



December 16th, 2024

Ecosystems for Microscopy

Microscopy Hackathon

Rama Vasudevan¹

¹Center for Nanophase Materials Sciences, Oak Ridge
National Laboratory, Oak Ridge TN



U.S. DEPARTMENT OF
ENERGY

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY



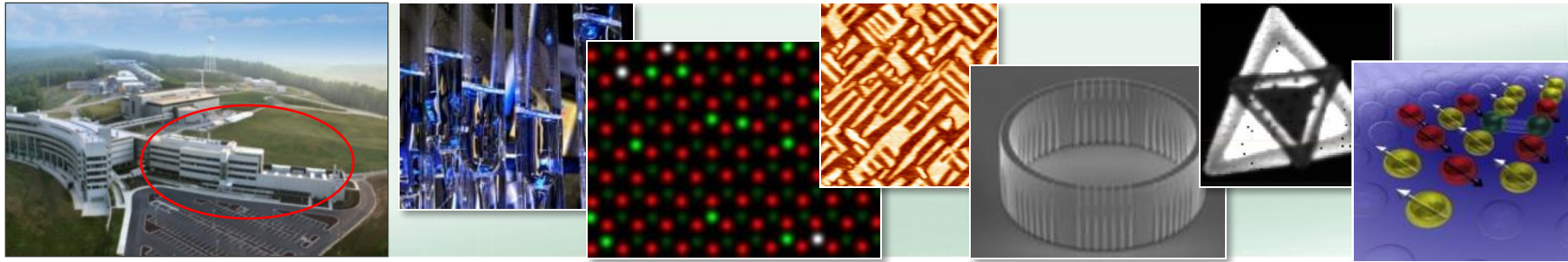
CNMS is a national user facility with a mission to advance nanoscience

- About CNMS:

- Unlike many user facilities, you don't need to have samples to apply for time
- Two calls per year for continuous access; anytime for short-term projects
- Simple 2-page proposal
- Free access to laboratories, equipment and expertise if you agree to publish
- Proposal deadlines: early May and mid-October
- Joint proposals with neutron sources (SNS, HFIR)

- Research areas:

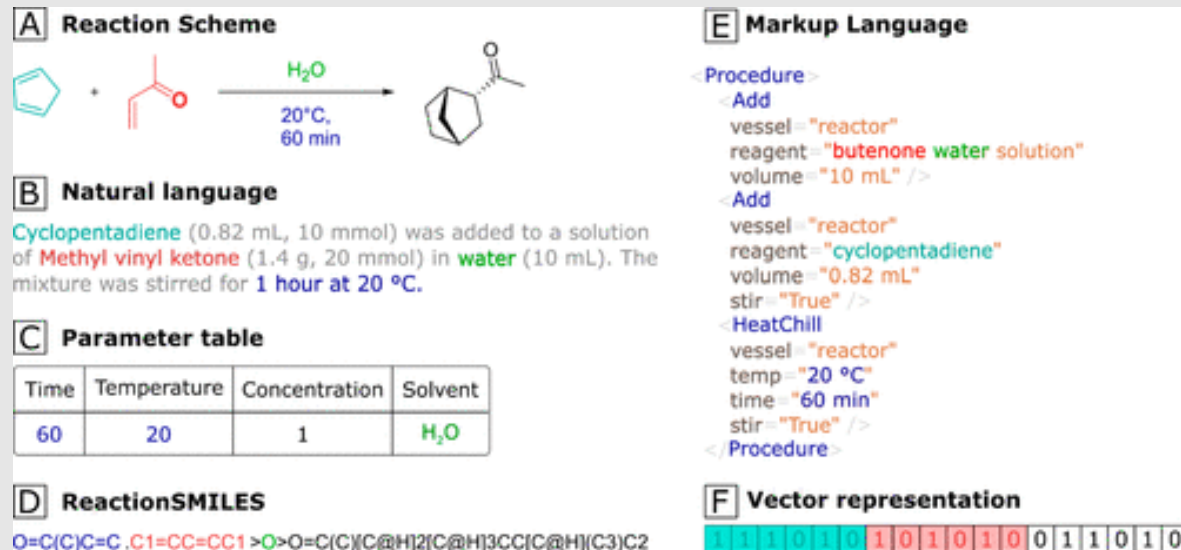
- **Synthesis** – 2D, precision synthesis, selective deuteration
- **Nanofabrication** – direct-write, microfluidics, cleanroom
- **Advanced Microscopy** – AFM, STM, aberration-corrected TEM/STEM, atom-probe tomography
- **Functional Characterization** – laser spectroscopy, transport, magnetism, electromechanics
- **Theory and Modelling** – including gateway to leadership-class high performance computing



CNMS is a Nanoscale Science Research Center supported by the U.S. Department of Energy, Office of Science, Scientific User Facilities Division

Abstractions make the (automated) world go round

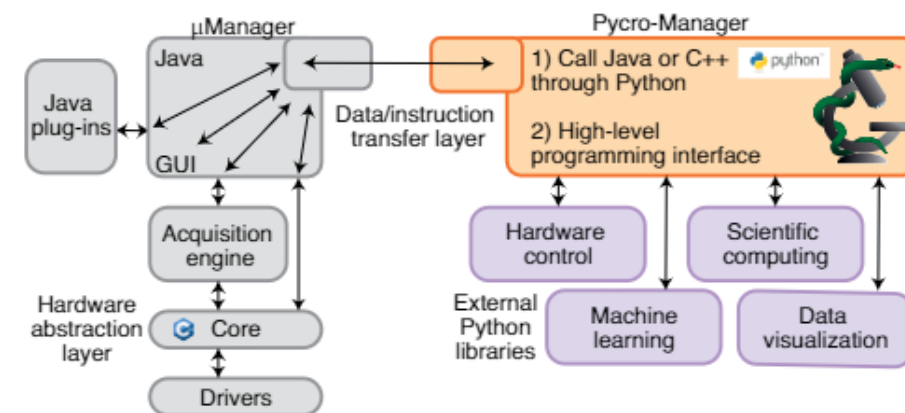
“Chemputation” – Digital chemistry



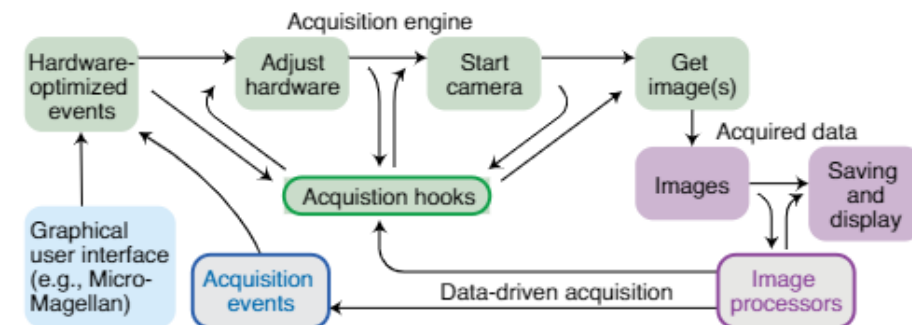
Hammer, Leonov, Bell and Cronin, JACS 1, 1572 (2021)

A complete programming language for chemistry that can run on open hardware

PycroManager



b Pycro-Manager high-level programming interface



Pinkard, et al. Nat. Methods. 18, 226 (2021)

Systems for automated microscopy

Software Infrastructure



Welcome to AECroscopy

Get Started

Get Started

Experiment

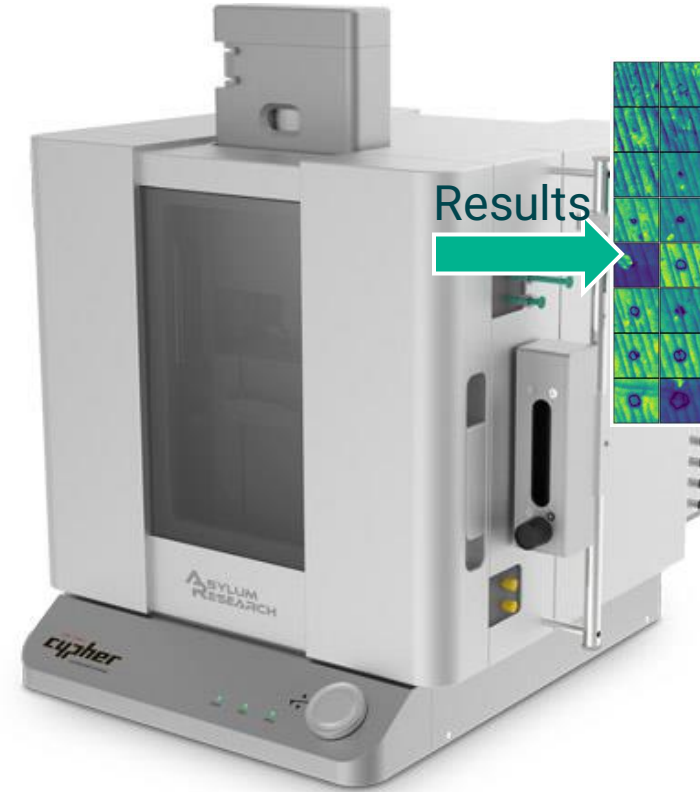
Experiments

Step 4. Do a BEPFM at the whole experiment area

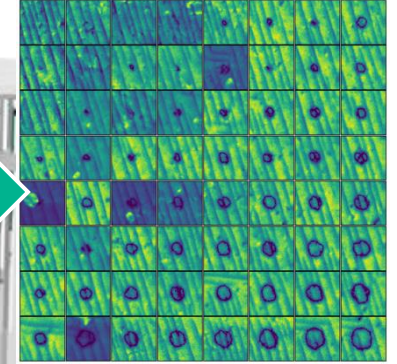
```
dset_pfm, dset_chns, dset_cs = newexp.raster_scan(raster_params_dict = {"scan_pixel": 32, "scan_y_start": -1.0, "scan_y_stop": 1.0},
                                                file_name = "pfm_whole", ploton = False)

f, (ax1, ax2, ax3, ax4, ax5, ax6) = plt.subplots(1, 6, figsize = (30, 5), dpi = 100)
ax1.imshow(dset_pfm[:, :, 0])
ax2.imshow(dset_pfm[:, :, 1])
ax3.imshow(dset_pfm[:, :, 2])
ax4.imshow(dset_pfm[:, :, 3])
ax5.imshow(dset_chns[0, :, :])
ax6.imshow(dset_chns[1, :, :])
plt.show()
```

Deploy



Results



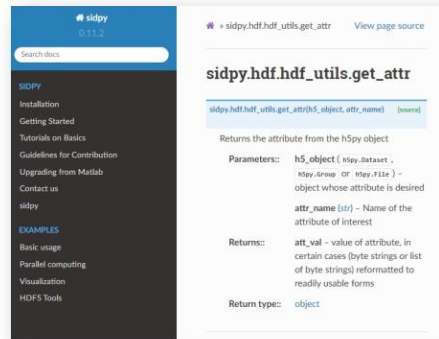
Y. Liu et al. Small Methods 2301740 (2024)



An ecosystem for microscopy data ingestion, analytics and visualization



pycroscopy
A general-purpose package for microscopy imaging and spectroscopy data analytics, including registration, image cleaning, unmixing, etc.



Vasudevan et al. Advanced Theory and Simulations 6, 2300247 (2023)



- Standardized data model
- In-built processing and viz utilities

scifireaders
For ingesting a variety of microscopy files for output to sidpy dataset objects

pyusid
Python package for reading and visualizing our universal spectral imaging dataset format

pynsid
Python package for reading and visualizing our N-dimensional spectral imaging dataset format

sidpy
Python utilities for storing, visualizing and fitting Spectroscopic Imaging Data

bglib
Utilities to analyze, fit and visualize Band - Excitation and G - mode imaging data primarily for CNMS SPM Users

atomai
Deep learning toolkit for analysis of atomically resolved imaging and spectroscopy datasets

stemtools
Python based codes for analysis of 4D-STEM and aberration - corrected vanilla STEM datasets

pytemlib
Python tools for simulation, registration, analysis and visualization of TEM datasets

Pycroscopy



An ecosystem for microscopy data ingestion, analytics and visualization



pycroscopy

A general-purpose package for microscopy imaging and spectroscopy data analytics, including registration, image cleaning, unmixing, etc.

scifireaders

For ingesting a variety of microscopy files for output to sidpy dataset objects

pyusid

Python package for reading and visualizing our universal spectral imaging dataset format

pynsid

Python package for reading and visualizing our N-dimensional spectral imaging dataset format

sidpy

Python utilities for storing, visualizing and fitting Spectroscopic Imaging Data

bglib

Utilities to analyze, fit and visualize Band - Excitation and G - mode imaging data primarily for CNMS SPM Users

atomai

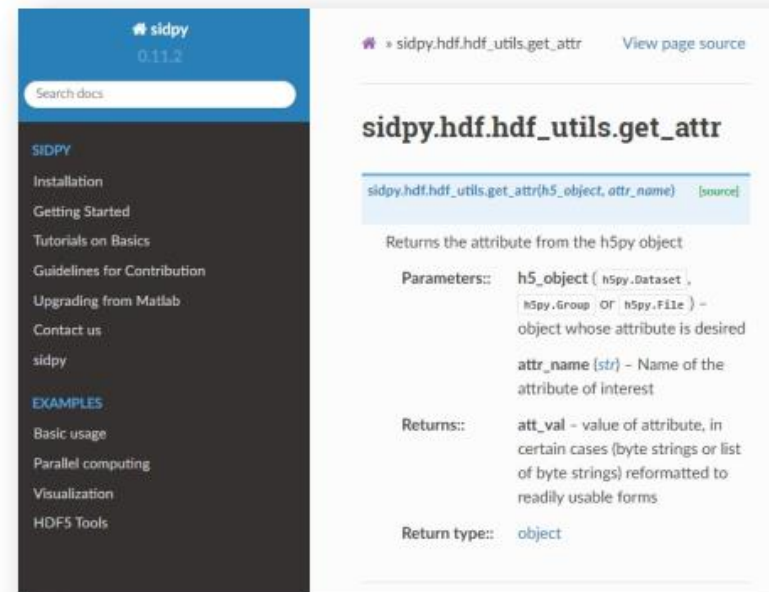
Deep learning toolkit for analysis of atomically resolved imaging and spectroscopy datasets

stemtools

Python based codes for analysis of 4D-STEM and aberration - corrected vanilla STEM datasets

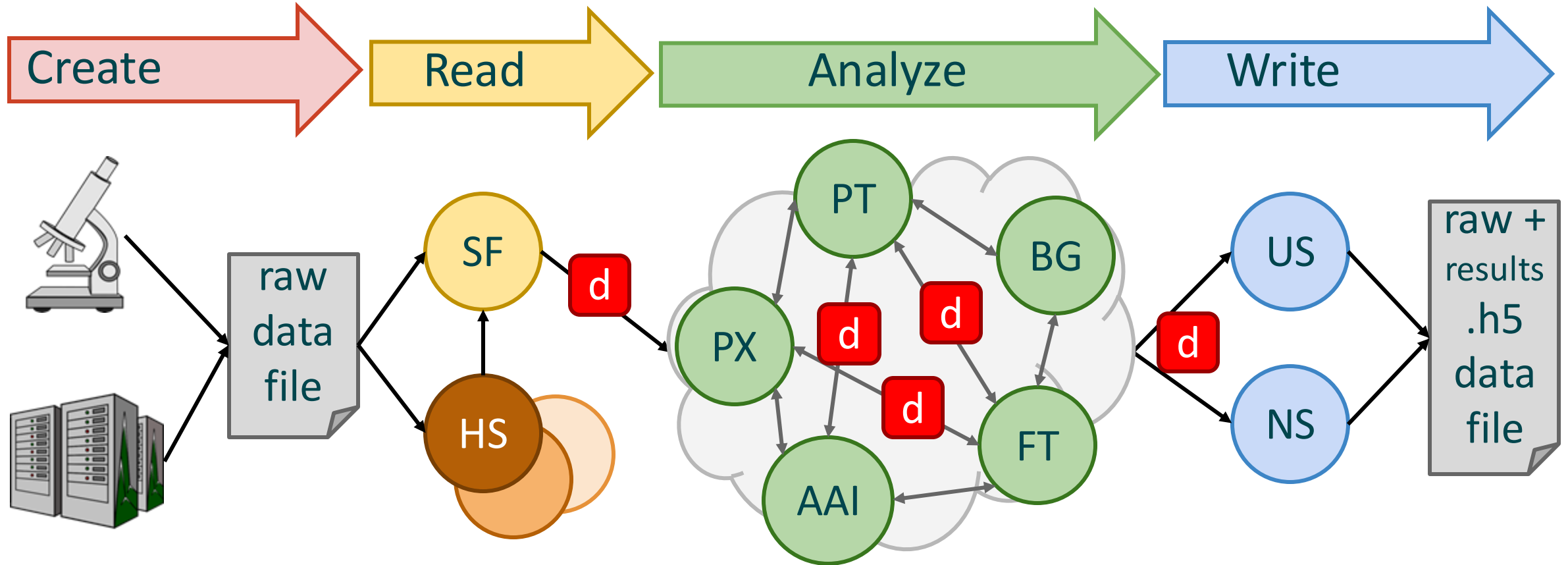
pytemlib

Python tools for simulation, registration, analysis and visualization of TEM datasets



[Github.com/pycroscopy](https://github.com/pycroscopy)

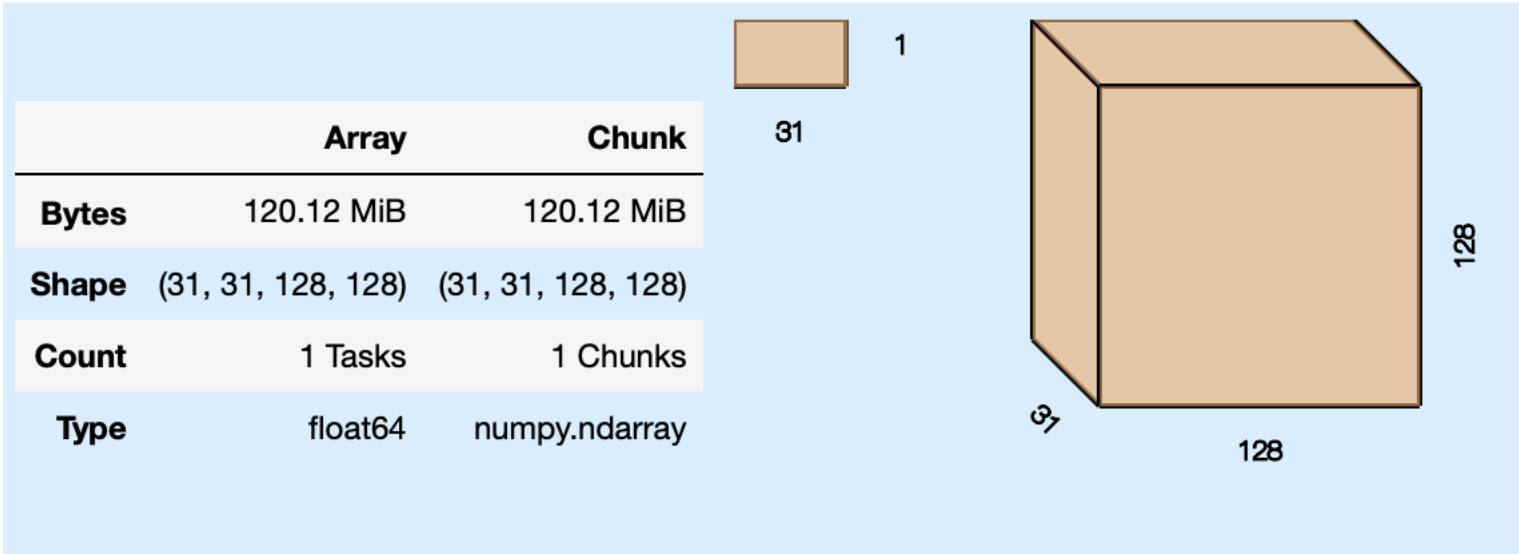
Pycroscopy philosophy



Data from measurements or simulations are read into `sidpy.Dataset` (d) objects directly by **SciFiReaders** (SF). 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).

NSID Model (implemented as sidpy.Dataset)

Dataset Object built on top of dask arrays

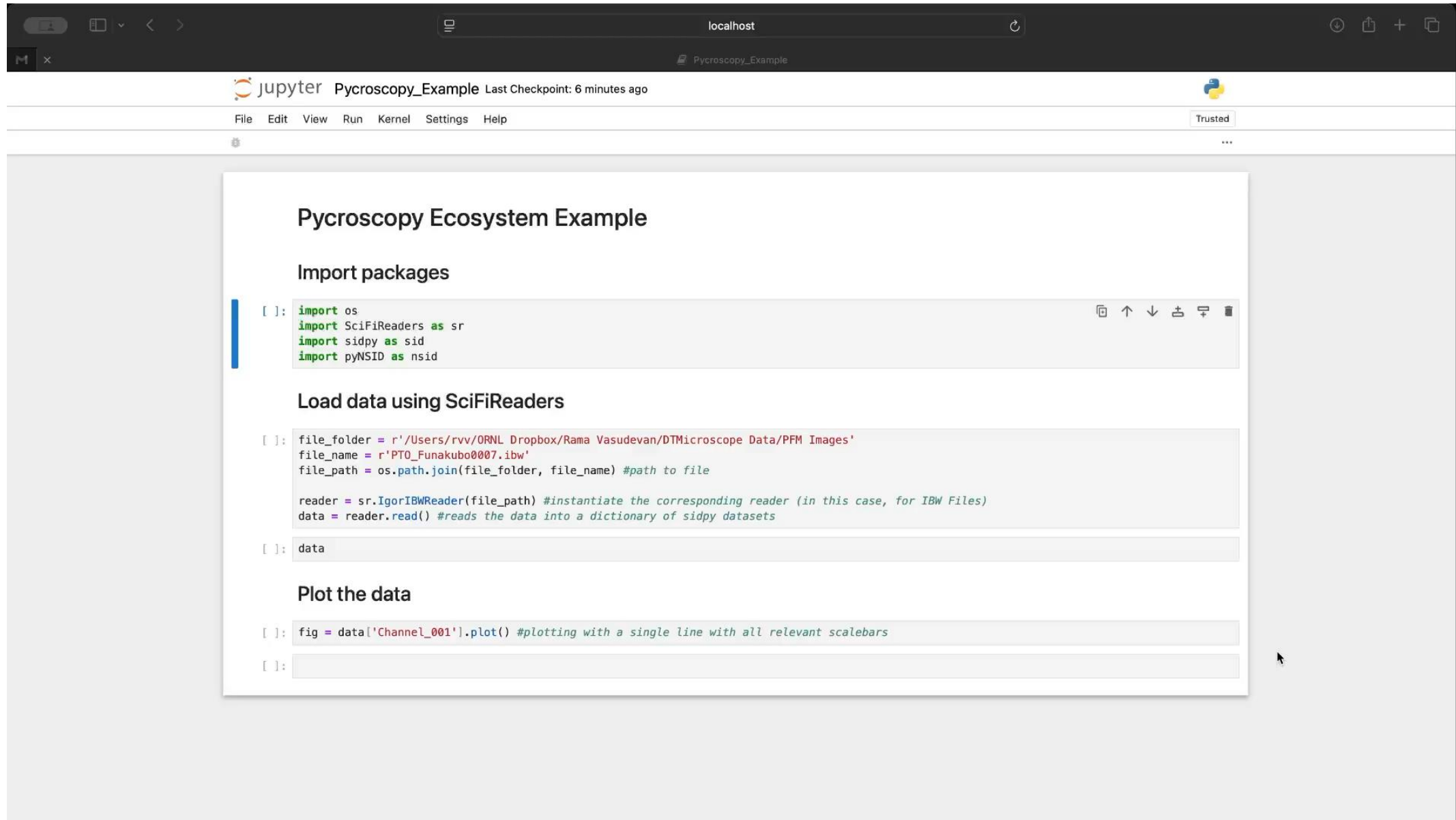


Benefits of the model:

- Easy to understand
- sidpy takes care of plotting (dataset.plot())
- Can easily perform parallel computations
- Easy to push to file including metadata
- Useful for data pipelining

- Maintain N-dimensional form
- All of the advantages of dask (large sizes, parallel compute)
- Additional data given for each dimension of dataset, such as name, quantity, units
- Metadata stored in dictionary
- Can be readily pushed to hdf5 files

Example: Load data into a sidpy dataset and plot it



The screenshot shows a Jupyter Notebook titled "Pycroscopy_Eample" running on a "localhost" browser. The notebook has a menu bar with "File", "Edit", "View", "Run", "Kernel", "Settings", and "Help". A "Trusted" button is visible in the top right. The notebook content is divided into three sections: "Import packages", "Load data using SciFiReaders", and "Plot the data".

Pycroscopy Ecosystem Example

Import packages

```
[ ]: import os
import SciFiReaders as sr
import sidpy as sid
import pyNSID as nsid
```

Load data using SciFiReaders

```
[ ]: file_folder = r'/Users/rvv/ORNL Dropbox/Rama Vasudevan/DTMicroscope Data/PFM Images'
file_name = r'PT0_Funakubo0007.ibw'
file_path = os.path.join(file_folder, file_name) #path to file

reader = sr.IgorIBWReader(file_path) #instantiate the corresponding reader (in this case, for IBW Files)
data = reader.read() #reads the data into a dictionary of sidpy datasets
```

[]: data

Plot the data

```
[ ]: fig = data['Channel_001'].plot() #plotting with a single line with all relevant scalebars
```

```
[ ]:
```

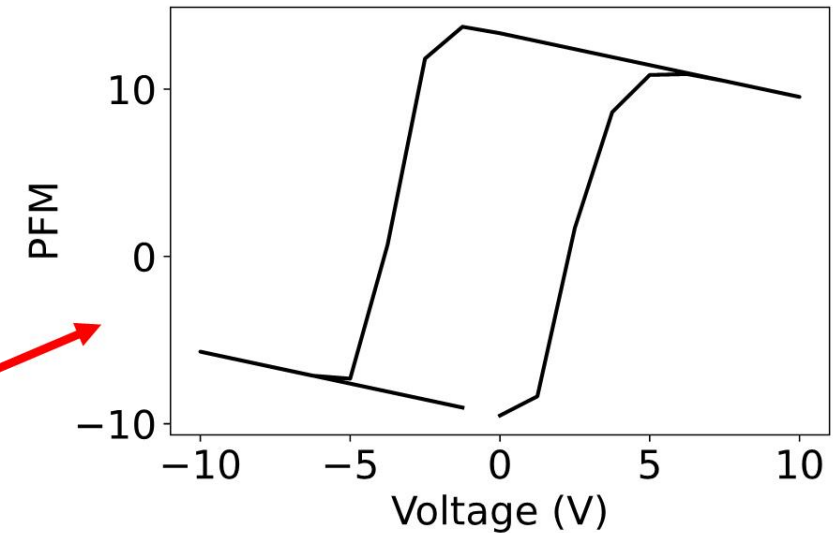
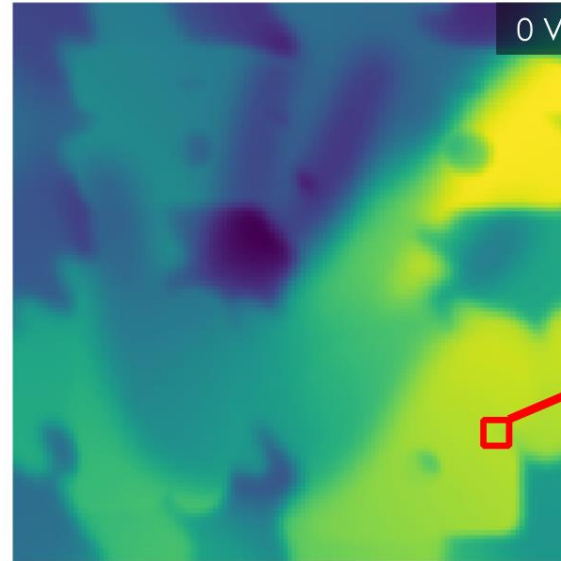
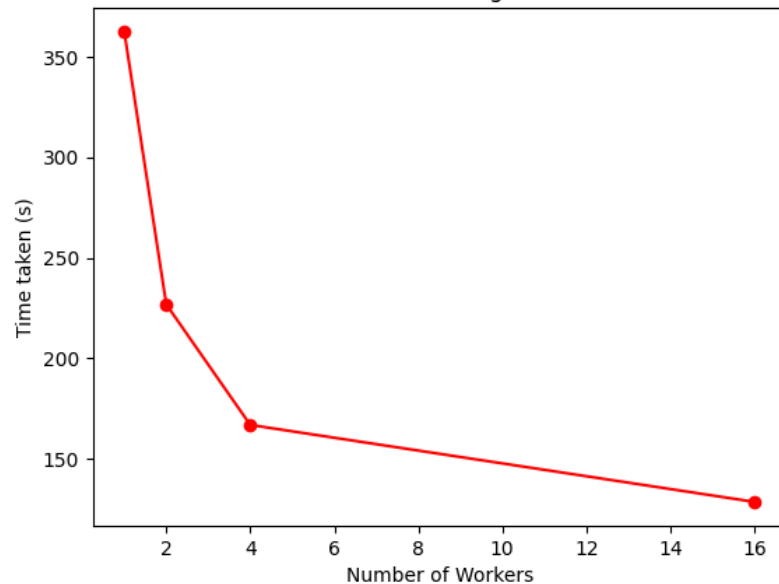

Dask backend means we can enable rapid parallel fitting

```
pserver = ProcessServer(server_loc = 'remote')
```

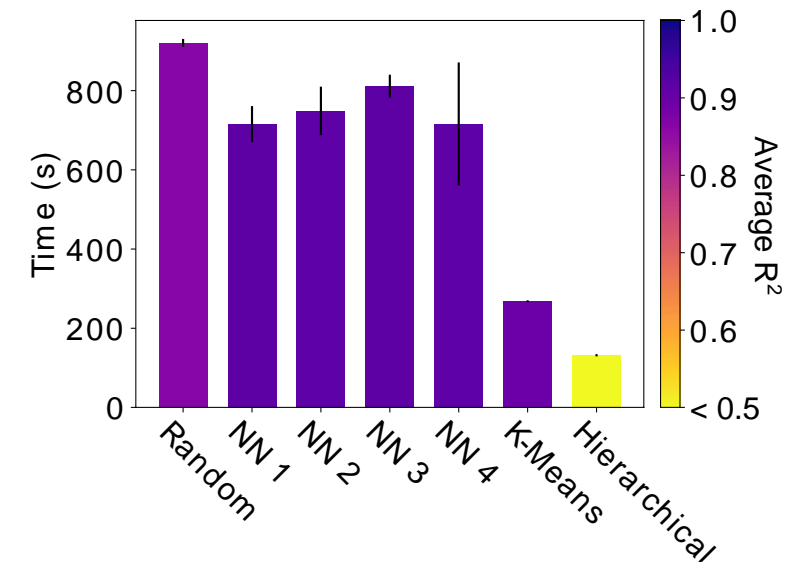
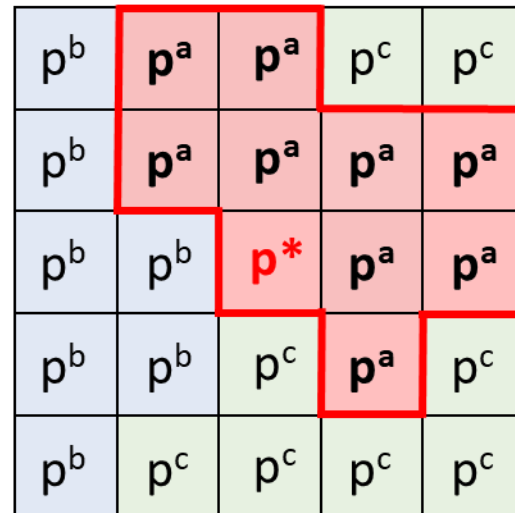
Please start the server at the remote location if it is not already running.

```
times = []
for num_workers in [1,2,4,16]:
    t0 = time.time()
    process_args = {'num_workers':num_workers,
                    'ind_dims':[0,1],
                    'return_cov':False,
                    'return_fit':False}
    processed_data = pserver.run_process(beline_sidpy,
                                        'SHO Fit',
                                        **process_args )
```

SHO Fitting

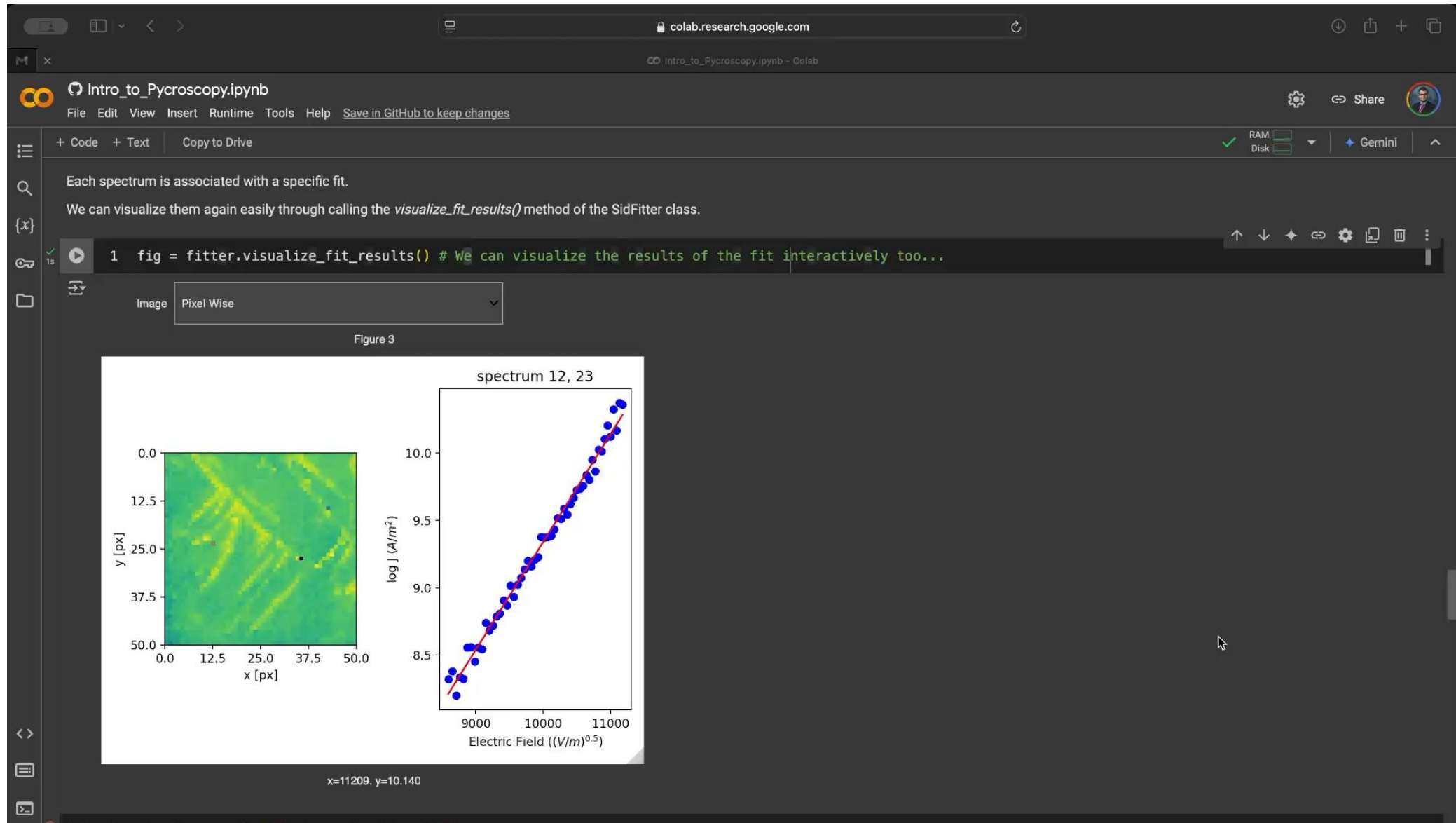


K-Means

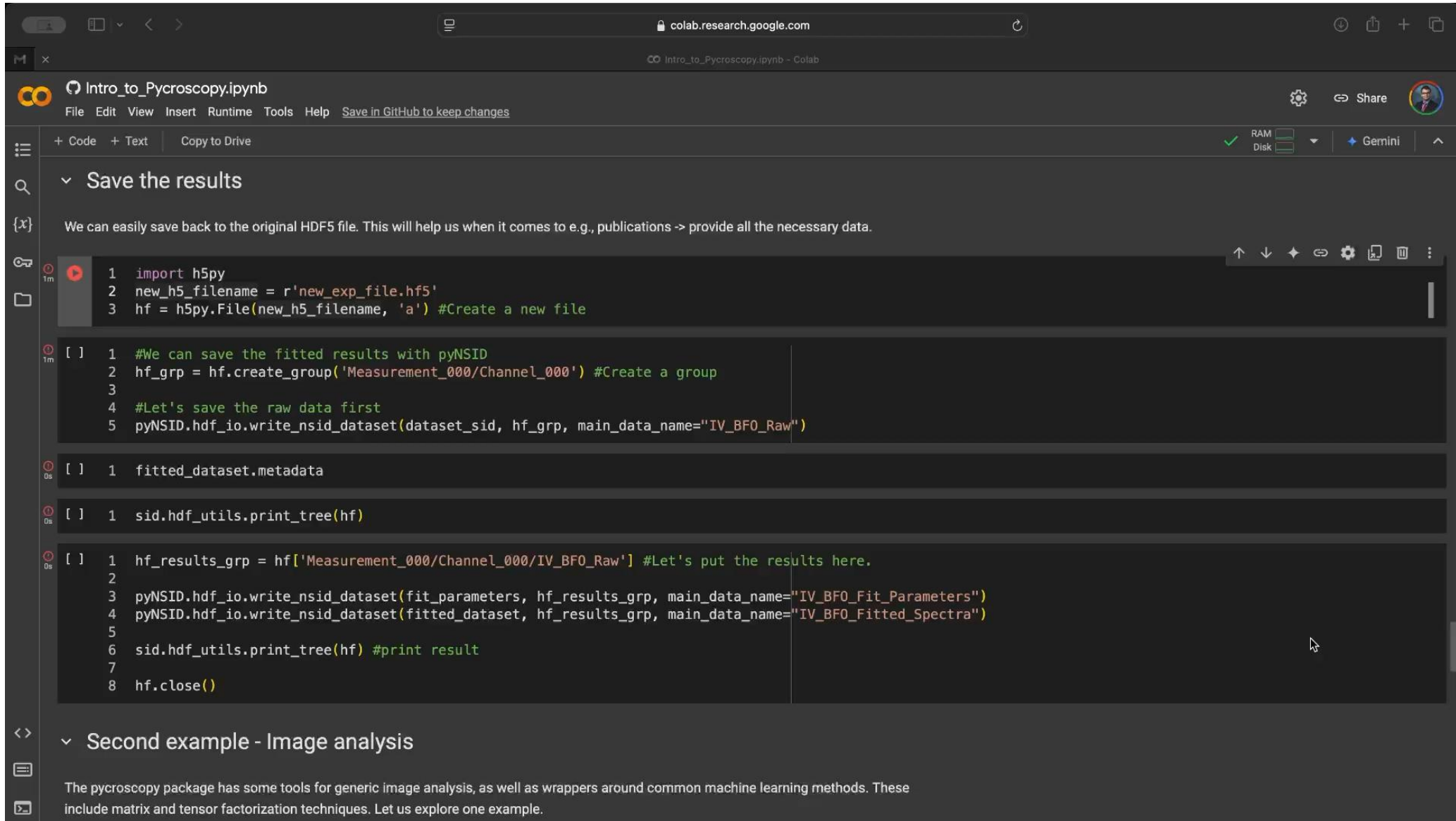


N. Creange et al., MLST (2021)

Example: Save and reload data from hdf5 files



Example: Save and reload data from hdf5 files



The screenshot shows a Google Colab notebook interface. The browser address bar is `colab.research.google.com`. The notebook title is `Intro_to_Pycroscopy.ipynb`. The menu bar includes `File`, `Edit`, `View`, `Insert`, `Runtime`, `Tools`, and `Help`. A link to `Save in GitHub to keep changes` is present. The left sidebar shows icons for file explorer, search, and other notebook functions. The top right shows system status (RAM, Disk) and a `Gemini` button. The main content area is titled `Save the results` and contains a text block and four code cells.

We can easily save back to the original HDF5 file. This will help us when it comes to e.g., publications -> provide all the necessary data.

```
1 import h5py
2 new_h5_filename = r'new_exp_file.hf5'
3 hf = h5py.File(new_h5_filename, 'a') #Create a new file
```

```
[ ] 1 #We can save the fitted results with pyNSID
2 hf_grp = hf.create_group('Measurement_000/Channel_000') #Create a group
3
4 #Let's save the raw data first
5 pyNSID.hdf_io.write_nsid_dataset(dataset_sid, hf_grp, main_data_name="IV_BF0_Raw")
```

```
[ ] 1 fitted_dataset.metadata
```

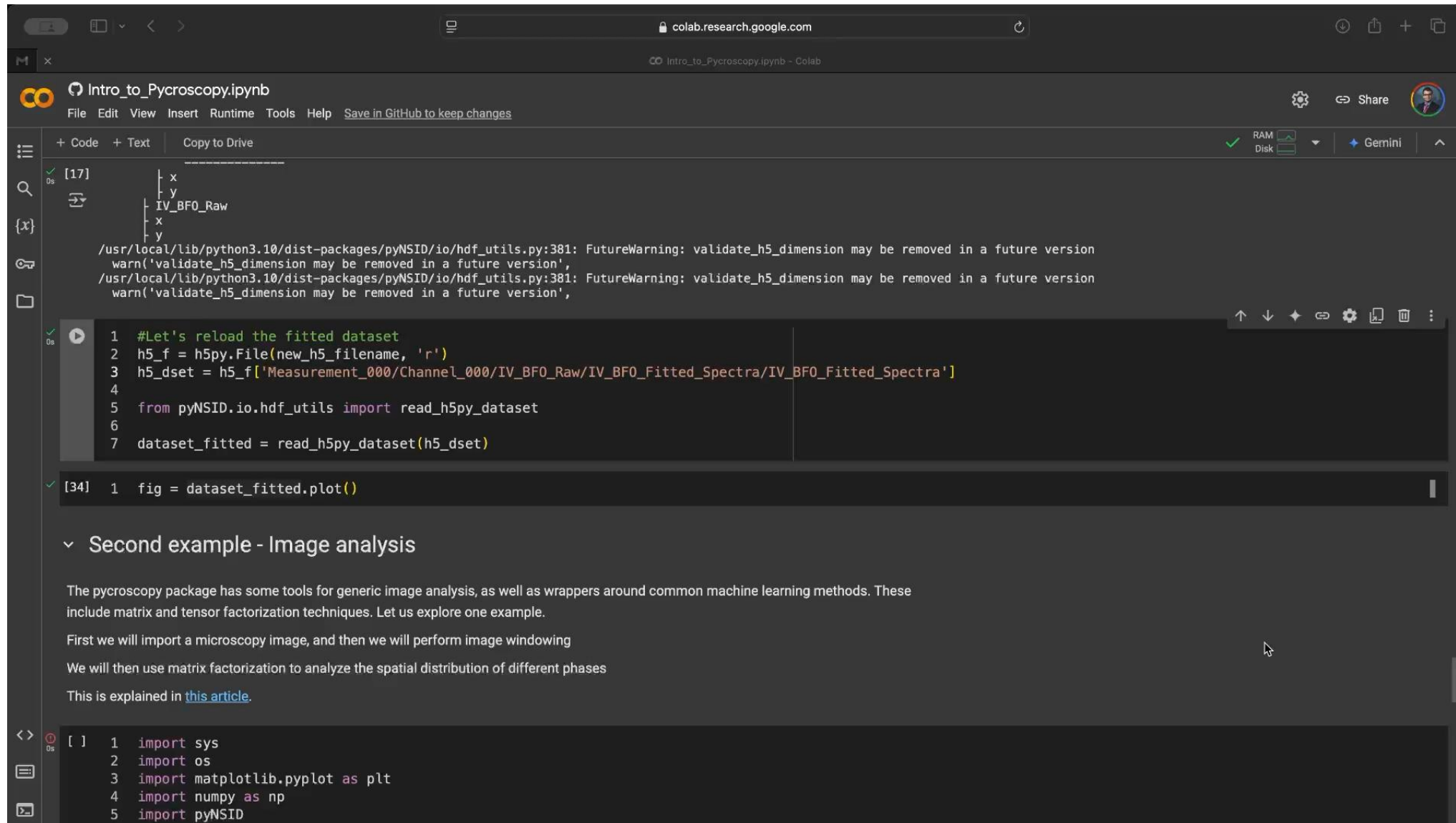
```
[ ] 1 sid.hdf_utils.print_tree(hf)
```

```
[ ] 1 hf_results_grp = hf['Measurement_000/Channel_000/IV_BF0_Raw'] #Let's put the results here.
2
3 pyNSID.hdf_io.write_nsid_dataset(fit_parameters, hf_results_grp, main_data_name="IV_BF0_Fit_Parameters")
4 pyNSID.hdf_io.write_nsid_dataset(fitted_dataset, hf_results_grp, main_data_name="IV_BF0_Fitted_Spectra")
5
6 sid.hdf_utils.print_tree(hf) #print result
7
8 hf.close()
```

<> `Second example - Image analysis`

The pycroscopy package has some tools for generic image analysis, as well as wrappers around common machine learning methods. These include matrix and tensor factorization techniques. Let us explore one example.

Example: Save and reload data from hdf5 files



The screenshot shows a Google Colab notebook interface. The top bar includes the Colab logo, the notebook title 'Intro_to_Pycroscopy.ipynb', and a 'Share' button. Below the top bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a link to 'Save in GitHub to keep changes'. The left sidebar contains icons for file management, search, and a file explorer. The main area displays code cells. The first cell, labeled '[17]', shows a directory tree structure and two FutureWarning messages from pyNSID. The second cell, labeled '[34]', contains Python code to reload a fitted dataset. The third cell, labeled '[34]', contains a single line of code to plot the dataset. Below the code cells is a section titled 'Second example - Image analysis' with explanatory text and a link to an article. The bottom of the notebook shows a code cell with import statements for sys, os, matplotlib.pyplot, numpy, and pyNSID.

```
[17] In [17]:
      | x
      | y
      | IV_BF0_Raw
      | x
      | y
      |
      | /usr/local/lib/python3.10/dist-packages/pyNSID/io/hdf_utils.py:381: FutureWarning: validate_h5_dimension may be removed in a future version
      | warn('validate_h5_dimension may be removed in a future version',
      | /usr/local/lib/python3.10/dist-packages/pyNSID/io/hdf_utils.py:381: FutureWarning: validate_h5_dimension may be removed in a future version
      | warn('validate_h5_dimension may be removed in a future version',

1 #Let's reload the fitted dataset
2 h5_f = h5py.File(new_h5_filename, 'r')
3 h5_dset = h5_f['Measurement_000/Channel_000/IV_BF0_Raw/IV_BF0_Fitted_Spectra/IV_BF0_Fitted_Spectra']
4
5 from pyNSID.io.hdf_utils import read_h5py_dataset
6
7 dataset_fitted = read_h5py_dataset(h5_dset)

[34] In [34]: 1 fig = dataset_fitted.plot()
```

Second example - Image analysis

The pycroscopy package has some tools for generic image analysis, as well as wrappers around common machine learning methods. These include matrix and tensor factorization techniques. Let us explore one example.

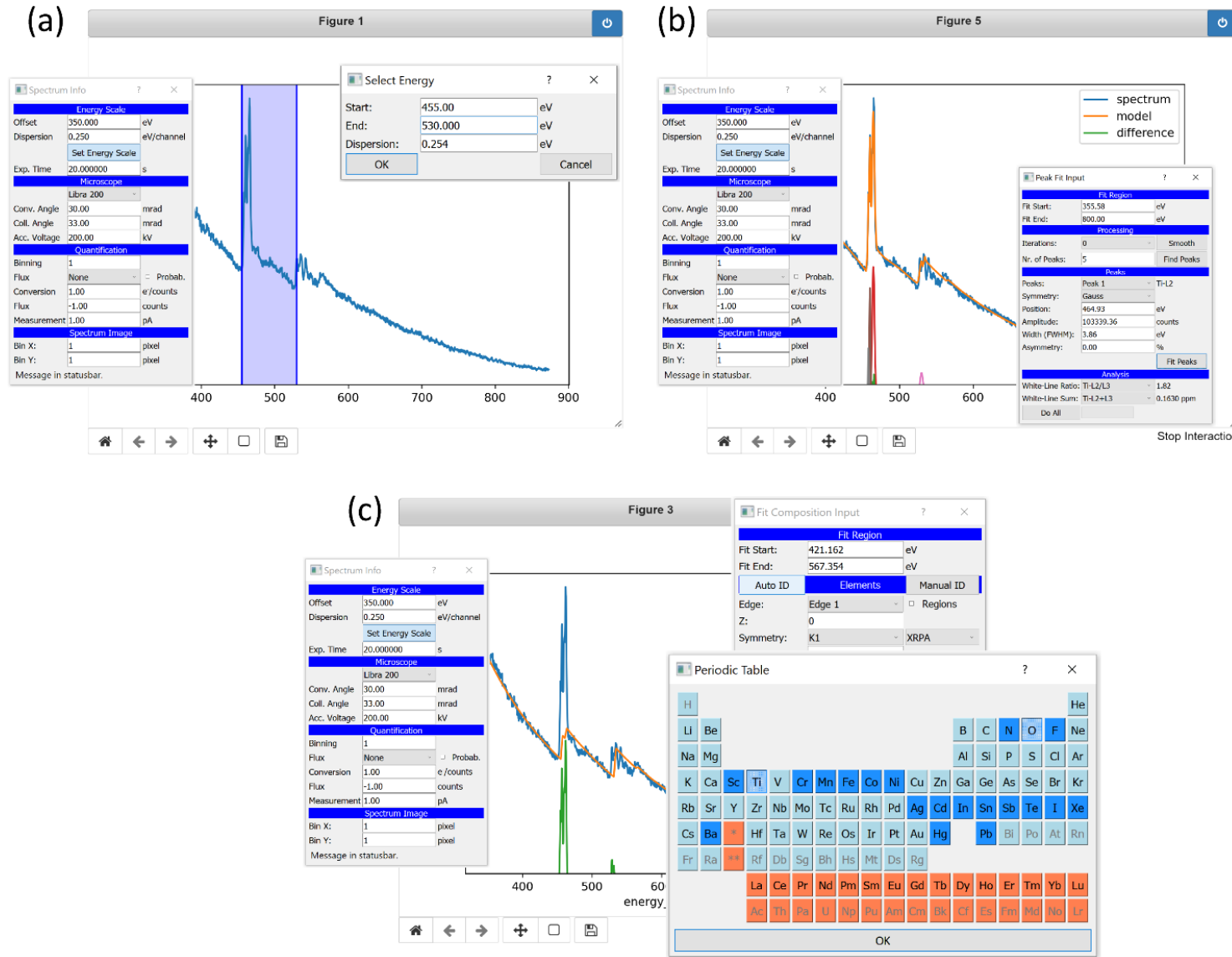
First we will import a microscopy image, and then we will perform image windowing

We will then use matrix factorization to analyze the spatial distribution of different phases

This is explained in [this article](#).

```
[ ] In [ ]: 1 import sys
           2 import os
           3 import matplotlib.pyplot as plt
           4 import numpy as np
           5 import pyNSID
```

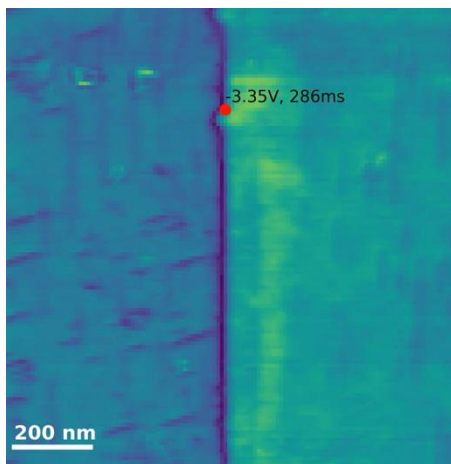
Domain-specific analysis available within the ecosystem



Domain-specific analysis, e.g. can analyze electron microscopy data in pyTEMlib, or atomic force microscopy data in BGLib, etc.

Automated workflows enable materials manipulation

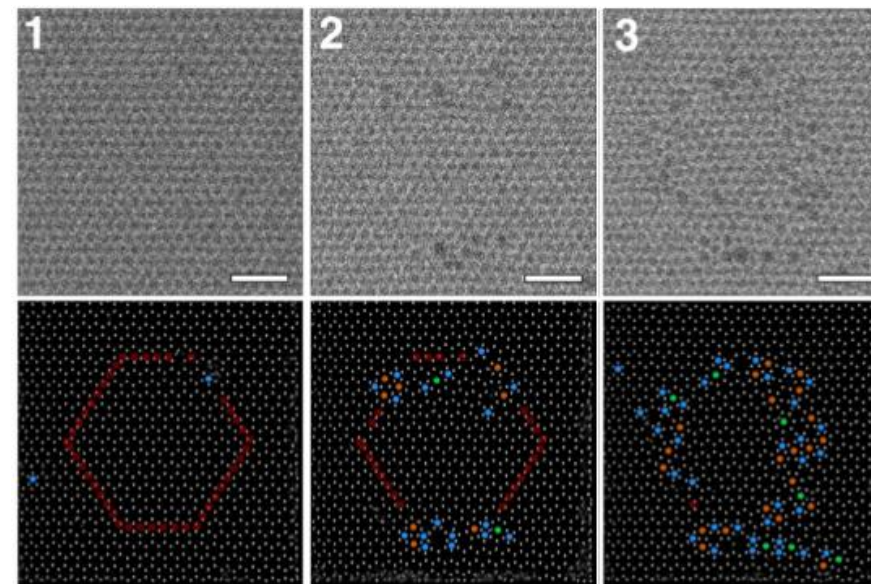
SPM: Autonomous Wall Manipulation



B. Smith et al. RSC Dig. Disc. (2024)

- Automated transition acquisition: write and perturb domain walls with voltages from the SPM tip
- Train physics-guided neural networks for prediction and digital twin, for arbitrary wall configurations and pulse parameters
- Use digital twin to train AI Agents for autonomous wall manipulation

STEM: Workflows for Defect Writing



K. Roccapiore et al. ACS Nano 16, 17116 (2022)

- Graphene imaging – steer beam in desired path, image again, find defects with computer vision, avoid defective areas, repeat
- Fully automated ‘avoidance patterning’ workflow to create defective regions on demand

DTMicroscope on Github

Rather than need a real microscope for development, we have created several ‘digital twin’ microscopes that you can utilize

This will ensure minimal changes are required when porting the methods developed herein, onto the full microscope for real-time autonomous steering

main

10 Branches

0 Tags

Go to file

Code

ramav87

Merge pull request #30 from pycroscopy/updated_data

975fb1b · 3 days ago

152 Commits

DTMicroscope	test updates	last month
assets	update: STEM datasets_links.md	last month
data	add: CoAg data	3 weeks ago
notebooks	additional notebooks	3 days ago
.DS_Store	additional notebooks	3 days ago
.gitignore	Merge branch 'utk' of https://github.com/pycroscopy/DT...	last month
LICENSE	added files	4 months ago
README.md	Initial commit	4 months ago
__init__.py	added files	4 months ago
setup.py	structural changes	last month

README

MIT license

DTMicroscope

Digital Twin Microscope

Some Hackathon notes

- Use the Slack channels for communication. If you have a question, feel free to unmute or directly message us in Slack (preferred) or Zoom
- If you don't know what to do or are stuck, we have several helpers available. We will direct you to a breakout room for further discussion
- The key point is to join a group tackling a problem, and work on that problem over the course of the hackathon. If you need ideas you can again feel free to ask us – there are several computer scientists and machine learning experts on hand
- The point is to have fun and learn something cool – we are building the future of microscopy with the merging of automation, software ecosystems, and machine learning capabilities.



Questions?
Enjoy the hackathon!
