

¹ BM3DORNL: High-Performance BM3D Denoising for Neutron Tomography

³ **Chen Zhang**  ^{1¶}, **Jean-Christophe Bilheux**  ², **Dmitry Ganyushin**¹, and **Pete Peterson**  ¹

⁵ 1 Computing and Computational Sciences Directorate, Oak Ridge National Laboratory, Oak Ridge, TN,
⁶ USA 2 Neutron Sciences Directorate, Oak Ridge National Laboratory, Oak Ridge, TN, USA ¶
⁷ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 12 January 2026

Published: unpublished

License

Authors of papers retain copyright
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](#)).¹⁶

¹⁵ 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 over 500× faster processing than comparable BM3D implementations while maintaining cross-platform compatibility including native Apple Silicon support.

²³ Statement of Need

²⁴ 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](#)).

³² 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.

39 State of the Field

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

45 [Figure 1](#) shows the benchmark input data: a synthetic sinogram (720×725 pixels) with
 46 simulated ring artifacts and the corresponding clean ground truth. All benchmark results
 47 are from Linux x86_64 to enable comparison with bm3d-streak-removal, which provides only
 48 x86_64 binaries. We compared eight methods: four BM3DORNL variants (streak, generic,
 49 multiscale, and Fourier-SVD), three TomoPy algorithms (wavelet-Fourier, sorting-fitting, and
 50 sorting-based), and the original bm3d-streak-removal. The multiscale variant implements
 51 the pyramid approach from Mäkinen et al. ([Mäkinen et al., 2021](#)), while Fourier-SVD is a
 52 lightweight FFT-guided method that achieves comparable quality at $50\times$ faster speeds.

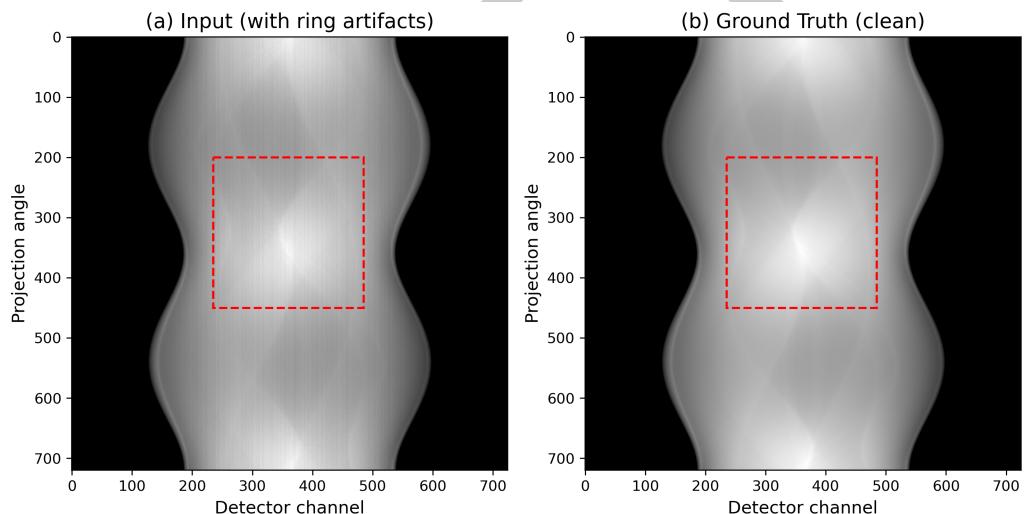


Figure 1: Benchmark input data. (a) Input sinogram with simulated ring artifacts. (b) Ground truth (clean sinogram). The dashed rectangle indicates the crop region shown in [Figure 2](#).

53 **Speed comparison:** [Figure 3\(a\)](#) shows processing times ($n=100$ runs on Linux x86_64).
 54 Fourier-SVD is the fastest method at 0.017 seconds, achieving $2300\times$ speedup over bm3d-
 55 streak-removal (39.9 seconds). The BM3D-based methods span a performance range: standard
 56 BM3DORNL processes sinograms in 0.078 seconds (511 \times faster than bm3d-streak-removal),
 57 while multiscale BM3DORNL takes 0.475 seconds (84 \times faster) due to pyramid processing.
 58 TomoPy methods achieve processing times of 0.28–0.35 seconds. Cross-platform performance
 59 differs substantially: TomoPy runs 2.5–3 \times slower on Apple Silicon than Linux x86_64 (likely
 60 due to unoptimized C extensions), whereas BM3DORNL runs at comparable speeds (within
 61 10%) across platforms due to its Rust backend.

62 **Quality analysis:** [Figure 2](#) shows cropped results from all eight methods with their difference
 63 images (result minus ground truth). The benchmark reveals that quality metrics alone do not
 64 tell the full story. TomoPy SF achieves the highest SSIM (0.987, see [Figure 3\(b\)](#)), but difference
 65 image analysis shows residual vertical stripes in its output. This occurs because SSIM uses
 66 local windows (7×7 or 11×11 pixels), and narrow vertical stripes affect few pixels per window.
 67 Notably, all BM3DORNL variants achieve excellent SSIM scores (0.943–0.951), with both
 68 multiscale BM3DORNL and Fourier-SVD reaching 0.951. Fourier-SVD accomplishes this at
 69 0.017 seconds, demonstrating that aggressive artifact removal and high-quality reconstruction

70 are achievable without computationally expensive multi-scale processing. The difference images
 71 reveal that all BM3DORNL variants produce clean vertical patterns, indicating successful
 72 artifact removal, while TomoPy methods leave faint residual artifacts.

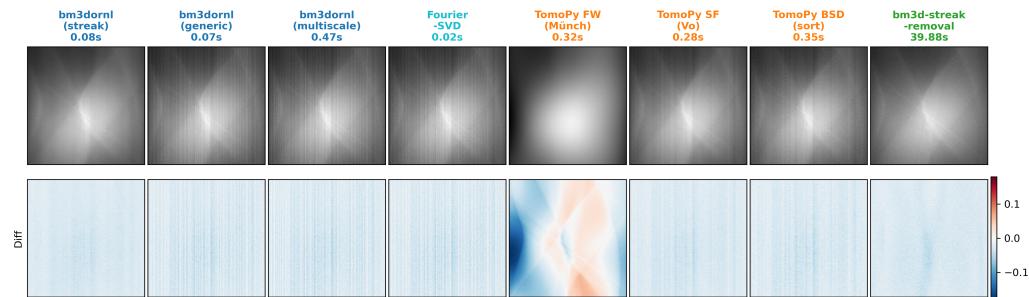


Figure 2: Method comparison. Top row: Cropped results from each method with processing times. Bottom row: Difference images (result minus ground truth) using a zero-centered RdBu colormap—blue indicates artifact removal, red indicates added artifacts. Data from Linux x86_64.

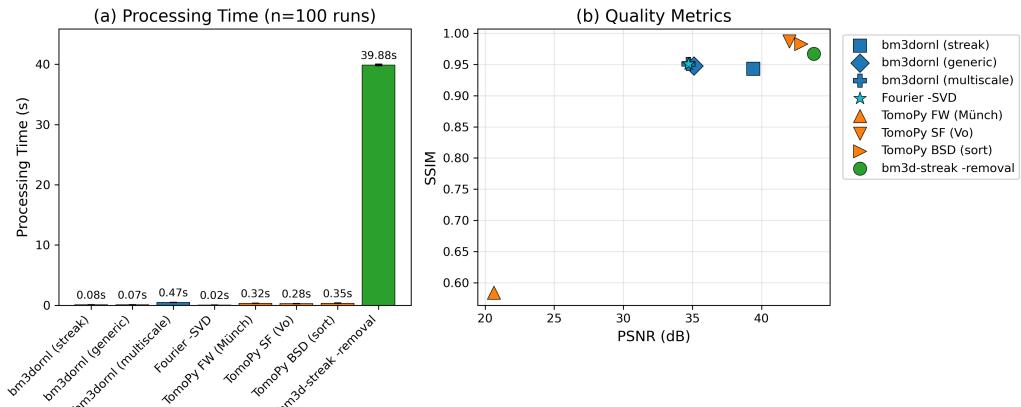


Figure 3: Performance metrics (Linux x86_64, n=100 runs). (a) Processing time comparison on linear scale, showing Fourier-SVD's 2300× speedup and standard BM3DORNL's 511× speedup over bm3d-streak-removal. (b) Quality metrics (PSNR vs SSIM) showing trade-offs between methods. BM3DORNL variants (blue/cyan), TomoPy (orange), bm3d-streak-removal (green).

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

77 Software Design

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

81 Key optimizations:

- 82 **Integral image pre-screening:** Before computing expensive patch distances, the block
 83 matching stage uses integral images to compute mean and norm bounds in O(1) time.
 84 Patches that fail these bounds are skipped, eliminating approximately 80% of distance
 85 calculations.

86 ▪ **FFT with SIMD acceleration:** BM3DORNL defaults to FFT-based transforms with
 87 batched row/column passes and specialized 8×8 fast paths. FFT plans are computed
 88 once and shared across threads via Arc, and a direct-mapped transform cache eliminates
 89 redundant computations. An alternative Walsh-Hadamard path is available for
 90 multiplication-free processing.

91 ▪ **SIMD-optimized block matching:** The 8×8 patch distance computation uses platform-
 92 specific SIMD instructions with a contiguous-memory fast path, reducing the block
 93 matching inner loop to near-hardware-limit throughput.

94 ▪ **Tile-owned aggregation:** Sinograms are processed in tiles that fit in L2/L3 cache, with
 95 SIMD-optimized weight accumulation. Per-worker scratch buffers are reused across
 96 kernel runs, minimizing memory allocation overhead.

97 The library provides a minimal Python API:

```
from bm3dornl import bm3d_ring_artifact_removal

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

98 **GUI application:** BM3DORNL includes a native GUI built with the egui framework for
 99 Rust, installable via pip install bm3dornl[gui] or Homebrew on macOS. The GUI enables
 100 interactive parameter tuning, side-by-side comparison with difference visualization, HDF5/TIFF
 101 file loading with dataset browsing, and real-time processing feedback with a fast-mode toggle.
 102 At 0.08 seconds per sinogram, scientists can explore parameter space interactively at over 12
 103 frames per second—something impractical with bm3d-streak-removal's 40-second processing
 104 time.

105 Research Impact

106 BM3DORNL is being integrated into processing pipelines at the VENUS and MARS beamlines
 107 at Oak Ridge National Laboratory. The library provides multiple performance tiers: Fourier-
 108 SVD enables real-time parameter exploration at 0.017 seconds per sinogram ($2300\times$ faster
 109 than bm3d-streak-removal), standard BM3DORNL offers robust denoising at 0.078 seconds
 110 ($511\times$ speedup), and multiscale BM3DORNL provides maximum quality at 0.475 seconds
 111 ($84\times$ speedup). For batch processing, a 1000-sinogram dataset that would take 11 hours
 112 with bm3d-streak-removal completes in 17 seconds (Fourier-SVD), 78 seconds (standard
 113 BM3DORNL), or 8 minutes (multiscale BM3DORNL).

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

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

¹²³ AI Usage Disclosure

¹²⁴ Generative AI tools (Claude) were used for code assistance and documentation drafting. All
¹²⁵ AI-generated content was reviewed, tested, and validated by the authors.

¹²⁶ Acknowledgements

¹²⁷ This research used resources at the Spallation Neutron Source, a DOE Office of Science User
¹²⁸ Facility operated by Oak Ridge National Laboratory. This work is supported by the U.S.
¹²⁹ Department of Energy under Contract No. DE-AC05-00OR22725.

¹³⁰ References

- ¹³¹ Aarle, W. van, Palenstijn, W. J., Cant, J., Janssens, E., Bleichrodt, F., Dabravolski, A.,
¹³² De Beenhouwer, J., Batenburg, K. J., & Sijbers, J. (2016). Fast and flexible x-ray
¹³³ tomography using the ASTRA toolbox. *Optics Express*, 24(22), 25129–25147. <https://doi.org/10.1364/OE.24.025129>
- ¹³⁵ Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. (2007). Image denoising by sparse 3-d
¹³⁶ transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8),
¹³⁷ 2080–2095. <https://doi.org/10.1109/TIP.2007.901238>
- ¹³⁸ Gürsoy, D., De Carlo, F., Xiao, X., & Jacobsen, C. (2014). TomoPy: A framework for
¹³⁹ the analysis of synchrotron tomographic data. *Journal of Synchrotron Radiation*, 21(5),
¹⁴⁰ 1188–1193. <https://doi.org/10.1107/S1600577514013939>
- ¹⁴¹ Mäkinen, Y., Marchesini, S., & Foi, A. (2021). Ring artifact reduction via multiscale nonlocal
¹⁴² collaborative filtering of spatially correlated noise. *Journal of Synchrotron Radiation*, 28(3),
¹⁴³ 876–888. <https://doi.org/10.1107/S1600577521001910>
- ¹⁴⁴ Miqueles, E. X., Rinkel, J., O'Dowd, F., & Bermúdez, J. S. V. (2014). Generalized titarenko's
¹⁴⁵ algorithm for ring artefacts reduction. *Journal of Synchrotron Radiation*, 21(6), 1333–1346.
¹⁴⁶ <https://doi.org/10.1107/S1600577514016919>
- ¹⁴⁷ Münch, B., Trtik, P., Marone, F., & Stampanoni, M. (2009). Stripe and ring artifact
¹⁴⁸ removal with combined wavelet—fourier filtering. *Optics Express*, 17(10), 8567–8591.
¹⁴⁹ <https://doi.org/10.1364/OE.17.008567>
- ¹⁵⁰ Vo, N. T., Atwood, R. C., & Drakopoulos, M. (2018). Superior techniques for eliminating
¹⁵¹ ring artifacts in x-ray micro-tomography. *Optics Express*, 26(22), 28396–28412. <https://doi.org/10.1364/OE.26.028396>