doi:10.1017/S1431927619002770
Microsc. Microanal. 25 (Suppl 2), 2019

Tomviz: Open Source Platform Connecting Image Processing Pipelines to GPU Accelerated 3D Visualization

Marcus D. Hanwell[1*], Christopher J. Harris[1], Alessandro Genova[1], Jonathan Schwartz[2], Yi Jiang[3] and Robert Hovden[2]

[1.] Kitware, Scientific Computing, Clifton Park, NY, USA.
[2.] University of Michigan, Materials Science, Ann Arbor, MI, USA.
[3.] Argonne National Laboratory, Advanced Photon Source Facility, Lemont, IL, USA.
* Corresponding author: marcus.hanwell@kitware.com

As the use of 3D tomography in materials science progresses, the detector densities increase, resolutions are improved, and the sheer volume of data collected increases, the community must develop innovative software platforms. The *tomviz* open source project has been developed over the past five years to offer a powerful image processing pipeline connected to a GPU accelerated 3D visualization engine that works on all major operating systems. The platform offers state-of-the-art reproducible processing pipelines developed collaboratively by Kitware software experts and University of Michigan researchers to address the needs of transmission electron microscopy tomography. Now *tomviz* is expanding to offer capabilities relevant in related fields of tomography (i.e. computed tomography). From the very beginning the emphasis has been on the development of a powerful image processing/reconstruction pipeline coupled with a highly accelerated 3D rendering engine featuring a user-friendly interface [1].

Tomviz provides a pipeline that runs inside the graphical user interface (GUI) in a background thread to enable long-running tasks without freezing the program (Fig. 1). Some operators in the pipeline are developed in C++, but most are implemented in Python that can be edited inside the program. The mechanism of embedding Python, operating on views of data from C++, and synchronizing with the graphical thread offer an intuitively simple interface that successfully abstracts many of the complexities of asynchronous task execution, GPU programming, and visualization of the results seamlessly. Once volumetric data is ready to be visualized, the rendering engine offers several visualization modules. These operate on the raw and/or processed data, with options to split the 3D view and lock cameras between two or more perspectives to facilitate comparisons. The hardware accelerated rendering engine uses code developed to execute on the graphics card to efficiently render large 3D volumes using multiple techniques including isosurfaces, volume rendering, slices, outlines and thresholds coupled with quantitative measurement modules such as scale cubes and rulers.

The pipeline offers a rich set of options to from Python operators to graphical controls from within the 3D visualization. Every action taken from reading in raw image stacks, executing the image processing tools, adding visualization modules and specifying lighting parameters can all be recorded in state files (Fig. 2a). This offers the capability of sharing all data, processing steps, and any editing in a transparent format based on the JSON standard in wide use today. Anyone can download the open source Tomviz program, state file and data to reproduce figures, make changes, and examine all steps offering a unique insight into research.

Recent work has been extending the in-application pipeline execution to offer out of process execution using Docker to aid in reproducibility. A pipeline runner abstracts out the differences between in and out of process execution, creating new possibilities for batch execution of pipelines once they have been validated on very large data sets along with improved reproducibility thanks to the Docker platform.

Going beyond simple execution of operations and visualization of the output has been challenging but was spurred on by the needs of researchers to "see" what was happening as long-running reconstructions proceed or as data is acquired. These "live" pipelines have required significant extension of the synchronization to offer programming interfaces to indicate when updated data is available. Figure 2b-e highlights this unique GUI feature for a reconstructed Co_2P nanoparticle [2]. The software infrastructure has been developed to then update the visualizations based on the updated data, and even re-run downstream pipeline elements. The ultimate goal is to offer real-time updates as operations proceed or data is acquired which must be balanced against the overhead of copying large data from the processing thread to the graphical thread [3].

References:

[1] BDA Leven et al., Microscopy Today **1** (2018), p. 12.
[2] H Zhang et al., Nano Letters, **11** (2011), p. 188.
[3] The authors acknowledge funding from the DOE Office of Science contract DE-SC0011385..

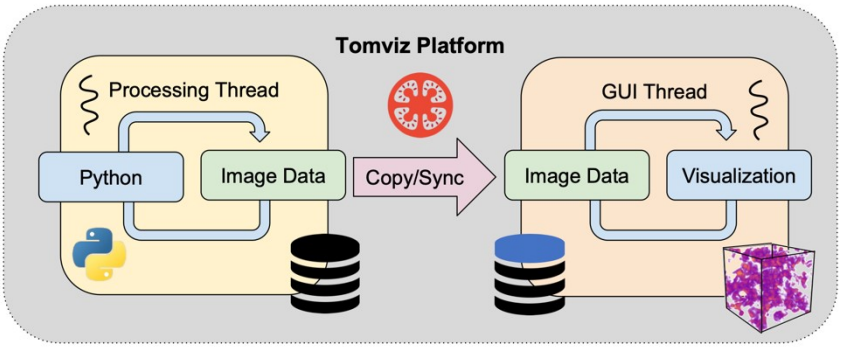

Figure 1. The internal architecture that simultaneously processes python scripts and visualizations in the GUI. As the python thread executes tomography reconstructions, data is concurrently transferred to the GUI thread to seamlessly update visualization(s) for the user.

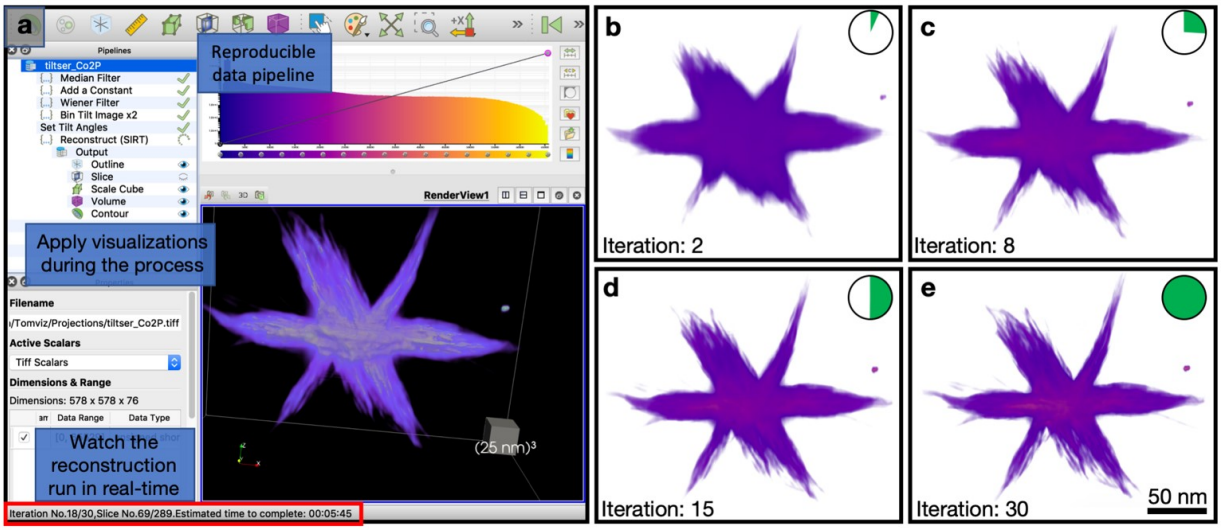

Figure 2. Demonstration of real-time, reproducible processes provided by *tomviz's* graphical interface. a**,** The tomviz platform demonstrating a real-time reconstruction. The pipeline tracks all preprocessing operations such as Wiener filtering. As the reconstruction proceeds, visualization modules can be seamlessly manipulated. b-e, Live volume rendering of a Co_2P nanoparticle reconstructed with Simultaneous Iterative Reconstruction Technique.

https://doi.org/10.1017/S1431927619002770