Design & Testing

Project 2: MATLAB to Python Application Translation

Team members:
Garrett Cook
Lucas Towers
Jasmine Mishra
Jose Pena Revelo

Document Overview:

1. Software Description

There are currently three MATLAB tools created by the NRC laboratory that are used to analyze and calibrate electron microscope data. The new software will have the same functionality as the MATLAB tools, but translated into Python. The client wants to switch their software to Python because it's more portable, user friendly, and readable, among various other advantages.

The three tools being translated are named qEELS, Nanomi Optics, and Alignment Software. The qEELS software takes a spectrogram image and outputs calibrated energy loss. The Nanomi Optics software optimizes electron microscope optics settings based on user provided parameters. The Alignment Software takes electron microscope nanoparticle images and aligns and optimizes them for tracking them around each image.

2. User groups

- NRC researchers at University of Alberta

User persona:

Name: Lakshmi
Age: 32
Pronouns: she/her

Laksmi works as an electron microscope laboratory assistant researcher at the University of Alberta with the National Research Council of Canada as a graduate student. Her responsibilities in the laboratory include using the microscope to look at specimens, taking images with the microscope, processing microscope data, and using MATLAB tools.

She enjoys meticulous procedures; when she is in the laboratory, there is always a particular way of performing procedure and this is how she likes to perform her duties. Along with this, she hates changing her routine. When the procedures are heavily changed, she gets frustrated and she feels like it is a waste of time to learn new ways of doing the same thing. Lakshmi strives to be an efficient worker, so that she can make the most out of her time in the lab. She likes to learn new ways to make her use of a particular technology faster, so she doesn't need to learn how to use a new one. Lakshmi wants to learn more about nanoparticles through the use of electron microscope technology. In the long term, she hopes to work more with the Electron Energy Loss team in her lab. She also hopes that her laboratory will be able to make some real changes in the advancement of the use of electron microscopes.

2.1. Usage scenarios

Name:	qEELS
Description:	Calculate calibrated energy loss from a spectrogram image.
Flow of Events:	1. The user opens the software. 2. The user uploads the spectrogram image. 3. The user indicates the location of features on the spectrogram image. 4. The user indicates that the software will detect fitted peaks of surface plasmon and bulk plasmon. 5. The user requests the calibrated energy loss axis and transfer axis.
Pre-conditions:	● The user must have the software open.

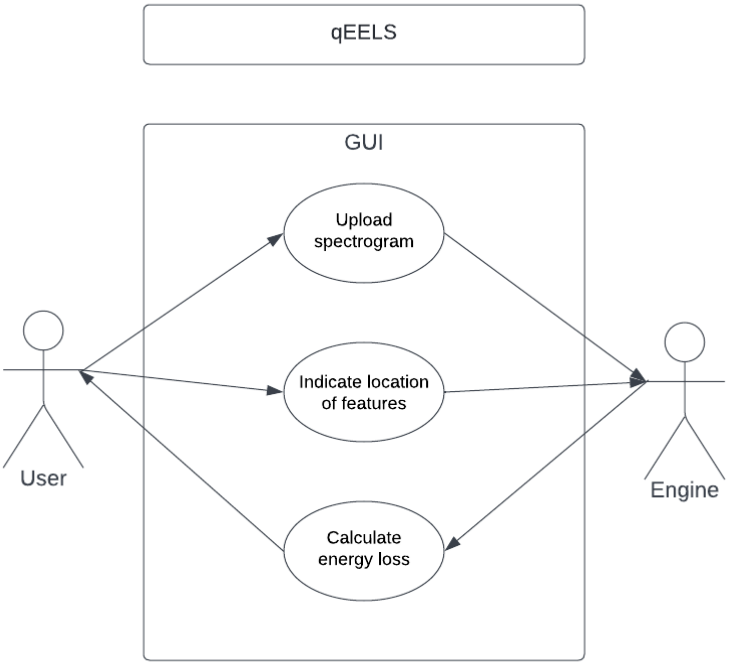
	● The user must have an appropriate spectrogram image. ● The user has a knowledge of spectrogram images.
Post-Conditions:	The energy loss axis and transfer axis is calculated and displayed in an image.

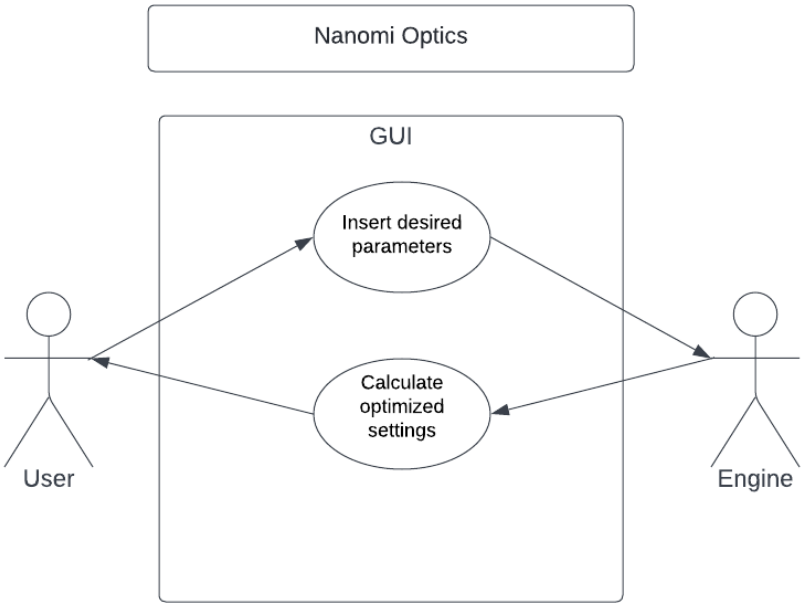
Name:	Nanomi Optics
Description:	Optimize electron microscope optics settings.
Flow of Events:	1. The user opens the software. 2. User inputs their desired parameters for optics 3. User request optimized lens settings
Pre-conditions:	● The user must have the software open. ● The user has knowledge of electron microscope settings.
Post-Conditions:	Optimized settings are displayed to the user.

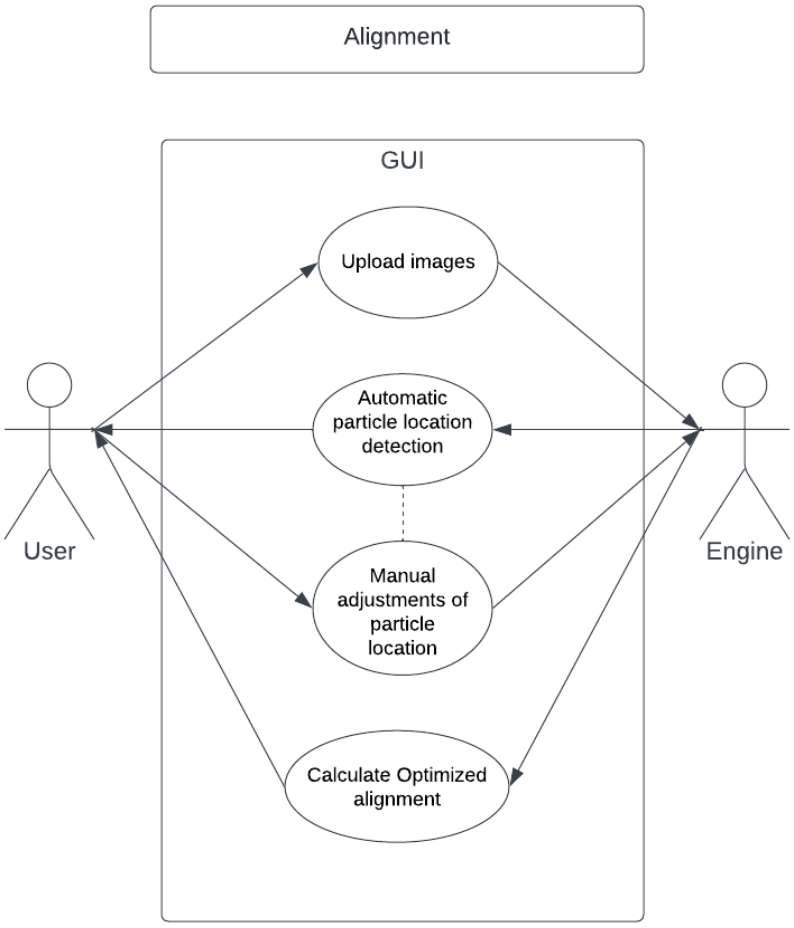
Name:	Alignment Software
Description	Align nanoparticle images.
Flow of Events:	1. The user opens the software. 2. The user uploads nanoparticle images. 3. The user selects the location of the particle on the image. 4. The user requests aligned sequence images from software.
Pre-conditions:	● The user must have the software open. ● The user has appropriate nanoparticle images. ● The user has knowledge of nanoparticle images.
Post-Conditions:	An aligned sequence of images is optimized and calculated.

2.2. Use Case Diagrams

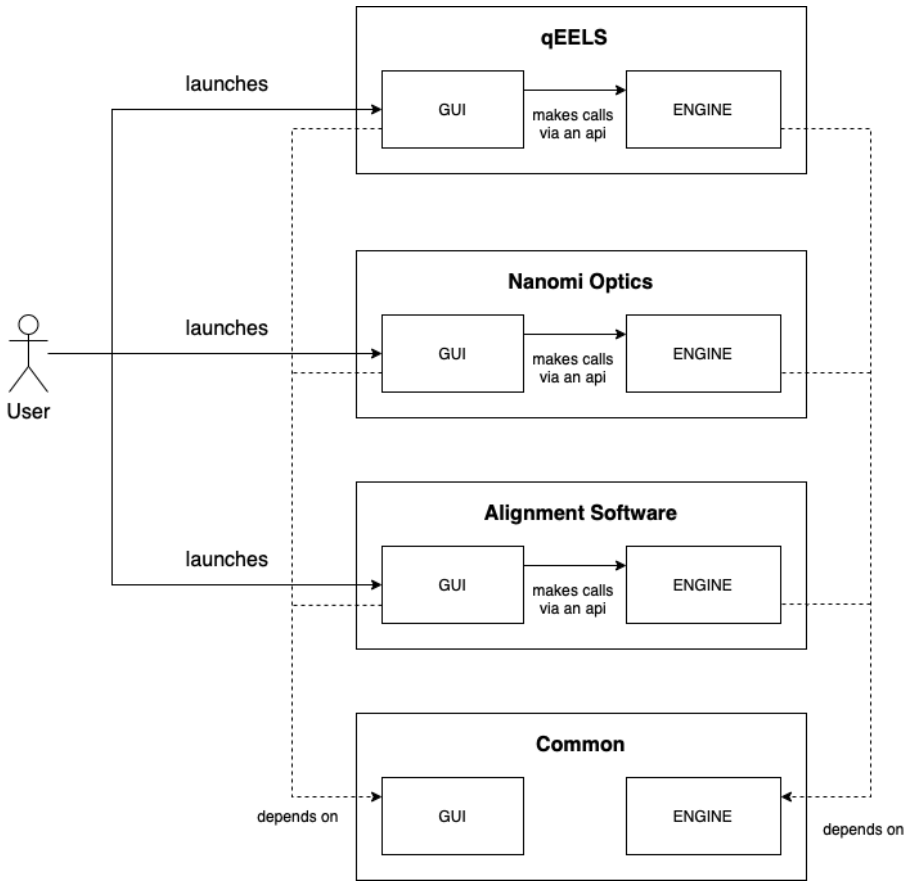
In all three pieces of software, the user will interact via GUI. The calculations and/or optimization will occur in the engine and report results back to the user in the GUI, by graph, image, or numerical values. Each software has one main usage, hence why there is one separate diagram for each.






Alignment
GUI
Upload images
Automatic particle location detection
Manual adjustments of particle location
Calculate Optimized alignment
User
Engine

3. System Architecture



Each of the three software being converted will have its own top-level package. Code from specific software should not import functionality from another software. Instead, functionality needed in multiple software should be refactored into the *common* top-level package.

Each *engine* should include all of the core logic and processing behind the software. The engine packages must be thoroughly tested. The engine packages must also have zero dependencies on the *gui* packages or gui libraries.

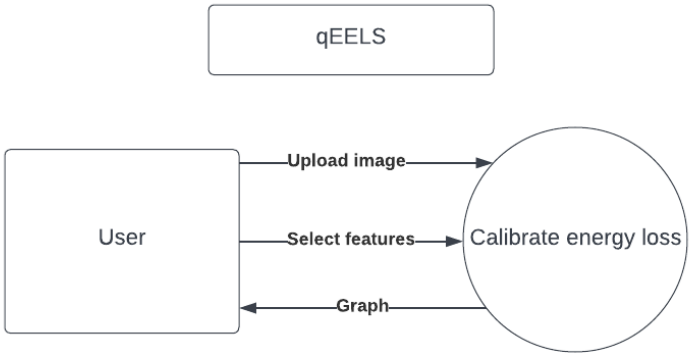
Each *gui* package should not include any data processing and should instead delegate all of that in calls to its respective engine. Each type of call made from a *gui* to an *engine* package should be tested. The *gui* should focus only on handling user interaction.

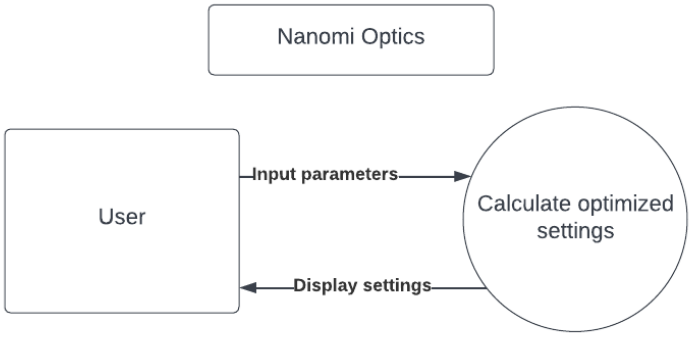
4. Diagrams

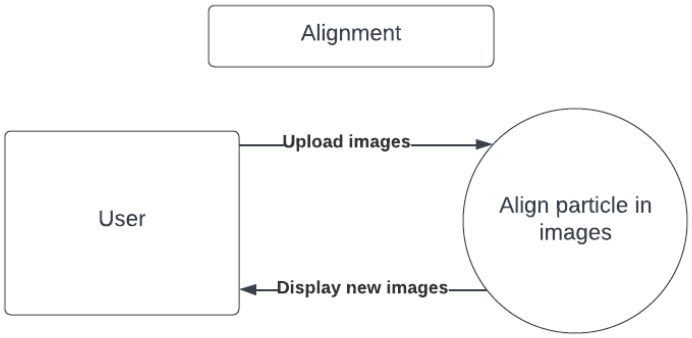
4.1. Data Flow Diagrams

Level 0 (High-Level)

The following three diagrams are high-level descriptions of the processes that each of the electron microscope software perform:



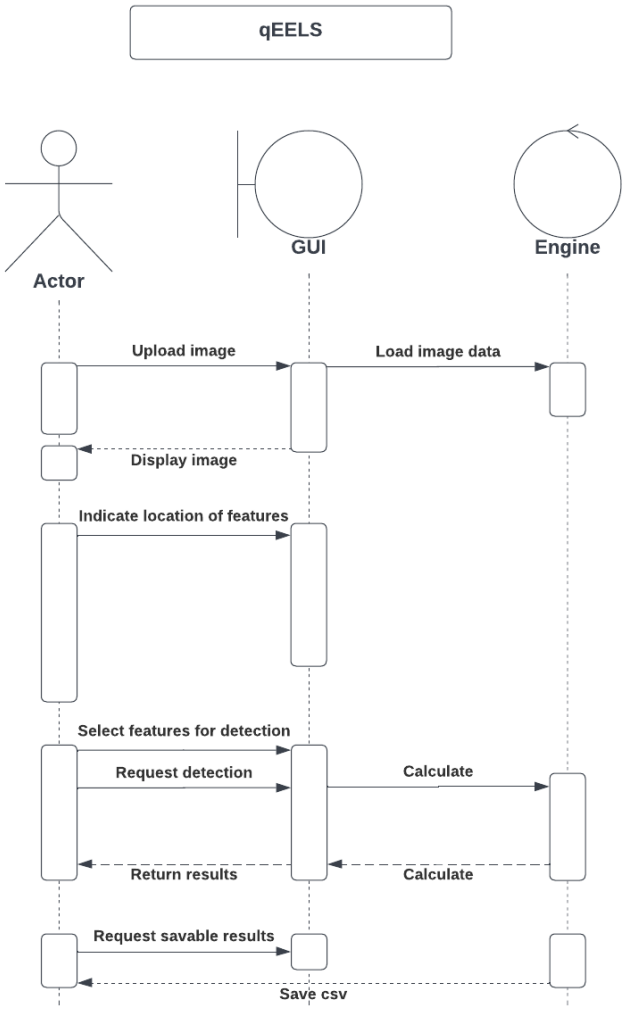




4.2. Dynamic Models

4.2.1. Sequence Diagrams

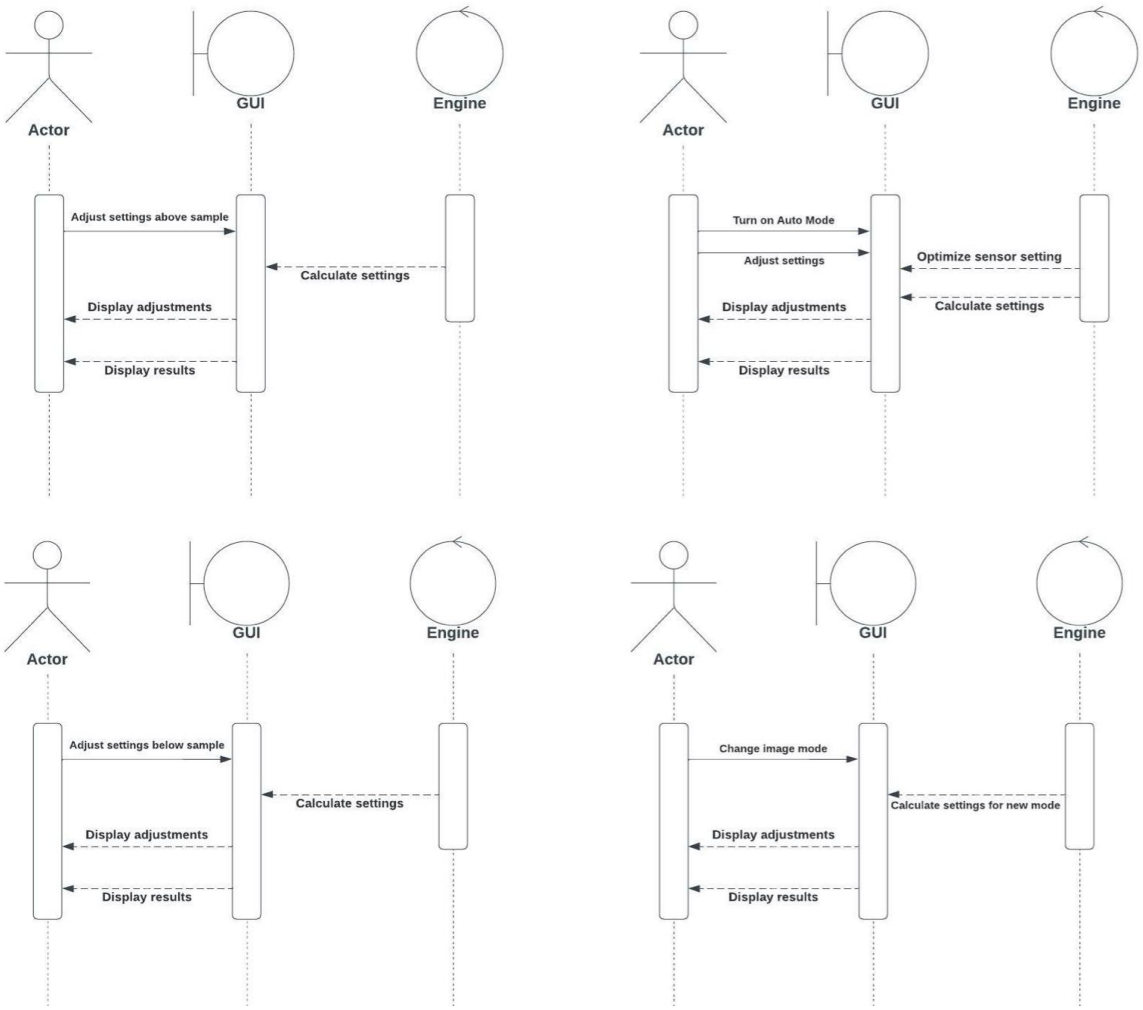
Sequence diagram of the user interface and the operations in the qEELS software:



There are several steps in calibrating energy loss, which include:

1. **Upload image**: uploading the image of the spectrogram to the software
    a. **Display Image**: the image will be displayed on the GUI
2. **Indicate location of features**: the user will indicate the location of each feature on the image
3. **Select features for detection:** the user will select which features they would like to include in the detection process
4. **Request detection:** the user will request the calculation of the energy loss based on the selected features
    a. **Return results**: the GUI will display the results as an image to the user
5. **Request savable results:** the user is now able to save the results in csv
    a. Save csv: the user will now have a csv with all results saved on their machine

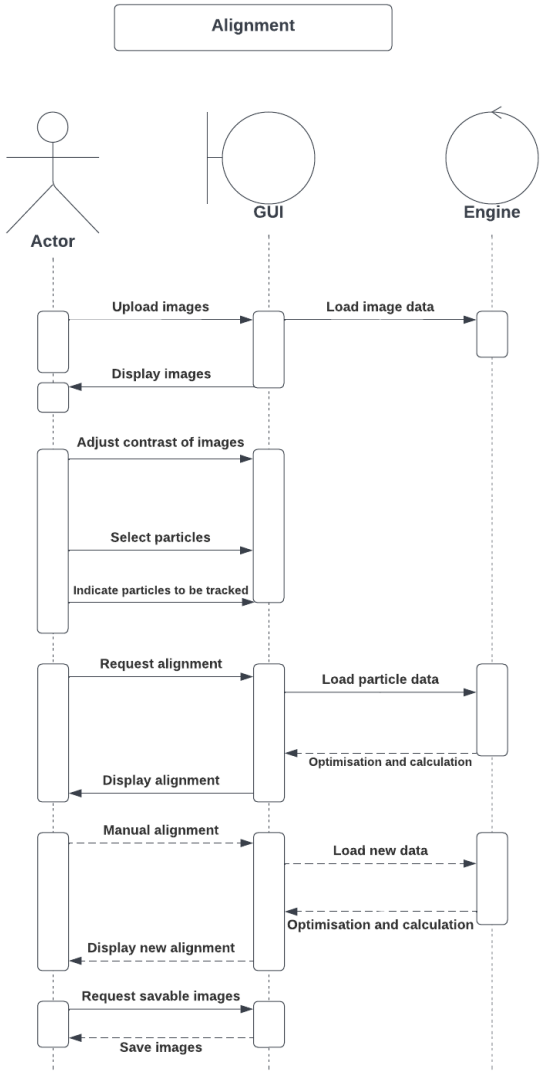
Sequence diagram of the user interface and the operations in the Nanomi Optics software. Note that the actions performed by the user in the Nanomi Optics software can be done in any preferred order, which is why there are several diagrams.



There are several steps in calculating optimized electron microscope settings, which include:

- **Adjust settings above sample**: changing the focal lengths of the sensors that sit above the sample
- **Select image mode:** the user is able to select diffraction or image mode to view the electron microscope settings
- **Turn on Auto mode:** turning on auto mode enables one of the sensors' (below the sample) settings' to be optimized while other sensors are being adjusted
- **Adjust settings below sample:** changing to focal lengths of the sensors that sit below the sample
- **Display results:** display of changing settings, and/or the optimized settings in auto mode
- **Display adjustments:** display of changing features on the diagram of the electron microscope

Sequence diagram of the user interface and the operations in the Alignment Software:

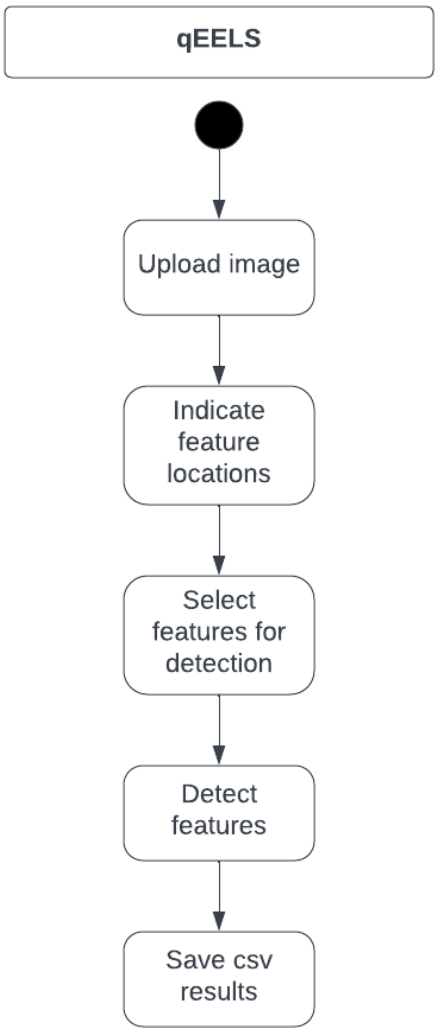


There are several steps in aligning nanoparticle images, which include:

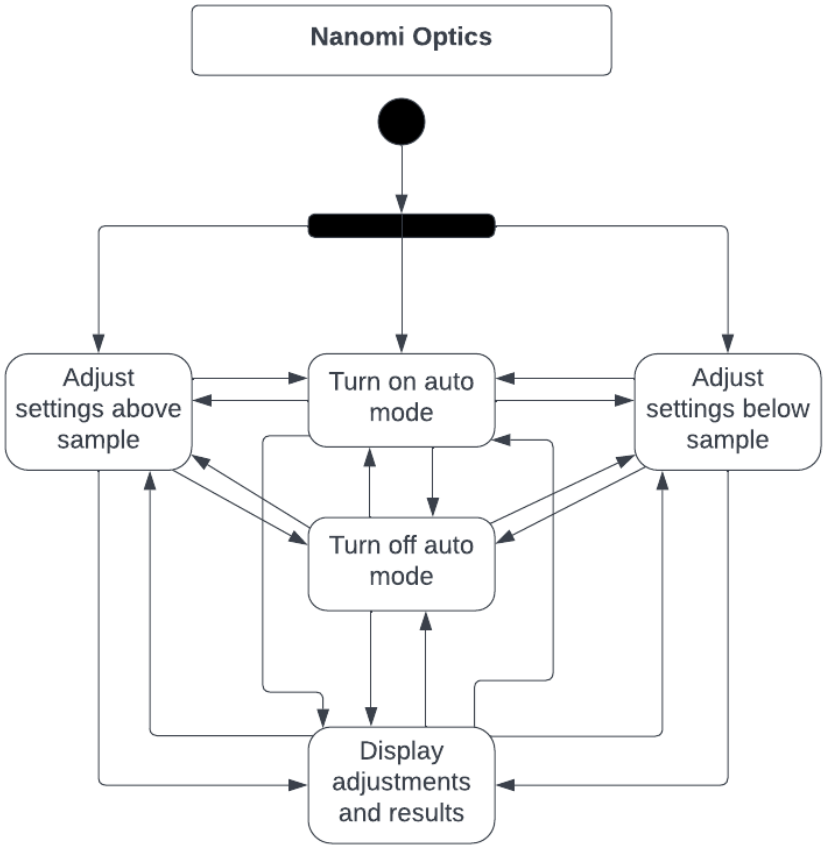
1. **Upload images**: uploading the images of the nanoparticles to the software
    a. **Display Images:** the images will be displayed on the GUI
2. **Adjust contrast of images**: adjust the contrast of the images for better viewing, and later optimisation
3. **Select particles**: the user selects all the particles in the image
4. **Indicate particles to be tracked**: indication of which particles are wanting to be tracked
5. **Request alignment:** when all parameters are set, the user requests optimized alignment
    a. **Display alignment**: the optimized aligned images are displayed to the user
6. **Manual alignment**: accepts manual adjustments of the location of particles
    a. **Display alignment**: the optimized aligned images are displayed to the user
7. **Request savable images**: when the user is satisfied with their images, they request to save
    a. **Save images**: the images are saved onto the user's machine

4.2.2. Activity Diagrams

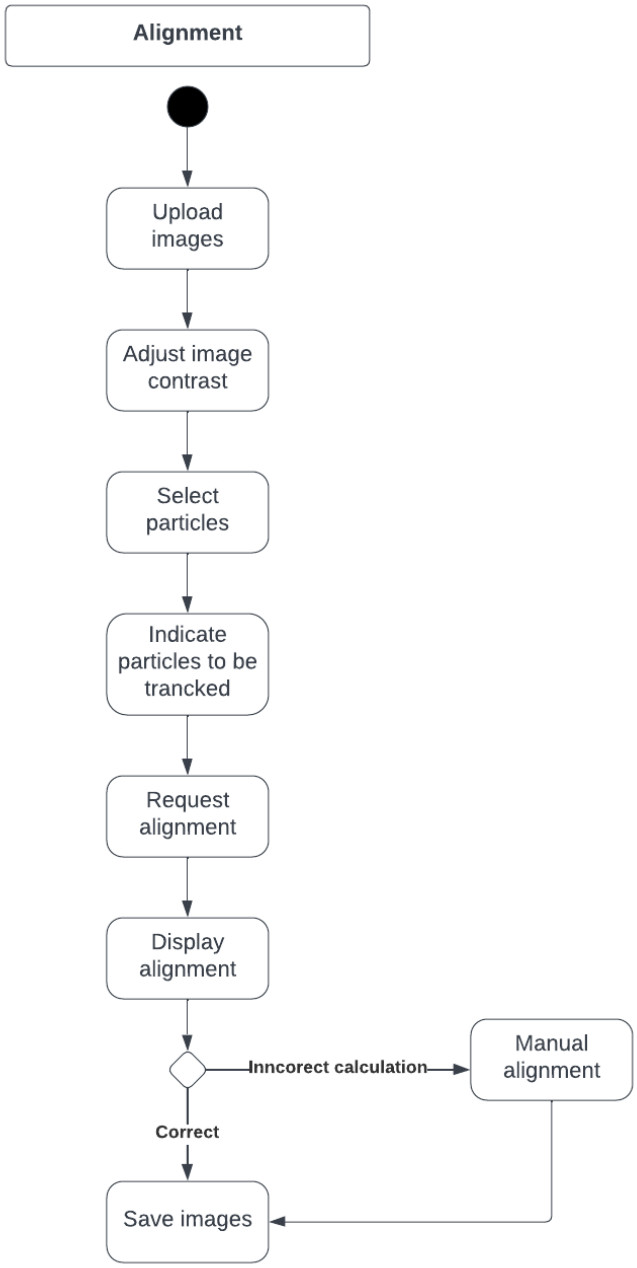
The following diagrams are descriptions of the activities that a user will go through while using each software:



The activities in qEELS must be performed in this order. The user will upload the image, indicate feature locations, select features for detection, detect features, and save the results.



The Nanomi Optics software has a very free sequence of activities. The user can adjust settings above sample, adjust settings below sample, or turn on auto mode at any time they want to. If auto mode is turned on, then they can turn off auto mode. After every adjustment that is made in the settings, the adjustments and results are displayed to the user.

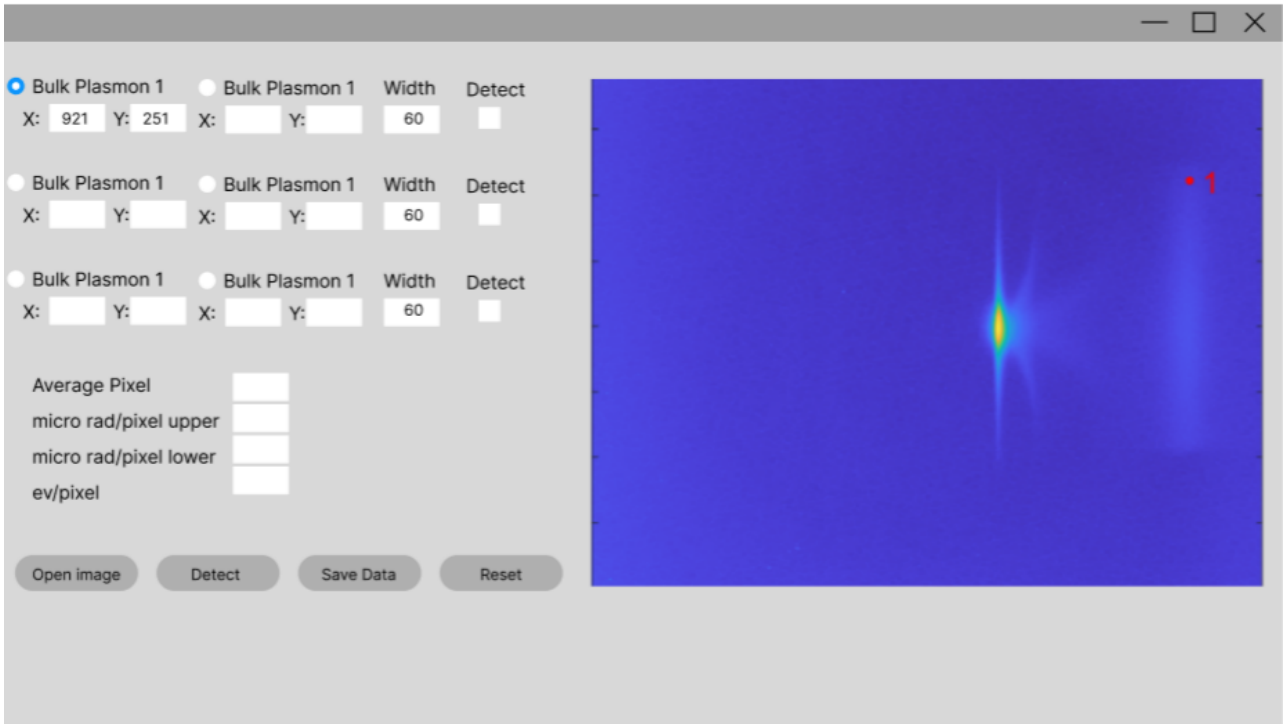


The activities for the alignment software must be done in this order. The user will upload their images, adjust the image contrast, select particles, indicate which particles are to be tracked, request alignment, and display alignment. If the alignment is displayed incorrectly, the user can manually adjust the alignment. Then the user will save the aligned images.

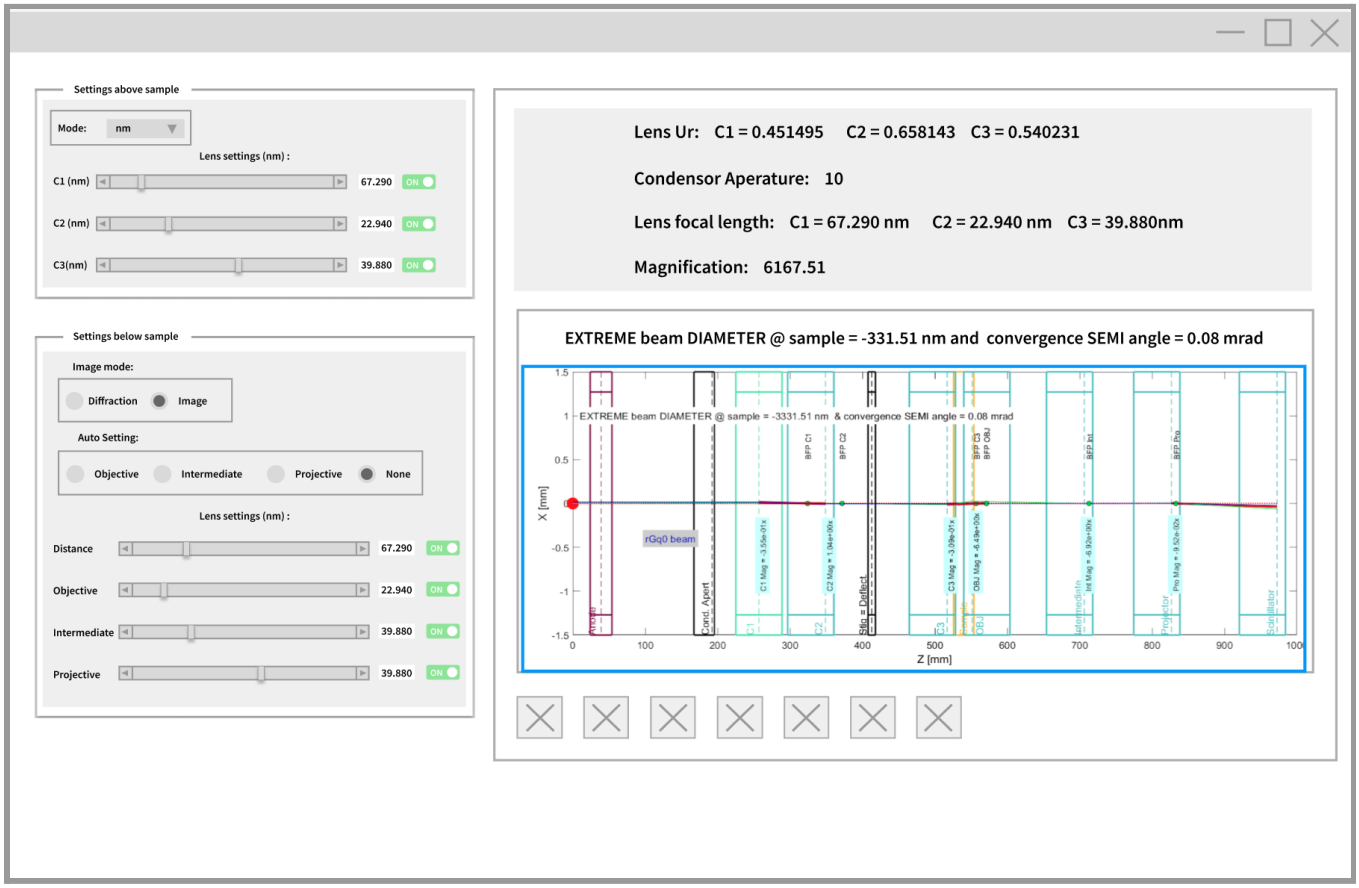
5. UI Mockups

The new software GUIs are closely based on the legacy MATLAB script GUIs to provide a sense of familiarity to the user, whilst still updating the design for improved HCI.

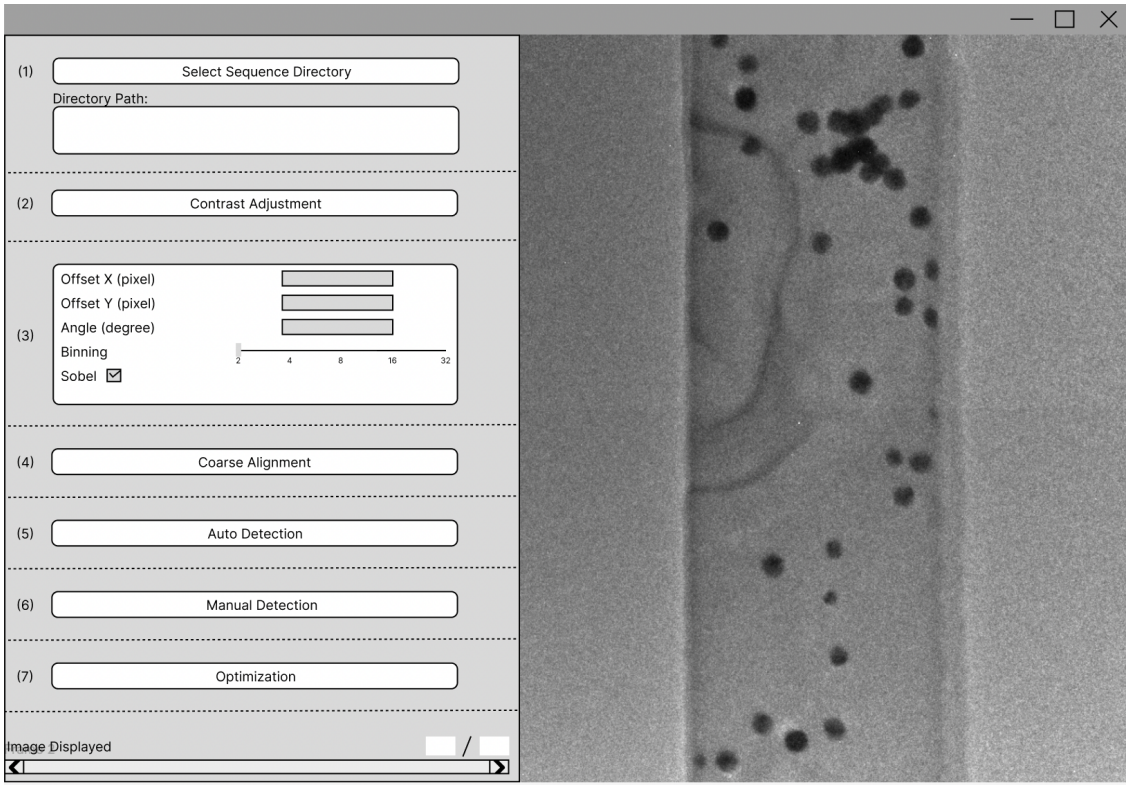
qEELS:

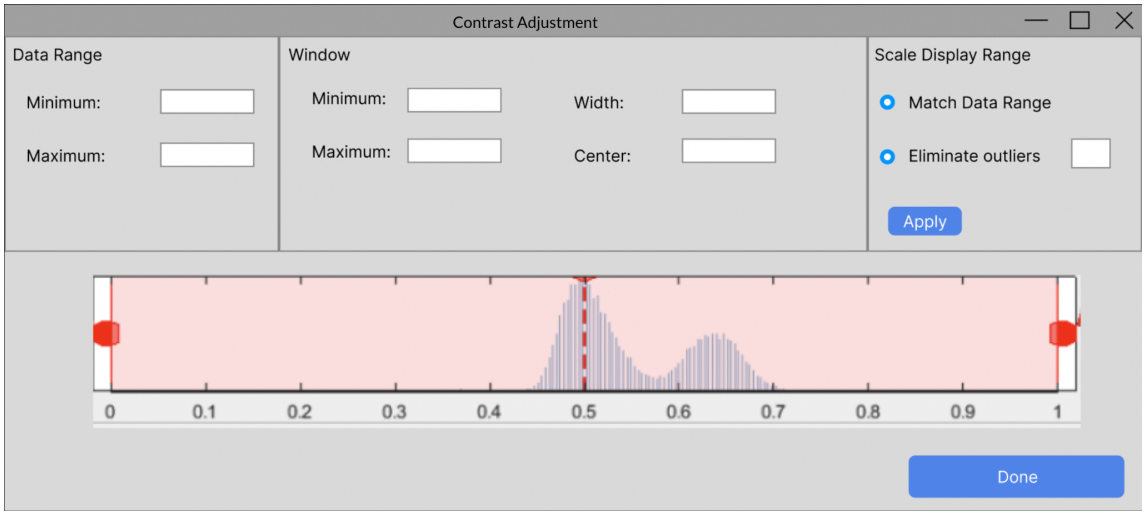


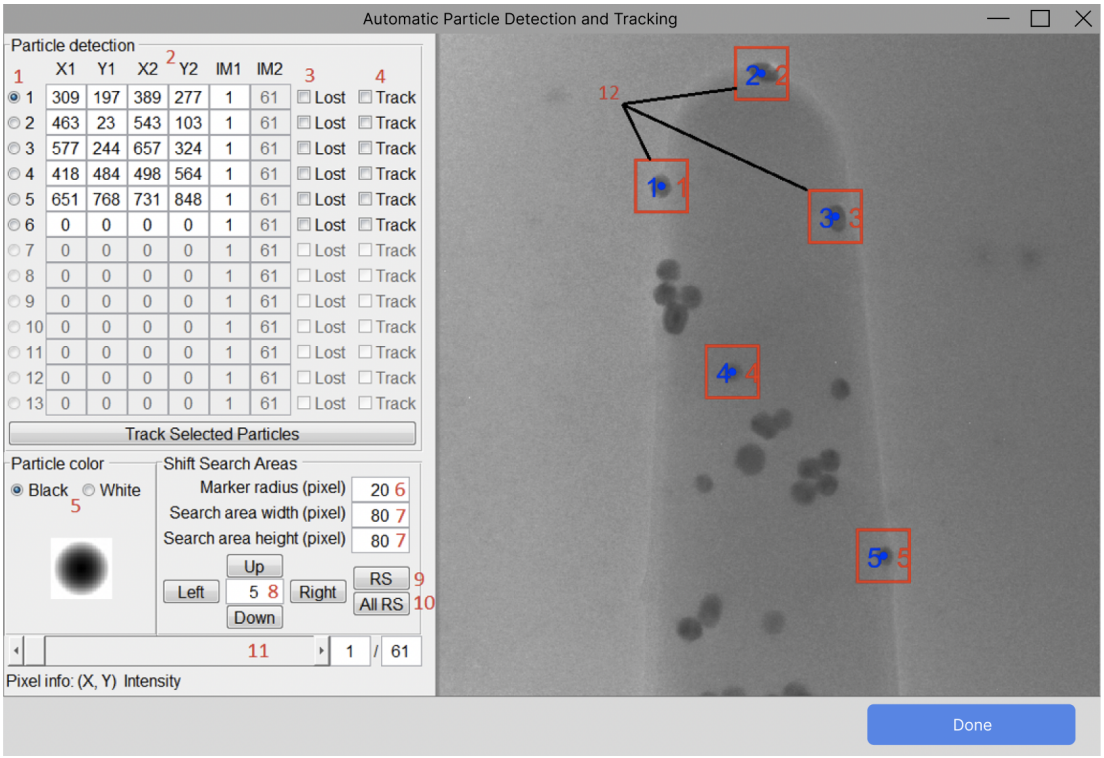
Nanomi optics:

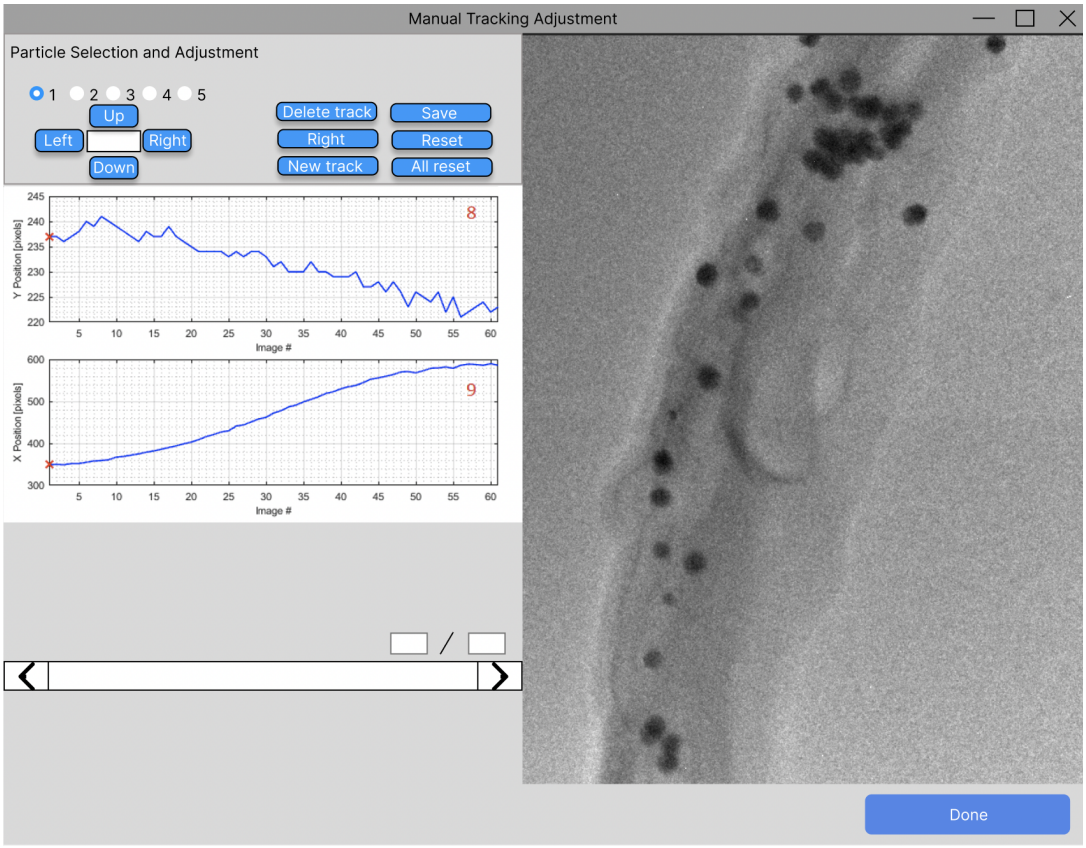


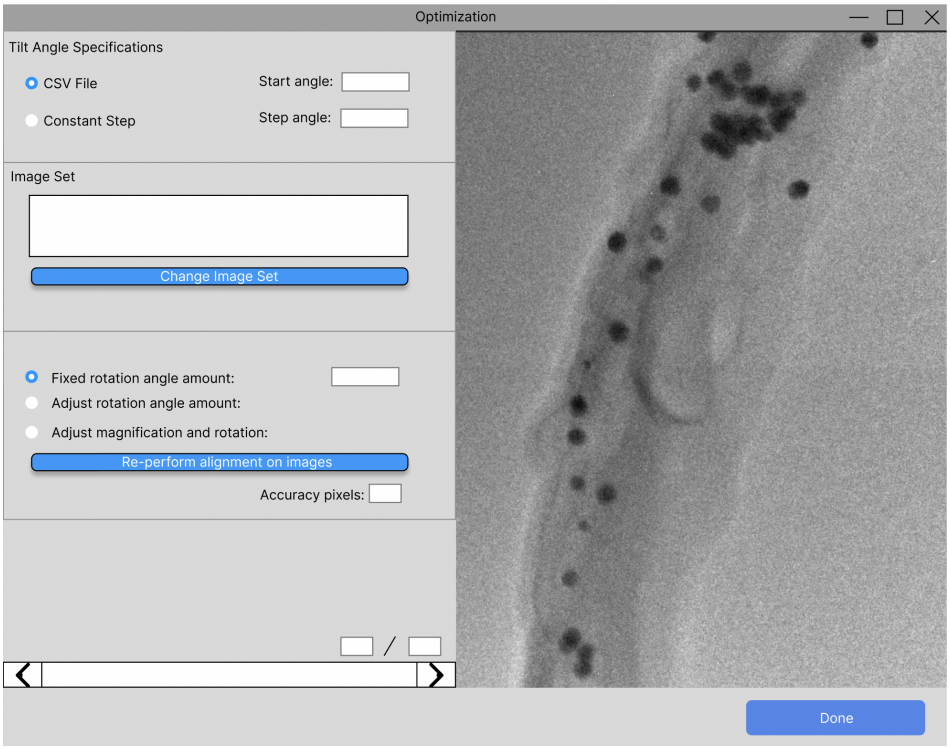
Alignment Software:






Automatic Particle Detection and Tracking
Particle detection
1
X1 Y1 X2 2 Y2 IM1 IM2 3 4
1 309 197 389 277 1 61 Lost Track
2 463 23 543 103 1 61 Lost Track
3 577 244 657 324 1 61 Lost Track
4 418 484 498 564 1 61 Lost Track
5 651 768 731 848 1 61 Lost Track
6 0 0 0 0 1 61 Lost Track
7 0 0 0 0 1 61 Lost Track
8 0 0 0 0 1 61 Lost Track
9 0 0 0 0 1 61 Lost Track
10 0 0 0 0 1 61 Lost Track
11 0 0 0 0 1 61 Lost Track
12 0 0 0 0 1 61 Lost Track
13 0 0 0 0 1 61 Lost Track
Track Selected Particles
Particle color
Black White
5
Shift Search Areas
Marker radius (pixel) 20 6
Search area width (pixel) 80 7
Search area height (pixel) 80 7
Up
Left 5 8 Right
Down
RS 9
All RS 10
11
1 / 61
Pixel info: (X, Y) Intensity
12
Done


Manual Tracking Adjustment
Particle Selection and Adjustment
1 2 3 4 5
Up
Left Right
Down
Delete track Save
Right Reset
New track All reset
8
Y Position [pixels]
Image #
9
X Position [pixels]
Image #
Done



6. Technical Specification

The following lists are not exhaustive and more will likely be expanded as the need for functionality is discovered.

Programming Languages:

- Python

Python Libraries:

- Tkinter (GUI)
- Matplotlib (Chart Rendering)
- Scipy (Optimization and other algorithms)
- Numpy (Efficient numeric arrays)
- Pillow (Image processing and transforms)
- Flake8 (Linting)
- Pytest (Unit Testing)

Supporting Technologies:

- Github
- Github Actions (CI/CD)
- Github Projects (Project Management)

7. Test Plan

We will take several measures to ensure that the code we produce is of good quality. We will set up Github Actions workflows to run all tests automatically when code changes are proposed. In our repository, there will be a few restrictions so that we can double check our code. Pushing code directly to the main branch will be blocked. Attempting to merge a pull request without all the tests passing will be blocked. All pull requests will require an approving review form from at least one other team member before merging. Pull requests will not be approved unless they include tests that demonstrate appropriate functionality. All code will be linted with Flake8 and if any warnings or errors are detected, merging will be blocked. We will write our unit tests with Pytest.

To make our project testable, we will structure our project with testing in mind. The project's GUI will be difficult to test without excessive mocking, because there are not any well-supported frameworks for testing tkinter GUIs. Because of this, anything tightly integrated with the GUI will also be difficult to test. Therefore, we will isolate backend logic from the GUI to make sure that it can be tested thoroughly. The backend and GUI portions of each program will be in separate Python packages. The backend will not be allowed to make any calls or references to the GUI or import any GUI related package. The GUI must only make calls to the backend via clearly defined and well tested interfaces.

Even with an isolated backend, several components of the three software being converted will be non-trivial to test. For example, the Nanomi Optics and Alignment Software use optimization algorithms, which could vary from the output of the legacy software since we will be using different tools. Additionally, optimization may produce results that are not the exact same run-to-run or machine-to-machine. For example, the Alignment Software makes transformations to images that minimize the displacement of particles between frames. For these scenarios, we will need to use threshold tests to determine if a proper amount of optimization has occurred. So, rather than testing for a specific output transformed image, we would instead test whether the particle's movement was successfully minimized within an acceptable threshold in the output image within a threshold.

Another example where testing will be non-trivial is in calculations that involve floating-point numbers. All three of the software use floating numbers extensively. Real numbers cannot always be represented exactly by a floating-point number, so there will be some loss in precision when operations are performed. This loss of precision may not be consistent between the legacy software and the new software. This is because tools that perform the same mathematical operations in the new software may not do calculations in the exact same order

as the legacy software, resulting in very slight differences in results. To solve this we will again, like in the optimization problems, use thresholds to allow an acceptable range of answers.

We also need to create tests that verify functions that have very large binary inputs and outputs. An example of this is functions that return large images as outputs. For example, the Alignment software will have functions that perform image operations such as contrast adjustment and kernel convolution which will produce large outputs. One approach could be to have an exact copy of the expected output, but this would be difficult to hand craft. Instead, for testing image operations we will use smaller sample images for test cases which will make crafting test cases easier.