18/6620.

[17] M. Frigo and S. G. Johnson. FFTW library, 2004. URL `www.fftw.org`.

[18] R. Asai and A. Vladimirov. Intel Cilk Plus for complex parallel algorithms: "Enormous Fast Fourier Transforms" (EFFT) library. *Parallel Comput.*, 48:125–142, 2015. doi:10.1016/j.parco.2015.05.004.

[19] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users Guide*. SIAM, Philadelphia, 1999. doi:10.1137/1.9780898719604.

[20] T. Knopp and T. M. Buzug. *Magnetic Particle Imaging: An Introduction to Imaging Principles and Scanner Instrumentation*. Springer, Berlin/Heidelberg, 2012. doi:10.1007/978-3-642-04199-0.

[21] P. Vogel, M. A. Rückert, P. Klauer, S. Herz, T. Kampf, T. A. Bley, and V. C. Behr. Real-time 3D Dynamic Rotating Slice-Scanning Mode for Traveling Wave MPI. *Intern. J. Magnetic Particle Imaging*, 3(2):1706001, 2016. doi:10.18416/ijmpi.2017.1706001.

[22] J. Fung, F. Tang, and S. Mann. Mediated Reality using Computer Graphics Hardware for Computer Vision. In *Proceedings of the International Symposium on Wearable Computing 2002 (ISWC2002)*, pages 83–89, 2002.