The Journal of Open Source Software

PyStack3D: A python package for fast image stack correction

Patrick Quéméré [1] **and Thomas David** [2]

1 Univ. Grenoble Alpes, CEA, Leti, Grenoble, France **2** Univ. Grenoble Alpes, CEA, LITEN, Grenoble, France

DOI: 10.21105/joss.07079

Software

- Review
- Repository
- Archive

Editor: Marcel Stimberg

Reviewers:

- @kasasxav
- @xiuliren
- @sklumpe

Submitted: 18 June 2024
Published: 23 September 2024

Summary

Three-dimensional reconstruction from 2D image stacks is a crucial technique in various scientific domains. For instance, acquisition techniques like Focused Ion Beam Scanning Electron Microscopy (FIB-SEM) leverage this approach to visualize complex structures at the nanoscale. However, creating a "clean" 3D stack often requires image corrections to remove artifacts and inconsistencies, particularly for volume segmentation, a crucial process for 3D quantitative data analysis.

Here we present `PyStack3D`, a Python open-source library, that aims at performing several image 'cleaning' tasks (Figure 1) in the most integrated and efficient manner possible.

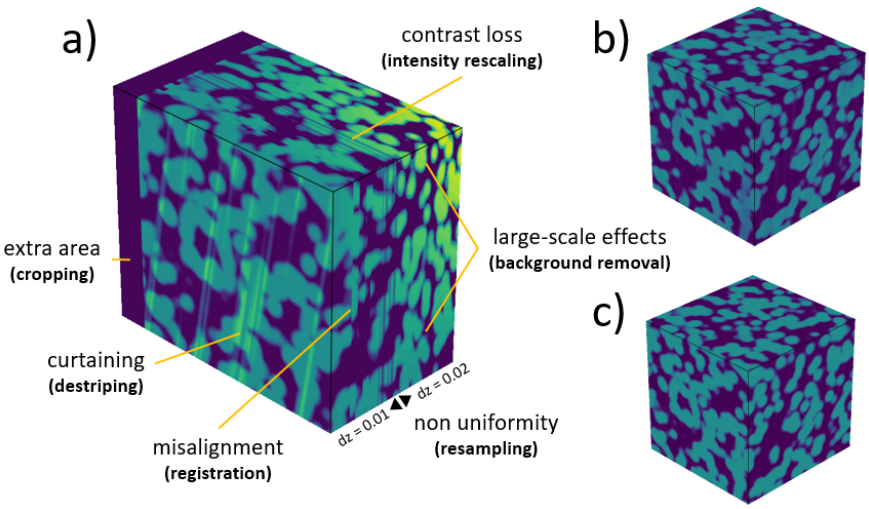

Figure 1: a) Synthetic stack with different types of defects and related processing. b) Corrected stack with `PyStack3D`. c) Ground truth.

Statement of need

Accurate 3D reconstruction is crucial for extracting detailed features across various imaging techniques. In **life sciences**, for instance, this includes identifying cellular organelles, understanding tissue architecture or studying protein localization. In **energy materials**, precise imaging is necessary for analyzing porous structures, mapping catalyst particles or assessing battery electrode interfaces. Various imaging methods, such as confocal microscopy, light sheet microscopy, and electron tomography, often introduce distortions or misalignments due to factors like optical aberrations, sample movement or inconsistent illumination. These issues

become even more pronounced with FIB-SEM (Höflich et al., 2023), where artifacts from the milling process and variations in sample preparation can further complicate the 3D stack.

Effective correction of these distortions is essential for reliable segmentation and accurate feature extraction (Osenberg et al., 2023; Spehner et al., 2020).

Statement of field

Certainly, one of the most widely used open-source software for performing image stack corrections is the Fiji software (Schindelin et al., 2012), a distribution of ImageJ (Schneider et al., 2012). Written in **Java**, this software offers numerous macros for the analysis and processing of 2D and 3D images. Unfortunately, not all the macros needed to perform the stack corrections exist, and the existing macros do not all support multiprocessing, which can lead to processing times of several hours for stacks composed of thousands of images (see Appendix).

As an alternative, codes written in **Python** like `Hifiem` (Kreinin et al., 2023), `PolishEM` (Fernandez et al., 2020) or `Napari` (Sofroniew et al., 2024) have been developed in recent years to achieve processing times of just a few minutes thanks to multiprocessing capabilities. `PyStack3D`, whose project started in 2020, is part of this trend. Designed to be executed as a workflow, `PyStack3D` aims to enable users to easily manage the automation of such workflows. With the quickly obtained results, users can easily readjust the parameters, and restart the processing if needed.

Implementation

In `PyStack3D`, to reduce the memory footprint, images (called "slices") are loaded and processed one by one either on a single processor or across multiple processors, depending on the user's machine capabilities.

The `PyStack3D` workflow is made up of multiple processing steps, specified in a `.toml` parameter file, and executed in the order desired by the user.

The processing steps currently offered by `PyStack3D` are:

- **cropping** to reduce the image field of view to the user's ROI (Region Of Interest)
- **background removal** to reduce, from 2D or 3D polynomial approximations, large-scaled brightness and contrast variations issued for instance from shadowing or charging effects in FIB-SEM images acquisition
- **intensity rescaling** to homogenize the 'gray' intensity distribution between successive slices and smooth out abrupt intensity jumps that can occur due to, for instance, variations in the beam source.
- **registration** to correct the image misalignment due to shifting, drift, rotation, ... during the image acquisition (based on the `PyStackReg` package, Thevenaz et al., 1998)
- **destriping** to minimize artefacts like stripes or curtain effects typically found in FIB-SEM images, based on the `PyVSNR` package (Fehrenbach et al., 2012; Pavy & Quéméré, 2024), or wavelet decomposition (Münch et al., 2009)
- **resampling** to correct non-uniform spatial inter-slice distances and enable correct 3D volume reconstructions
- **final cropping** to eliminate artefacts potentially produced near the edges during the image processing or to select another ROI at the end.

At the end of each process step, `PyStack3D` provides statistical profiles like evolution of minimum, maximum, and mean values for each slice, and relevant visualizations specific to the processing performed. In addition, 3D and 2D plots (cut-planes) akin to those shown in Figure 1 and Figure 2, respectively, can be produced.

Note that the processing can be carried out on multiple channels corresponding to images issued from multiple detectors, typically useful in the context of FIB-SEM input data. Moreover, when working with a Zeiss microscope, some metadata issued from the equipment can be automatically incorporated in the input `.toml` parameter file.

In conclusion, `Pystack3D` has been designed to evolve over time and accommodate new process steps. Its code structure has been crafted to seamlessly integrate new functionalities, leveraging multiprocessing capabilities.

Acknowledgements

This work, carried out on the CEA-Platform for Nanocharacterisation (PFNC), was supported by the "Recherche Technologique de Base" program of the French National Research Agency (ANR).

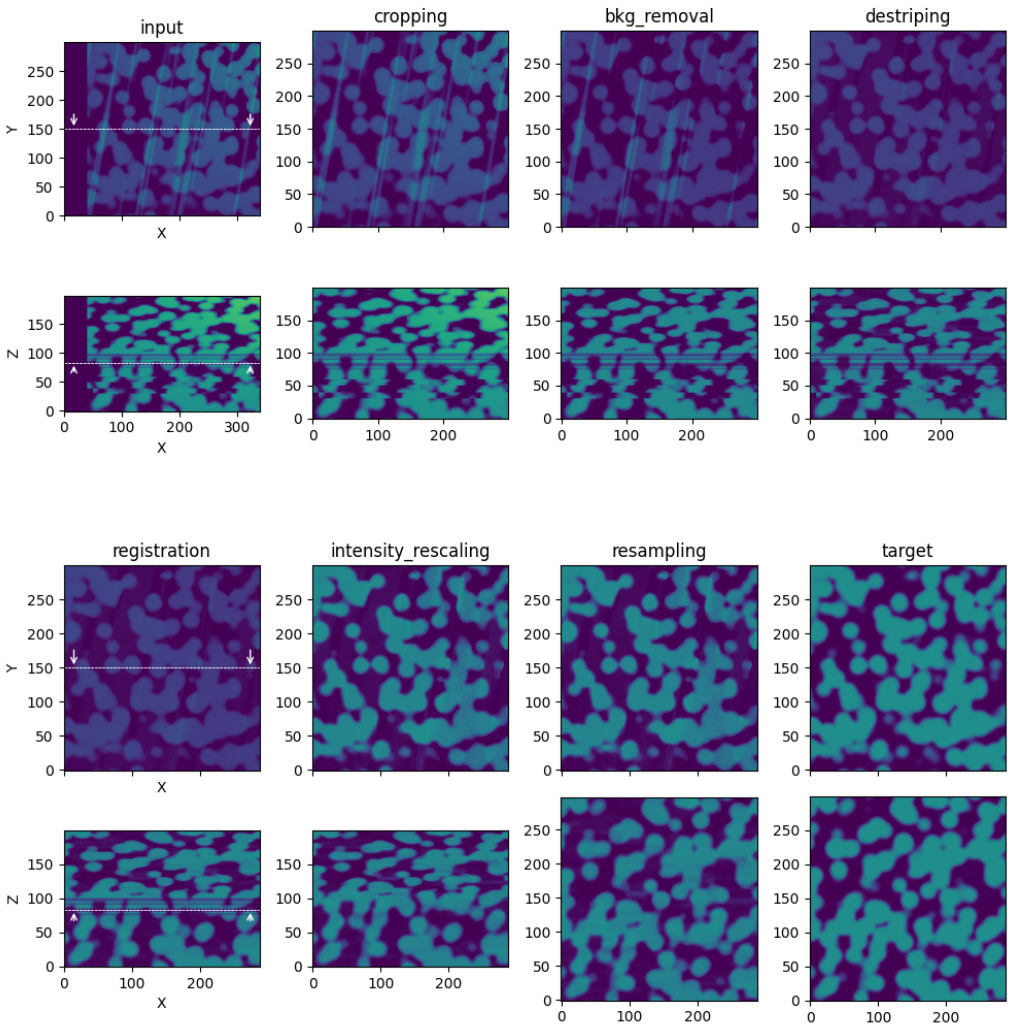

Figure 2: Cut-planes related to the different process steps applied to the stack presented in the Figure 1.

Appendix

Processing time for a stack composed of 2000 slices (from ex_real_stack_perf.py)

Process step	Fiji (s)	PyStack3D (s)
cropping	750	30
bkg_removal (2D / 3D)	250 / -	70 / 40*
destriping	22400	700**
registration	5400	25
intensity_rescaling	-	25
resampling	-	10

image size: 4224 x 4224 before cropping / 2000 x 2000 after cropping.

Machine: Linux - **32 CPUs** Intel(R) Xeon(R) Platinum 8362 CPU @ 2.80GHz.

(*) in 3D the polynomial coefficients are calculated only once, unlike in 2D, where the coefficients are recalculated for each slice.

(**) 120s with a GPU Nvidia A-100.

References

Fehrenbach, J., Weiss, W., & Lorenzo, C. (2012). Variational algorithms to remove stationary noise. Application to microscopy imaging. *IEEE Image Processing*, *21*(10), 4420–4430. https://doi.org/10.1109/TIP.2012.2206037

Fernandez, J.-J., Torres, T. E., Martin-Solana, E., Goya, G. F., & Fernandez-Fernandez, M.-R. (2020). PolishEM: image enhancement in FIB–SEM. *Bioinformatics*, *36*(12), 3947–3948. https://doi.org/10.1093/bioinformatics/btaa218

Höflich, K., Hobler, G., Allen, F. I., Wirtz, T., Rius, G., McElwee-White, L., Krasheninnikov, A. V., Schmidt, M., Utke, I., Klingner, N., & others. (2023). Roadmap for focused ion beam technologies. *Applied Physics Reviews*, *10*(4). https://doi.org/10.1063/5.0162597

Kreinin, Y., Gunn, P., Chklovskii, D., & Wu, J. (2023). High-fidelity image restoration of large 3D electron microscopy volume. *bioRxiv*. https://doi.org/10.1101/2023.09.14.557785

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

Osenberg, M., Hilger, A., Neumann, M., Wagner, A., Bohn, N., Binder, J. R., Schmidt, V., Banhart, J., & Manke, I. (2023). Classification of FIB/SEM-tomography images for highly porous multiphase materials using random forest classifiers. *Journal of Power Sources*, *570*, 233030. https://doi.org/10.1016/j.jpowsour.2023.233030

Pavy, K., & Quéméré, P. (2024). *PyVSNR 2.0.0* (Version 2.0.0). Zenodo. https://doi.org/10.5281/zenodo.10623640

Schindelin, J., Arganda-Carreras, I., & Frise, E. (2012). Fiji: An open-source platform for biological-image analysis. *Nature Methods*, *9*, 676–682. https://doi.org/10.1038/nmeth.2019

Schneider, C. A., Rasband, W. S., & Eliceiri, K. W. (2012). NIH image to ImageJ: 25 years of image analysis. *Nature Methods*, *9*(7), 671–675. https://doi.org/10.1038/nmeth.2089

Sofroniew, N., Lambert, T., Bokota, G., Nunez-Iglesias, J., Sobolewski, P., Sweet, A., Gaifas, L., Evans, K., Burt, A., & Doncila Pop, D. (2024). *napari: a multi-dimensional image viewer for Python* (Version v0.5.2). Zenodo. https://doi.org/10.5281/zenodo.13309520

Spehner, D., Steyer, A. M., Bertinetti, L., Orlov, I., Benoit, L., Pernet-Gallay, K., Schertel, A., & Schultz, P. (2020). Cryo-FIB-SEM as a promising tool for localizing proteins in 3D. *Journal of Structural Biology*, *211*(1), 107528. https://doi.org/10.1016/j.jsb.2020.107528

Thevenaz, P., Ruttiman, U. E., & Unser, M. (1998). A pyramid approach to subpixel registration based on intensity. *IEEE Transactions on Image Processing*, *7*(1), 27–41. https://doi.org/10.1109/83.650848