JOURNAL OF SYNCHROTRON RADIATION
ISSN 1600-5775

tomoCAM: fast model-based iterative reconstruction via GPU acceleration and non-uniform fast Fourier transforms

Dinesh Kumar,[a,b]* Dilworth Y. Parkinson[b,c] and Jeffrey J. Donatelli[a,b]

[a]Mathematics Department, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, [b]Center for Advanced Mathematics for Energy Research Applications, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, and [c]Advanced Light Source, Lawrence Berkeley National Laboratory, Berkeley, CA, USA. *Correspondence e-mail: dkumar@lbl.gov

Received 31 March 2023
Accepted 12 October 2023

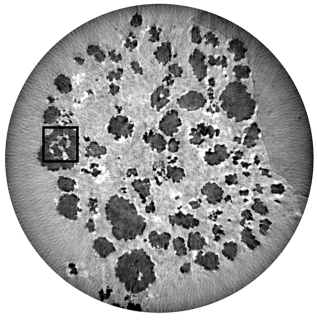
Edited by A. Momose, Tohoku University, Japan

Keywords: X-ray tomography; micro-CT; synchrotron tomography; GPU; MBIR; nano-CT; tomographic reconstruction.

X-ray-based computed tomography is a well established technique for determining the three-dimensional structure of an object from its two-dimensional projections. In the past few decades, there have been significant advancements in the brightness and detector technology of tomography instruments at synchrotron sources. These advancements have led to the emergence of new observations and discoveries, with improved capabilities such as faster frame rates, larger fields of view, higher resolution and higher dimensionality. These advancements have enabled the material science community to expand the scope of tomographic measurements towards increasingly *in situ* and *in operando* measurements. In these new experiments, samples can be rapidly evolving, have complex geometries and restrictions on the field of view, limiting the number of projections that can be collected. In such cases, standard filtered back-projection often results in poor quality reconstructions. Iterative reconstruction algorithms, such as model-based iterative reconstructions (MBIR), have demonstrated considerable success in producing high-quality reconstructions under such restrictions, but typically require high-performance computing resources with hundreds of compute nodes to solve the problem in a reasonable time. Here, *tomoCAM*, is introduced, a new GPU-accelerated implementation of model-based iterative reconstruction that leverages non-uniform fast Fourier transforms to efficiently compute Radon and back-projection operators and asynchronous memory transfers to maximize the throughput to the GPU memory. The resulting code is significantly faster than traditional MBIR codes and delivers the reconstructive improvement offered by MBIR with affordable computing time and resources. *tomoCAM* has a Python front-end, allowing access from *Jupyter*-based frameworks, providing straightforward integration into existing workflows at synchrotron facilities.

1. Introduction

Micro- and nano-tomography using synchrotron technology is crucial in uncovering the inner makeup of modern materials, particularly in dynamic settings. Its diverse applications include: the investigation of the fractures and deterioration of ceramic matrix composites, which are novel lightweight materials used in jet engines that operate under high temperatures and pressure (Forna-Kreutzer *et al.*, 2021); the study of the flow of oil, brine and carbon dioxide through rocks (Walsh *et al.*, 2014); and the analysis of dendrite formation in batteries, which causes capacity reduction and eventual failure (Dienemann *et al.*, 2023). Many synchrotron micro-computed tomography (micro-CT) facilities now have cameras that can acquire many-megapixel images at thou-



sands of frames per second (MacDowell *et al.*, 2012; Nikitin *et al.*, 2022; Ge *et al.*, 2018; Thuering *et al.*, 2011; Mokso *et al.*, 2017). These advances in instrumentation have encouraged users to push the boundaries of what can be imaged at synchrotron beamlines.

An increasing number of investigators are conducting *in situ* (Larson & Zok, 2018; French *et al.*, 2022) and *in operando* (Kulkarni *et al.*, 2020; Dienemann *et al.*, 2023) measurements. Typically the initial technique attempted for micro-CT reconstructions is filtered back-projection (FBP)[1], which is available in *tomopy* (Gürsoy *et al.*, 2014), *tomocupy* (Nikitin, 2023), *ASTRA* (van Aarle *et al.*, 2015, 2016) and *TIGRE* (Biguri *et al.*, 2016, 2020). However, in the case of many dynamic experiments, where the specimen under observation is changing rapidly, it is generally not possible to capture sufficient projections to satisfy angular Shannon sampling conditions (Crowther *et al.*, 1970) and overcome the noise. FBP is not a suitable option in such situations. The reconstructions obtained through this method tend to have excessive noise levels and exhibit streaking artifacts, making it difficult or even impossible to carry out further analysis.

As an alternative, iterative methods, such as the simultaneous iterative reconstruction technique (SIRT) (Tarantola & Valette, 1982) and model-based iterative reconstruction (MBIR) (Venkatakrishnan *et al.*, 2013; Mohan *et al.*, 2014), aim to mitigate these shortcomings. In recent years, significant efforts have been made to develop reconstruction packages that provide access to iterative reconstruction algorithms with total variation constraints. This is primarily driven by the increasing demand for high-quality images in various imaging modalities, such as computed tomography (CT), magnetic resonance imaging and optical microscopy. Some of the prominent efforts include *SVMBIR*, *TIGRE*, *ASTRA*, *ToMoBAR* (Kazantsev & Wadeson, 2020) and *Core Imaging Library (CIL)* (Jørgensen *et al.*, 2021). *SVMBIR* is a multi-threaded implementation of MBIR with a Python front-end (SVMBIR, 2020). *ASTRA* and *TIGRE* provide an array of CPU- and GPU-based algorithms for direct inversion and iterative algorithms with total variation constraints. *ToMoBAR* and *CIL* focus primarily on regularized iterative methods for datasets with sparse projection data.

These iterative algorithms formulate the reconstruction as an optimization problem. The solution is obtained through an iterative process that aims to minimize the mismatch between the measured data and a forward model (Radon transform) of a digital representation of the sample. This iterative approach enables the incorporation of prior knowledge, such as total variation constraints, into the optimization process, as demonstrated in various studies (Trampert & Leveque, 1990; Zhang *et al.*, 2014; Venkatakrishnan *et al.*, 2013; Mohan *et al.*, 2014). However, current CPU-based implementations of MBIR typically require a large compute cluster to achieve turnaround times that are comparable with data collection times. This not only adds extra time to the experiment-to-analysis loop but also places an additional burden on material scientists, who must acquire a new set of expertise in using a compute cluster. This paper introduces *tomoCAM*, a GPU-accelerated implementation of MBIR that is based on the non-uniform fast Fourier transforms (NUFFT) approach (Greengard & Lee, 2004; Fessler & Sutton, 2003; Dutt & Rokhlin, 1993), which significantly reduces compute time complexity while maintaining accuracy. With the computational power provided by modern GPU devices and the relatively affordable cost of computer memory, it has become possible to perform these reconstructions on a single machine within a reasonable amount oftime.

To design our GPU-accelerated algorithm and implementation, we build Radon and back-projection operators based on NUFFT. We also leverage highly optimized cuFFT libraries that are native to the CUDA software development kit (Vingelmann & Fitzek, 2020). We follow the mathematical outline laid out by Venkatakrishnan *et al.* (2013) and Mohan *et al.* (2014) to add a total variation constraint, which helps in reducing noise while preserving the sharp edges. An important feature of our implementation is the flexibility to introduce a different constraint. The choice of the constraint is not limited by the algorithm design.

We test our computational framework through a series of numerical experiments on known phantoms and experimental data made publicly available through Tomobank (Carlo *et al.*, 2018). We compare the reconstructions with those obtained from FBP (Gürsoy *et al.*, 2014) and *SVMBIR* (SVMBIR, 2020), a publicly available CPU-based package.

In our numerical experiments, we found that: (i) when compared with FBP, the reconstruction quality produced by *tomoCAM* was superior with less noise and required a lower number of projections, and (ii) additionally, we observed that *tomoCAM* was around 15 times faster on a single machine than *SVMBIR* and 22 times faster than *TIGRE* (IRN-TV-CGLS).

Finally, we provide a Python front-end that exposes *tomoCAM* functionality to the widely used *NumPy* package (Harris *et al.*, 2020), using *pybind11* (Jakob *et al.*, 2017). This makes it easy to integrate *tomoCAM* into existing workflows. The code is freely available at https://github.com/lbl-camera/tomocam.

2. Radon transform and NUFFT

This section provides a brief overview of the fundamental concepts related to tomography, including the Radon transform (as well as its adjoint) and its connection to the Fourier transform. To perform tomographic measurements, a series of images, referred to as projections, are captured at various angles by rotating either the camera or the sample under observation.

[1] Although some authors prefer to distinguish filtered back-projection computed in real space from gridrec, it is common for gridrec to be considered a specific implementation of filtered back-projection in Fourier space (*e.g.* Barutcu *et al.*, 2021), as they are both based on approximating the inverse Radon transform through integration of filtered projection data over the projection angle (Kak & Slaney, 2001). In this work, we take the latter convention.

2.1. Radon transform

The Radon transform is fundamental to any tomographic reconstruction. It transforms a function $f(\mathbf{x}, z)$, $\mathbf{x} \in \mathbb{R}^2$, $z \in \mathbb{R}$, to $Rf(t, \hat{\boldsymbol{n}}, z)$, $t \in \mathbb{R}$, $\hat{\boldsymbol{n}} \in \mathbb{S}^1$ through a line integral (2), see Fig. 1. Given a set of oriented lines $\ell_{t,\hat{\boldsymbol{n}}}$ defined as

$$\ell_{t,\hat{\boldsymbol{n}}} = \left\{ \mathbf{x} : \langle \mathbf{x}, \hat{\boldsymbol{n}} \rangle = t \right\} = \left\{ t\hat{\boldsymbol{n}} + s\hat{\boldsymbol{n}}_\perp : s \in \mathbb{R} \right\}, \tag{1}$$

where $\hat{\boldsymbol{n}}_\perp$ is the direction of the X-ray beam, $\hat{\boldsymbol{n}}$ is perpendicular to the beam in same plane as ℓ, and t is the distance to ℓ from the origin. The Radon transform Rf of function f is defined as

$$Rf(t, \hat{\boldsymbol{n}}, z) = \int_{\ell_{t,\hat{\boldsymbol{n}}}} f(\mathbf{x}, z) = \int_{-\infty}^{\infty} f(t\hat{\boldsymbol{n}} + s\hat{\boldsymbol{n}}_\perp, z)\, \mathrm{d}s. \tag{2}$$

Tomographic measurements can be accurately modeled as the Radon transform of the sample density represented by f. It is the inversion of equation (2) that reconstructs f from the data, and is of primary importance in tomographic reconstruction. By the central slice theorem, the Fourier transform of Rf in direction $\hat{\boldsymbol{n}}$ is equivalent to the slice of the Fourier transform of f along $\hat{\boldsymbol{n}}$, *i.e.*

$$\begin{aligned} \mathbb{F}_1[Rf](k, \hat{\boldsymbol{n}}, z) :&= \int_{-\infty}^{\infty} \exp\left(-2\pi i k t\right) Rf(t, \hat{\boldsymbol{n}}, z)\, \mathrm{d}t \\ &= \left[\mathbb{F}_2 f\right](k\hat{\boldsymbol{n}}, z), \end{aligned} \tag{3}$$

where z is the dimension along the axis of rotation and $\mathbb{F}_d$ denotes the d-dimensional Fourier transform. The Radon transform and its adjoint are two-dimensional operators that are applied slice-by-slice on three-dimensional data. For simplification, we will drop the z dependency from the subsequent notations. Assuming f and $\mathbb{F}_1 f$ are integrable everywhere, the inverse of the Radon transform (2) is given by

$$\begin{aligned} f(\mathbf{x}) = &\int_0^{\pi} \int_{-\infty}^{\infty} \exp\left(2\pi i k \langle \mathbf{x}, \hat{\boldsymbol{n}}(\theta) \rangle\right) \\ &\times \int_{-\infty}^{\infty} \exp\left(-2\pi i k t\right) y\left[t, \hat{\boldsymbol{n}}(\theta)\right] \mathrm{d}t\, |\, k\, |\, \mathrm{d}k\, \mathrm{d}\theta, \end{aligned} \tag{4}$$

where we denote the θ dependency as $\hat{\boldsymbol{n}}(\theta) = (\cos\theta, \sin\theta)$.

It is computationally very expensive to exactly compute equation (4). In practice, (4) is efficiently approximated with a NUFFT (Gürsoy *et al.*, 2014) or directly estimated in real space through the use of various filters such as Shepp–Logan (Shepp & Logan, 1974), Ram–Lak (Ramachandran & Lakshminarayanan, 1971) and Butterworth (Butterworth, 1930), which approximate and weight by the Fourier sampling density, hence the name 'filtered back-projection' (Epstein, 2007; Jørgensen & Lionheart, 2021; Candes, 2021).

These filters are additionally designed to dampen out the higher Fourier frequencies. This is the most commonly used method in the reconstruction of tomographic data, in part because of the sheer speed by which the inversion can be performed. However, in cases when the view is partially blocked, or the specimen is evolving, it may not be possible to collect enough projections to sufficiently sample the Fourier space. In such cases, the FBP results in poor image quality.

2.2. NUFFT

On a discrete uniform grid, if a sufficient number of projections are available, inversion of the Radon transform entails computing Fourier coefficients along radial lines using a one-dimensional fast Fourier transform (FFT), followed by two-dimensional backward Fourier transforms from a non-uniform polar grid $\{\mathbf{k}_j = k_j(\cos\theta_j, \sin\theta_j)\}$ onto a Cartesian grid $\{(x_n, y_n)\}$, which can be represented as the summation

$$f_n = \sum_{j=1}^{M} c_j \exp\left[2\pi i k_j (x_n \cos\theta_j + y_n \sin\theta_j)\right], \quad n \in [1, N], \tag{5}$$

where c_j is the Fourier coefficient at $\mathbf{k}_j$, N is the number of discrete points that represent the sample density f on a uniform Cartesian grid, and M is the number of polar grid points representing the projection data. However, directly computing equation (5) is computationally expensive, as the complexity is $\mathcal{O}(MN)$. Data taken at synchrotron light sources can usually reach up to $M = \mathcal{O}(10^{10})$ pixels, and the final reconstructed image size N has a similar order or magnitude for the final reconstructed image.

NUFFTs offer a precise and efficient method for computing equation (5). This method involves first computing the Fourier coefficients on a polar grid using a sequence of one-dimensional FFTs along radial lines. Then, the computed coefficients are convolved with a compactly supported spreading kernel φ, and this convolution is evaluated on a uniform grid (see Fig. 2). Subsequently, an inverse Fourier transformation is performed on the convolution values on the uniform grid, followed by division by the Fourier transform $\hat{\varphi}$ of the kernel, *i.e.*

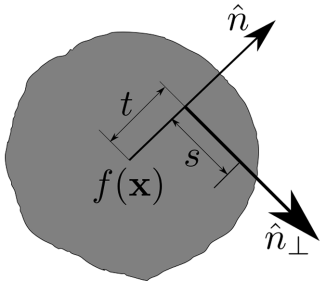


Figure 1
The Radon transform of f is its line integral along each line perpendicular to $\hat{\boldsymbol{n}}$.

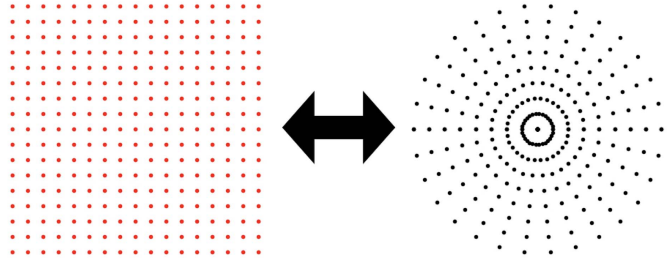


Figure 2
The NUFFT is used to transform intensity on a uniform grid to its Fourier transform on a polar grid, and *vice versa*.

$$c_j = \sum_t \rho(t, \theta_j) \exp\left(-2\pi i t k_j\right), \tag{6}$$

$$F_r = \sum_{\|\mathbf{k}_r^c - \mathbf{k}_j\| < W} c_j \varphi\left(\mathbf{k}_r^c - \mathbf{k}_j\right), \tag{7}$$

$$f_n \simeq \hat{\varphi}^{-1} \mathbb{F}_2^{-1}(F), \tag{8}$$

where $\{\mathbf{k}_r^c\}$ is a Cartesian grid, $\mathbb{F}_2$ is the 2D Fourier transform, equations (6) and (8) are computed via FFTs, and W is the spreading width of the convolution in equation (7). For an appropriately chosen kernel, the NUFFT has an error of ϵ if W is chosen to span approximately $w = \log_{10}(1/\epsilon)$ grid points per dimension. The computational complexity of equations (6)–(8) is $\mathcal{O}(M \log M_t + w^2 N + N \log N)$, where M_t is the number of points in the radial direction of the polar grid. Since $w^2 \ll M$, this results in a massive speedup compared with the direct computation of equation (5). The Radon transform can similarly be computed by performing the above steps in reverse order. In this work we have used the cuFINUFFT (Shih *et al.*, 2021) library to compute NUFFTs. For a detailed discussion on the topic, we refer the reader to Dutt & Rokhlin (1993), Fessler & Sutton (2003), Greengard & Lee (2004), Barnett *et al.* (2019) and Barnett (2021).

3. Model-based iterative reconstruction

An alternate approach to FBP methods is to rely on iterative methods, such as MBIR. Although these methods have longer processing times, they produce better quality reconstructions when compared with FBP methods. This is especially noticeable when a smaller number of projections are available. This is because iterative methods are able to incorporate *a priori* information as a constraint on the optimization process. We refer the reader to ASTM (2019) for a more detailed discussion. Iterative methods seek a solution f by minimizing the difference between its Radon transform and projection data b, *i.e.*

$$f = \arg\min_f \underbrace{\|Rf - b\|^2}_{\mathcal{G}} + \underbrace{g(f)}_{\mathcal{H}}. \tag{9}$$

Here we set up the objective function as a least-squared problem. The target is to iteratively search for f that minimizes the ℓ^2-norm of difference between Rf and b while penalizing violation of the constraint by g. Now we differentiate equation (9) with respect to f and equate the result to 0. The gradient of the first term is

$$\nabla \mathcal{G} = R^*(Rf - b) \tag{10}$$

where R^* is the adjoint of equation (2),

$$R^*\rho(\mathbf{x}) = \int_0^{\pi} \int_{-\infty}^{\infty} \exp\left(2\pi i k \langle \mathbf{x}, \hat{\boldsymbol{n}}(\theta)\rangle\right) \times \int_{-\infty}^{\infty} \exp\left(-2\pi i t k\right) \rho\left(t, \hat{\boldsymbol{n}}(\theta)\right) \mathrm{d}t\, \mathrm{d}k\, \mathrm{d}\theta, \tag{11}$$

which is simply equation (4) without the scaling $|k|$. The operators R and R^* can be efficiently computed using NUFFT.

In the results presented here, we choose $\mathcal{H}$ to be a total variation penalty in equation (9). We follow the mathematical approach presented by Venkatakrishnan *et al.* (2013) and Mohan *et al.* (2014) to model $\mathcal{H}$ as a q-generalized Gaussian Markov random field (qGGMRF),

$$g_m = \sum_n w_{mn} h_{mn}, \quad \forall\, n \in \{n \mid \|m - n\|_\infty \leq 1\}, \tag{12}$$

$$h_{mn} = \frac{\left(|f_m - f_n|/\sigma\right)^2}{c + \left(|f_m - f_n|/\sigma\right)^{2-p}}, \tag{13}$$

where h is defined over a 1-hop neighborhood of m, with m and n being integer coordinates on the three-dimensional uniform grid. The weights w_{mn} are the Gaussian weights that partition the unity and are inversely proportional to the distance between m and n. Hyper-parameters c, p and σ are used to control the strength of the penalty term. The term $\mathcal{H}$ is an algebraic expression, and can easily be differentiated.

In this work, we have used a monotonic accelerated gradient method with restart detailed by Giselsson & Boyd (2014), but it is possible to use other optimizers.

4. Implementation

When it comes to implementing software solutions, performance is a critical factor. In this section we discuss some of the important implementation details that have a significant impact on the performance of *tomoCAM*. These include factors such as memory management and hiding PCIe latency efficient GPU caching. To achieve both high performance and user-friendliness, we utilize a blend of C++, CUDA and Python. The data structures of *tomoCAM* are implemented in C++, while most of the mathematical functions are coded using CUDA. To efficiently handle large datasets, a two-tier partition scheme is employed to seamlessly stream data into and out of GPU memory. To address the vast number of pixels in a typical synchrotron micro- or nano-CT sinogram, which can exceed $\mathcal{O}(10^{10})$, we have carefully optimized the memory usage in the implementation of *tomoCAM*. For instance, to minimize the memory footprint, we pass large arrays that contain frequently accessed data such as the most recent solution, projection data and gradient as references rather than copies, which is the default behavior in C++. We avoid allocating two large arrays by updating the values in-place. This reduces the number of large allocations to six, saving 25% RAM. We have implemented various strategies to minimize the memory footprint, including the following:

(i) Quantities are never stored as complex numbers in the host memory. This additionally helps with the amount of data copied to and from the GPU memory.

(ii) Instead of duplicating data, partitions contain pointers to memory locations in the parent array.

(iii) Gradients are updated in place when computing the total variation constraint.

(iv) Projection data are reordered into sinogram form for fast contiguous readouts.

4.1. GPU optimizations

While GPUs are highly efficient in performing complex calculations, the latency over the PCIe bus remains a significant bottleneck for GPU-accelerated software implementations. To minimize runtime and maximize throughput from CPU to GPU memory, we employ a combination of techniques. These include asynchronous transfers, OpenMP threads and a two-tier data partitioning scheme. The partitioning is done along the axis of rotation, with the data first divided into as many partitions as there are available GPU devices. Each partition is then further subdivided into smaller chunks, with the optimal size depending on the GPU device's available memory. The sub-partitions are streamed to GPU memory, and to minimize the memory footprint they do not create deep copies of the data. Fig. 3 provides an overview of this process. By utilizing these techniques, we can significantly reduce the impact of the PCIe bottleneck and achieve higher performance in our GPU-accelerated software implementations. Some of the other optimizations and features of *tomoCAM* include the following:

(i) Since the axis of rotation may not be aligned with the center of the image, we use the Fourier shift property to efficiently move the rotation axis to the center of the image.

(ii) We use OpenMP threads to parallelly launch level-1 partitions on all the available GPUs, as well as to stream data into GPU memory.

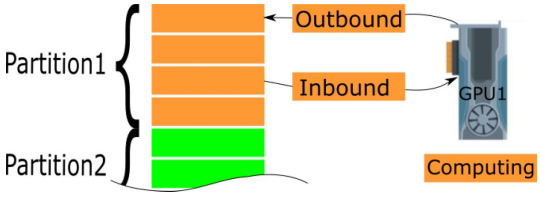


Figure 3
Large arrays are partitioned along the axis of rotation using a two-tier partitioning scheme. Larger, tier-1, partitions are per GPU. Smaller, tier-2, partitions (orange and green rectangles) are asynchronously streamed into GPU to overlap data transfers and computations.

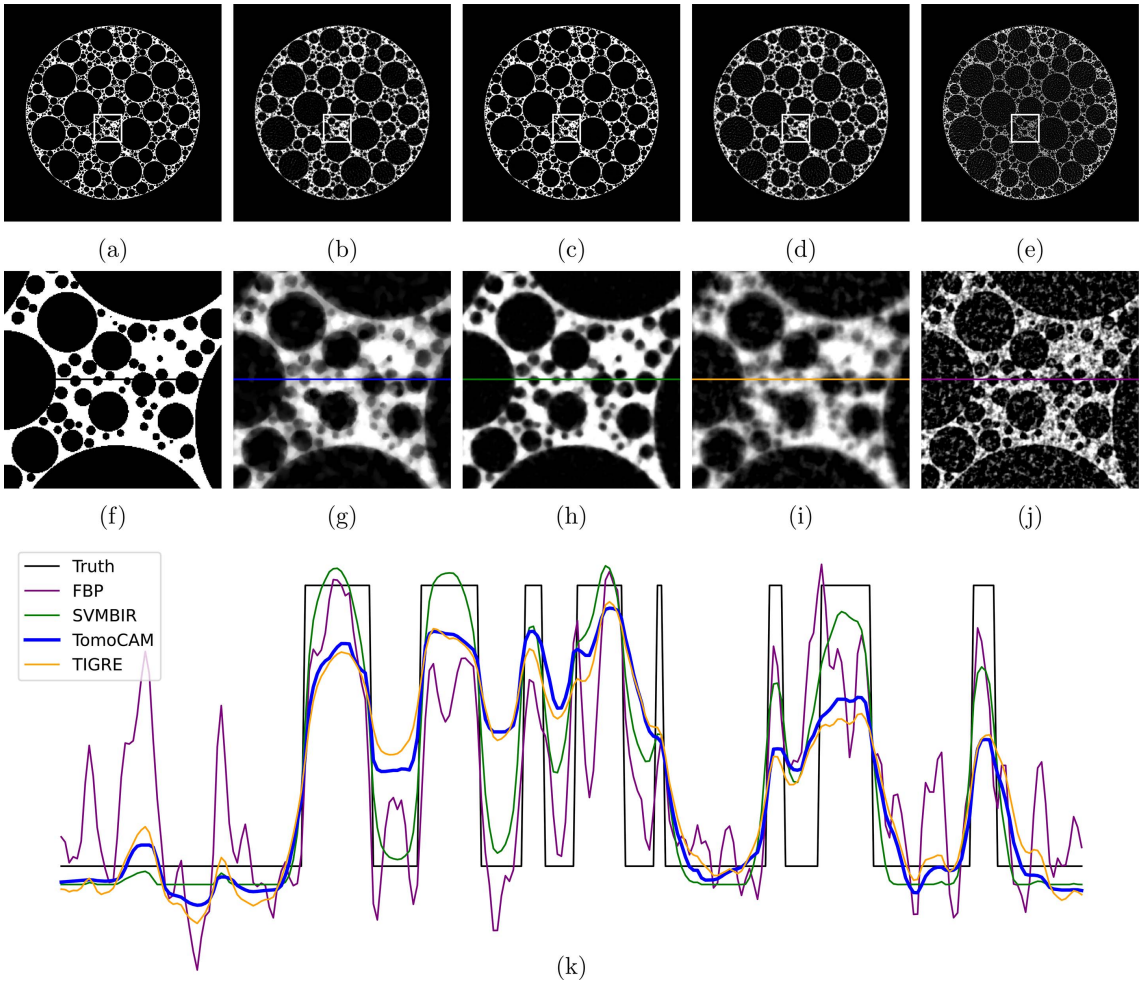


Figure 4
A compassion of reconstruction methods for a foam phantom with 128 projections. (*a*) Ground truth, (*b*) *tomoCAM*, (*c*) *SVMBIR*, (*d*) *TIGRE* (IRN-TV-CGLS) and (*e*) gridrec. Panels (*f*), (*g*), (*h*), (*i*) and (*j*) are the zoomed-in regions of interest represented by the boxes in (*a*), (*b*), (*c*), (*d*) and (*e*), respectively, and (*k*) displays the line profiles on (*f*), (*g*), (*h*), (*i*) and (*j*). Reconstructions using *tomoCAM*, *SVMBIR* and *TIGRE* (IRN-TV-CGLS) result in images with low noise when compared with gridrec. The relative root-mean-square error values when compared against the ground truth for gridrec, *SVMBIR*, *TIGRE* and *tomoCAM* are 0.95, 0.33, 0.53 and 0.47, respectively. Here *tomoCAM* is about 9× faster than *SVMBIR*, and 20× faster than *TIGRE* (IRN-TV-CGLS).

(iii) To improve the cache efficiency, we utilize GPUs' `__shared__` memory to store data that are accessed multiple times, such as when computing the 'total variation' constraint.

(iv) A Python front-end and *NumPy* compatibility are provided via the *pybind11* project (Jakob *et al.*, 2017).

5. Numerical experiments

We tested *tomoCAM* with publicly available phantoms and measured datasets. Here, we present a comparison of reconstructed results using *tomoCAM*, *SVMBIR*, *TIGRE* (Biguri *et al.*, 2016, 2020) and FBP using gridrec available in the *Tomopy* package (Gürsoy *et al.*, 2014). Each reconstruction and line profile (B) is scaled with s and shifted with Δ, where $s, \Delta = \arg\min_{s,\Delta} \|A - sB + \Delta\|$ to the ground truth (A) before plotting. In the case of experimental data, we rescale reconstructions from *SVMBIR* and gridrec with the one obtained from *tomoCAM*. The total variation constraint used in *SVMBIR* is slightly different from the one used in *tomoCAM* [see the theory section of SVMBIR (2020)]. *SVMBIR* uses ten nearest neighbors, while *tomoCAM* uses 26 of them, to evaluate equation (12). We believe parameters can be fine-tuned for *tomoCAM* and *SVMBIR* to produce equivalent results. The primary comparison with *SVMBIR* is to demonstrate performance gains, rather than comparing two different constraints or image quality. All the tests were carried out on a single machine with (i) 2 × Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz; (ii) 4 × Tesla P100 GPUs; (iii) 128 GB RAM.

In the first experiment, we compare the reconstruction of a foam phantom from all three codes. A foam phantom and its projection data of size (128 × 16 × 2048) was generated using the *foam_ct_phantom* package (Pelt *et al.*, 2022). A full slice from the phantom in Fig. 4(*a*) is compared with the reconstruction obtained from each code (*SVMBIR*, *tomoCAM* and

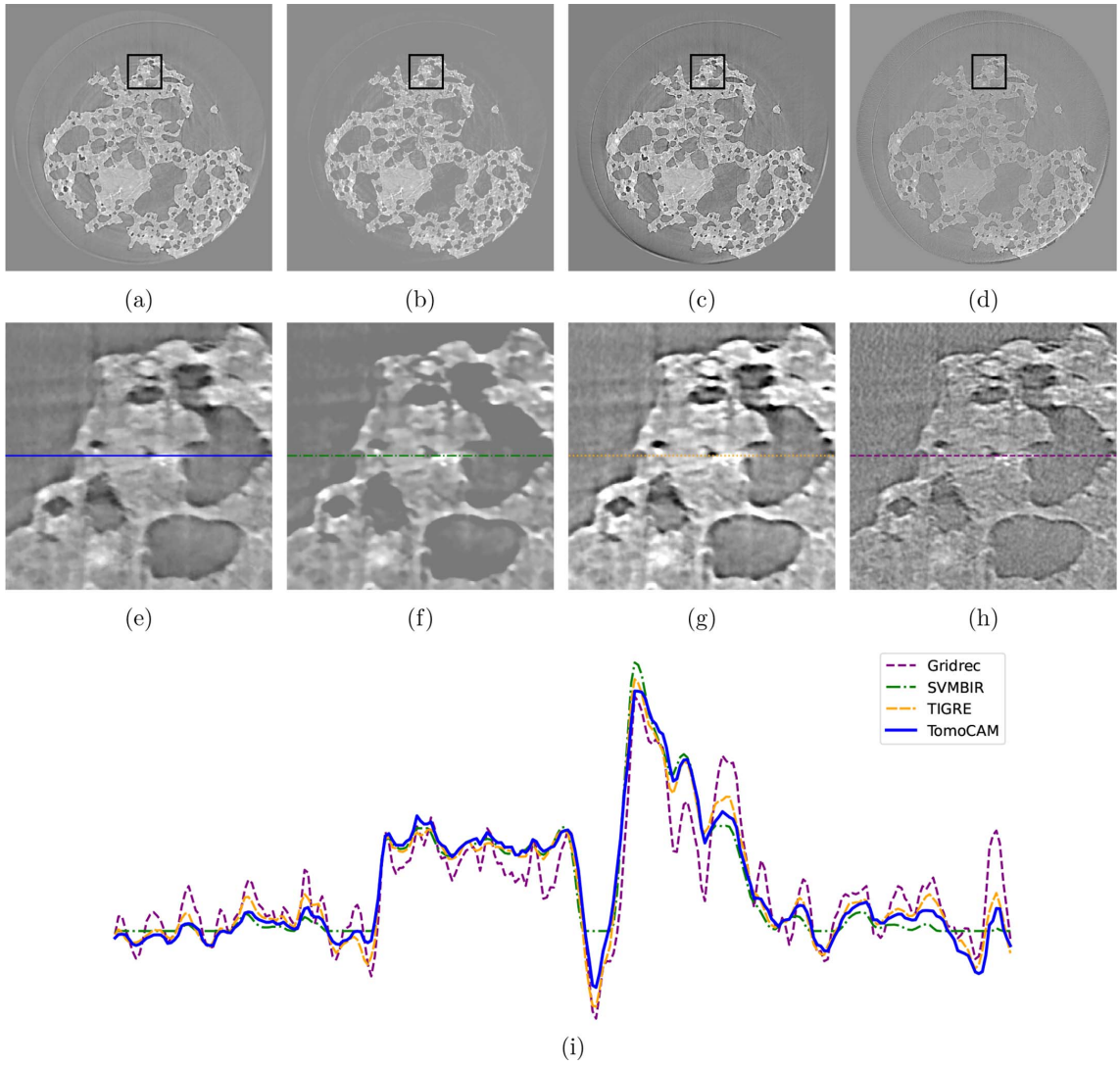


Figure 5
Reconstructions for Tomobank dataset ID 25, micro-CT *in situ* study of rock permeability with 400 projections, with (*a*) *tomoCAM*, (*b*) *SVMBIR*, (*c*) *TIGRE* (IRN-TV-CGLS) and (*d*) gridrec. Panels (*e*), (*f*), (*g*) and (*h*) are zoomed-in regions of interest represented by the boxes in (*a*), (*b*), (*c*) and (*d*), respectively. Panel (*i*) displays the line profiles for (*e*), (*f*), (*g*) and (*h*). A circular mask was applied to all the reconstructions. While *tomoCAM*, *SVMBIR* and *TIGRE* (IRN-TV-CGLS) do an excellent job at suppressing the noise when compared with gridrec, *tomoCAM* is approximately 15× faster than *SVMBIR* and 22× faster than *TIGRE* (IRN-TV-CGLS).

gridrec) in Figs. 4(*b*)–4(*e*). This is followed by zoomed-in regions of each image in Figs. 4(*f*)–4(*j*). A line profile from each of the zoomed-in regions is then compared in Fig. 4(*k*). It is evident from the results that both *tomoCAM* and *SVMBIR* are effective at suppressing the noise. One major advantage of *tomoCAM* is that it can obtain equivalent results in an order-of-magnitude faster time.

Next, we evaluate the reconstruction of two experimental datasets obtained from diverse synchrotron light sources that are accessible through Tomobank (Carlo *et al.*, 2018) (see Figs. 5 and 6). For each dataset, the available number of projections is notably lower than what is typically expected, which follows the general rule of thumb that it should be as many as the number of pixel columns in the camera sensor.

Using Beer–Lambert's law (Swinehart, 1962), b in equation (9) is defined as $-\log(I/I_0)$, where I is the measured intensity and I_0 is the beam intensity without the sample blocking the view. The selection of hyper-parameters can influence the quality of reconstruction. However, in practice $p = 1.2$ (Mohan *et al.*, 2014) and $c = 0.0001$ have shown good performance across multiple datasets. The strength of the qGGMRF constraint is controlled by the parameter σ. Lower values of σ increase the contribution of the constraint, resulting in smoother profiles, while higher values decrease the contribution. We evaluate a spectrum of σ values on a small subset of dataset, in order to ascertain the hyper-parameter value that yields a desirable quality of data reconstruction, see Fig. 7. Without a quantitative measure of reconstruction quality, we rely on a human expert to determine the fitness of the reconstruction. In order to automate this process, we are currently working on developing new machine-learning approaches to recommend optimal hyper-parameters based on the features of the data.

For gridrec we chose the Butterworth filter with order 2, and the cutoff frequency was set to 0.25, which is typical for a synchrotron tomographic reconstruction. For

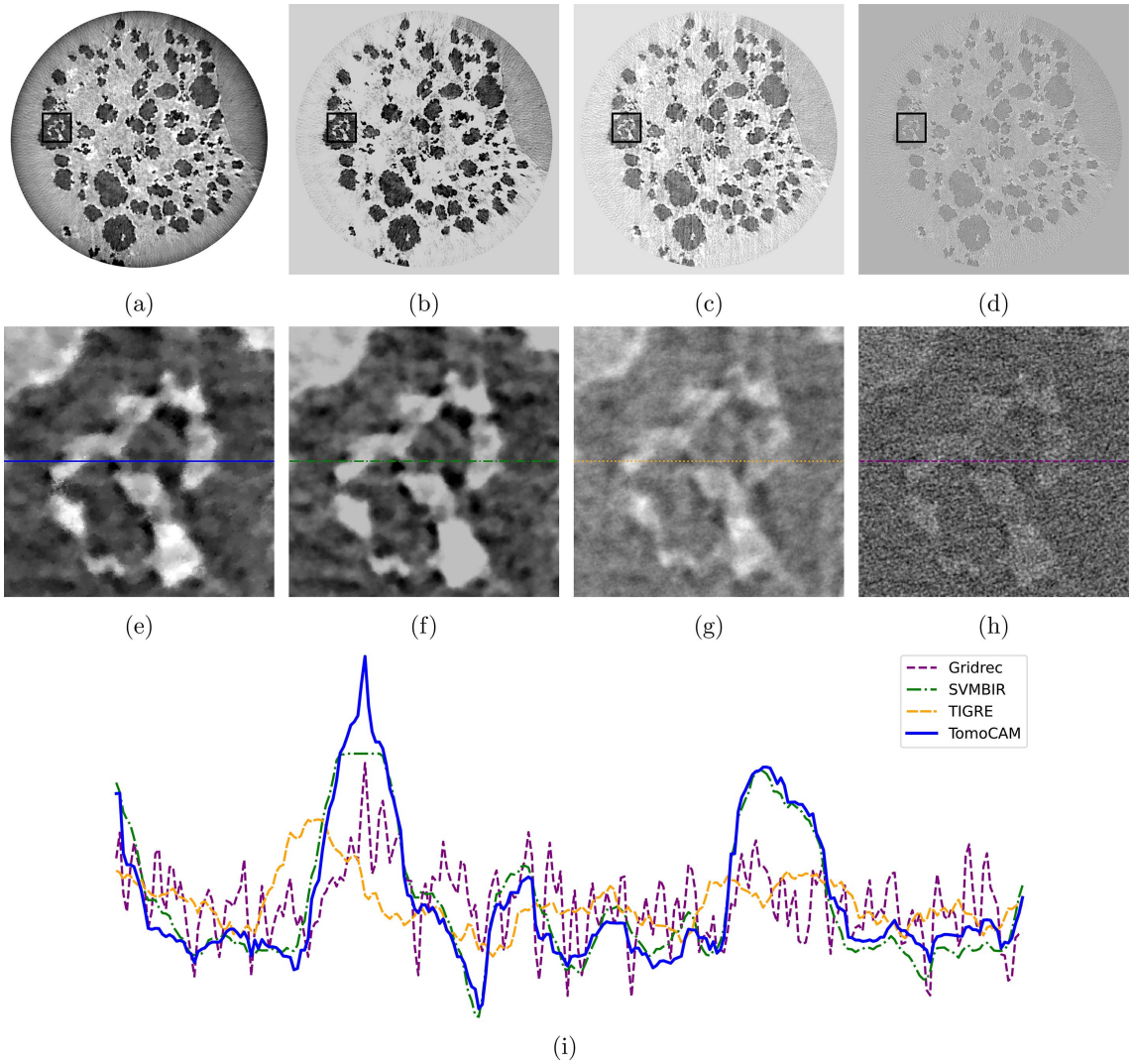


Figure 6
Reconstructions for Tomobank dataset ID 86, nano-CT data with sparse projection angles using 202 projections, with (*a*) *tomoCAM*, (*b*) *SVMBIR*, (*c*) *TIGRE* (IRN-TV-CGLS) and (*d*) gridrec. Panels (*e*), (*f*), (*g*) and (*h*) are zoomed-in regions of interest represented by the boxes in (*a*), (*b*), (*c*) and (*d*), respectively. Panel (*i*) displays the line profiles for (*d*), (*e*), (*f*) and (*h*). A circular mask was applied to all the reconstructions. While *tomoCAM* and *SVMBIR* both do an excellent job at suppressing the noise when compared with gridrec, *tomoCAM* is approximately 15× faster. Also, *tomoCAM* is 22× faster than *TIGRE* (IRN-TV-CGLS).

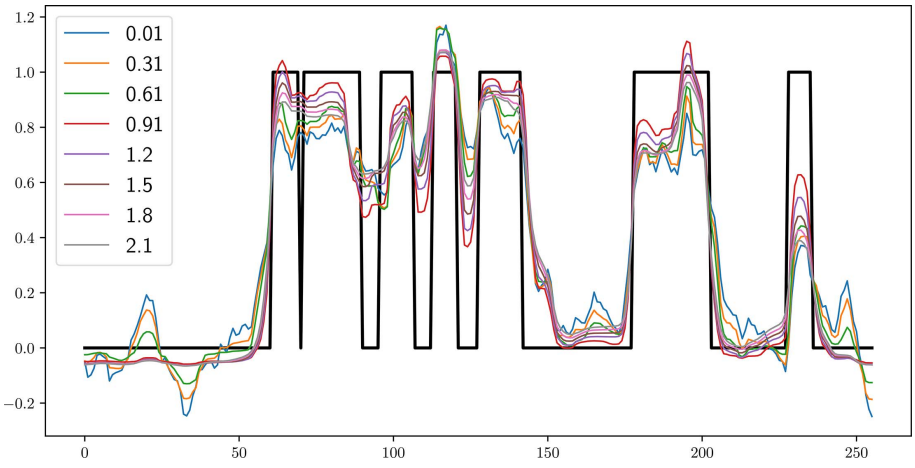


Figure 7
A small subset of slices were reconstructed using a range of hyper-parameters σ to determine an optimal reconstruction quality.

SVMBIR we choose hyper-parameters as $T = 1$ and $\sigma_x = \{0.98, 2.1, 1.1\} \times 10^{-4}$ for the phantom, Tomobank dataset id 25 (TB-25) and Tomobank dataset id 86 (TB-86), respectively, in order to produce similar quality reconstructions as *tomoCAM*. Fig. 8 shows the convergence rates of the foam phantom, TB-25 and TB-86 samples.

We follow a similar pattern to Fig. 4 for plotting images and line profiles. The first row of images depicts full slices, followed by zoomed-in regions, and then a line profile is taken from the middle of each zoomed-in region. Given their mathematical similarity, we expect that conducting a thorough hyper-parameter search would yield comparable outcomes from both *tomoCAM* and *SVMBIR*. Table 1 shows a comparison of the time taken by each code. Small discrepancies between *tomoCAM* and *SVMBIR* are due to the difference in hyper-parameters, as well as how each of the packages implements the qGGMRF constraint. We use equation (12) directly, whereas *SVMBIR* uses a surrogate function to approximate it. We would also like to note that *TIGRE* uses a fundamentally different approach to enforce the total variation constraint (Biguri *et al.*, 2016). We have used the default hyper-parameters for reconstructions with *TIGRE* (IRN-TV-CGLS). We assume that a systematic search for hyper-parameters will produce similar quality results as *tomoCAM* and *SVMBIR*.

We have also implemented a hierarchical reconstruction method that iteratively refines the solution by starting at a low resolution and gradually increasing the resolution. This reduces the number of optimization iterations required at higher resolution, leading to significant speedups. In the example shown in Fig. 9, for a foam phantom with 512 projections of 512 × 2048 resolution, we apply this hierarchical approach. The final reconstruction resolution is (512 × 2048 × 2048). We started with projection images down-sampled to one-quarter of the original resolution, running for 60 iterations. This was followed by two cycles of up-sampling the solution, with reconstruction at the half resolution running for 30 iterations; and finally 15 iterations at the full resolution. The hierarchical method is five times faster than running at full resolution, *i.e.* 809 versus 4090 s for similar convergence values. We are working on making further improvements to the hierarchical method before it is made publicly available.

Table 1
A comparison of the time taken to reconstruct various datasets; iterative methods *tomoCAM*, *SVMBIR* and *TIGRE* were timed for 100 iterations.

		Reconstruction time (s)			
Data set	Size	*tomoCAM*	*SVMBIR*	*TIGRE*	gridrec
Phantom	(128, 16, 2048)	93	810	1820	0.21
TB-25	(400, 128, 2048)	862	12730	18833	2.53
TB-86	(202, 128, 2448)	1210	14273	26147	3.73

6. Conclusions

In this work, we have presented *tomoCAM*, a new GPU-accelerated software for reconstructing high-quality tomographic images. *tomoCAM* is capable of running model-based iterative reconstructions for large datasets with relatively modest hardware requirements, within a reasonable time. The resulting reconstructed images have lower noise when compared with the prevalent FBP methods, while being an order of magnitude faster than CPU-only MBIR implementations.

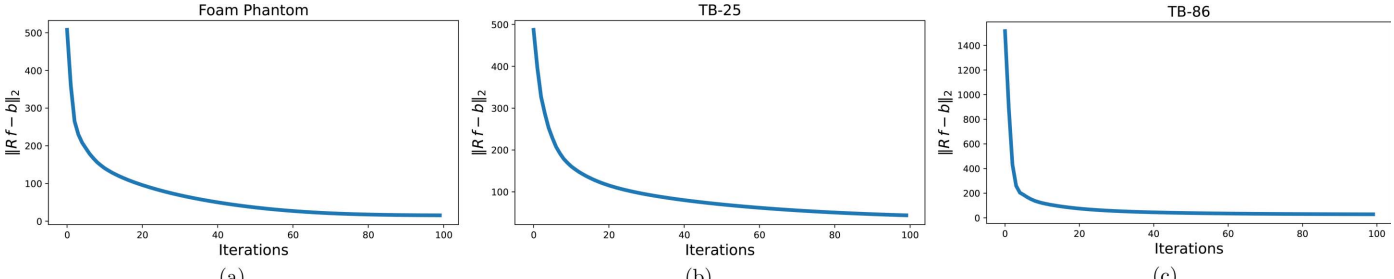


Figure 8
L2 error versus iterations for (*a*) a foam phantom, (*b*) TB-25 and (*c*) TB-86. Although the optimizer converges quickly, the resolution of smaller features may need extra iterations.

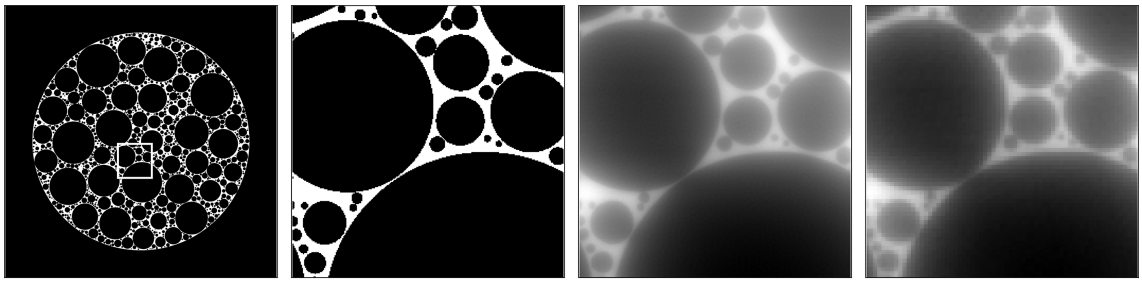


Figure 9
Hierarchical method. The optimization process begins with lower-resolution projection data. The result from this lower-resolution reconstruction is then up-sampled and serves as the initial point for the next cycle, which needs fewer iterations. Panel (*a*) represents the ground truth, and panel (*b*) is an area of interest within (*a*). Panel (*c*) is the same region of interest from a 100-iteration reconstruction at full resolution, and panel (*d*) shows the reconstruction achieved through the hierarchical method. The hierarchical method takes one-fifth of the time required for reconstruction at the full resolution to achieve the same convergence.

A Python-based front-end has been created for *tomoCAM*, which is specifically designed to receive *NumPy* arrays as both input and output for reconstructions. This facilitates seamless integration of *tomoCAM* into the existing workflows of beamline scientists. Although the use of MBIR is particularly advantageous in cases where there is a scarcity of available projection data, the current implementation of MBIR is quite time-consuming. Consequently, this is the primary reason why beamline scientists do not utilize MBIR even when it is advantageous. *tomoCAM* overcomes this problem, thus making MBIR reconstruction more practical, by the following:

(i) Improving efficiency: the run time has been reduced by an order of magnitude, making it faster than previous MBIR versions.

(ii) Reducing hardware requirements: it can run on machines as small as an individual desktop with a GPU, making it more accessible.

(iii) Simplifying hyper-parameter search: *tomoCAM*'s speed makes it easier to search for hyper-parameters, allowing for faster and more efficient experimentation.

(iv) Enhancing compatibility: the implementation provides a Python interface, which makes it easy to integrate with existing workflows that use FBP.

Acknowledgements

This work was supported by the Center for Advanced Mathematics for Energy Research Applications (CAMERA). This research used resources of the Advanced Light Source, which is a DOE Office of Science User Facility under contract no. DE-AC02-05CH11231. We thank S. V. Venkatakrishnan of Oak Ridge National Laboratory for numerous valuable discussions. We would also like to thank our colleagues J. A. Sethian, Z. Hu and K. Pande for reviewing the manuscript.

Funding information

The following funding is acknowledged: Advanced Scientific Computing Research and Basic Energy Sciences programs of the Office of Science of the Department of Energy (DOE) (Award No. DE-AC02-05CH11231).

References

Aarle, W. van, Palenstijn, W. J., Cant, J., Janssens, E., Bleichrodt, F., Dabravolski, A., De Beenhouwer, J., Joost Batenburg, K. & Sijbers, J. (2016). *Opt. Express*, **24**, 25129–25147.

Aarle, W. van, Palenstijn, W. J., De Beenhouwer, J., Altantzis, T., Bals, S., Batenburg, K. J. & Sijbers, J. (2015). *Ultramicroscopy*, **157**, 35–47.

ASTM (2019). *ASTM E1441-19 – Standard Guide for Computed Tomography (CT)*. American Society for Testing and Materials, West Conshohocken, PA, USA.

Barnett, A. (2021). *Appl. Comput. Harmon. Anal.* **51**, 1–16.

Barnett, A., Magland, J. & af Klinteberg, L. (2019). *SIAM J. Sci. Comput.* **41**, C479–C504.

Barutcu, S., Aslan, S., Katsaggelos, A. K. & Gürsoy, D. (2021). *Sci. Rep.* **11**, 17740.

Biguri, A., Dosanjh, M., Hancock, S. & Soleimani, M. (2016). *Biomed. Phys. Eng. Expr.* **2**, 055010.

Biguri, A., Lindroos, R., Bryll, R., Towsyfyan, H., Deyhle, H., Harrane, I. E., Boardman, R., Mavrogordato, M., Dosanjh, M., Hancock, S. & Blumensath, T. (2020). *J. Parallel Distrib. Comput.* **146**, 52–63.

Butterworth, S. (1930). *Wireless Eng.* **7**, 536–541.

Candes, E. (2021). *Math 262/CME 372 Applied Fourier Analysis and Elements of Modern Signal Processing*, https://candes.su.domains/teaching/math262/.

Crowther, R. A., DeRosier, D. J. & Klug, A. (1970). *Proc. R. Soc. London A*, **317**, 319–340.

De Carlo, F., Gürsoy, D., Ching, D. J., Batenburg, K. J., Ludwig, W., Mancini, L., Marone, F., Mokso, R., Pelt, D. M., Sijbers, J. & Rivers, M. (2018). *Meas. Sci. Technol.* **29**, 034004.

Dienemann, L. L., Geller, L. C., Huang, Y., Zenyuk, I. V. & Panzer, M. J. (2023). *Appl. Mater. Interfaces*, **15**, 8492–8501.

Dutt, A. & Rokhlin, V. (1993). *SIAM J. Sci. Comput.* **14**, 1368–1393.

Epstein, C. L. (2007). *Introduction to the Mathematics of Medical Imaging*, 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics.

Fessler, J. & Sutton, B. (2003). *IEEE Trans. Signal Process.* **51**, 560–574.

Forna-Kreutzer, J. P., Ell, J., Barnard, H., Pirzada, T. J., Ritchie, R. O. & Liu, D. (2021). *Mater. Des.* **208**, 109899.

French, J., Dahlkamp, C., Befus, E. & Czabaj, M. W. (2022). *Compos. Sci. Technol.* **224**, 109453.

Ge, M., Coburn, D. S., Nazaretski, E., Xu, W., Gofron, K., Xu, H., Yin, Z. & Lee, W.-K. (2018). *Appl. Phys. Lett.* **113**, 083109.

Giselsson, P. & Boyd, S. P. (2014). *Proceedings of the 53rd IEEE Conference on Decision and Control (CDC 2014)*, 15–17 December 2014, Los Angeles, CA, USA, pp. 5058–5063. IEEE.

Greengard, L. & Lee, J.-Y. (2004). *SIAM Rev.* **46**, 443–454.

Gürsoy, D., De Carlo, F., Xiao, X. & Jacobsen, C. (2014). *J. Synchrotron Rad.* **21**, 1188–1193.
Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. E. (2020). *Nature*, **585**, 357–362.
Jakob, W., Rhinelander, J. & Moldovan, D. (2017). *pybind11*, https://github.com/pybind/pybind11.
Jørgensen, J. S., Ametova, E., Burca, G., Fardell, G., Papoutsellis, E., Pasca, E., Thielemans, K., Turner, M., Warr, R., Lionheart, W. R. B. & Withers, P. J. (2021). *Philos. Trans. R. Soc. A.* **379**, 20200192.
Jørgensen, J. S. & Lionheart, W. R. B. (2021). *Computed Tomography: Algorithms, Insight, and Just Enough Theory*, ch. 6, pp. 73–103.
Kak, A. C. & Slaney, M. (2001). *Principles of Computerized Tomographic Imaging.* Philadelphia: Society for Industrial and Applied Mathematics.
Kazantsev, D. & Wadeson, N. (2020). *CT Meeting 2020: Proceedings of the 6th International Conference on Image Formation in X-ray Computed Tomography*, 3–7 August 2020, Regensburg, Germany, pp. 450–453.
Kulkarni, D., Normile, S. J., Connolly, L. G. & Zenyuk, I. V. (2020). *J. Phys. Energy*, **2**, 044005.
Larson, N. M. & Zok, F. W. (2018). *Composites Part A*, **107**, 124–134.
MacDowell, A. A., Parkinson, D. Y., Haboub, A., Schaible, E., Nasiatka, J. R., Yee, C. A., Jameson, J. R., Ajo-Franklin, J. B., Brodersen, C. R. & McElrone, A. J. (2012). *Proc. SPIE*, **8506**, 850618.
Mohan, K. A., Venkatakrishnan, S. V., Drummy, L. F., Simmons, J., Parkinson, D. Y. & Bouman, C. A. (2014). *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*, 4–9 May 2014, Florence, Italy, pp. 6909–6913. IEEE.
Mokso, R., Schlepütz, C. M., Theidel, G., Billich, H., Schmid, E., Celcer, T., Mikuljan, G., Sala, L., Marone, F., Schlumpf, N. & Stampanoni, M. (2017). *J. Synchrotron Rad.* **24**, 1250–1259.
Nikitin, V. (2023). *J. Synchrotron Rad.* **30**, 179–191.
Nikitin, V., Tekawade, A., Duchkov, A., Shevchenko, P. & De Carlo, F. (2022). *J. Synchrotron Rad.* **29**, 816–828.
Pelt, D. M., Hendriksen, A. A. & Batenburg, K. J. (2022). *J. Synchrotron Rad.* **29**, 254–265.
Ramachandran, G. N. & Lakshminarayanan, A. V. (1971). *Proc. Natl Acad. Sci. USA*, **68**, 2236–2240.
Shepp, L. A. & Logan, B. F. (1974). *IEEE Trans. Nucl. Sci.* **21**, 21–43.
Shih, Y., Wright, G., Andén, J., Blaschke, J. & Barnett, A. H. (2021). *arXiv*:2102.08463.
SVMBIR (2020). *Super-Voxel Model Based Iterative Reconstruction (SVMBIR)*, https://github.com/cabouman/svmbir.
Swinehart, D. F. (1962). *J. Chem. Educ.* **39**, 333.
Tarantola, A. & Valette, B. (1982). *Rev. Geophys.* **20**, 219–232.
Thuering, T., Modregger, P., Grund, T., Kenntner, J., David, C. & Stampanoni, M. (2011). *Appl. Phys. Lett.* **99**, 041111.
Trampert, J. & Leveque, J. (1990). *J. Geophys. Res.* **95**, 12553–12559.
Venkatakrishnan, S. V., Drummy, L. F., Graef, M. D., Simmons, J. P. & Bouman, C. A. (2013). *Proc. SPIE*, **8657**, 86570A.
Vingelmann, P. & Fitzek, F. H. (2020). CUDA, release 10.2.89. NVIDIA, CA, USA (https://developer.nvidia.com/cuda-toolkit).
Walsh, S. D., Mason, H. E., Du Frane, W. L. & Carroll, S. A. (2014). *Int. J. Greenhouse Gas Control*, **22**, 176–188.
Zhang, R., Thibault, J.-B., Bouman, C. A., Sauer, K. D. & Jiang Hsieh, (2014). *IEEE Trans. Med. Imaging*, **33**, 117–134.