doi:10.1017/S143192761800329X
Microsc. Microanal. 24 (Suppl 1), 2018

Dragonfly as a Flexible Platform for Interpreting and Processing Hyperspectral and other High-dimensional Images

Nicolas Piche[1], Francis Cote[1], Eric Yen[1], Mike Marsh[2]

[1.] Object Research Systems. Montreal, Canada.
[2.] Object Research Systems. Denver, USA

High-dimensional imaging experiments produce results that are data-rich but frustratingly challenging to interpret. Those challenges arise, in large part, due to the lack of easy software tools to display and properly analyze any dataset whose domain spans more than four dimensions. We describe here our extensions to the Dragonfly platform to make the visualization and quantitative analysis of high-dimensionality images user-friendly and accessible for non-experts.

Ordinary imaging records a single signal at spatially discrete positions and encodes that digitized signal as a scalar value at every position on a regularly sampled spatial array. Here, we use the terms high-dimensional image, spectrum image, and hyperspectral image interchangeably to describe imaging experiments where the spatial array may be 2D, 3D, or 4D (3D-space plus time), but where instead of capturing a single scalar value at every spatial position, a broad signal array is recorded. That signal array can be an energy spectrum as in the cases of energy electron loss spectroscopy (EELS) or energy dispersive x-ray spectroscopy (EDS). Or that signal might be a 2D array as in electron backscattered diffraction (EBSD) or ptychography. These experiments have become more routine, but the available processing software is often arcane and has limited functionality.

First we deal with hyperspectral visualization. Normal image visualization techniques display pixels and use color to convey signal intensity for the rendered image. The most naive, but still useful, solution for spectral images is to display a conventional 2D or 3D spatial rendering of a single component of the signal array but provide the user an interface for selecting which component is rendered. Similar to visualizing a 3D image by displaying a single 2D Z-slice and letting the user drag a slider for Z-position, visualizing EDS or EELS spectral images is easily done by letting the user have a slider for the energy-axis; we call this spectral slicing. A slightly more advanced solution is to define an interval of spectral components (e.g. an energy window) over which the signal is integrated; we refer to this as spectral windowing. A third solution is to perform a principal components analysis (PCA) over the spectral dimension for all pixels, take the first eigenvector as a set of coefficients, and then sum the signal over all components and render the weighted sum; the last approach is a highly efficient way of reducing a multi-dimensional domain into a single-dimensional axis of greatest variance. All three of these techniques are nicely implemented in the Cornell Spectrum Imager [1]. What all of these solutions have in common is taking a spectral array signal and reducing it to single scalar value.

We generalize the techniques described above by defining a software pattern. In Dragonfly, any Python function that takes a signal array as input and returns a scalar value can be used for spectral reduction. We implement all three of the above solutions within the constraints of this pattern but also document the platform so others can implement their own spectral reduction techniques. In the case of spectrum slicing, only one slider is required in the user interface. For spectral windowing, two sliders easily define the lower and upper bounds on the spectrum axis; it is common to also employ spectral background characterization and correction for this kind of windowing technique, but we describe this elsewhere. For reduction by PCA, the only user input required is which eigenvector (e.g. first, second, etc.) to use for the coefficients. This same pattern is easily extended for signals that are comprised of 2D arrays. As an example we show here a spectral reduction interface that presents the user a 2D image and lets the user manually paint a binary pattern which defines a non-contiguous 2D integration "window."

Second we address hyperspectral analysis. A typical solution for analysis is to provide a user interface positioning a zero-dimensional probe in the spatial domain and displaying a 1D plot of the spectral signal for that position. We provide the same, but we also support 2D plots for 2D signal arrays, and we provide for 2D, 3D, and 4D averaging probes in additional to zero-dimensional probes.

Each spectral reduction described earlier is dynamic so users can continuously vary the free parameters of the Python function and visualize the results in real time. But those dynamic renderings are made persistent when the user names the output as a new image channel in the Dragonfly workspace. Thereafter, it's accessible like any other normal imported 2D, 3D, or 4D image; each spectral reduction, therefore, behaves like virtual experimental image. For example, the user might process spectral data in order to yield multiple chemical channels and overlay them for visualization or feed them into integrative segmentation tools such as Dragonfly's convolutional neural network filters or machine-learning Segmentation Trainer [2].

Strengths of Dragonfly include its Python functionality and its integration of image visualization, image segmentation, and quantitative analysis tools for multiple image channels. By tying these features together with flexible hyperspectral reduction and analysis, we establish a unified solution that is easy to use and extend. We support the import of high-dimensional datasets encoded in the open pycroscopy file format [3] and expect access to Dragonfly to complement that effort for standardizing hyperspectral data processing.

References:

[1] P Cuerva *et al*, Microscopy Today **21** (2013), pp 40-45.
[2] N Piche *et al.*, Microsc Microanal **23** (2017), Suppl 1, pp. 246-7.
[3] S Jesse *et al.*, Scientific Reports **6** (2016), No: 26348, pp 1-8.