International Journal on Magnetic Particle Imaging
Vol 11, No 1, Suppl 1, Article ID 2503048, 2 Pages

Proceedings Article

GPU Accelerated Multi-Patch Reconstruction

Niklas Hackelberg[a,b,*] · Marija Boberg[a,b] · Tobias Knopp[a,b,c]

[a]Section for Biomedical Imaging, University Medical Center Hamburg-Eppendorf, Hamburg, Germany
[b]Institute for Biomedical Imaging, Hamburg University of Technology, Hamburg, Germany
[c]Fraunhofer Research Institution for Individualized and Cell-based Medical Engineering IMTE, Lübeck, Germany
[*]Corresponding author, email: niklas.hackelberg@tuhh.de

Abstract

The multi-patch approach in magnetic particle imaging is used to capture large field of views. System-matrix-based image reconstruction for this approach often considers a joint system of equations to minimize artifacts. Due to the prohibitive size of this inverse problem, reconstructions rely on iterative algorithms that do not need to keep the entire system matrix in memory. This work shows a graphical processing unit accelerated implementation of a generalized multi-patch operator. The achieved runtime improvements allow for multi-patch reconstructions using different optimization algorithms, which in turn allow for a flexible choice of regularization terms.

I. Introduction

An important area in the field of system-matrix-based reconstruction for magnetic particle imaging (MPI) is multi-patch reconstruction. In order to obtain large field of views (FOV), the FOV is divided into smaller patches, which are then scanned sequentially. The patches are often considered as a joint problem of the linear system of equations to avoid artifacts. Such a joint approach is taken with the generalized multi-patch reconstruction described in [1], which uses clusters of similar patches and exploits the sparse nature of the system equations.

This joint multi-patch reconstruction approach is implemented in the open-source Julia package MPIReco.jl [2] with an efficient implementation for the Kaczmarz algorithm with l_2 regularization. A variety of different optimization algorithms and different regularization terms have been investigated for single-patch MPI. However, the problem size of multi-patch has been prohibitive in both memory and runtime requirements. As such the generalized approach with Kaczmarz and the approach of [3] have been used in multi-patch MPI so far.

An important factor in runtime improvements is graphics processing unit (GPU) acceleration, which has been investigated for single-patch MPI [4]. Kaczmarz has poor GPU performance due to data dependencies that cannot be parallelized for one measurement. Unlike, single-patch reconstructions, multi-patch cannot make use of existing linear algebra functions of a GPU back-end, but instead requires custom GPU functions.

In this work we present a GPU accelerated generalized multi-patch operator based on custom GPU kernels. This operator was added to MPIReco.jl together with general GPU acceleration support for different GPU vendors, such as NVIDIA and AMD cards. The achieved runtime improvements allow for usage of optimization algorithms other than Kaczmarz such as conjugated gradient applied to the normal equation (CGNR) while retaining comparable performance. These algorithms in turn allow for a more flexible choice of regularization terms.

II. Methods and materials

Iterative solvers often require the matrix-vector product of an operator, as well as its adjoint. The forward and adjoint operation of the generalized multi-patch operator

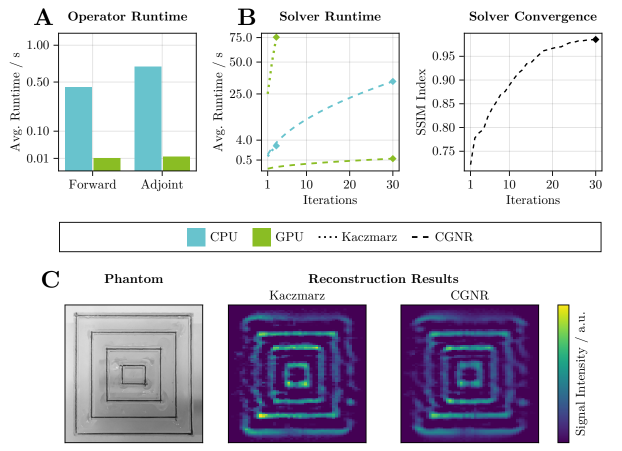

Figure 1: A multi-patch phantom was reconstructed using Kaczmarz with 3 iterations as described in [1] and CGNR with 30 iterations on CPU and GPU. Subfigure A shows the runtime of the forward and adjoint application of the operator on a CPU and a GPU. Subfigure B shows the iteration-dependent runtime of CGNR and Kaczmarz and the SSIM index of CGNR compared with the final reconstruction of Kaczmarz.

are given by eq. (12) and (13) in [1].

To implement these equations an operator needs to store system matrices and several mappings per patch. These mappings translate a given row of the operator to a patch and a system matrix row, as well as provide sparse index mappings to elements of the matrix row and measurement vector. This way sparsity along the rows of the operator can be achieved. The same mappings can be reused for the adjoint operation, where a column of the adjoint operator refers to a row of a system matrix.

GPUs are programmed with functions that are called kernels, which are executed in parallel on hundreds of threads. Our custom kernels are implemented with the Julia package KernelAbstractions.jl[1], which allows one to write kernels that can target different GPU backends. Such custom kernels are able to access the appropriate system matrices and mappings of our operator. The forward operation can be parallelized along the rows with groups of threads performing a parallel sparse dot product. The adjoint operation features a data dependency, as different threads may access the same result element. However, as long as the element is updated in an atomic manner, this can still be parallelized along the columns.

To investigate the runtime improvement we replicated a reconstruction from [1]. All executions are repeated 20 times and we report the average runtime. We use 9 system matrices for the reconstruction which we map to 15 patches. We consider 1953 frequency components and a grid of $49 \times 21 \times 86$, resulting in a sparse operator of size 29295×88494. More measurement details can be found in [1].

To see the runtime improvements of the GPU kernels, we apply both the forward and the adjoint individually. We also perform reconstructions with the operator using the Conjugate Gradient Normal Residual (CGNR) algorithm with 30 iterations and compare its runtime and SSIM index of the reconstructed image to the Kaczmarz result from [1] which used 3 iterations.

[1]https://github.com/JuliaGPU/KernelAbstractions.jl

III. Results and discussion

The results are shown in Figure 1. As can be seen in subfigure A, the forward and adjoint operation could be accelerated by roughly one order of magnitude. CGNR's GPU-accelerated reconstruction achieves a fourfold speedup compared to Kaczmarz's CPU reconstruction.

IV. Conclusion

In this work we have shown the GPU acceleration of a generalized multi-patch operator. The speedup of the operator is slightly offset by the slower convergence and higher computational complexity of CGNR. Nonetheless, with GPU acceleration we can now consider different solvers with more complex regularization terms with improved or comparable runtime to state-of-the-art Kaczmarz based reconstructions.

Author's statement

This work was supported in part by the German Research Foundation (DFG) under Grant KN 1108/2-2. Conflict of interest: Authors state no conflict of interest.

References

[1] M. Boberg, T. Knopp, P. Szwargulski, and M. Möddel. Generalized mpi multi-patch reconstruction using clusters of similar system matrices. *IEEE transactions on medical imaging*, 39(5):1347–1358, 2019.

[2] T. Knopp, P. Szwargulski, F. Griese, M. Grosser, M. Boberg, and M. Möddel. MPIReco.jl: Julia package for image reconstruction in MPI. *International Journal on Magnetic Particle Imaging IJMPI*, 5(1-2), 2019.

[3] L. Zdun, M. Boberg, and C. Brandt. Joint multi-patch reconstruction: Fast and improved results by stochastic optimization. *International Journal on Magnetic Particle Imaging IJMPI*, 8(2), 2022.

[4] K. N. Quelhas, M.-A. Henn, R. Farias, W. L. Tew, and S. I. Woods. Gpu-accelerated parallel image reconstruction strategies for magnetic particle imaging. *Physics in Medicine & Biology*, 69(13):135005, 2024.