doi:10.1017/S143192761900134X
Microsc. Microanal. 25 (Suppl 2), 2019

Nion Swift: Open Source Image Processing Software for Instrument Control, Data Acquisition, Organization, Visualization, and Analysis Using Python.

Chris Meyer[1*], Niklas Dellby[1], Jordan A. Hachtel[3], Tracy Lovejoy[1], Andreas Mittelberger[1], and Ondrej Krivanek[1,2]

[1.] Nion R&D, 11511 NE 118th St., Kirkland, WA 98034, USA
[2.] Department of Physics, Arizona State University, Tempe, AZ 85287, USA
[3.] Center for Nanophase Materials Sciences, Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA
* Corresponding author: cmeyer@nion.com

Advances in imaging and spectroscopy techniques have introduced the ability to generate vast amounts of data as measured in time, space, and other conditions. Spectrum imaging, for example, can generate hundreds of gigabytes in minutes and 4D STEM diffraction can generate data even faster. The data typically includes related components such as HAADF images, EELS data, and instrumentation data. Simple visualization may include slicing data spatially at a specific energy, looking at an EELS spectrum integrated over an area, or selecting specific regions in diffraction pattern to access atomic scale potentials and electric fields from a 4D STEM diffraction data set. Processing and analysis stretch from simple tasks such as energy axis alignment and data splicing to more involved procedures such as principal component analysis, sparse data set in-painting, and automated quantification of typically noisy data sets. The visualization, processing, and analysis tools must be able to be shared, reported upon, and reused months or years later. To verify and trust this workflow, the tools must be readily available and transparent in their functionality. From initial conception to now, we have focused on this functionality and these priorities to develop Nion Swift [1].

As compared to other image processing applications, Nion Swift provides several unique capabilities. The main display area, a tiled workspace where images and plots are displayed, is easily configured for the workflow and is persistent, making it easy to return to later. In addition, Nion Swift tracks relationships between data. Data acquisition stores metadata defining the same acquisition parameters, reference frame, session, site, sample, and even location on the sample. It also includes a rich description of its NumPy-compatible data with time, collection, data, and element indexes. Finally, in addition to being able to import and export individual pieces of data, Nion Swift tracks sets of data and relationships, processing, analysis, and visualizations within them, storing them in a single portable file based on HDF5.

While Python is not optimized for speed, and there are issues with concurrency and utilization of multiple processors, Nion Swift has been engineered to move heavy duty processing into optimized libraries like NumPy, making performance comparable to equivalent software written in C++. The latency from data acquisition to display for a 2048x2048 image can be under 50 ms and the throughput can be as high as 20 frames per second. High performance drawing is achieved using a native Qt based UI which also utilizes multiple threads. The Python code maintains and coordinates the data acquisition and flow, storage, processing, analysis, and visualization, but leaves processing to multi-threaded, optimized Python libraries. Also, the Python ecosystem itself is continually improving performance and Nion Swift is positioned to quickly take advantage of these improvements. For even higher processing

speeds, Swift will be able interface to software running on graphics processing units (GPUs, e.g. [2]).

Since Nion Swift is a foundation on which other projects can be built, we have put considerable effort into ensuring that a stable application programming interface (API) is available for plug-in package developers. The goal is to allow plug-in packages to run without modification for several years and to minimize maintenance requirements for plug-in packages that have an even longer lifetime. We plan to accomplish this by publishing an API with a well defined deprecation policy, using standard Python libraries such as NumPy and SciPy where possible, using a strict versioning policy, and making nearly everything open source and published on GitHub for review.

As an example of Nion Swift processing, Fig. 1 shows 4D STEM diffraction data set with associated field magnitude and direction calculations (top row), an EELS SI data set with associated background subtracted spectrum calculation (middle row), and an FFT with mask and simulated EELS multi-acquire (bottom row).

More information about Nion Swift is available at [3].

References:
[1] C.E. Meyer et al., Microsc. Microanal. 20 (Suppl 3, 2014) 1108-1109.
[2] R.S. Pennington, F. Wang and C.T. Koch, Ultramicroscopy 141 (2014) 32-37.
[3] http://nion.com/swift

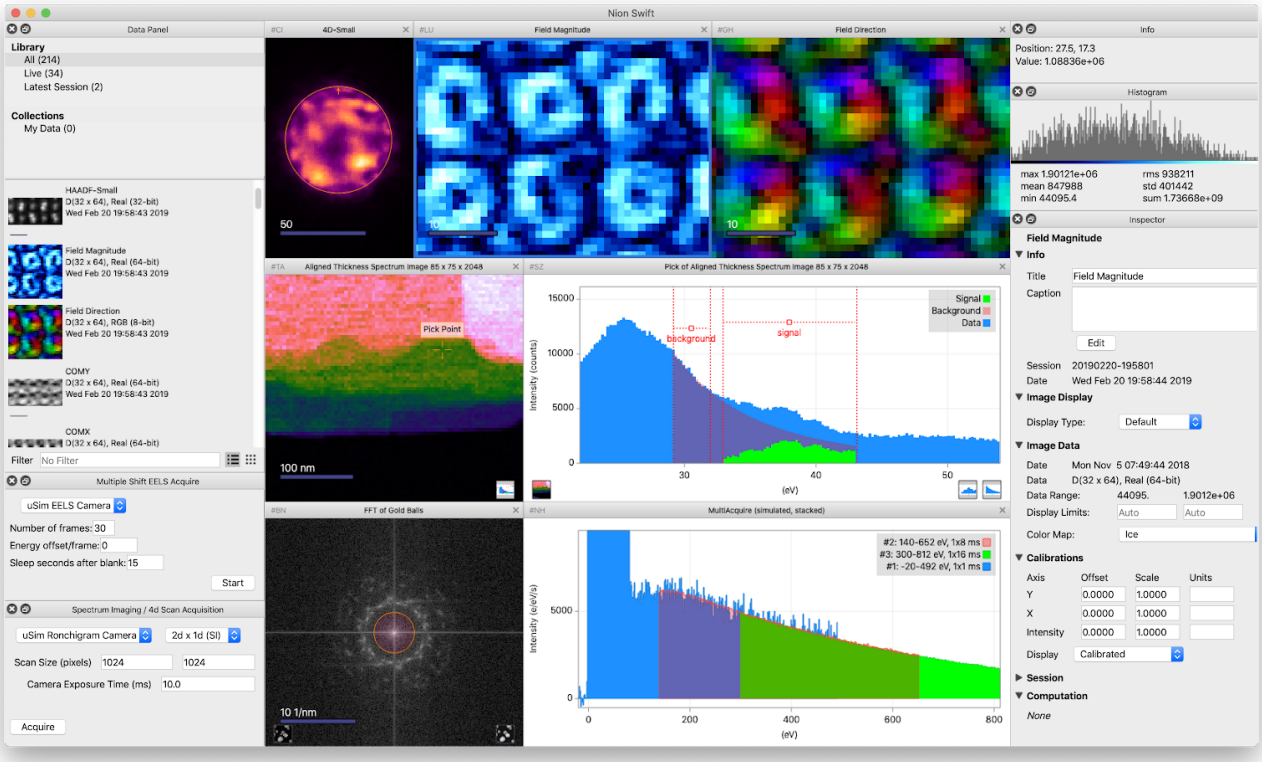

Figure 1. Nion Swift screenshot showing library, data panel, workspace with images and line plot, histogram, and inspectors.

https://doi.org/10.1017/S143192761900134X Published online by Cambridge University Press