MATLAB to Python Translation: Electron Microscope Tools

COSC 499: Software Engineering

Introduction

- Converting 3 tools from MATLAB to Python:
  - qEELS
  - NanoMi Optics
  - Tomography Alignment software

National Research Council
NRC·CNRC
Canada

National Institute for Nanotechnology

UNIVERSITY OF ALBERTA

Motivation

- Python’s growing popularity
- Proprietary nature of MATLAB
- Software engineering

National Research Council
NRC·CNRC
Canada

National Institute for Nanotechnology

UNIVERSITY OF ALBERTA

User Group

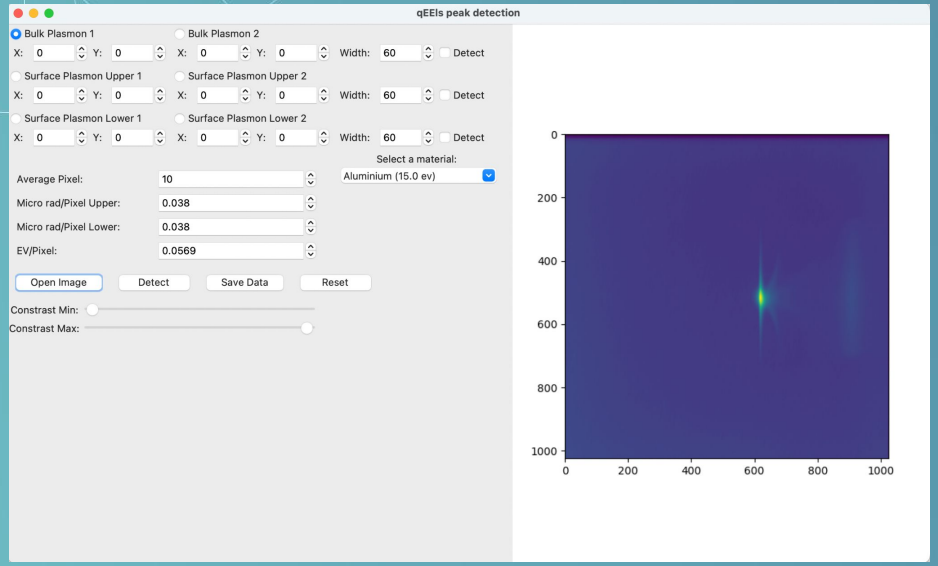
- NRC researchers at University of Alberta
- Expertise in nanotechnology
- Familiar with the MATLAB softwares

National Research Council
NRC·CNRC
Canada

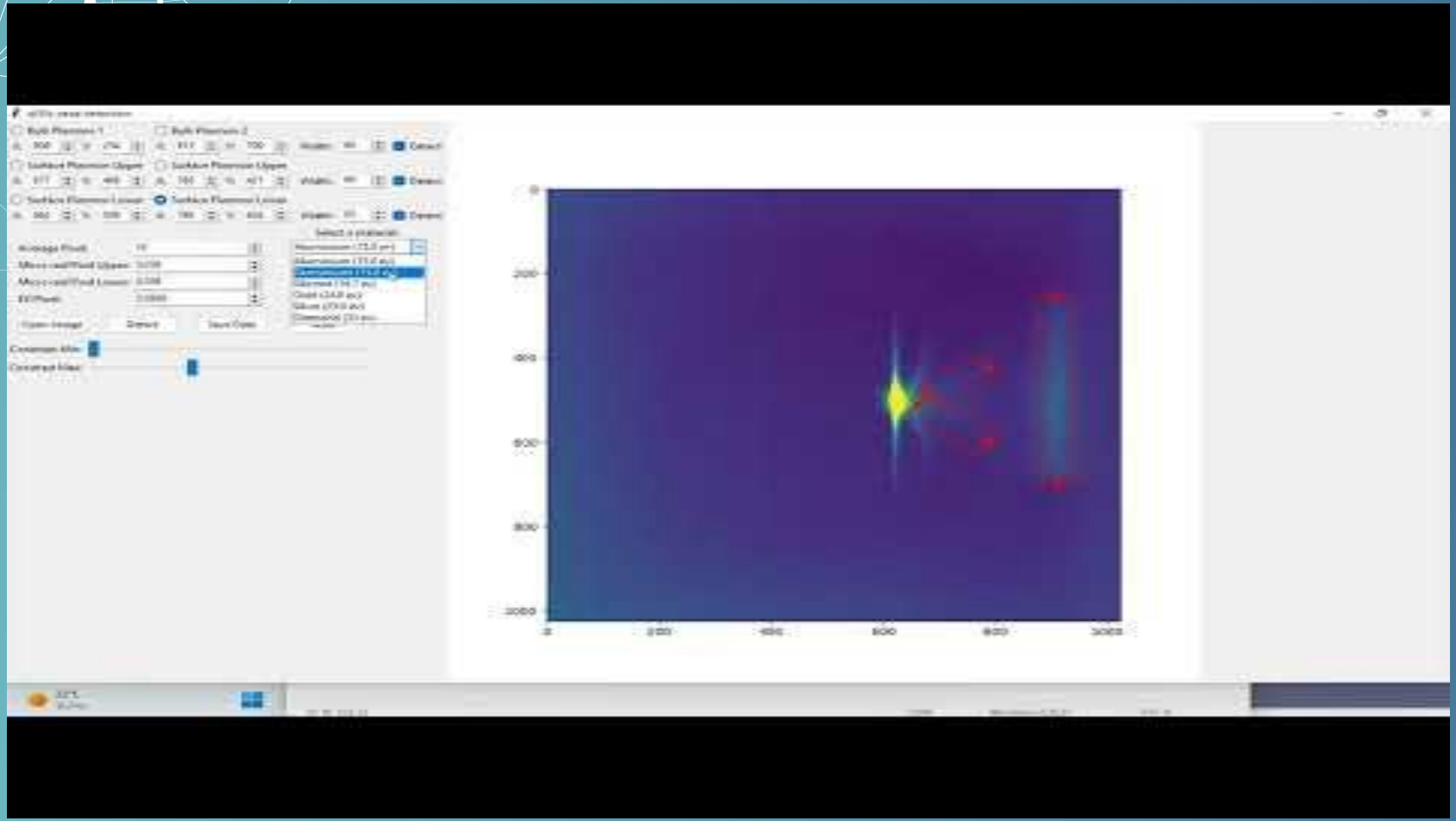
National Institute for Nanotechnology

UNIVERSITY OF ALBERTA

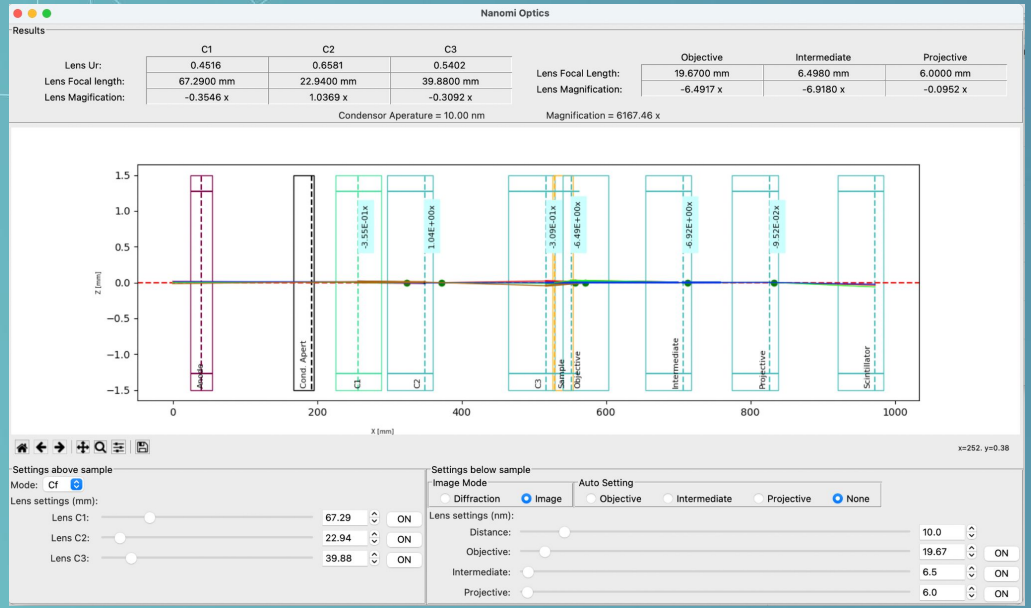
qEELS



- Opens a two-dimensional spectrum
- Detects and displays the location of peaks
- Calculates calibrated energy loss



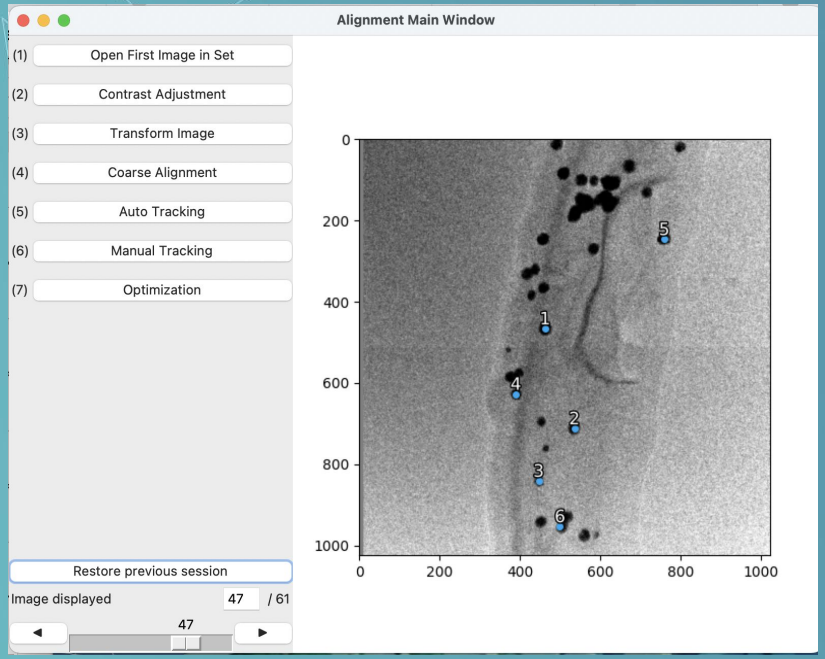
NanoMi Optics



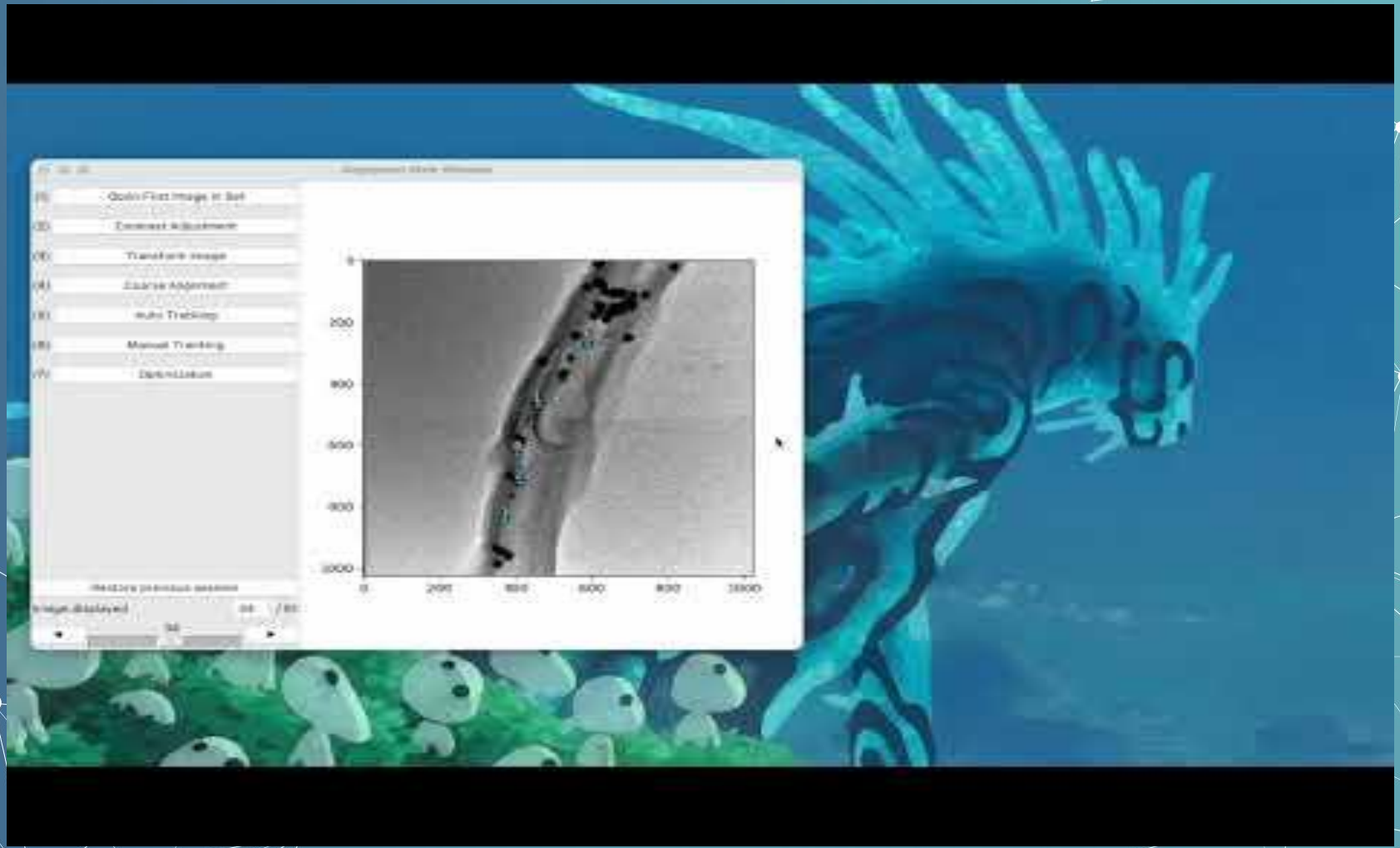
- Allows user to control lens settings
- Draws a diagram of the lenses and beams
- Can optimizes lens settings
- Shows results in a table


Results
C1
C2
C3
Lens Id:
0.4510
0.8581
0.5402
Lens Focal length:
67.2900 mm
22.9400 mm
33.0930 mm
Lens Magnification:
-0.3549 x
1.0368 x
-0.3082 x
Objective
Intermediate
Projective
Lens Focal Length:
Lens Magnification:
Condenser Aperature = 10.00 mm
Magnification = 6167.46 x
Settings above sample
Mode:
Lens settings (mm):
Lens C1:
67.29
Lens C2:
22.94
Lens C3:
33.09
Settings below sample
Image Mode
Diffraction
Image
Auto Setting
Objective
Intermediate
Projective
None
Lens settings (mm):
Condenser:
10.0
Objective:
16.87
Intermediate:
6.0
Projective:
6.0
ON
Reset
Save

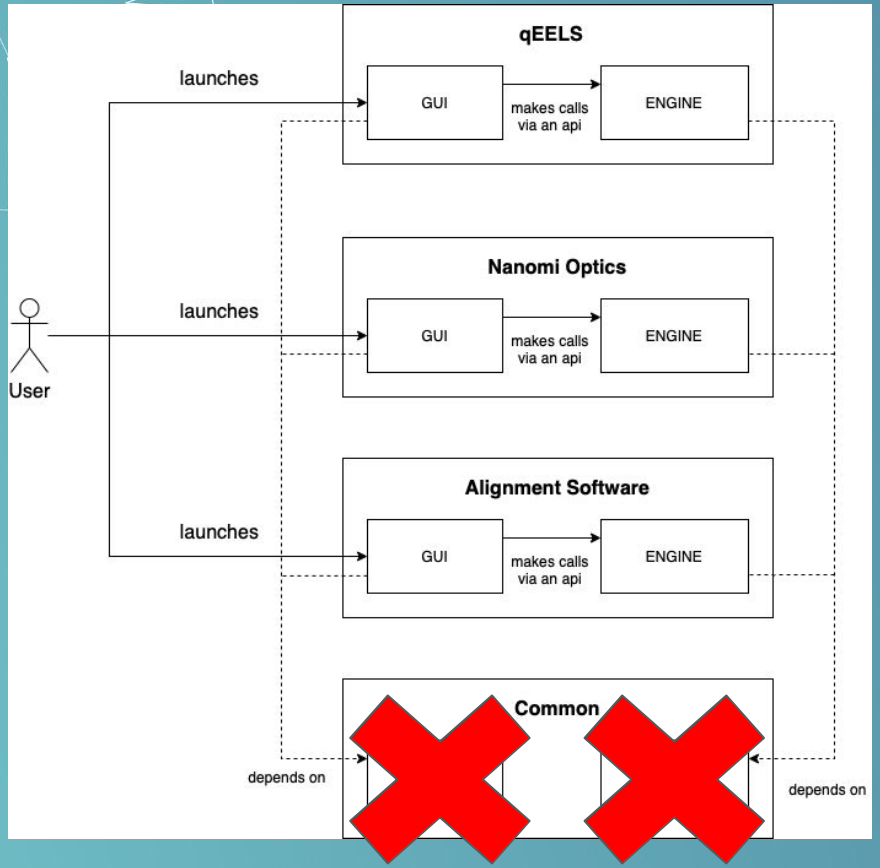
Alignment Software



- Opens unaligned sequences of images captured by electron microscopes
- Aligns images using particle tracking and optimization algorithms
- Outputs aligned sequences of images


Auto Tracking
Manual Tracking
0
200
400
600
800
1000
0
200
400
600
800
1000

Design: Architecture



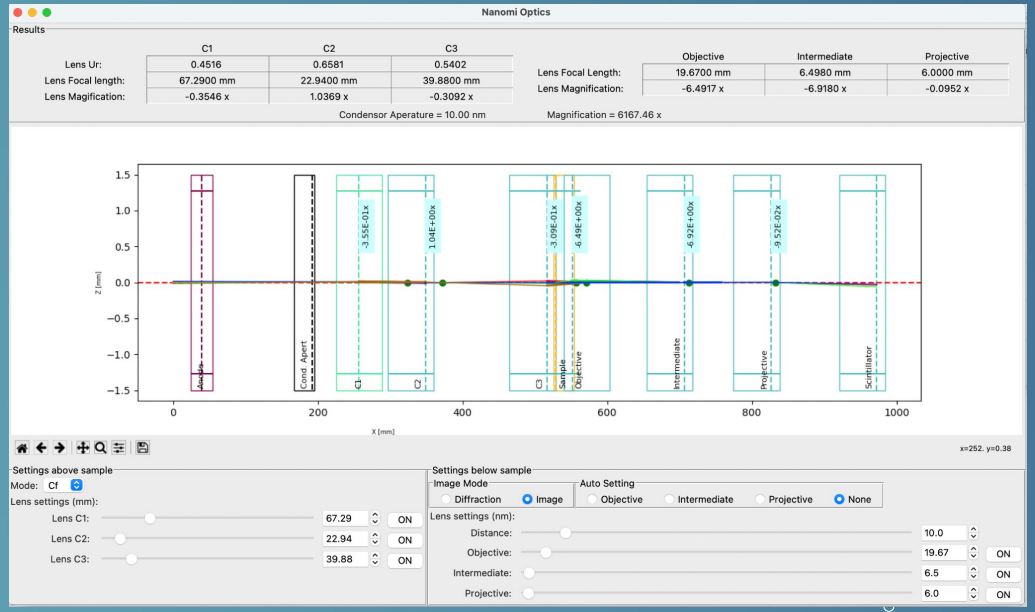
- Each software has a GUI and Engine
- GUI make calls to the Engine
- Engine **does not** make calls to GUI

Design: Engines

- SciPy & NumPy
  - Fast hardware accelerated operations
  - Optimization framework
  - Image transformation
  - Kernel convolution
  - Interpolation
- Large number of application specific math functions
- Custom CSV Input/Output methods
- Custom DM3 Image encoder/decoder

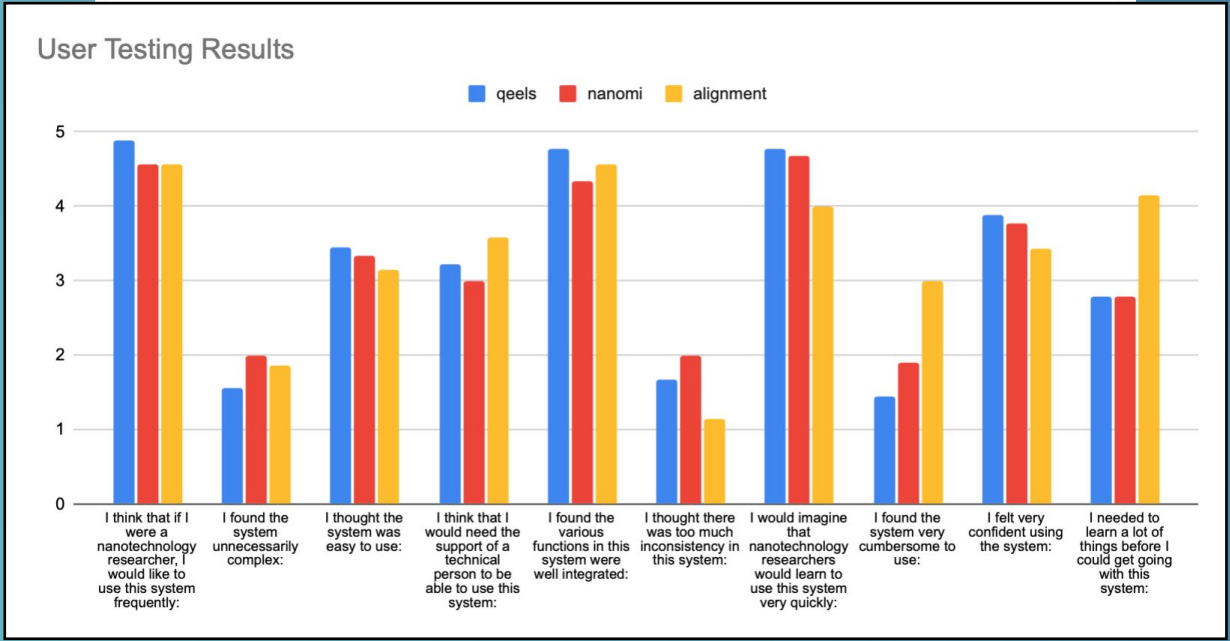
Design: GUIs

- Tkinter
  - Window management
  - Widgets and layout
- Matplotlib
  - Rendering graphs
  - Rendering Diagrams
  - Rendering images
  - Marking up images
- Custom input validation and linking code



Usability Testing

Software	Score
qEELS	77.78
NanoMi Optics	72.50
Tomography Alignment Software	65.00



Testing

- Acceptance tests
  - Is GUI is accepted by the user group?
- Unit tests
  - Does Python engine produce the same results as Matlab?

REQUIREMENTS	Type of Test UN: Unit Testing A: Acceptance Testing	Pass or Fail P: pass F: fail	Contributor GC: Garrett Cook JM: Jasmine Mishra JP: Jose Pena Revelo LT: Lucas Towers
Functional Requirements			
qEELS			
Software will load the spectrogram from a PRZ file	UN	P	GC
Software will render points and boxes of indicated features on the spectrogram	A	P	GC
Software will detect fitted peaks of surface and bulk plasmon	UN	P	GC
Software will calculate calibrated energy loss axis and transfer axis	UN	P	GC
Software will output results to CSV format	UN	P	GC
Software will be functionally equivalent to legacy software	A	P	GC
Nanomi Optics			
Software will render lenses on diagram	A	P	JM
Software will calculate upper beam ray	UN	P	JP
Software will calculate beam ray through projective, intermediate, and objective lenses	UN	P	JP
Software will render upper beams on diagram	A	P	JM, JP
Software will render lower beams on diagram	A	P	JP
Software will display results in a table	A	P	JM, JP
Software will calculate optimized settings for projective lens	UN	P	JP
Software will calculate optimized settings for intermediate lens	UN	P	JP
Software will caluclate optimized settings for objective lens	UN	P	JP
Software will be at least as functional as legacy sofware	A	P	JP
Alignment Software			
Software will load DM3 images	UN	P	LT
Software will perform automatic contrast adjustment	UN	P	LT
Software will perform translation, rotation and scaling of frames	UN	P	LT
Software will perform coarse alignment of frames with cross-corelation	UN	P	LT
Software will automatically detect the location of particles with kernel convolution	UN	P	LT
Software will calculate optimized alignment of frames	UN	P	LT
Software will output intermediary alignment info to csv format	UN	P	LT
Software will write DM3 images	UN	P	LT
Software will be at least as functional as legacy sofware	A	P	LT

Unit Testing

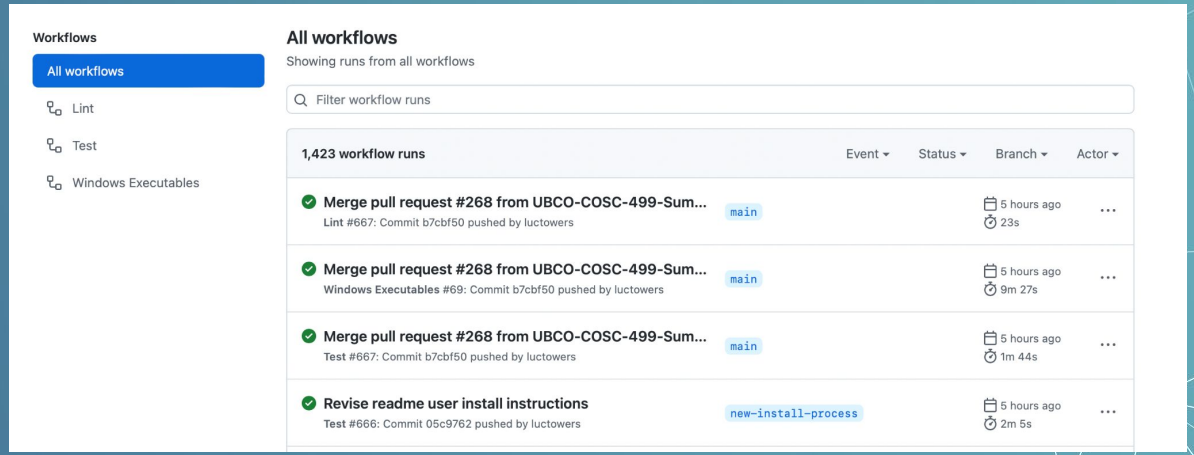
- Pytest unit testing framework
- NumPy's unit testing library


```
============================================== test session starts ==============================================
platform darwin -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: /Users/luctowers/Documents/ubco/cosc499/matlab-to-python-application-translation-project2-nrc
collected 60 items

nrcemt/alignment_software/test/test_csv_io.py ....                                                    [  6%]
nrcemt/alignment_software/test/test_dm3.py ...                                                        [ 11%]
nrcemt/alignment_software/test/test_file_discovery.py ...                                             [ 16%]
nrcemt/alignment_software/test/test_img_io.py ...                                                     [ 21%]
nrcemt/alignment_software/test/test_img_processing.py ............                                    [ 41%]
nrcemt/alignment_software/test/test_optimization.py .......                                           [ 53%]
nrcemt/alignment_software/test/test_particle_tracking.py ....                                         [ 60%]
nrcemt/nanomi_optics/test/test_lens_static_methods.py ..                                              [ 63%]
nrcemt/nanomi_optics/test/test_lower_lenses_math.py .                                                 [ 65%]
nrcemt/nanomi_optics/test/test_optimization.py .                                                      [ 66%]
nrcemt/nanomi_optics/test/test_upper_lenses_math.py ...                                               [ 71%]
nrcemt/nanomi_optics/test/test_ur_conversion.py ..                                                    [ 75%]
nrcemt/qeels/test/test_peak_detection.py ............                                                 [ 95%]
nrcemt/qeels/test/test_prz.py ..                                                                      [ 98%]
nrcemt/qeels/test/test_results.py .                                                                   [100%]

============================================== 60 passed in 5.31s ===============================================
```

CI/CD

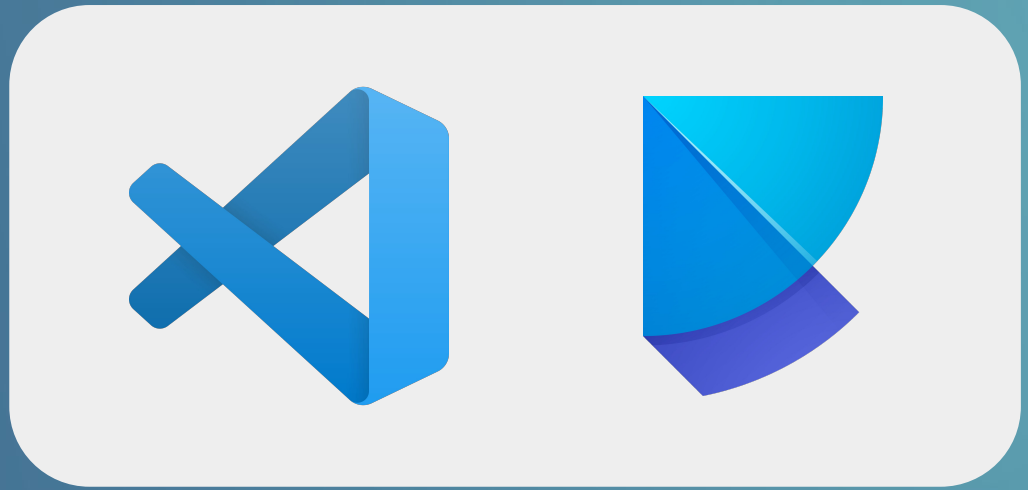
- Automatically run all unit tests
- Automatically lint test all code
- Automatically build window executables

GitHub Actions

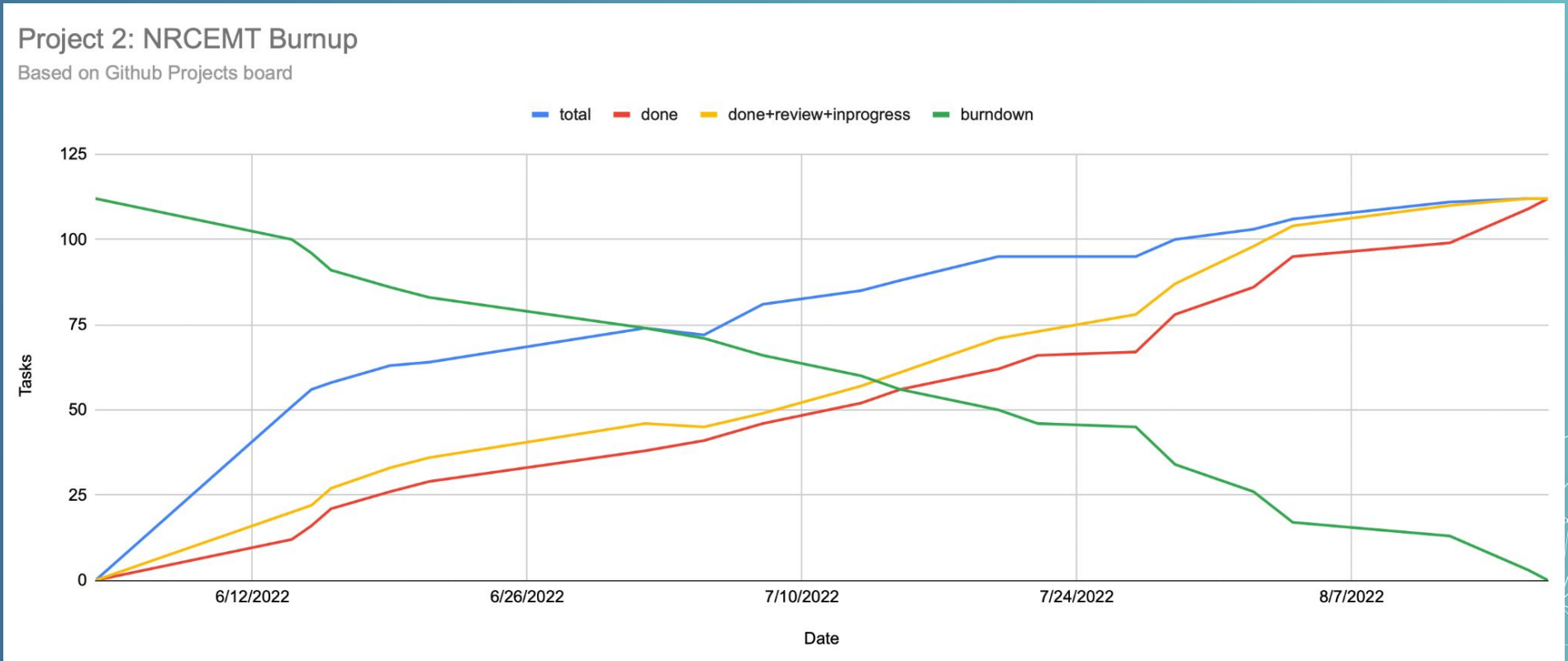


Development Environment

- Visual Studio Code
- Python Poetry Virtual Environments

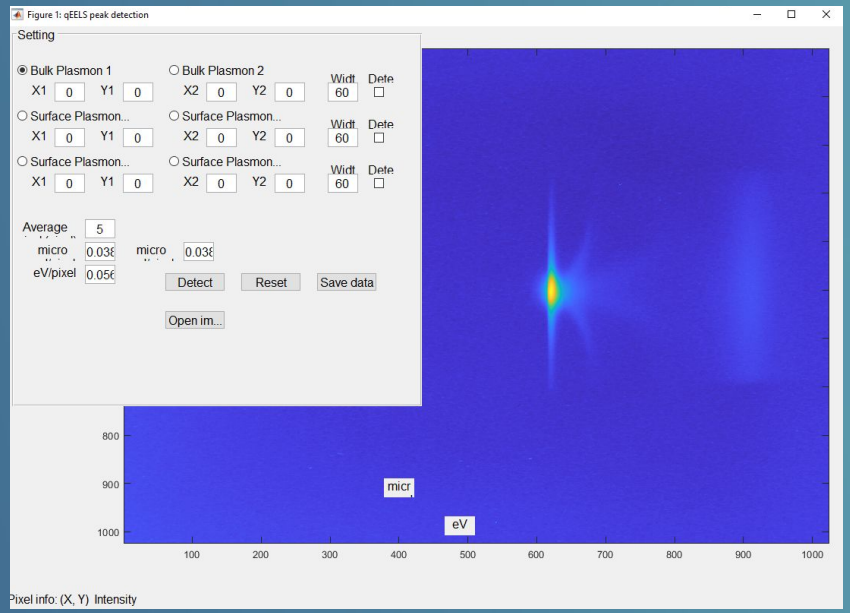


Project Management

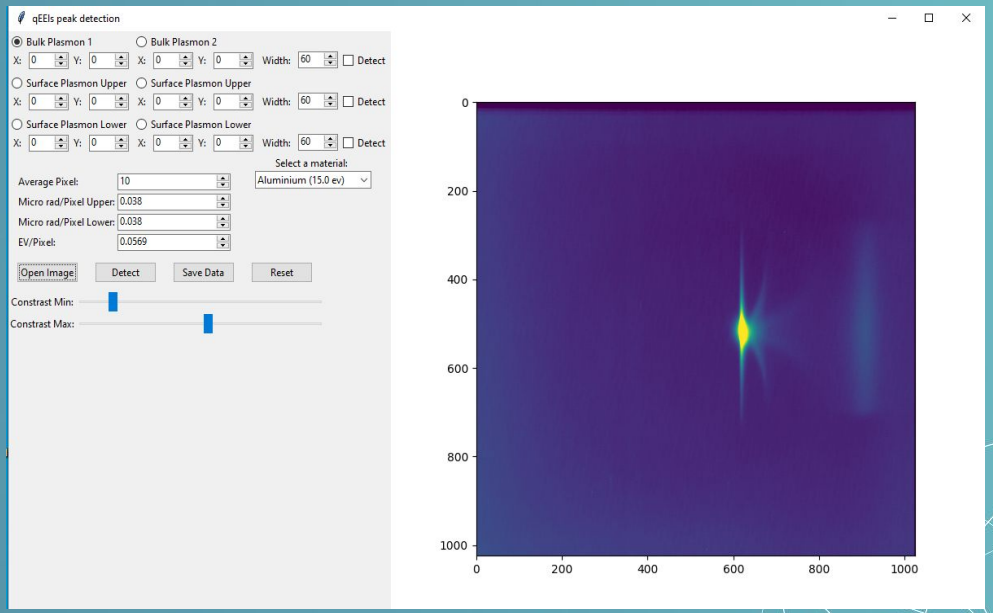


Combined time worked: ~700 Hours

Additional Features: qEELS GUI

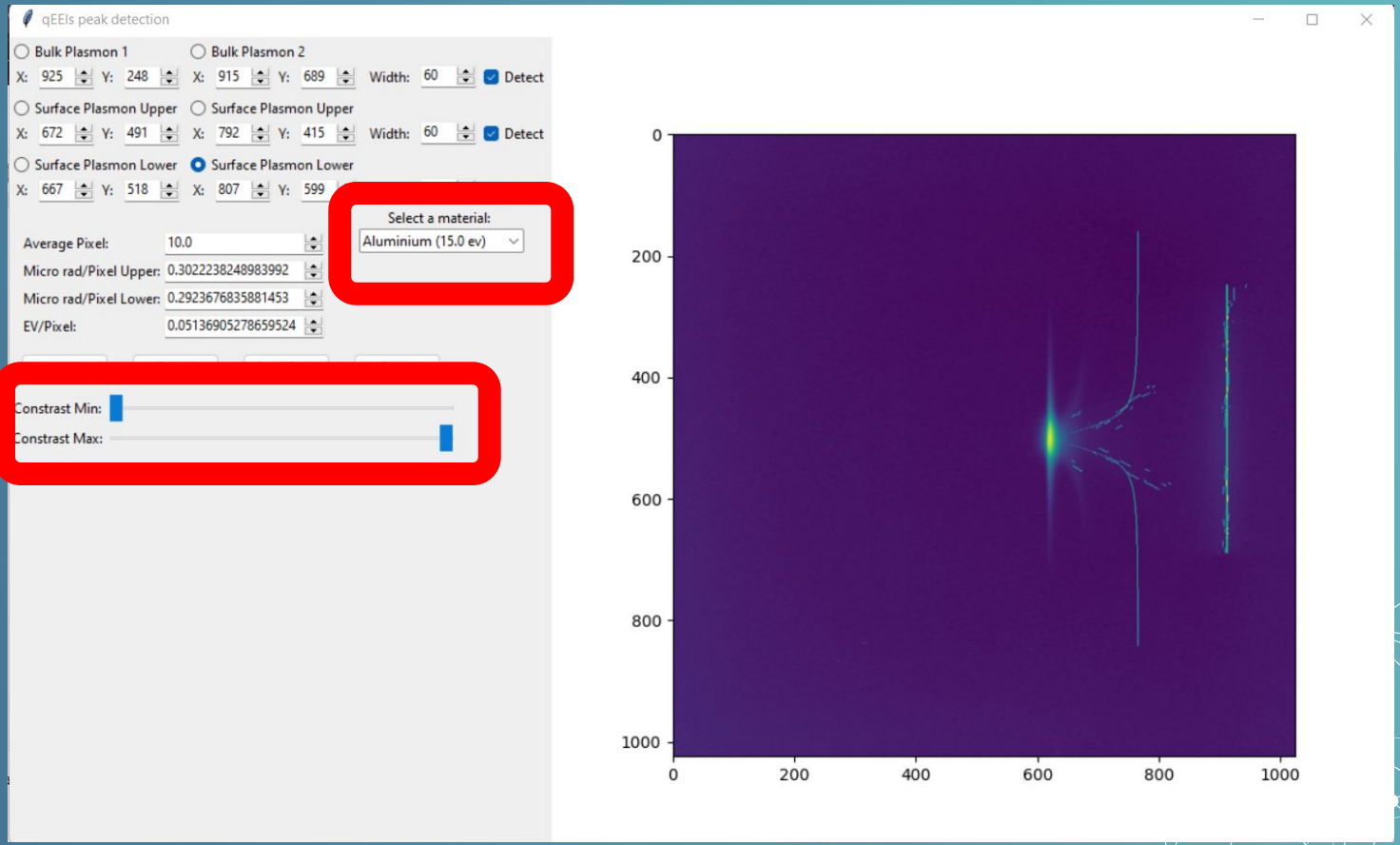


MATLAB



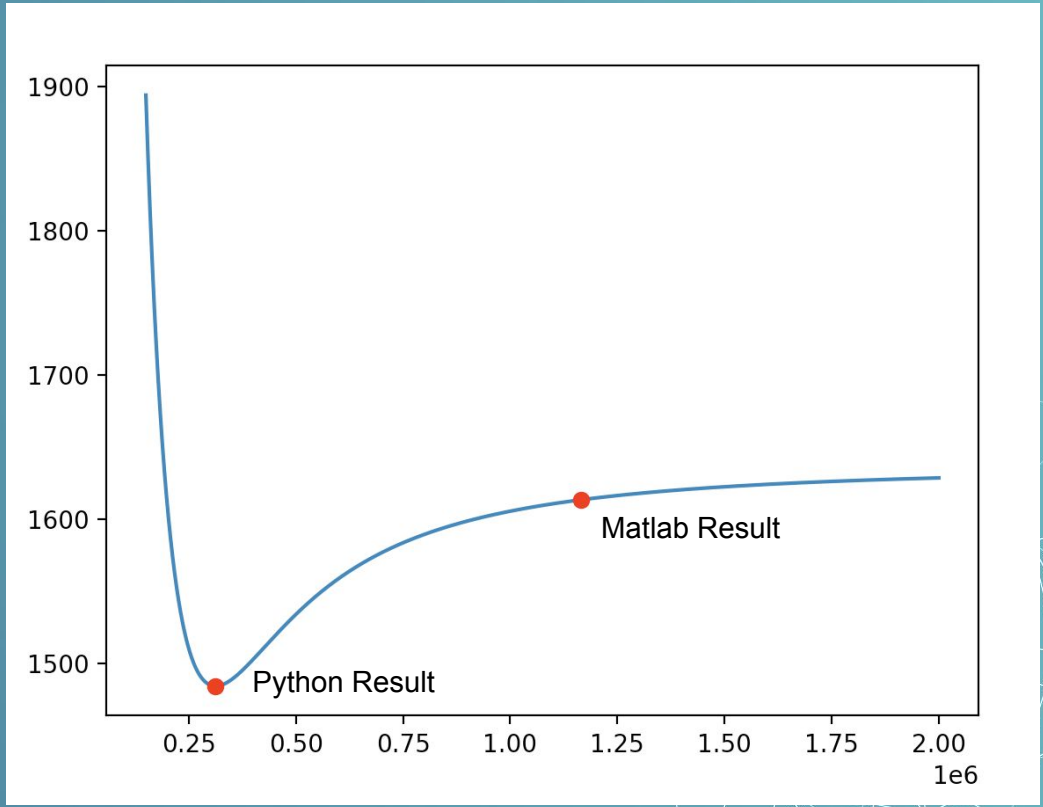
Python

Additional Features: qEELS GUI



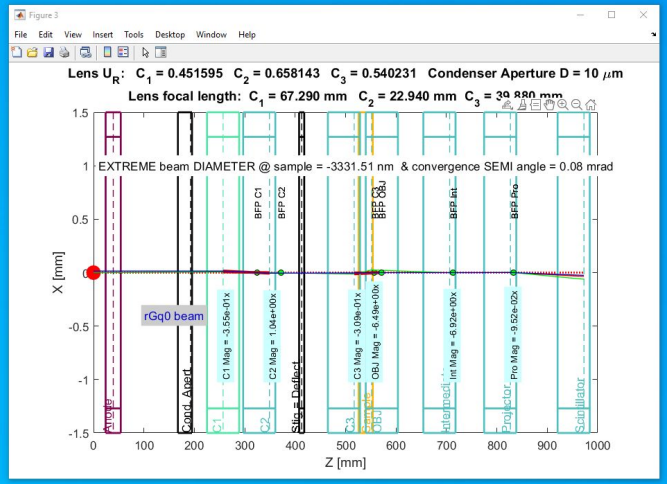
Additional Features: qEELS

- Improved optimization

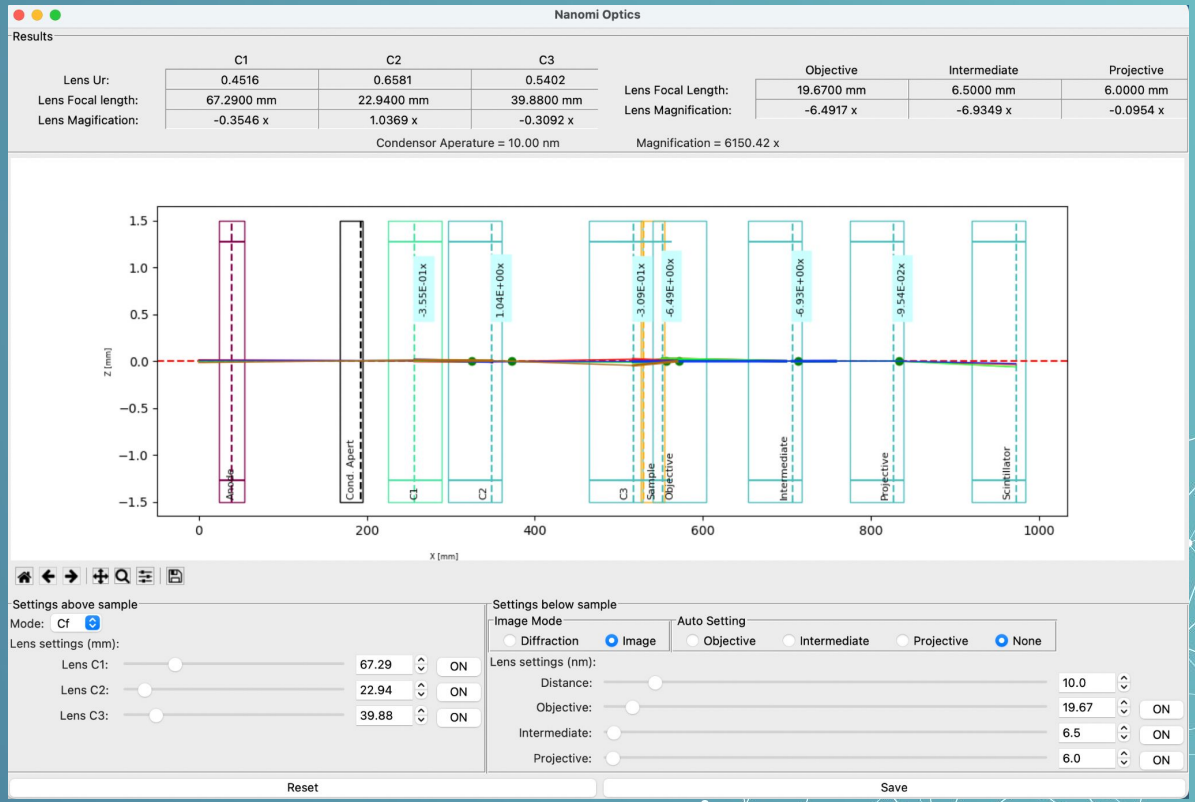


Additional Features: NanoMi GUI

MATLAB



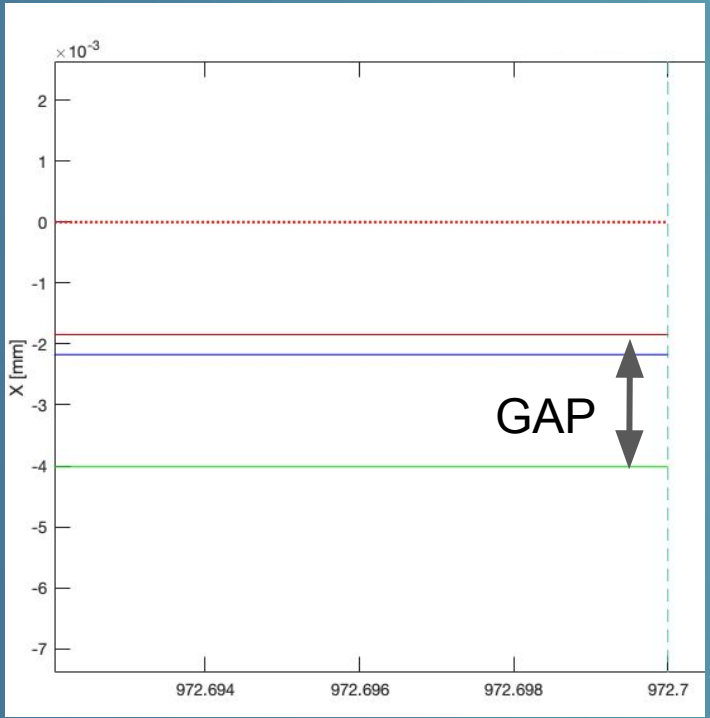
Python



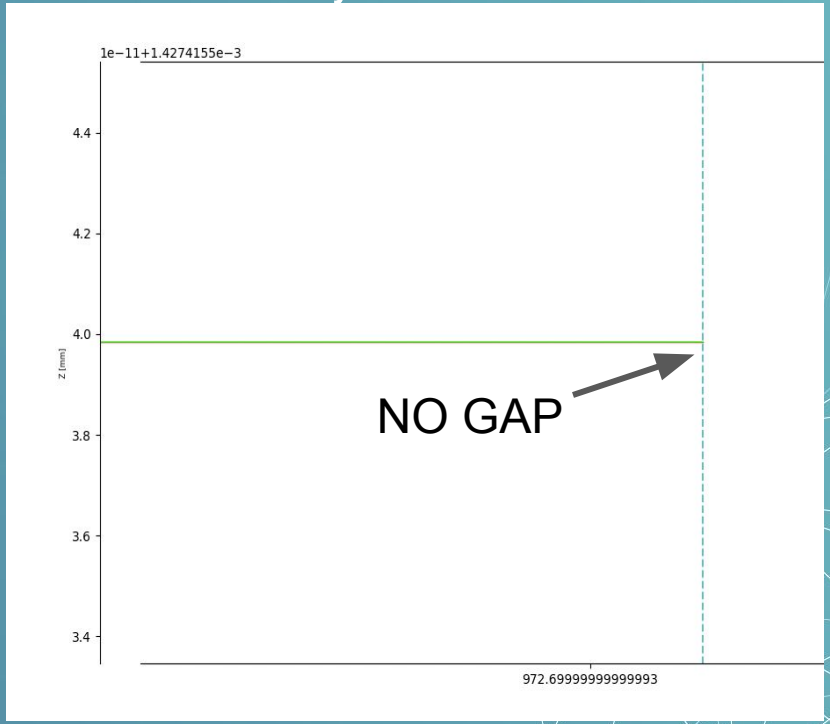
Additional Features: NanoMi

- Improved lens optimization, Diffraction mode:

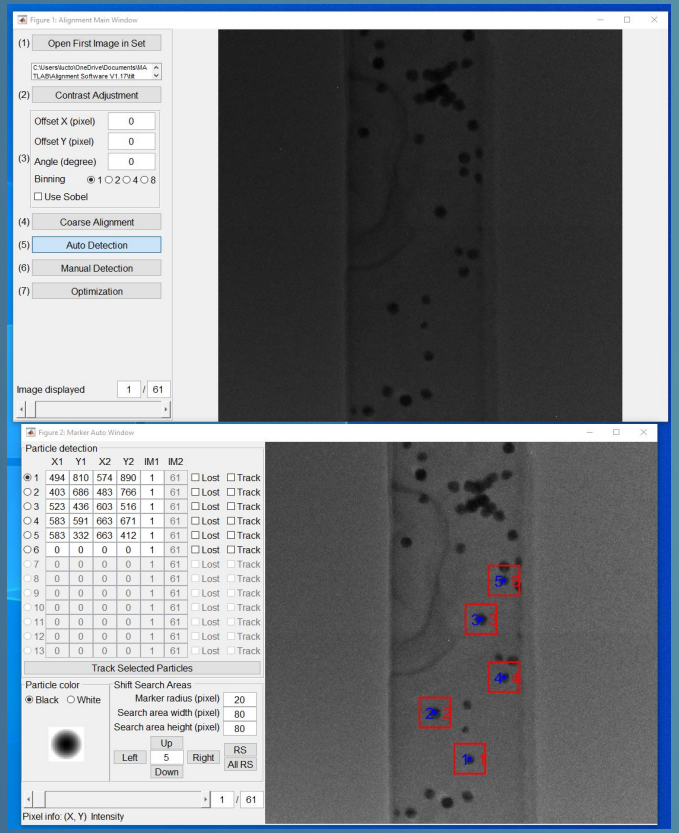
MATLAB



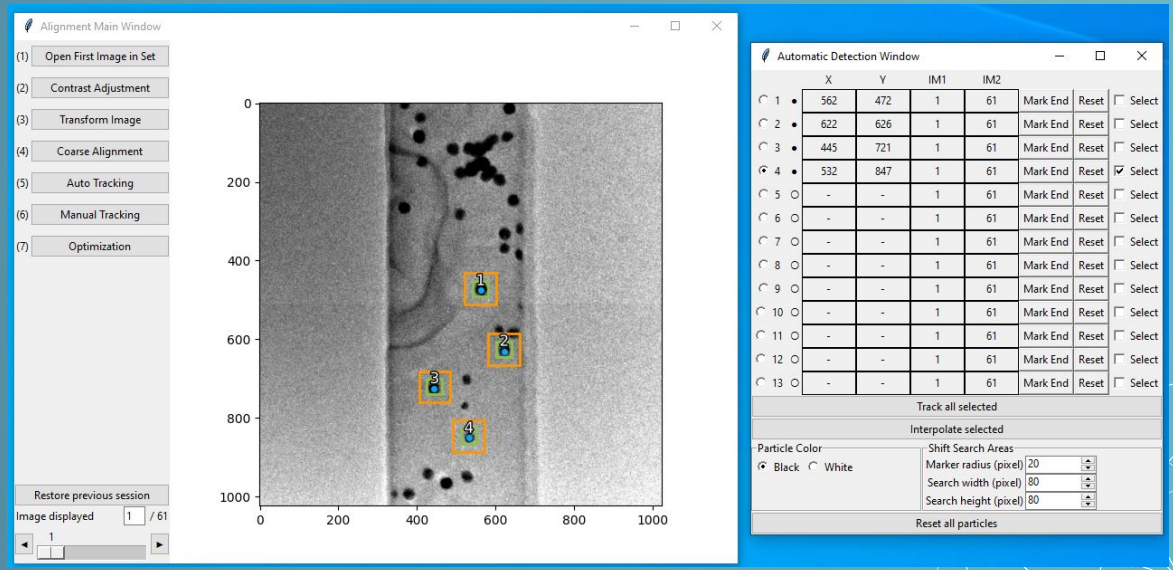
Python



Additional Features: Alignment GUI



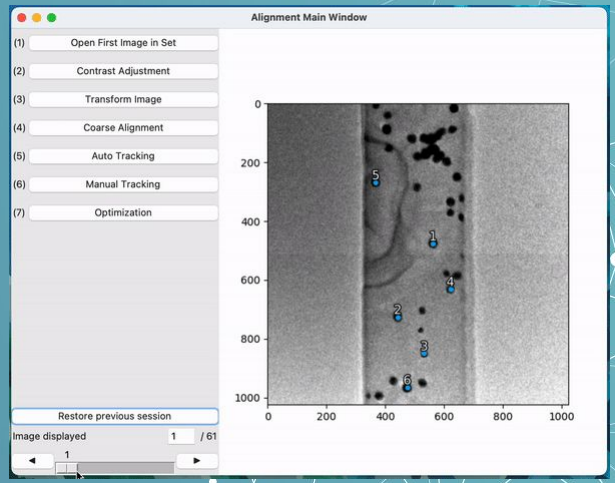
MATLAB



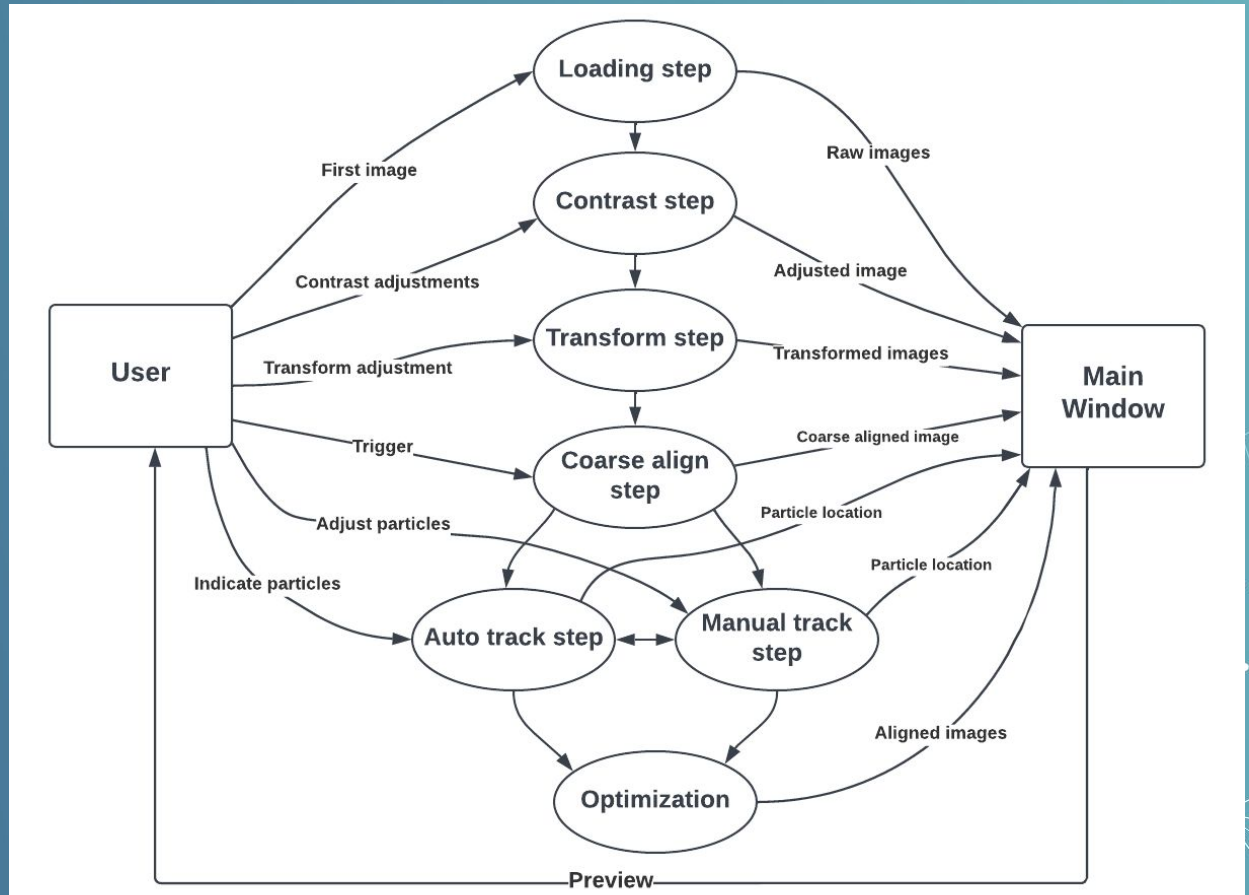
Python

Additional Features: Alignment

- Improved contrast adjustment
- Improved responsiveness
- Transforms are viewable before coarse alignment
- Transforms can be controlled by sliders
- Images don’t have sections chopped off after multiple transforms
- Overhauled automatic tracking workflow
- Interpolation for Automatic and Manual Detection
- Restore previous session button



Dataflow: Alignment



Technical Challenges

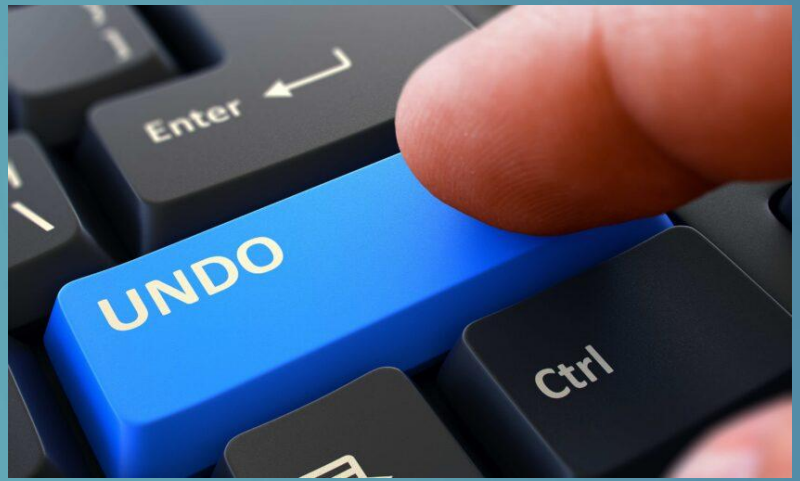
- Understanding complex math with few comments and nondescript variable names
- Learning new libraries and tools
- Matching and testing functionality between MATLAB and Python libraries
- Installing and running software on lab computers with restricted access

Lessons Learned

- Working in a professional setting with a client
- Collaboration using a Git and Kanban can workflow
- Using scientific libraries: SciPy, NumPy
- Debugging and testing complex math functions
- Working around the unique quirks of everybody’s python environments

What we would do different?

- Consider alternatives to tKinter
- Reconsider the order in which GUI/Engine components were developed



Summary

- Project status:
  - Finished
- Project hand over:
  - User Manuals
  - Technical Documentation

Q & A