1172
doi:10.1017/S1431927620017213
Microsc. Microanal. 26 (Suppl 2), 2020

Data Processing Using Python in DigitalMicrograph

Benjamin Miller[1] and Stephen Mick[2]

[1]Gatan, Inc., Gilbert, Arizona, United States, [2]Gatan, Pleasanton, California, United States

Processing and visualizing data after a transmission electron microscopy (TEM) experiment is an essential step toward generating scientific conclusions. DigitalMicrograph or Gatan Microscopy Suite (GMS) is a commonly used software for collecting electron microscopy data, since it controls all Gatan cameras, spectrometers, and detectors [1]. GMS also has extensive capabilities for visualizing and processing collected data, but since methods for data processing are constantly changing many techniques still cannot be applied using the standard user interface. To supplement the user-interface with additional flexibility and power, GMS has always had scripting capabilities (typically called DM-scripts [1]). One great advantage of scripting within GMS is that the results of scripted acquisition or processing can then be visualized and manipulated using the standard user interface. However, the scripting language has been GMS-specific. While this means that it is uniquely tailored to the needs of microscopists, it also has limited the use of other open-source libraries within GMS during TEM data collection. Gatan has now integrated Python scripting into GMS adding the capability to edit and run Python (v 3.7) code directly from the scripting interface starting in GMS 3.4.0 [2]. With an integrated Python API, GMS enables a Python code to interact with all the native Gatan objects such as images, windows, tags, and even cameras and microscopes. Python libraries and packages can also be imported and used, unlocking endless possibilities for capture, processing, visualization, and analysis during and after data collection. There are several ways in which Python scripting embedded in GMS is superior to using GMS and Python separately. These include live processing and visualization, analysis at the microscope, taking advantage of large datasets/data-streams, scripted data acquisition, and reduced data import/export and conversion. This presentation will provide examples of each of these possibilities. While detailed processing and visualization may take place long after data collection, it can be beneficial to perform initial processing and visualization rapidly during a session on the microscope. This initial analysis enables users to determine whether good data is being collected, the experimental conditions need to be modified, or additional data collection is needed. Enabling the user to do this assessment while the experiment is still in progress can save time, especially for in-situ experiments since the next experimental step at the microscope is often decided by detecting and interpreting dynamic changes happening in the sample. An example of a processing function which has been applied to the live view is one in which the Python code extracts a 1D profile from the image as a function of time and presents this as a kymograph. This has been applied to the data in Figure 1 where the profile shown in part A is measured live as the data is played back. The resulting kymograph is shown in part F, where a clear boundary with slope representing the average growth rate can be observed. A similar, but more complicated radial profile kymograph can be applied to diffraction data (or a diffractogram) over time to observe changes in crystallinity [3]. GMS already has many built in capabilities for data processing and visualization, but most data analysis (which typically results in a simple plot or a single numerical value) is still performed either manually using simple tools, or using more sophisticated tools and techniques implemented in ImageJ, MATLAB, Python, etc. Almost always, this analysis takes place after the user is no longer using the microscope to collect data, sometimes days, weeks, or months later. Python scripting in GMS means that advanced data analysis on at least a subset of the data can be performed at the microscope. An example of this is shown in Figure 1, where dendrite growth during a single second is measured from 75 frames using an image segmentation technique implemented with functions from scikit-image. The input data (Figure 1 A and B) was noisy, because it was acquired with only 5.7 e/pix/frame, so it was pre-processed within the same script. The

segmented result for selected frames is shown in Figure 1C, and 1D additionally gives a colored overlay on top of the filtered last frame. Figure 1 E shows the resulting measurement where both the length and area of the dendrite are plotted over time. On a PC with an Intel Core i7 8850H @ 2.6 GHz, this took < 10 s, or 0.4 s per million image pixels. Modern cameras like the K3 IS used to collect the 1 second video in Figure 1, produce large volumes of data. It is not uncommon for a single dataset to be tens or hundreds of GB, and to generate over a TB of data in a single TEM session. It is not practical to manually analyze such a large amount of data to produce scientifically meaningful results. For this reason, the large volume of data has the potential to be a burden. However, given the increasing accessibility of machine learning and artificial-intelligence-based techniques for data processing and analysis, the large raw data volume can also be an advantage. An example of this is given in Figure 2, where non-negative matrix factorization has been applied to a 4D STEM dataset to find several components. One of these components (Figure 2 B) clearly shows precipitates in the matrix which are difficult to see in the raw data. Integration of Python scripting into GMS enables powerful processing and analysis algorithms to be implemented on data during and immediately after acquisition. It also enables expanded visualization and even scripted acquisition. Since Python is a widely adopted scripting language [4], its integration into GMS makes TEM data processing accessible to far more researchers. Powerful open-source tools that are readily implemented with Python libraries can be applied to rapidly answer questions at the microscope.[5]

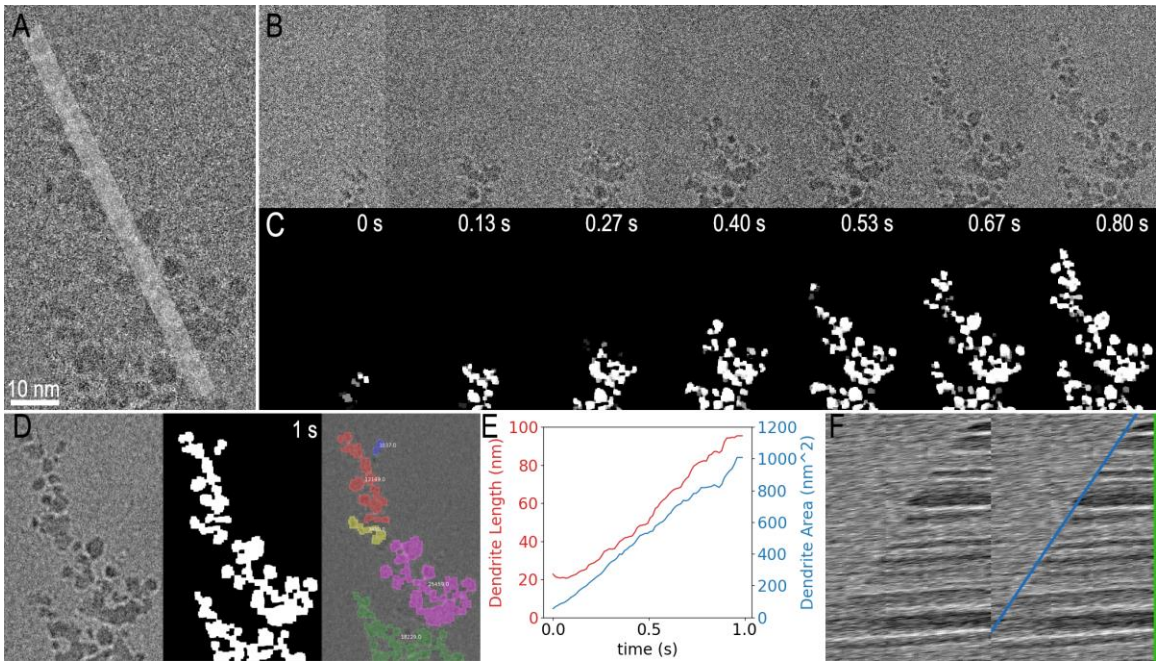

Figure 1. Processing and Analysis at the Microscope. Data from a 1s, video captured with K3 IS counting camera. Raw data from final frame is shown in A) along with the profile used to produce the kymograph in F). A selection of the 75 raw frames, are shown in B), with the corresponding segmented images in C). The final (filtered) frame is shown in D, along with the corresponding segmented frame and a color overlay on the filtered frame of connected segmented areas. The final measured values for dendrite length and area as a function of time are given as a plot in E). F) shows the kymograph produced from the profile in A) sampled over time, as well as the same kymograph with a blue line corresponding to the average linear growth rate and a vertical green line indicating which profile was calculated from the frame in A)

https://doi.org/10.1017/S1431927620017213 Published online by Cambridge University Press

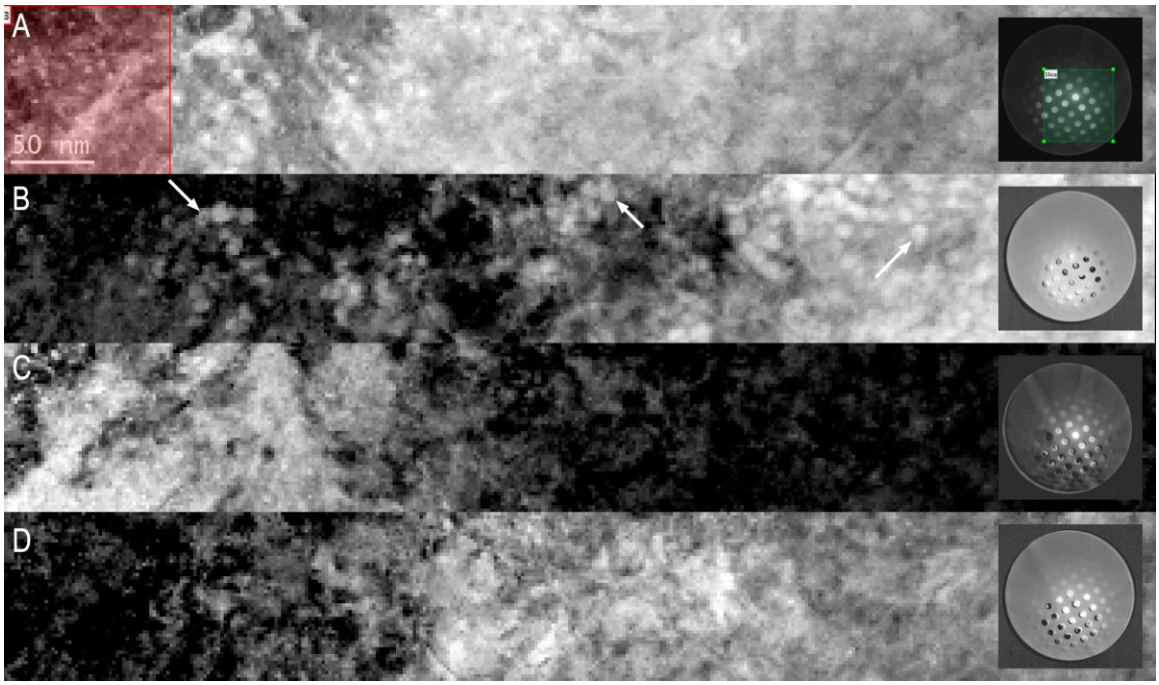

Figure 2. Processing Large Data Volumes. A view in GMS of a 4D dataset collected with a K3 IS counting camera is shown in A), where a sum over the diffraction patterns in the red region is shown in the inset. The green area in the inset gives the area of the diffraction patterns which is summed to yield the intensity in the image. B-D show maps based on fitting the data with components (insets) found using non-negative matrix factorization. 10 components were generated, but 3 are displayed. The map generated with the component in B) makes precipitate particles visible, as indicated by arrows. C) and D) show components which predominate in the thin and thick regions of the sample respectively.

References

[1] Schaffer, Bernhard. (2016). Digital Micrograph. In Transmission Electron Microscopy Diffraction, Imaging, and Spectrometry, Carter, Barry, Williams, David B. (Eds.), pp. 167-196. Springer International Publishing.

[2] https://www.gatan.com/installation-instructions

[3] Miller, B., Mick, S. (2019). Real-Time Data Processing using Python in DigitalMicrograph. Microscopy and Microanalysis, 25.S2, 234-235.

[4] https://www.tiobe.com/tiobe-index Accessed Feb 2020

[5] Data Courtesy of: Xiaobing Hu, Kun He, Paul Smeets, Roberto dos Reis, and Vinayak Dravid, Northwestern University. This work made use of the EPIC, Keck-II, and SPID facilities of Northwestern University's NUANCE Center, which has received support from the Soft and Hybrid Nanotechnology Experimental (SHyNE) Resource (NSF ECCS-1542205); the MRSEC program (NSF DMR-1720319) at the Materials Research Center; the International Institute for Nanotechnology (IIN); the Keck Foundation; and the State of Illinois, through the IIN.