Pycroscopy: Software ecosystem for scientific data ingestion, analytics and visualization

Suhas Somnath[1], Rama Vasudevan[1], Maxim Ziatdinov[1], Debangshu Mukherjee[1], Rajiv Giridharagopal[2] and Gerd Duscher[3]

[1]Oak Ridge National Laboratory, [2]University of Washington at Seattle, [3]University of Tennessee at Knoxville

github.com/pycroscopy

Introduction

Data-driven scientific discovery necessitates the development of appropriate software, tools, and infrastructure that connect scientific instruments with computational resources. Most scientific domains are facing similar challenges highlighted below:

- **Growing data sizes** -> need computing to scale to compute clusters
- Increasing **data complexity** and multiple (**proprietary**) **data formats** -> need open, general, origin-agnostic standards for data models & formats
- **Disjoint communities duplicating** software development **efforts** -> need open, general, multi-disciplinary, interoperable, and extensible software
- **Expensive** / **incapable commercial software** -> need transparency to logic and algorithms, applying cutting edge algorithms and artificial intelligence to domain sciences, and provide software free of cost.
- **Accessibility** -> need to make complex analyses and technologies user-friendly to the average researcher and keep code well documented with examples to foster / encourage next generation of developers.

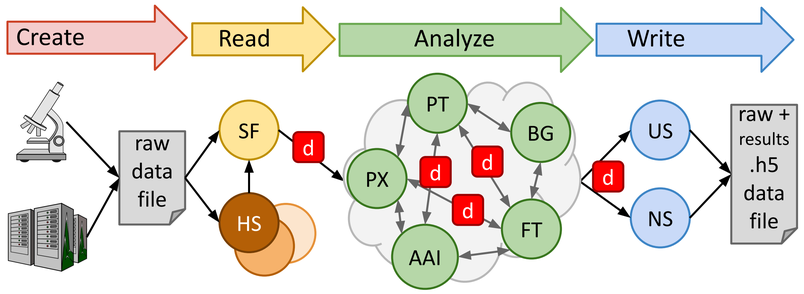
These factors warrant a concerted effort across disciplines to develop, share, and make available software tools to researchers. Here we present an ecosystem of open-source and ***interoperable python packages*** for reading, storing, analyzing and visualizing scientific data along with multiple general models for representing scientific data.

Packages

- **Scientific applications packages**
  - Transmission Electron Microscopy (TEM)
    - `pyTEMlib` (PT) - Physics model-based TEM data quantification
    - `stemtool` - Analysis of scanning TEM (STEM) data
  - Artificial Intelligence - deep and machine learning learning
    - `AICrystallographer` (AC)- Segmentation and classification of atomically resolved images of crystalline materials
    - `AtomAI` (AAI) - Deep learning tools for classification, segmentation and regression of microscopy imaging data
  - Atomic Force Microscopy (AFM) -
    - `BGlib` (BG) - Analysis of Band Excitation and General-Mode data
    - `FFTA` (FT) - Fast free transient analysis of AFM frequency data
  - `Pycroscopy` (PX) - Domain-agnostic scientific data analytics
- **General science packages**
  - `SciFiReaders` (SR) - Read information from instrument data files
- **Data Infrastructure**
  - Abstract data representation models
    - N-Dimensional Spectroscopy and Imaging Data (NSID) - For data with N-dimensional form
    - Universal Spectroscopy and Imaging Data (USID) - For data with or without N-dimensional form
  - `pyNSID` (NS) - Python interface to NSID HDF5 files
  - `pyUSID` (US) - Python interface to USID HDF5 files
  - `sidpy` - General plotting, data storage, and management

Philosophy

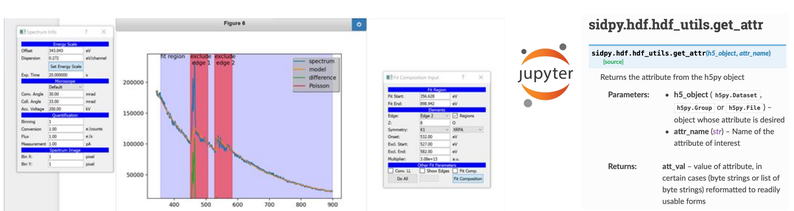
- **Modular** - Family of interoperable and extensible packages.
- **Consistency** - Packages exchange information with each other via `sidpy.Dataset` objects (d) which are wrappers over `dask.Array`
- **Accessible** - Low barrier for entry for users and developers - single line to create `Dataset` objects. Datasets are familiar, intuitive, scientific numeric arrays like numpy. Interactive Jupyter Notebooks and web applications for disseminating scientific narratives and pipelines. Packages are well documented and come with tutorials.
- **Standardized** - `NSID` and `USID` are open standards for representing data regardless of origin, dimensionality, and size. Algorithms are not tied to specific instruments or modalities.
- **Scalable** - Data are written to Hierarchical Data Format (HDF5) which scale from kB to TB and beyond. Pycroscopy packages are built on Dask, mpi4py, PyTorch, Keras, which can use multiple CPUs and GPUs
- **Traceable** - Raw data, intermediate data, and even figures can be written to the same data file and linked to capture provenance



Data from measurements or simulations are read into `sidpy.Dataset` (d) objects directly by `SciFiReaders` (SF), or through external packages like `HyperSpy` (HS). Data are processed using multiple science packages in the Pycroscopy ecosystem that interoperate via `Dataset` objects. `Dataset` objects are written to HDF5 files via `pyUSID` (US) or `pyNSID` (NS).

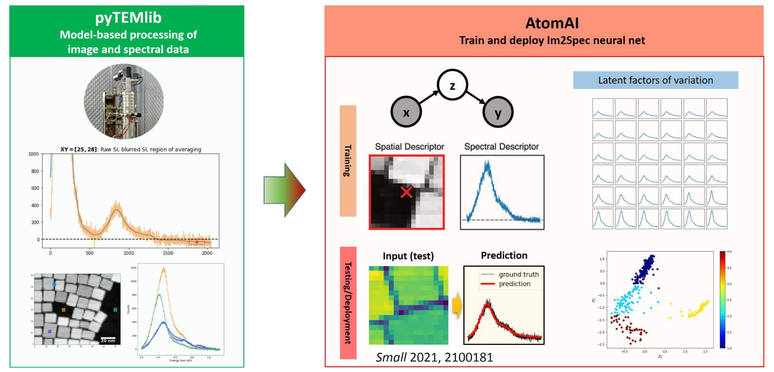
Getting Started

- Learn more about each package by visiting - github.com/pycroscopy
- Read our rich and extensive documentation
- Install any package via pip. e. g. - `pip install pyNSID`
- Try out our many interactive Jupyter Notebooks (example below)
- Join our team at our weekly hackathons on Fridays at 3 PM (Eastern)



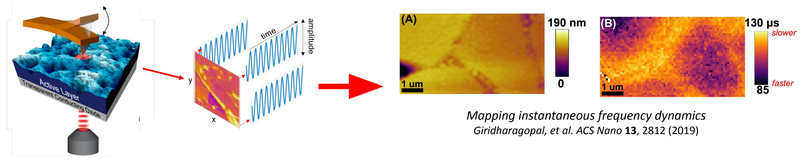
Pycroscopy Enabled Science

Deep Learning for Electron Microscopy



First, **pyTEMlib** performs image alignment, background subtraction, and basic data inspection. Then **AtomAI** uses the processed data to train the *Im2Spec* neural net for predicting spectra from images and analyzing structure-property relationships.

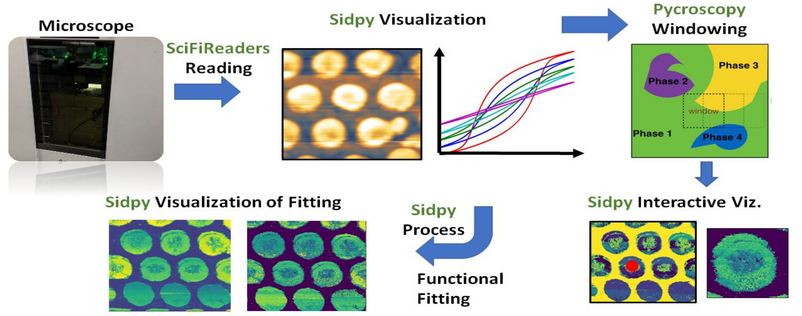
Parallel Processing for Optical Mapping



Mapping instantaneous frequency dynamics
Giridharagopal, et al. ACS Nano **13**, 2812 (2019)

The `FFTA` package uses `pyUSID` to process all acquired spectra in parallel in addition to writing the raw and processed data into a `USID` formatted HDF5 file.

Signal Processing Workflow



Spectral imaging data visualized using `.plot()` method of `sidpy.Dataset`. Functional fitting of signals in parallel via `sidpy.Process` (leveraging `Dask` framework underneath). Texture analysis ivia `pycroscopy`'s image windowing.

Acknowledgements

Pycroscopy was developed at the Center for Nanophase Materials Sciences, which is a DOE Office of Science User Facility. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the US DOE. We are grateful for the many contributors to Pycroscopy