doi:10.1017/S1431927614007272
Microsc. Microanal. 20 (Suppl 3), 2014

Using Nion Swift for Data Collection, Analysis and Display

C.E. Meyer[1], N. Dellby[1], Z. Dellby[1], T.C. Lovejoy[1], M.C. Sarahan[1], G.S. Skone[1], and O.L. Krivanek[1]
[1] Nion Co., 1102 8th St., Kirkland, WA 98033, USA

Nion Swift is an open-source software platform for the collection, processing, quantification, visualization, and management of scientific data. Built on the popular Python programming language [1], Swift is written from the ground up to be extensible at every level. Although initially developed as software for controlling Nion electron microscopes and collecting and processing data from them, Swift has been designed to serve as a resource for the scientific community across a range of applications. It includes:

- Open source code based on Python
- Instrument control and data collection
- Data visualization, processing, and analysis
- Data management, storage, and retrieval based on metadata
- Multi-platform support: Windows and Mac OS presently, Linux in the future

When developing data collection, processing, and other modules, Nion plans to use the same APIs as those accessible by everyone else. Almost all capabilities available to Nion will also be available to others. The guiding idea is that if the software is available to the broad scientific community to develop further, it will advance more rapidly and become generally more useful.

Some base-level instrument control functions are protected and not fully accessible to outside users, so that users cannot, for instance, run a defective software module that vents the electron microscope column while the beam is on. However, controls needed for data collection, such as those relating to the microscope electron-optical modes, beam position, etc., are all available.

Swift builds on an ever-growing library of Python tools such as NumPy and SciPy [2]. Since it is open source, users can examine the data workflow down to the source code level. All data is stored in standard formats such as TIFF or Python "pickle". Internally, Swift uses Python classes and, at a lower level, NumPy arrays. Use of standard formats and data structures ensures that data collected in Swift can be used interchangeably with Python and other libraries.

Swift is designed to track data from collection to presentation, including relationships between data. All data is tagged with metadata such as details of the sample, the current user, and the specifics of the instrument. If an elemental map is produced from a spectrum image, Swift remembers from what larger data set the elemental map was produced, even if the elemental map is transferred to a colleague on another computer or network. It is then possible to return to the original data for adjustments if needed.

Processing in Swift can be performed "live", so that the user can inspect the end results during an experimental session, and, if needed, make immediate adjustments to the experimental parameters. If an elemental map is degraded due to incorrect gain normalization of the EELS detector or poor signal-to-noise ratio, the problem can be noticed and rectified in the middle of the session. The user can add new processing with Python and have multiple "live" processing and analysis items such as histograms, statistics, FFTs, or elemental maps, all visible during data collection.

The data storage and retrieval capabilities are likewise fast and automated. Searching for that perfect image from last month is designed to be as easy as searching for an email. Swift provides powerful capabilities for sorting and filtering collected data. Data can be located by session, sample, user, and other metadata. The metadata system can be extended using Python and fully integrated into the user interface.

A community of Swift users is now contributing collection, processing and visualization modules to the software, a process that is helped by a well-defined submission procedure and guidelines for documenting the contributions. Examples of contributed modules will be shown at the meeting.

Fig. 1 shows a screenshot of Swift acquiring a high-angle annular dark field (HAADF) STEM image at 60 keV, and Fourier-filtering it live. The Fourier filter is a double Gaussian one described in [3], and it can be adjusted by using the sliders shown in the bottom right corner, or by typing in new values of (σ_1, σ_2 and W_2. The left side of the window is used for navigating the available data items, while specific information about the selected data item is provided on the right side. The center area can be configured in many different ways to visualize data effectively. More information about Swift is available at [4].

[1] http://www.python.org/
[2] http://www.numpy.org/ and http://www.scipy.org/
[3] O.L. Krivanek et al., Ultramicroscopy **110** (2010) 935-945.
[4] http://nion.com/swift/
[5] We are grateful to Prof. P.E. Batson for the use of the Nion UltraSTEM HERMES at Rutgers U.

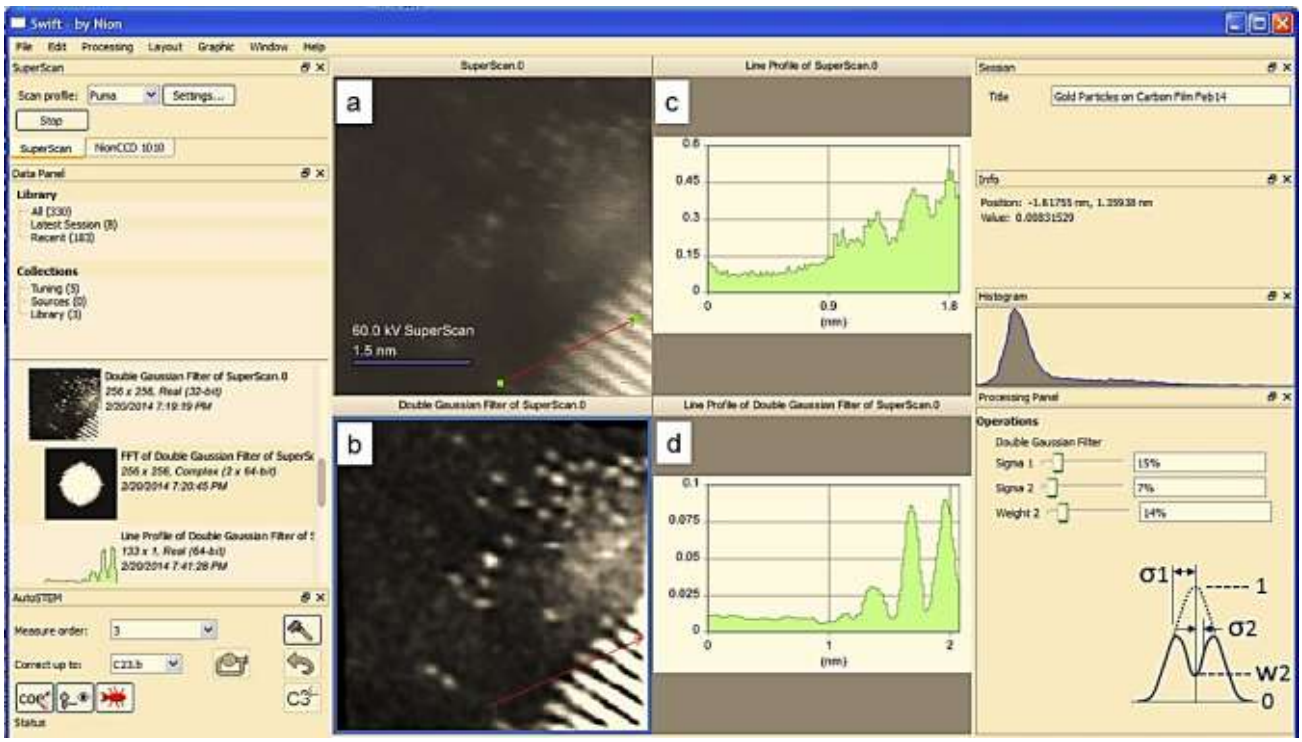

Fig 1. a) Swift acquiring a STEM HAADF image of the edge of a gold particle and of single Au atoms, b) Fourier-filtering it live with the filter shown at lower right (σ_1 = 15% f_N, σ_2 = 7% f_N, W_2 = 0.14, where f_N is the Nyquist freq.), c) live line profiles through (a), d) live line profile through (b).

https://doi.org/10.1017/S1431927614007272 Published online by Cambridge University Press