The Journal of Open Source Software

BM3DORNL: High-Performance BM3D Denoising for Neutron Tomography

Chen Zhang[1¶], **Jean-Christophe Bilheux**[2], **Dmitry Ganyushin**[1], and **Pete Peterson**[1]

1 Computing and Computational Sciences Directorate, Oak Ridge National Laboratory, Oak Ridge, TN, USA **2** Neutron Sciences Directorate, Oak Ridge National Laboratory, Oak Ridge, TN, USA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

BM3DORNL is a high-performance Python library for denoising neutron and X-ray tomography data using a modified Block-Matching and 3D Filtering (BM3D) algorithm. The library provides two denoising modes: a generic mode for standard noise removal, and a specialized streak mode optimized for removing vertical streak patterns in sinograms that manifest as ring artifacts in reconstructed images. Built with a Rust backend and Python bindings via PyO3, BM3DORNL achieves 160× faster processing than comparable BM3D implementations while maintaining cross-platform compatibility including native Apple Silicon support.

Statement of Need

Ring artifacts are a persistent challenge in neutron and X-ray computed tomography, arising from variations in detector pixel response, beam intensity fluctuations, and systematic errors (Münch et al., 2009). These artifacts appear as concentric rings in reconstructed images, obscuring sample structure and degrading quantitative analysis. Pre-reconstruction streak removal addresses these artifacts directly in sinograms, avoiding the spatial spreading that occurs during reconstruction. Existing approaches include wavelet-Fourier filtering (Münch et al., 2009), polynomial fitting (Vo et al., 2018), and sorting-based methods (Miqueles et al., 2014), with implementations available in TomoPy (Gürsoy et al., 2014).

Mäkinen et al. (Mäkinen et al., 2021) demonstrated that applying BM3D (Dabov et al., 2007) across multiple scales achieves superior artifact removal by exploiting the self-similarity of streak patterns. However, their bm3d-streak-removal implementation is closed-source, restricted to non-commercial use, and provides only x86_64 binaries incompatible with Apple Silicon and modern Python versions. BM3DORNL provides the neutron imaging community with an open-source, MIT-licensed, high-performance implementation that enables both high-throughput batch processing and interactive parameter tuning.

The Journal of Open Source Software

State of the Field

Several software packages implement pre-processing streak removal for tomography: TomoPy (Gürsoy et al., 2014) provides wavelet-Fourier filtering (Münch et al., 2009) and Vo's sorting/fitting methods (Vo et al., 2018); ASTRA Toolbox (Aarle et al., 2016) offers GPU-accelerated preprocessing; and bm3d-streak-removal (Mäkinen et al., 2021) implements multiscale BM3D but remains closed-source and platform-limited.

Figure 1 shows the benchmark input data: a synthetic sinogram (720$\times 725 pixels) with simulated ring ar streak − removal, which provides only x86_64 binaries. We compared eight methods : four BM3DORNL variants (streak, generic, multiscale, and Fourier−SVD), three TomoPy alg Fourier, sorting − fitting, and sorting − based), and the original bm3d − streak − removal. The multiscale variant implements the pyramid approach from Mkinen et al. [@mkinen20 SVD is a lightweight FFT − guided method that achieves comparable quality at 50\times$ faster speeds.

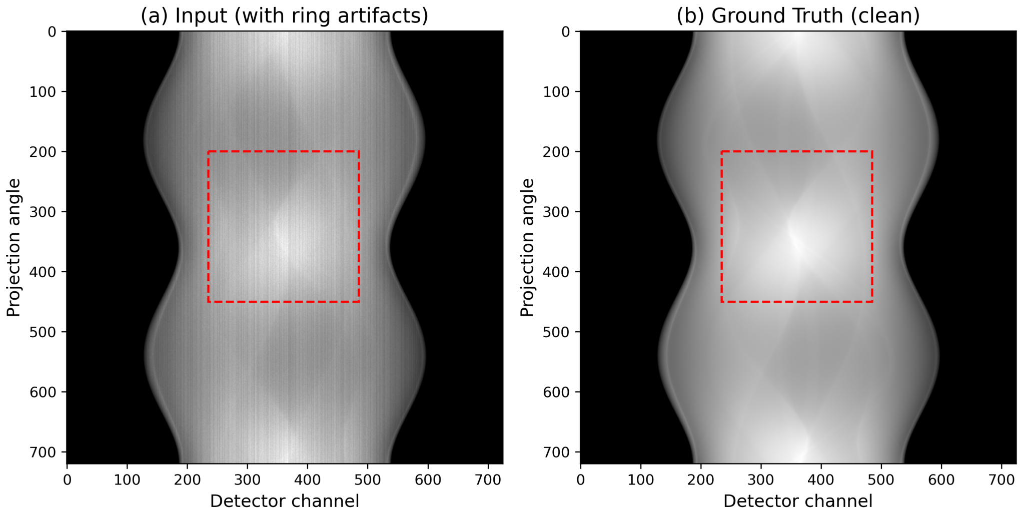

Figure 1: Benchmark input data. (a) Input sinogram with simulated ring artifacts. (b) Ground truth (clean sinogram). The dashed rectangle indicates the crop region shown in Figure 2.

Speed comparison: Figure 3(a) shows processing times (n=100 runs on Linux x86_64). Fourier-SVD is the fastest method at 0.017 seconds, achieving 2300× speedup over bm3d-streak-removal (39.9 seconds). The BM3D-based methods span a performance range: standard BM3DORNL processes sinograms in 0.4 seconds (100× faster than bm3d-streak-removal), while multiscale BM3DORNL takes 0.9 seconds (44× faster) due to pyramid processing. TomoPy methods achieve comparable processing times (0.7–0.8 seconds). Cross-platform performance differs substantially: TomoPy runs 2.5–3× slower on Apple Silicon than Linux x86_64 (likely due to unoptimized C extensions), whereas BM3DORNL runs at comparable speeds (within 10%) across platforms due to its Rust backend.

Quality analysis: Figure 2 shows cropped results from all eight methods with their difference images (result minus ground truth). The benchmark reveals that quality metrics alone do not tell the full story. TomoPy SF achieves the highest SSIM (0.987, see Figure 3(b)), but difference image analysis shows residual vertical stripes in its output. This occurs because SSIM uses local windows (7$\times 7 or 11 \times$11 pixels), and narrow vertical stripes affect few pixels per window. Interestingly, both multiscale BM3DORNL and Fourier-SVD achieve excellent SSIM scores (0.951) while maintaining fast processing times—Fourier-SVD accomplishes this at 0.017 seconds, demonstrating that aggressive artifact removal and high-quality reconstruction are achievable without computationally expensive multi-scale processing. The difference images reveal that all BM3DORNL variants produce clean vertical patterns, indicating successful artifact removal, while TomoPy methods leave faint residual artifacts.

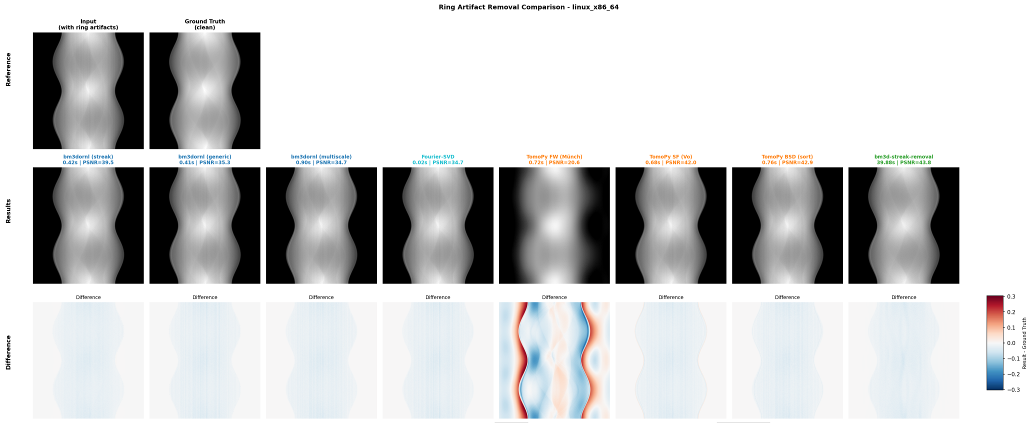

Figure 2: Method comparison. Top row: Cropped results from each method with processing times. Bottom row: Difference images (result minus ground truth) using a zero-centered RdBu colormap—blue indicates artifact removal, red indicates added artifacts. Data from Linux x86_64.

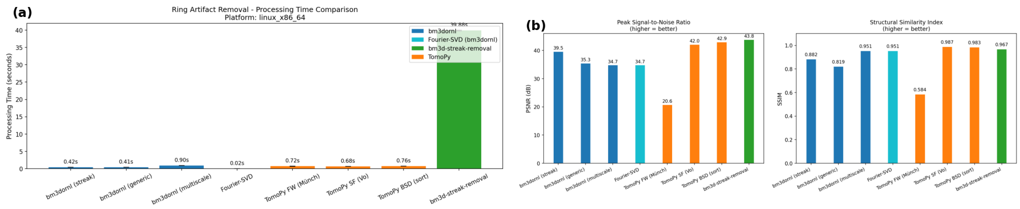

Figure 3: Performance metrics (Linux x86_64, n=100 runs). (a) Processing time comparison on linear scale, showing Fourier-SVD's 2300× speedup and standard BM3DORNL's 100× speedup over bm3d-streak-removal. (b) Quality metrics (PSNR vs SSIM) showing trade-offs between methods. BM3DORNL variants (blue/cyan), TomoPy (orange), bm3d-streak-removal (green).

Platform support: bm3d-streak-removal is unavailable on Apple Silicon (x86_64 binaries only), incompatible with Python 3.11+, and restricted to non-commercial use. BM3DORNL provides native performance on all platforms, supports Python 3.12+, and uses the MIT license for unrestricted commercial use.

Software Design

BM3DORNL employs a hybrid Python-Rust architecture: the core algorithm is implemented in Rust using the rayon crate for work-stealing parallelism, with Python bindings via PyO3 for seamless NumPy integration.

Key optimizations:

- **Integral image pre-screening:** Before computing expensive patch distances, the block matching stage uses integral images to compute mean and norm bounds in O(1) time. Patches that fail these bounds are skipped, eliminating approximately 80% of distance calculations.
- **Walsh-Hadamard transform:** For the default 8$×$8 patches, BM3DORNL uses Walsh-Hadamard transforms instead of FFT. This is multiplication-free (only additions and

subtractions), cache-friendly with fixed 64-element buffers, and computed in-place without memory allocation.

- **Pre-computed FFT plans:** FFT plans are computed once and shared across threads via Arc, avoiding expensive plan initialization on each invocation (measured 45% speedup).
- **Batched stack processing:** Sinograms are processed slice-by-slice to control memory usage, ensuring each slice fits in L2/L3 cache while minimizing memory allocation through buffer reuse.

The library provides a minimal Python API:

```python
from bm3dornl import bm3d_ring_artifact_removal

cleaned = bm3d_ring_artifact_removal(
    sinogram,
    mode="streak",
    sigma=0.0, # default to automatic noise estimation
)
```

GUI application: BM3DORNL includes a native GUI built with the egui framework for Rust. The GUI enables interactive parameter tuning, side-by-side comparison with difference visualization, HDF5/TIFF file loading with dataset browsing, and real-time processing feedback. At 0.2–0.3 seconds per sinogram, scientists can explore parameter space interactively—something impractical with bm3d-streak-removal's 41-second processing time.

Research Impact

BM3DORNL is being integrated into processing pipelines at the VENUS and MARS beamlines at Oak Ridge National Laboratory. The library provides multiple performance tiers: Fourier-SVD enables real-time parameter exploration at 0.017 seconds per sinogram (2300× faster than bm3d-streak-removal), standard BM3DORNL offers robust denoising at 0.4 seconds (100× speedup), and multiscale BM3DORNL provides maximum quality at 0.9 seconds (44× speedup). For batch processing, a 1000-sinogram dataset that would take 11 hours with bm3d-streak-removal completes in 17 seconds (Fourier-SVD), 7 minutes (standard BM3DORNL), or 15 minutes (multiscale BM3DORNL).

The hybrid Rust-Python architecture demonstrates a modern approach to scientific software development: Rust provides memory safety and performance portability (single codebase compiles natively to arm64 and x86_64), while Python ensures integration with the NumPy/SciPy ecosystem. This pattern is increasingly valuable as the scientific computing community diversifies hardware platforms.

BM3DORNL fills a critical gap: the neutron imaging community needed an open-source, actively maintained BM3D implementation that works on modern platforms. The MIT license enables unrestricted use at national facilities, and native Apple Silicon support ensures scientists can use contemporary hardware for analysis workstations.

AI Usage Disclosure

Generative AI tools (Claude) were used for code assistance and documentation drafting. All AI-generated content was reviewed, tested, and validated by the authors.

Acknowledgements

This research used resources at the Spallation Neutron Source, a DOE Office of Science User Facility operated by Oak Ridge National Laboratory. This work is supported by the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

Aarle, W. van, Palenstijn, W. J., Cant, J., Janssens, E., Bleichrodt, F., Dabravolski, A., De Beenhouwer, J., Batenburg, K. J., & Sijbers, J. (2016). Fast and flexible x-ray tomography using the ASTRA toolbox. *Optics Express*, *24*(22), 25129–25147. https://doi.org/10.1364/OE.24.025129

Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. (2007). Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, *16*(8), 2080–2095. https://doi.org/10.1109/TIP.2007.901238

Gürsoy, D., De Carlo, F., Xiao, X., & Jacobsen, C. (2014). TomoPy: A framework for the analysis of synchrotron tomographic data. *Journal of Synchrotron Radiation*, *21*(5), 1188–1193. https://doi.org/10.1107/S1600577514013939

Mäkinen, Y., Marchesini, S., & Foi, A. (2021). Ring artifact reduction via multiscale nonlocal collaborative filtering of spatially correlated noise. *Journal of Synchrotron Radiation*, *28*(3), 876–888. https://doi.org/10.1107/S1600577521001910

Miqueles, E. X., Rinkel, J., O'Dowd, F., & Bermúdez, J. S. V. (2014). Generalized titarenko's algorithm for ring artefacts reduction. *Journal of Synchrotron Radiation*, *21*(6), 1333–1346. https://doi.org/10.1107/S1600577514016919

Münch, B., Trtik, P., Marone, F., & Stampanoni, M. (2009). Stripe and ring artifact removal with combined wavelet—fourier filtering. *Optics Express*, *17*(10), 8567–8591. https://doi.org/10.1364/OE.17.008567

Vo, N. T., Atwood, R. C., & Drakopoulos, M. (2018). Superior techniques for eliminating ring artifacts in x-ray micro-tomography. *Optics Express*, *26*(22), 28396–28412. https://doi.org/10.1364/OE.26.028396