



OPEN ACCESS

EDITED BY

Mario Petretta,
IRCCS SYNLAB SDN, Italy

REVIEWED BY

Pablo Galve,
Complutense University of Madrid, Spain
Kevin T. Chen,
National Taiwan University, Taiwan
Nikos Efthimiou,
Harvard Medical School, United States

*CORRESPONDENCE

Georg Schramm
✉ georg.schramm@kuleuven.be

RECEIVED 19 October 2023

ACCEPTED 07 December 2023

PUBLISHED 08 January 2024

CITATION

Schramm G and Thielemans K (2024)
PARALLELPROJ—an open-source framework
for fast calculation of projections in
tomography.
Front. Nucl. Med. 3:1324562.
doi: 10.3389/fnume.2023.1324562

COPYRIGHT

© 2024 Schramm and Thielemans. This is an open-access article distributed under the terms of the [Creative Commons Attribution License \(CC BY\)](https://creativecommons.org/licenses/by/4.0/). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

PARALLELPROJ—an open-source framework for fast calculation of projections in tomography

Georg Schramm^{1*} and Kris Thielemans^{2,3}

¹Department of Imaging and Pathology, Division of Nuclear Medicine, KU Leuven, Leuven, Belgium, ²Institute of Nuclear Medicine, University College London, London, United Kingdom, ³Centre for Medical Image Computing, University College London, London, United Kingdom

In this article, we introduce `parallelproj`, a novel open-source framework designed for efficient parallel computation of projections in tomography leveraging either multiple CPU cores or GPUs. This framework efficiently implements forward and back projection functions for both sinogram and listmode data, utilizing Joseph's method, which is further extended to encompass time-of-flight (TOF) PET projections. Our evaluation involves a series of tests focusing on PET image reconstruction using data sourced from a state-of-the-art clinical PET/CT system. We thoroughly benchmark the performance of the projectors in non-TOF and TOF, sinogram, and listmode employing multi CPU-cores, hybrid CPU/GPU, and exclusive GPU mode. Moreover, we also investigate the timing of non-TOF sinogram projections calculated in STIR (Software for Tomographic Image Reconstruction) which recently integrated `parallelproj` as one of its projection backends. Our results indicate that the exclusive GPU mode provides acceleration factors between 25 and 68 relative to the multi-CPU-core mode. Furthermore, we demonstrate that OSEM listmode reconstruction of state-of-the-art real-world PET data sets is achievable within a few seconds using a single consumer GPU.

KEYWORDS

Emission tomography (PET and SPECT), Positron emission tomography (PET), Image reconstruction, ray tracing algorithm, GPU (CUDA)

1 Introduction

For tomographic imaging techniques used in medicine, such as X-ray computed tomography (CT), positron emission tomography (PET) and single photon emission tomography (SPECT), image reconstruction results are usually expected within seconds or minutes after data acquisition, creating a severe computational challenge when reconstructing data from state-of-the-art systems using iterative algorithms (1). With new scanner generations, this challenge is steadily growing, since (i) the data size is increasing due to higher resolution detectors and scanners with bigger field of view (2), and (ii) more advanced (iterative) reconstruction algorithms are being used that try to exploit more information from the acquired data, which usually necessitates the calculation of a huge amount of projections. An example of the latter is the data-driven motion correction in PET (3) where, instead of reconstructing a single “static frame”, many very short time frames are reconstructed and subsequently used for motion estimation and correction. Another example for (ii) is the combination of deep learning and tomographic image reconstruction (4–6), using, e.g., unrolled networks, where during training a tremendous number of projections

also have to be calculated to evaluate the gradient of the data fidelity term across a mini batch in every training epoch.

For most tomographic image reconstruction algorithms, the bottleneck in terms of computation time is the evaluation of a linear forward model that describes the physics of the data acquisition process. In CT, and PET, the forward model includes the computation of many (weighted) line or volume integrals through an image volume, commonly called “projections” - which can be slow when executed on a single processor. Fortunately, for most reconstruction algorithms, the computation of projections can be executed in parallel on multiple processors, e.g., using multiple CPU-cores or one or more graphics processing units (GPUs). Note that the parallel evaluation of the adjoint of the forward model - commonly called “back projection” - is more demanding, since race conditions, where multiple threads/processes need to update the same memory location, can occur. In recent decades, the use of GPUs for faster calculation of projections in tomographic imaging has been studied extensively; see, e.g., (7–22) or the reviews (1, 23) for the use of GPUs in PET reconstruction. All of these articles conclude that the time needed to calculate forward and back projections on state-of-the-art GPUs is usually much shorter compared to using multiple CPU-cores.

Motivated by these findings and the recent availability of very powerful low- and high-level GPU programming frameworks such as CUDA and `cupy` (24), we developed a new open source research framework, called `parallelproj`, for fast calculations of forward and back projections in tomographic image reconstruction.

The objectives of the `parallelproj` framework are as follows:

- To provide an open-source framework for fast parallel calculation of time-of-flight (TOF) as well as non-TOF projections suited for tomographic image reconstruction in sinogram as well as listmode using multiple CPU-cores or GPUs.
- To provide an accessible framework that can be easily installed without the need for compilation of source code on all major operating systems (Linux, Windows, and macOS).
- To provide a framework that can be efficiently used in conjunction with `pytorch` (25) GPU arrays to facilitate research on tomographic imaging methods, including deep learning.

In light of the absence of an open-source framework that fully meets these criteria at the time of writing, this article introduces the new `parallelproj` framework and is structured as follows: We first review Joseph’s method for calculating projections, followed by a short overview of the design choices and implementation of `parallelproj`. Subsequently, we report the results of a few benchmark tests related to image reconstruction in PET with and without TOF information using sinograms or listmode (LM) before ending the article with a detailed discussion and conclusion. In this article, we focus on the performance of `parallelproj` projectors for non-TOF and TOF PET reconstruction problems. Note, however, that the non-TOF Joseph projectors could also be used in iterative CT reconstruction.

2 Materials and methods

2.1 Joseph’s method for projecting rays through voxel images

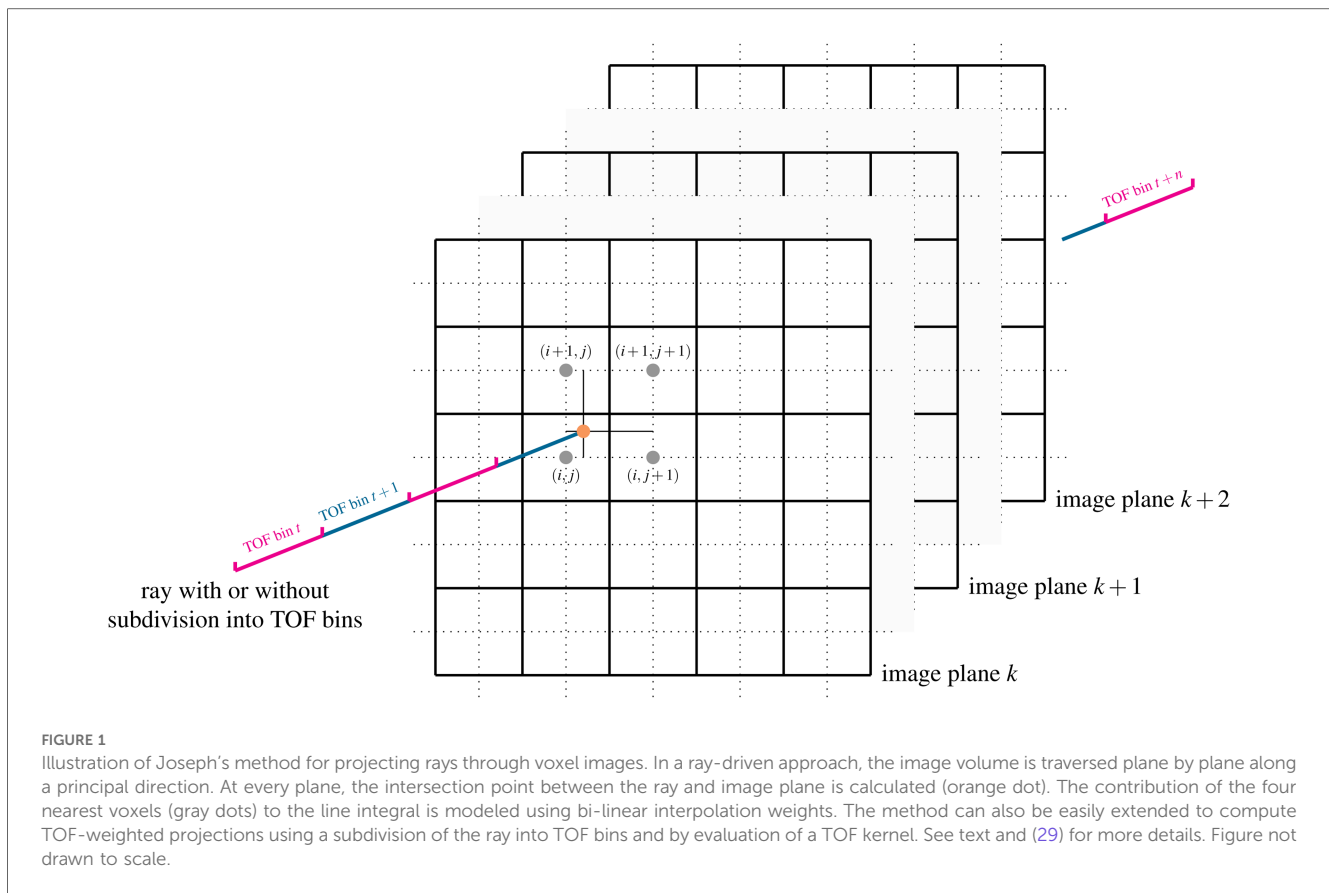
Besides Siddon’s method (26), Wu’s method (27) and the distance-drive method (28), Joseph’s method (29) is a very efficient and popular way to calculate projections in transmission and emission tomography. Rahmin et al. (30) have shown that while being only 20% slower than Siddon’s method, Joseph’s method leads to superior image quality in listmode PET reconstructions. The basic idea of the original Joseph method for calculating line integrals, which can be used to model non-TOF projections, is shown in Figure 1. For a given ray, the algorithm first determines the principal direction in the image that is most parallel to the ray and then steps through the image volume plane by plane along this principal direction. In every plane, the intersection point between the ray and the plane is calculated and the contribution of the image at that point to the line integral is approximated using bi-linear interpolation of the four nearest neighbors around the intersection point. In other words, only the four nearest neighboring voxels are contributing to the line integral and their contributions are given by the bi-linear interpolation weights. Finally, the contributions of all planes are added and corrected for the incidence angle of the ray, see (29) for more details.

An extension of the original Joseph method to calculate TOF-weighted projections is straightforward. For every voxel contributing to the line integral and every TOF bin along the ray, a TOF weight can be computed by evaluating a TOF kernel that is a function of the Euclidean distance between the voxel and the center of the TOF bin. The TOF kernel can be e.g., modeled as a Gaussian kernel - representing the TOF uncertainty of the detection system - convolved with a rectangular function - representing the width of the TOF bin, resulting in the evaluation of the difference of two error functions.

2.2 Design principles and implementation details

The application programming interface (API) to the `parallelproj` framework was designed such that:

- The input to the low-level projector functions are as generic as possible. In practice, that means that these functions take a list of coordinates representing the start and end point of the rays to be projected as input, making the low-level functions agnostic to specific scanner geometries (or symmetries). Thus, any scanner geometry can be modeled.
- Projections can be performed in non-TOF or TOF mode.
- In the TOF mode, optimized projections for sinogram and listmode are available. In the former, the contributions to all available TOF bins along a ray are computed while traversing the image volume plane by plane, whereas in the latter only



the contribution to one specific TOF bin (the TOF bin of a given listmode event) is evaluated.

- The back projections are the exact adjoint of the forward projections (matched forward and back projections).

Parallelization across multiple processors was implemented in two different ways. To enable parallelization across multiple CPUs, a first version of the `parallelproj` library was implemented using C and OpenMP (31) (`libparallelproj_c`). Furthermore, the exact same projector functions were implemented in CUDA to enable parallelization on one or multiple GPUs (`libparallelproj_cuda`). In the CUDA version, the input data is first transferred from the host to all available GPU(s) followed by the parallel execution of the projection kernels. After the kernel execution, the result is transferred back to the host. To handle race conditions, all implementations use atomic add operations in the back projections.

2.3 Availability of source code and precompiled libraries

`parallelproj` is an open-source project and its source code is available at <https://github.com/gschramm/parallelproj> under an MIT license. In addition to the sources, we also offer precompiled libraries (`libparallelproj_c` and `libparallelproj_cuda`) for all major operating systems (Linux, Windows, and macOS) and various recent CUDA

versions using the `conda-forge` package manager.¹ Depending on the presence or absence of supported CUDA devices and drivers, `conda-forge` automatically installs the matching library type. In addition to the precompiled libraries, the `parallelproj` package also includes the source file of the CUDA projection kernels such that they can be directly executed on GPU arrays using frameworks that allow for just-in-time compilation of CUDA kernels such as, e.g., `cupy` (24). Moreover, `parallelproj` also includes a minimal python interface to all projection functions that is compatible with the Python array API standard, enabling efficient projections and back projections of various compatible array classes (e.g., `numpy`, `cupy`, `pytorch` tensors).

2.4 `parallelproj` computation modes

Using the two aforementioned projection libraries, as well as the CUDA projections kernels, projections can be performed in the following three different computation modes:

1. **CPU mode:** Forward and back projections of image volumes (arrays) stored on the host (CPU) can be performed using

¹The CUDA version of the `parallelproj` library is not available for macOS.

`libparallelproj_c` where parallelization across all available CPU cores is performed using OpenMP.

2. **hybrid CPU/GPU mode:** Forward and back projections of image volumes (arrays) stored on the host can be performed using `libparallelproj_cuda` involving data transfer from the host to all available GPUs, execution of projection kernels on the GPUs, and transfer of the results back to the host.
3. **direct GPU mode:** Forward and back projections of image volumes (arrays) stored on a GPU can be performed by direct execution of the projection kernels using a framework that supports just-in-time compilation of CUDA kernels, such as `cupy` (24). In contrast to the hybrid CPU/GPU mode, memory transfer between host and GPU is avoided.²

2.5 Integration of `parallelproj` into STIR

Software for Tomographic Image Reconstruction (STIR) is open-source software for PET and SPECT reconstruction (32, 33). It is a well-established tool for research in scanner modeling and iterative reconstruction methods. Its modular design in C++ allows integrating external components such as projectors. We integrated `parallelproj` into STIR since version 5.0 as a user-selectable projector such that STIR users can benefit from the high performance of the `parallelproj`. STIR's `conda-forge` recipe depends on `parallelproj` and therefore installs the GPU or CPU version accordingly. Moreover, as STIR forms the basis for the PET and SPECT support in the open-source Synergistic Image Reconstruction Framework (SIRF) (34), this was modified to allow calling `parallelproj` from SIRF as well, making `parallelproj` usable for SIRF's advanced algorithms, including motion correction (35).

STIR uses `parallelproj` if the latter's libraries are found by CMake at compilation time. Currently, STIR always uses `libparallelproj_cuda` if present and `libparallelproj_c` otherwise. As `parallelproj` needs the end-points of the lines of responses, these are computed by the STIR interface based on its normal modeling of scanner geometry, defaulting to cylindrical scanners, but recently also accommodating block-cylindrical and arbitrary crystal locations. These computations are performed once at set-up time, and end-points are stored in `std::vectors` suitable for passing to the low-level routines of `parallelproj`. Since STIR 5.2, these arrays are filled in parallel using OpenMP, reducing the set-up time. However, this set-up time is not included in the timings below.

As STIR's data-structures store sinograms and images in CPU memory, this interface uses the hybrid CPU/GPU mode of `parallelproj`. STIR's design for projectors is optimized for low-memory requirements, projecting only small chunks of

sinogram data at the time, using OpenMP or the Message Passing Interface³ (MPI) for non-shared memory architectures. GPU computations however have best performance on larger data-sets. Therefore, the current implementation uses temporary objects to store the result of the forward projection, and data is then copied as necessary. This is similar to the previous integration of `NiftyPET` (19) into STIR. This creates an extra (small) overhead, which could be avoided in the future.

At the time of writing, neither the TOF nor listmode projectors of `parallelproj` have been integrated into STIR. We hope to complete this in the near future.

2.6 Benchmark tests

To evaluate the performance of the `parallelproj` projectors using the computation modes described above, we implemented a series of benchmark tests. All tests are related to a PET image reconstruction task and used the geometry and properties of a state-of-the-art GE Discovery MI TOF PET/CT scanner (36) with 20 cm axial FOV. This scanner has 36 detector "rings", where each "ring" has a radius of 380 mm and consists of 34 modules containing 16 detectors each such that there are $16 \times 34 \times 36 = 19,584$ detectors in total. A non-TOF emission sinogram for this scanner without any data size reduction ("span 1") has 415 radial elements, 272 views, and 1,292 planes, meaning that for a full non-TOF sinogram projection, $415 \times 272 \times 1,292 = 146 \times 10^6$ line integrals have to be evaluated. For TOF data, each line of response (LOR) is subdivided into 29 TOF bins using a TOF bin width of 169 ps (25.4 mm). The reported TOF resolution of the scanner is 385 ps (57.7 mm) FWHM (36). In the TOF projectors of `parallelproj`, the Gaussian TOF kernel is truncated beyond ± 3 standard deviations.

To evaluate the performance of `parallelproj` for projections in sinogram mode, we measured the time needed for a forward and back projection of a span 1 subset sinogram containing 8 equally spaced views in non-TOF and TOF mode. This is equivalent to the projection work required for an OSEM subset update (37–39) using 34 subsets in total, a setting that is used in many clinical reconstructions. Since it is known that the in-memory data order severely affects the computation time, especially on CUDA devices, we varied the order of the spatial axis of the sinogram, as well as the order of the image axis relative to the axial direction of the scanner (symmetry axis mode). In the sinogram order mode "PVR", the radial direction increased the fastest and the plane direction increased the slowest in memory. For the sinogram order mode "VRP", the plane direction increased the fastest and the view direction the slowest. By varying the order of the image axis, we could test the impact

²A requirement for the use of the direct GPU mode is the presence of enough GPU memory to store the input/output images and projections.

³<https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf>.

of different image volume memory layouts. For example, the symmetry axis mode “2” meant that the image volume memory increased the fastest in the axial direction of the scanner, while “0” or “1” meant that one of the transaxial directions increased the fastest. To test the integration of `parallelproj` projection libraries into STIR, the same non-TOF projection benchmarks tests were repeated using the timing tool included in STIR version 5.2. These tests were only performed in the hybrid CPU/GPU mode, since the exclusive GPU mode is currently not available in STIR. Note that the current STIR integration uses the “PVR” and “2” symmetry axis mode. Finally, we also compared the performance of the `parallelproj` projectors in pure GPU and hybrid CPU/GPU mode with the performance of the GPU projectors included in the NiftyPET python package v2.0.0 (40) using a complete forward and back projections of non-TOF sinograms of the Siemens mMR (41). Since NiftyPET uses a span 11 sinogram, and `parallelproj` so far only supports span 1 sinograms, we artificially limited the maximal ring difference in this `parallelproj` test to 7 to obtain a sinogram with approximately the same number of planes (NiftyPET sinogram 837 planes, `parallelproj` sinogram 904 planes). In all cases, an image of shape (344, 344, 127) with a voxel size of (2.08 mm, 2.08 mm, 2.03 mm) was used.

In addition to the sinogram projection tests, we also evaluated the performance of `parallelproj` for non-TOF and TOF projections in listmode as a function of the number of acquired listmode events. Instead of randomly generating the event coordinates, listmode events from an acquisition of a NEMA image quality phantom were used, which guaranteed a more realistic event distribution. In contrast to the projections in sinogram mode, where the ray directions and memory access are somehow ordered, they are random for unsorted listmode data. Similarly to the sinogram tests, the symmetry axis of the scanner was also varied. For all sinogram and listmode projection benchmarks, the coordinates of all LOR start and endpoints were precalculated such that the overhead of calculating the LOR coordinates was not included in these tests. All benchmarks were repeated 10 times and the mean and standard deviation of the results were calculated and visualized.

Finally, we also measured the time needed for a complete listmode OSEM iteration using 34 subsets as a function of the number of listmode events in the NEMA acquisition. The raw listmode data, including 40 million prompt events, as well as all quantitative corrections needed for reconstruction of the NEMA phantom acquisition are available online at <https://doi.org/10.5281/zenodo.8404015>.

All tests used an image of size (215, 215, 71), an isotropic voxel size of 2.78 mm, and were performed on a workstation including an AMD Ryzen Threadripper PRO 3955WX 16 core 32 thread CPU with 256 GB RAM, and an NVIDIA GeForce RTX 3,090 GPU with 24 GB RAM on Ubuntu 22.04 LTS using CUDA v11.2 and `parallelproj` v1.5.0. Note that `parallelproj`'s projectors also support non-isotropic voxel sizes. The scanner geometry, as well as the list mode OSEM algorithm, was implemented in a minimal proof-of-concept Python package available at <https://github.com/gschramm/parallelproj-benchmarks>, except for the STIR benchmark, where STIR's normal geometric modelling was

used. For the CPU and hybrid CPU/GPU mode, Python's `ctypes` module is used to project `numpy` arrays stored in CPU (host) memory using a minimal interface to the low-level projection functions defined in `libparallelproj_c` and `libparallelproj_cuda`. In GPU mode, the CUDA projection kernels were just-in-time compiled and directly executed on `cupy` GPU arrays. Due to the interoperability between `numpy` and `cupy` the same high-level listmode OSEM implementation could be used for both modes. Note that in the latter, all operations needed for the OSEM update were executed directly on the `cupy` GPU arrays, eliminating any memory transfer between the host and the GPU during OSEM updates. In all listmode OSEM reconstructions, a shift-invariant image-based resolution model was used, including a 3D isotropic Gaussian kernel of 4.5 mm FWHM.

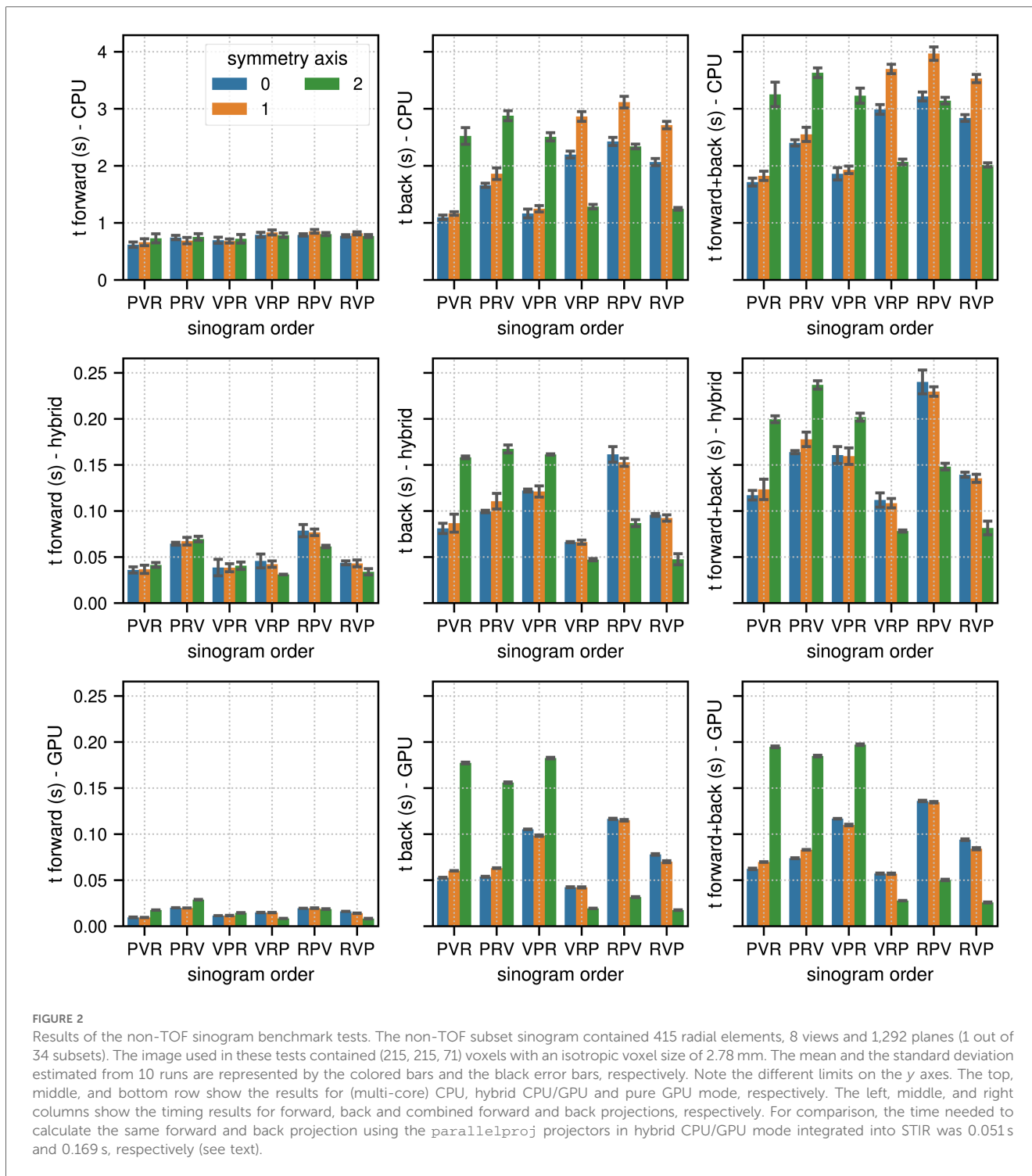
3 Results

Figures 2 and 3 show the results of the sinogram benchmarks in non-TOF and TOF mode, respectively. In non-TOF mode, the best results in terms of the summed time needed for the forward and back projection of one subset sinogram were (compute mode, sinogram order mode, scanner symmetry axis) 1.71 s for (CPU, PVR, 0), 0.078 s for (hybrid CPU / GPU, VRP, 2) and 0.025 s for (GPU, RVP, 2), meaning that the pure GPU mode was approximately 68x faster than the CPU mode and 3.1x faster than the hybrid mode. Benchmarking the timing of the same projections using `parallelproj` integrated into STIR using the hybrid CPU/GPU mode revealed very similar performance - 0.051 s for forward projection and 0.169 s for back projection - as compared to the corresponding results shown in the middle row of Figure 2.

Table 1 shows the timing results for the forward and back projection of a non-TOF sinogram of the Siemens mMR and demonstrates that the timing performance of `parallelproj` in both GPU compute modes is slightly superior to the NiftyPET's GPU projector.

In TOF mode, the corresponding results were 13.88 s for (CPU, PVR, 1), 0.45 s for (hybrid CPU/GPU, RVP, 2), and 0.21 s for (GPU, RVP, 2), which means that the pure GPU mode was approximately 66x faster than the CPU mode and 2.1x faster than the hybrid mode. As expected, especially for the back projections where atomic operations are used, the memory order in the sinogram as well as in the image has a substantial impact on the results. The ratios between the fastest and slowest results for the combined projection times in terms of sinogram order and symmetry axis (non-TOF, TOF mode) were (2.3, 1.4) in the CPU mode, (3.1, 2.0) in the hybrid mode, and (7.6, 3.6) in the GPU mode.

Figures 4 and 5 show the results of the listmode benchmarks in non-TOF and TOF mode, respectively. For 40×10^6 events, the best results in terms of time needed for the forward and back projection (non-TOF, TOF) were (36.3 s, 13.9 s) in CPU mode, (4.19 s, 0.94 s) in hybrid mode and (3.91 s, 0.56 s) in GPU mode. For 1.25×10^6 events, the best results in terms of time needed



for the forward and back projection (non-TOF, TOF) were (1.13 s, 0.45 s) in CPU mode, (0.13 s, 0.035 s) in hybrid mode, and (0.1 s, 0.017 s) in GPU mode. In non-TOF mode, the pure GPU mode was approximately 9.3× faster than the CPU mode and 1.3× faster than the hybrid mode. In TOF mode, the pure GPU mode was approximately 24.8× faster than the CPU mode and 1.7× faster than the hybrid mode. In contrast to the sinogram benchmark results, the impact of the scanner symmetry axis direction was small. In the CPU and GPU mode, the

increase in projection time as a function of the number of list-mode events was almost perfectly linear. In hybrid mode at low number of events, the scaling was non linear due to the overhead caused by the time needed for memory transfer.

Figure 6 shows the results for the timing of a complete TOF listmode OSEM iteration, including 34 subset updates, as well as a reconstruction of the NEMA image quality phantom data set using 40×10^6 total prompt events. The best results for (40×10^6 , 1.25×10^6) events were (23.82 s, 4.47 s) in CPU mode,

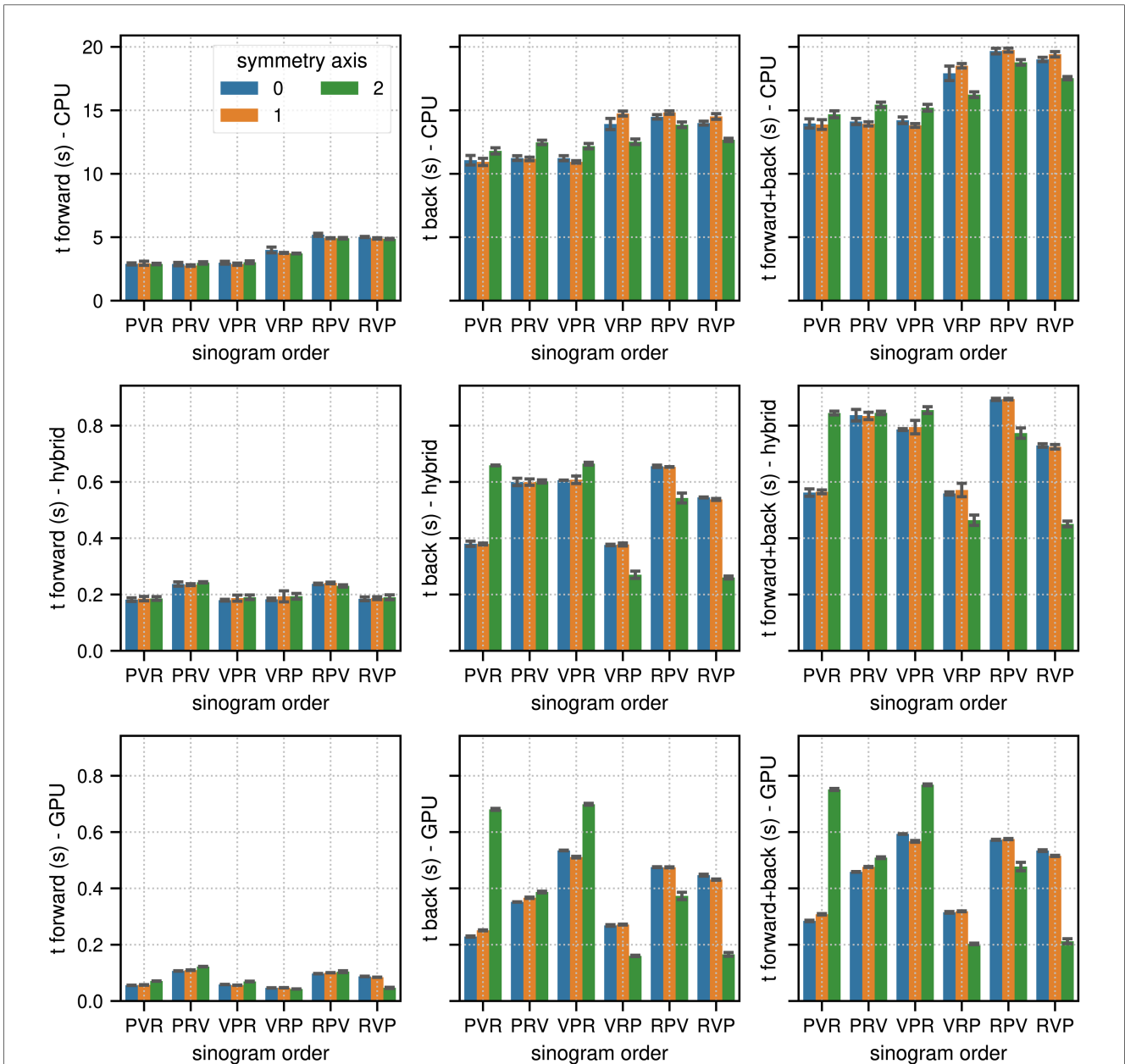


FIGURE 3 Same as Figure 2 for the results of the TOF sinogram benchmark tests. The TOF subset sinogram contained 415 radial elements, 8 views, 1,292 planes and 29 TOF bins with a width of 169 ps. The modeled TOF resolution was 375 ps.

TABLE 1 Timing results for the forward and back projection of a non-TOF sinogram of the Siemens mMR using parallelproj’s GPU and hybrid CPU/GPU mode, as well as NiftyPET’s projectors. See text for details.

projector	<i>t</i> forward (s)	<i>t</i> back (s)
parallelproj GPU mode	0.21	0.43
parallelproj hybrid CPU/GPU mode	0.62	0.95
NiftyPET	0.68	1.56

(9.17 s, 3.97 s) in hybrid mode, and (0.60 s, 0.057 s) in GPU mode, which means that for 40×10^6 events the pure GPU mode was approximately 40x faster than the CPU mode and 15x faster than the hybrid mode.

4 Discussion

All results shown in our article demonstrate once more that parallel computation of forward and back projections using a state-of-the-art GPU is substantially faster compared to parallelization using OpenMP on a state-of-the-art multicore CPU system. Certainly, the achievable GPU acceleration factor strongly depends on the computational problem itself (e.g., sinogram or listmode reconstruction) and the problem size. In our non-TOF and TOF sinogram and listmode benchmark tests, we observed GPU acceleration factors between 25 and 68.

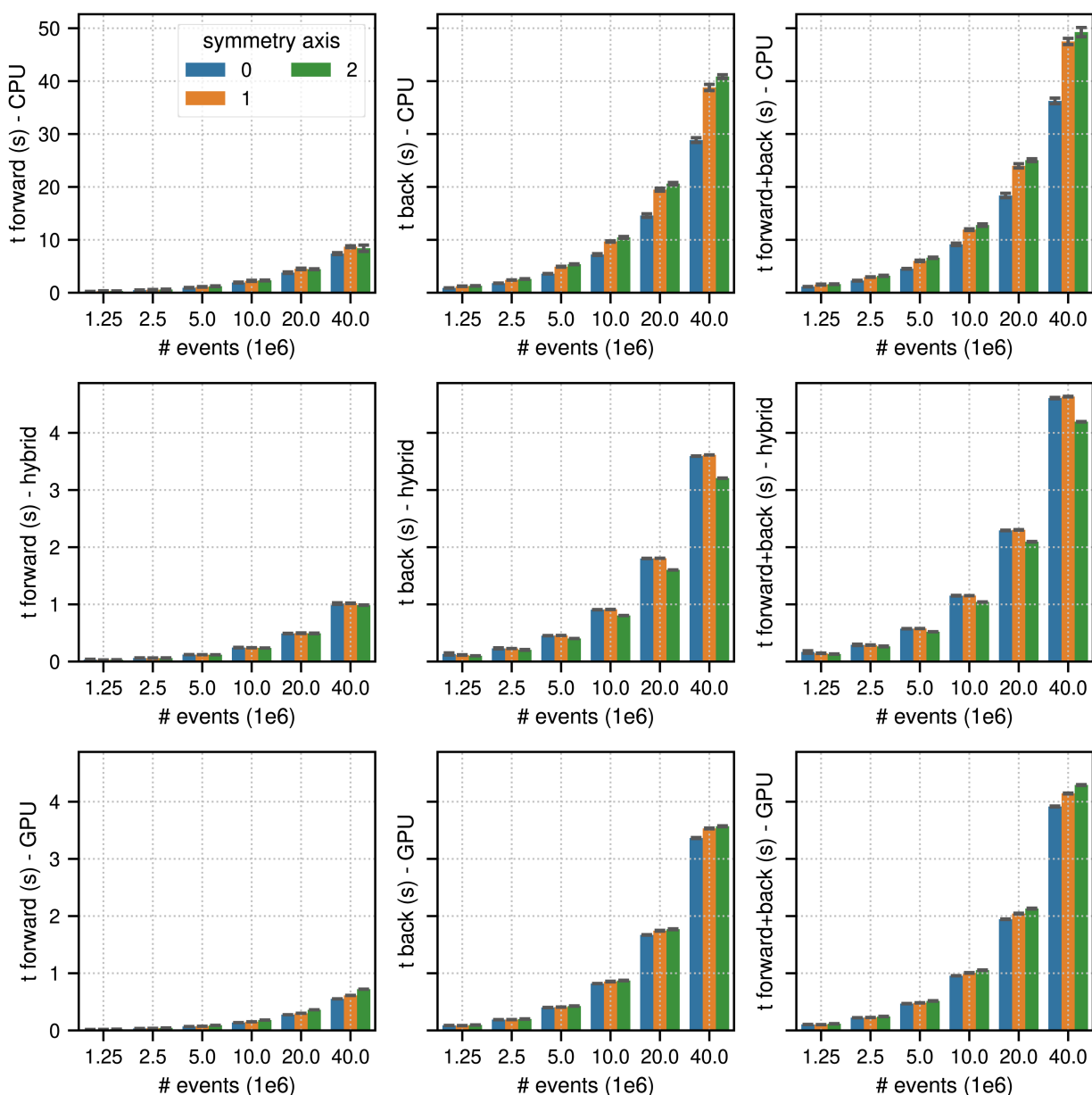


FIGURE 4

Same as Figure 2 for results of the non-TOF listmode benchmark tests for different number of listmode events. Note the different limits on the y axes and that the x-axis scale is non-linear.

One important aspect that emerged from our sinogram benchmark tests - where the projection data and memory access is ordered - is the fact that the projection times varied substantially when using different memory layouts (up to a factor of 7.6 in the GPU mode). This can be understood by taking into account that the amount of race conditions that are created during the back projection within a thread block heavily depends on the order and possible intersections of rays to be projected within that block. Note that in pure GPU mode, the time needed for sinogram forward projections also varied substantially across the different memory layouts, which is probably due to the way image memory is accessed and cached on CUDA GPUs.

Another interesting observation is the fact that in all compute modes the time needed to calculate TOF sinogram projections was

much longer than the times needed to calculate non-TOF sinogram projections, whereas the situation was reversed in listmode. For TOF sinogram projections, more floating point operations have to be computed compared to non-TOF sinogram projections due to the evaluations of the TOF kernels between the contributing voxels and a number of TOF bins. In listmode, however, the computational work needed to project a TOF event is much lower compared to projecting a non-TOF event. This is the case because a TOF listmode event detected in a specific TOF bin is only affected by a few voxels along the complete LOR in the image, where the number of affected voxels is inversely proportional to the TOF resolution of the scanner. That in turn means that with scanner TOF resolutions becoming better and better, the gap

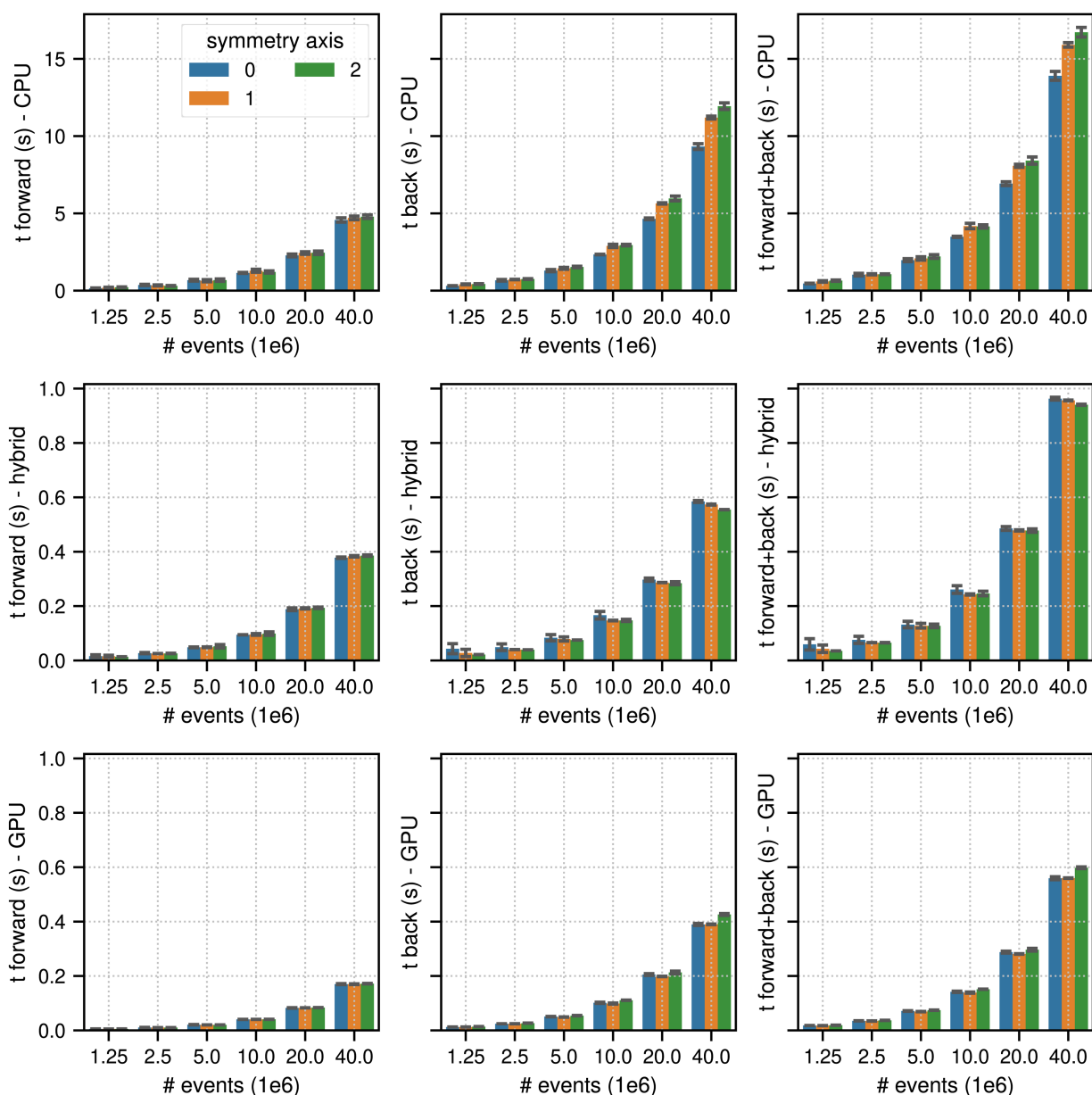


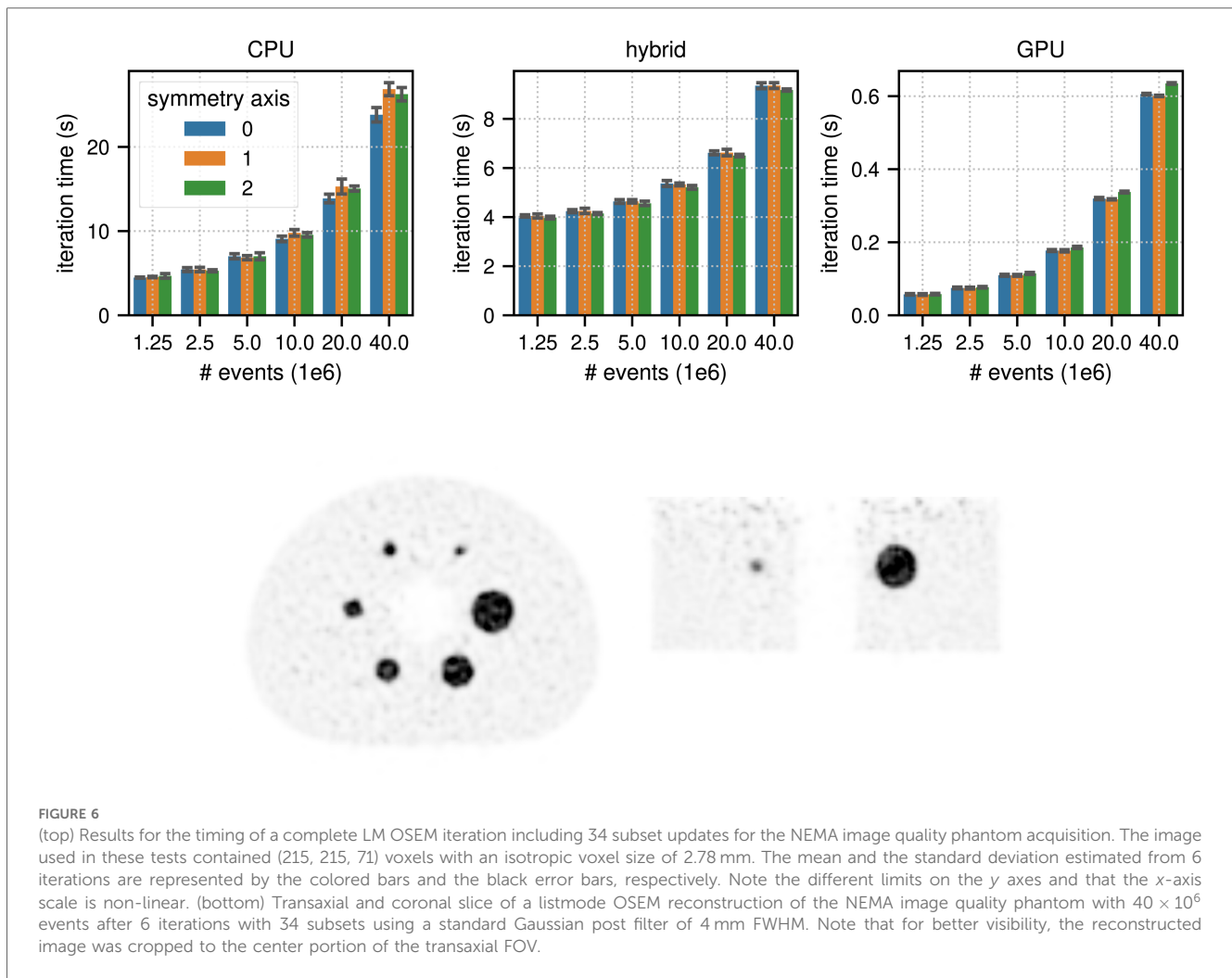
FIGURE 5
Same as Figure 2 for results of the TOF listmode benchmark tests for different number of listmode events. Note the different limits on the y axes and that the x-axis scale is non-linear.

between the TOF projection times in sinogram and listmode will become bigger and bigger, strongly favoring listmode processing. According to our experience, projection times in listmode are already much faster for most standard clinical acquisitions (except for very long static brain scans with high affinity tracers) on current PET systems with TOF resolutions between 250–400 ps.⁴ Extrapolating the timing results of one complete OSEM listmode

iteration of an acquisition with 40×10^6 counts in Figure 6, clinical listmode OSEM reconstructions of a single bed position of a standard static FDG whole-body acquisition using PET scanner with 20–25 cm axial FOV seem to be possible in a couple of seconds and could even be faster than the acquisition time.⁵

⁴An alternative way to further accelerate sinogram-based reconstructions is the use of dedicated sinogram rebinning techniques.

⁵This is obviously only true if all other necessary corrections, such as scatter estimation, can be performed very quickly as well. For PET scanners with a very long axial field of view (much higher sensitivity), the reconstruction times could be substantially longer.



A somewhat unexpected result was the fact that the gap in the TOF projection times between hybrid CPU/GPU and pure GPU mode was much bigger when timing the execution of a complete listmode OSEM iteration compared to the pure projection benchmark test when reconstruction 40×10^6 counts (approximately a factor of 15 vs. a factor of 1.7, respectively). After detailed profiling of a listmode OSEM iteration in hybrid mode, it became obvious that the total time spent for the 34 subset listmode forward and back projections (ca. 1.2 s) was short compared to the time needed to calculate all other operations necessary for the OSEM update. Profiling revealed that calculating all 68 Gaussian convolutions needed for image-based resolution modeling - performed on the CPU in hybrid compute mode - took approximately 2.3 s. An interesting lesson to be learned is that once very fast GPU-based projectors are used, it should always be double-checked whether other computational steps of any algorithm become new bottlenecks.

A natural prerequisite for running sinogram OSEM reconstruction is the availability of enough GPU memory to store the complete image volume, the emission sinogram, the

forward projection and the contamination sinogram. For TOF PET scanners with an 25 cm axial FOV and 400 ps TOF resolution, this means that ca. 40–50 GB of GPU memory is required, which is available on state-of-the-art server GPUs but can be challenging for consumer GPUs. Moreover, these memory requirements increase even further for PET systems with longer axial FOV and better TOF resolution. Note, however, that for systems with state-of-the-art TOF resolution, the memory requirements can be severely reduced when running OSEM in listmode. Moreover, the hybrid CPU/GPU mode of `parallelproj` allows “chunk-wise” calculations of projections and supports the use of multiple GPUs to be able to reconstruct sinogram data from long axial FOV PET systems.

An important limitation of our study is the fact that we only implemented and benchmarked Joseph’s projection method. Compared to other methods such as the distance-driven method, multiray models, or tube-of-response models, Joseph’s method is inherently faster. Consequently, projection times are expected to be somewhat slower for more advanced projectors, but a detailed investigation of more advanced projectors is beyond the scope of

this work and left for future research.⁶ Note, however, that according to our experience, combining Joseph's method with an image-based and/or sinogram-based resolution model can produce high-quality PET reconstructions.

Without a doubt, it is possible to further optimize the implementation of the `parallelproj` projectors, especially the CUDA implementation. As an example, we have decided not to use CUDA's texture memory, which could substantially accelerate the image interpolations needed in the Joseph forward projections, or be used to interpolate TOF kernel values based on a 1D lookup table which would also allow the use of non-Gaussian TOF kernels (42). The main reason for not using texture memory is the fact that it would only accelerate the forward projections since writing into texture memory is not possible and because reconstruction times are usually dominated by the back projections. Another way to further improve the listmode projection times is to pre-sort the listmode events to minimize race conditions during back projection, as e.g., shown in (10, 43).

The design of `parallelproj` allows it to be integrated into other reconstruction platforms, as illustrated here for STIR. However, for optimal performance, a re-design of the reconstruction platform might be required, as noted in Section 2.5. As shown in this paper, avoiding the overhead of copying data between CPU and GPU memory can have a substantial impact. In C++, this could be avoided by using CUDA managed pointers, for instance via the CuVec library.⁷ However, best performance requires implementing most operations such as numerical algebra and filtering directly in CUDA, as illustrated in this paper.

It is noteworthy that the current implementation of `parallelproj`'s Joseph projectors using arrays of LOR start and end coordinates is optimized towards (arbitrary) PET geometries. To calculate projections for reconstructing CT data acquired with a single moving source and a moving detector panel, more efficient implementations exploiting the known geometry between source and detector panel are possible.

Last but not least, it is worth highlighting that the python interface of `parallelproj` is compatible with the Python array API standard, enabling efficient projections and back projections of various compatible array classes (e.g., `numpy` CPU arrays, `cupy` GPU arrays, `pytorch` CPU and GPU tensors). This allows for a seamless integration of `parallelproj` into deep learning frameworks such as `pytorch` (25) for the development of neural networks including forward and back projection layers such as unrolled variational networks (44, 45).

⁶Since `parallelproj` is an open-source project, contributions of or discussions on more advanced projectors from the reconstruction community are more than welcome.

⁷<https://amypad.github.io/CuVec>.

5 Conclusion

`parallelproj` is an open-source and easy accessible research framework for efficient calculation of non-TOF and TOF projections in sinogram or listmode on multiple CPUs or state-of-the-art CUDA GPUs. Conventional and advanced research reconstructions (including deep learning) can be substantially accelerated by using the hybrid and pure GPU compute modes of this framework.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: <https://zenodo.org/records/8404015>.

Author contributions

GS: Conceptualization, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing; KT: Methodology, Software, Writing – original draft, Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article.

This work was supported by UK Research and Innovation [EP/T026693/1].

Acknowledgments

The integration of `parallelproj` into STIR and SIRF was funded by the UKRI EPSRC grant to the Collaborative Computational Platform in Synergistic Reconstruction for Biomedical Imaging (EP/T026693/1). The authors would like to thank Koen Michielsen for discussions on the Python array API standard.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

- Eklund A, Dufort P, Forsberg D, LaConte SM. Medical image processing on the GPU – past, present, future. *Med Image Anal.* (2013) 17(8):1073–94. doi: 10.1016/j.media.2013.05.008
- Zhang X, Zhou J, Cherry SR, Badawi RD, Qi J. Quantitative image reconstruction for total-body pet imaging using the 2-meter long explorer scanner. *Phys Med Biol.* (2017) 62(6):2465. doi: 10.1088/1361-6560/aa5e46
- Lamare F, Bousse A, Thielemans K, Liu C, Merlin T, Fayad H, et al. Pet respiratory motion correction: quo vadis? *Phys Med Biol.* (2022) 67(3):03TR02. doi: 10.1088/1361-6560/ac43fc
- Wang G, Ye JC, Mueller K, Fessler JA. Image reconstruction is a new frontier of machine learning. *IEEE Trans Med Imaging.* (2018) 37(2):1–14. doi: 10.1109/TMI.2018.2833635
- Wang G, Ye JC, De Man B. Deep learning for tomographic image reconstruction. *Nat Mach Intell.* (2020) 2(12):737–48. doi: 10.1038/s42256-020-00273-z
- Reader AJ, Corda G, Mehranian A, Costa-Luis C, Ellis S, Schnabel JA. Deep learning for pet image reconstruction. *IEEE Trans Radiat Plasma Med Sci.* (2021) 5(1):1–25. doi: 10.1109/TRPMS.2020.3014786
- Praxt G, Chinn G, Habte F, Olcott P, Levin C. Fully 3-d list-mode OSEM accelerated by graphics processing units. In: 2006 *IEEE Nuclear Science Symposium Conference Record*, vol. 4 (2006). p. 2196–202. doi: 10.1109/NSSMIC.2006.354350.
- Praxt G, Chinn G, Olcott PD, Levin CS. Fast, accurate, shift-varying line projections for iterative reconstruction using the GPU. *IEEE Trans Med Imaging.* (2009) 28(3):435–45. doi: 10.1109/TMI.2008.2006518
- Barker WC, Thada S, Dieckmann W. A GPU-accelerated implementation of the MOLAR PET reconstruction package. In: 2009 *IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*. (2009). p. 4114–9. doi: 10.1109/NSSMIC.2009.5402353.
- Cui J-y, Praxt G, Pevrhal S, Levin CS. Fully 3d list-mode time-of-flight PET image reconstruction on GPUs using CUDA. *Med Phys.* (2011) 38(12):6775–86. doi: 10.1118/1.3661998
- Herraz JL, España S, Cal-González J, Vaquero JJ, Desco M, Udías JM. Fully 3d GPU PET reconstruction. *Nucl Instrum Methods Phys Res A.* (2011) 648:S169–71. doi: 10.1016/j.nima.2010.12.043
- Herraz JL, España S, Cabido R, Montemayor AS, Desco M, Vaquero JJ, et al. GPU-based fast iterative reconstruction of fully 3-d PET sinograms. *IEEE Trans Nucl Sci.* (2011) 58(5):2257–63. doi: 10.1109/TNS.2011.2158113
- Kim KS, Ye JC. Fully 3d iterative scatter-corrected OSEM for HRRT PET using a GPU. *Phys Med Biol.* (2011-07) 56(15):4991. doi: 10.1088/0031-9155/56/15/021
- Zhou J, Qi J. Fast, efficient fully 3d PET image reconstruction using sparse system matrix factorization with GPU acceleration. *Phys Med Biol.* (2011) 56(20):6739. doi: 10.1088/0031-9155/56/20/015
- Chou C-Y, Dong Y, Hung Y, Kao Y-J, Wang W, Kao C-M, et al. Accelerating image reconstruction in dual-head PET system by GPU, symmetry properties. *PLoS One.* (2012) 7(12):e50540. doi: 10.1371/journal.pone.0050540
- Kinouchi S, Yamaya T, Yoshida E, Tashima H, Kudo H, Haneishi H, et al. GPU-based PET image reconstruction using an accurate geometrical system model. *IEEE Trans Nucl Sci.* (2012) 59(5):1977–83. doi: 10.1109/TNS.2012.2201953
- Cui J, Praxt G, Meng B, Levin CS. Distributed MLEM: an iterative tomographic image reconstruction algorithm for distributed memory architectures. *IEEE Trans Med Imaging.* (2013) 32(5):957–67. doi: 10.1109/TMI.2013.2252913
- Ha S, Matej S, Ispiryan M, Mueller K. GPU-accelerated forward and back-projections with spatially varying kernels for 3d DIRECT TOF PET reconstruction. *IEEE Trans Nucl Sci.* (2013) 60(1):166–73. doi: 10.1109/TNS.2012.2233754
- Markiewicz PJ, Thielemans K, Ehrhardt MJ, Jiao J, Burgos N, Atkinson D, et al. High throughput CUDA implementation of accurate geometric modelling for iterative reconstruction of PET data. In: 2014 *IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)* (2014). p. 1–4. DOI: 10.1109/NSSMIC.2014.7430963.
- Zhou J, Qi J. Efficient fully 3d list-mode TOF PET image reconstruction using a factorized system matrix with an image domain resolution model. *Phys Med Biol.* (2014) 59(3):541. doi: 10.1088/0031-9155/59/3/541
- Nassiri MA, Carrier J-F, Després P. Fast GPU-based computation of spatial multigrid multiframe LMEM for PET. *Med Biol Eng Comput.* (2015) 53(9):791–803. doi: 10.1007/s11517-015-1284-9
- Zeng T, Gao J, Gao D, Kuang Z, Sang Z, Wang X, et al. A GPU-accelerated fully 3d OSEM image reconstruction for a high-resolution small animal PET scanner using dual-ended readout detectors. *Phys Med Biol.* (2020) 65(24):245007. doi: 10.1088/1361-6560/aba6f9
- Depres P, Jia X. A review of GPU-based medical image reconstruction | Elsevier enhanced reader. *Eur J Med Phys.* (2017) 42:76–92. doi: 10.1016/j.ejmp.2017.07.024
- Okuta R, Unno Y, Nishino D, Hido S, Loomis C. Cupy: a numpy-compatible library for nvidia gpu calculations. In: *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)* (2017).
- Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: an imperative style, high-performance deep learning library. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors. *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc. (2019).
- Siddon RL. Fast calculation of the exact radiological path for a three-dimensional CT array. *Med Phys.* (1985) 12(2):252–5. doi: 10.1118/1.595715
- Wu X. An efficient antialiasing technique. *ACM SIGGRAPH Comput Graph.* (1991) 25(4):143–52. doi: 10.1145/127719.122734
- De Man B, Basu S. Distance-driven projection and backprojection in three dimensions. *Phys Med Biol.* (2004) 49(11):2463–75. doi: 10.1109/NSSMIC.2002.1239600
- Joseph PM. An improved algorithm for reprojecting rays through pixel images. *IEEE Trans Med Imaging.* (1982) 1(3):192–6. doi: 10.1109/TMI.1982.4307572
- Rahmim A, Cheng J-C, Blinder S, Cambarde M-L, Sossi V. Statistical dynamic image reconstruction in state-of-the-art high-resolution pet. *Phys Med Biol.* (2005) 50(20):4887. doi: 10.1088/0031-9155/50/20/010
- Dagum L, Menon R. Openmp: an industry standard api for shared-memory programming. *IEEE Comput Sci Eng.* (1998) 5(1):46–55. doi: 10.1109/99.660313
- Thielemans K, Tsoumpas C, Mustafovic S, Beisel T, Aguiar P, Dikaios N, et al. STIR: software for tomographic image reconstruction release 2. *Phys Med Biol.* (2012) 57(4):867–83. doi: 10.1088/0031-9155/57/4/867
- Martí-Fuster B, Falcon C, Tsoumpas C, Livieratos L, Aguiar P, Cot A, et al. Integration of advanced 3d SPECT modeling into the open-source STIR framework. *Med Phys.* (2013) 40(9):092502. doi: 10.1118/1.4816676
- Ovtchinnikov E, Brown R, Kolbitsch C, Pasca E, da Costa-Luis C, Gillman AG, et al. SIRF: synergistic image reconstruction framework. *Comput Phys Commun.* (2020) 249:107087. doi: 10.1016/j.cpc.2019.107087
- Brown R, Kolbitsch C, Delplancke C, Papoutsellis E, Mayer J, Ovtchinnikov E, et al. Motion estimation and correction for simultaneous PET/MR using SIRF and CIL. *Philos Trans R Soc Lond A.* (2021) 379(2204):20200208. doi: 10.1098/rsta.2020.0208
- Hsu DF, Ilan E, Peterson WT, Uribe J, Lubberink M, Levin CS. Studies of a next-generation silicon-photomultiplier-based time-of-flight PET/CT system. *J Nucl Med.* (2017) 58(9):1511–8. doi: 10.2967/jnumed.117.189514
- Dempster PA, Laird NM, Rubin DB, Lange K, Carson R, Shepp LA, et al. EM reconstruction algorithms for emission and transmission tomography. *J Comput Assist Tomogr.* (1984) 8(2):306–16. doi: 10.1109/TMI.1982.4307558
- Lange K, Carson R. EM reconstruction algorithms for emission and transmission tomography. *J Comput Assist Tomogr.* (1984) 8(2):306–16. PMID: 6608535.
- Hudson HM, Larkin RS. Ordered subsets of projection data. *IEEE Trans Med Imaging.* (1994) 13(4):601–9. doi: 10.1109/42.363108
- Markiewicz PJ, Ehrhardt MJ, Erlandsson K, Noonan PJ, Barnes A, Schott JM, et al. NiftyPET: a high-throughput software platform for high quantitative accuracy and precision PET imaging and analysis. *Neuroinformatics.* (2018) 16(1):95–115. doi: 10.1007/s12021-017-9352-y
- Delso G, Furst S, Jakoby B, Ladebeck R, Ganter C, Nekolla SG, et al. Performance measurements of the Siemens mMR integrated whole-body PET/MR scanner. *J Nucl Med.* (2011) 52(12):1914–22. doi: 10.2967/jnumed.111.092726
- Efthimiou N, Thielemans K, Emond E, Cawthorne C, Archibald SJ, Tsoumpas C. Use of non-Gaussian time-of-flight kernels for image reconstruction of Monte Carlo simulated data of ultra-fast PET scanners. *EJNMMI Phys.* (2020) 7(1):42. doi: 10.1186/s40658-020-00309-8
- Efthimiou N, Karp JS, Surti S. Data-driven, energy-based method for estimation of scattered events in positron emission tomography. *Phys Med Biol.* (2022) 67(9):095010. doi: 10.1088/1361-6560/ac62fc
- Mehranian A, Reader AJ. Model-based deep learning PET image reconstruction using forward-backward splitting expectation maximisation. *IEEE Trans Radiat Plasma Med Sci.* (2021) 5(1). doi: 10.1109/TRPMS.2020.3004408
- Adler J, Öktem O. Learned primal-dual reconstruction. *IEEE Trans Med Imaging.* (2018) 37(6):1322–32. doi: 10.1109/TMI.2018.2799231