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

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.

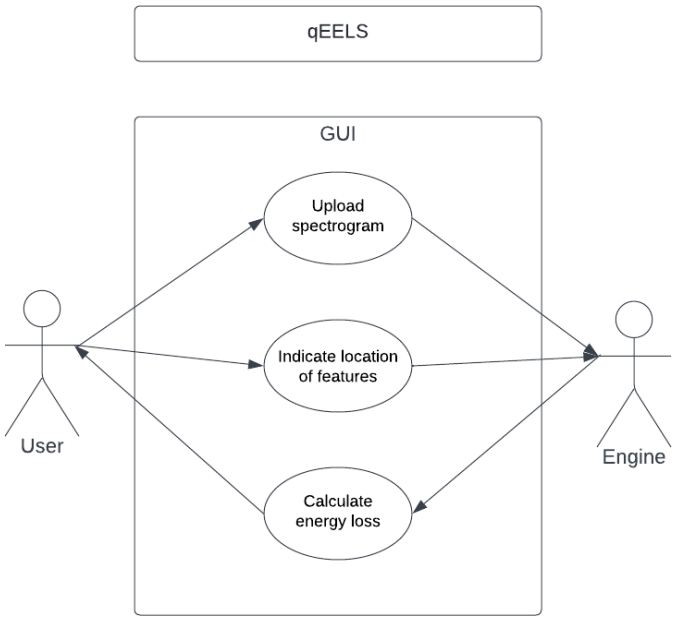
Name:	Nanomi Optics
Description:	Optimize electron microscope optics settings.
Flow of Events:	1. The user opens the software. 2. User inputs their desired parameters for optics 3. User request optimized lens settings
Pre-conditions:	● The user must have the software open. ● The user has knowledge of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

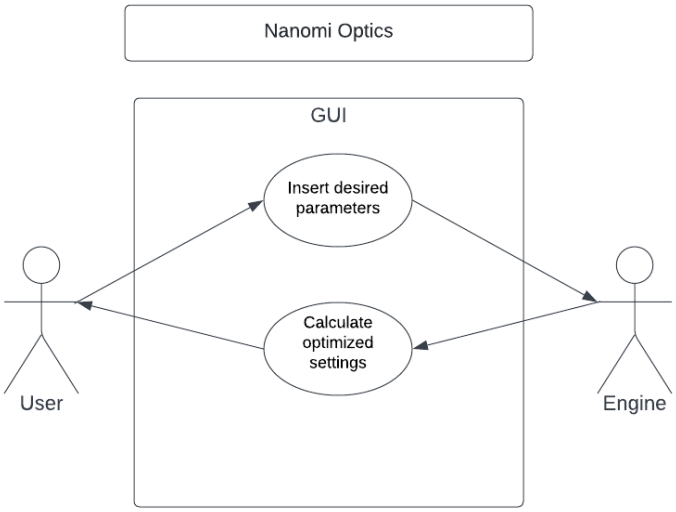
Name:	Alignment Software

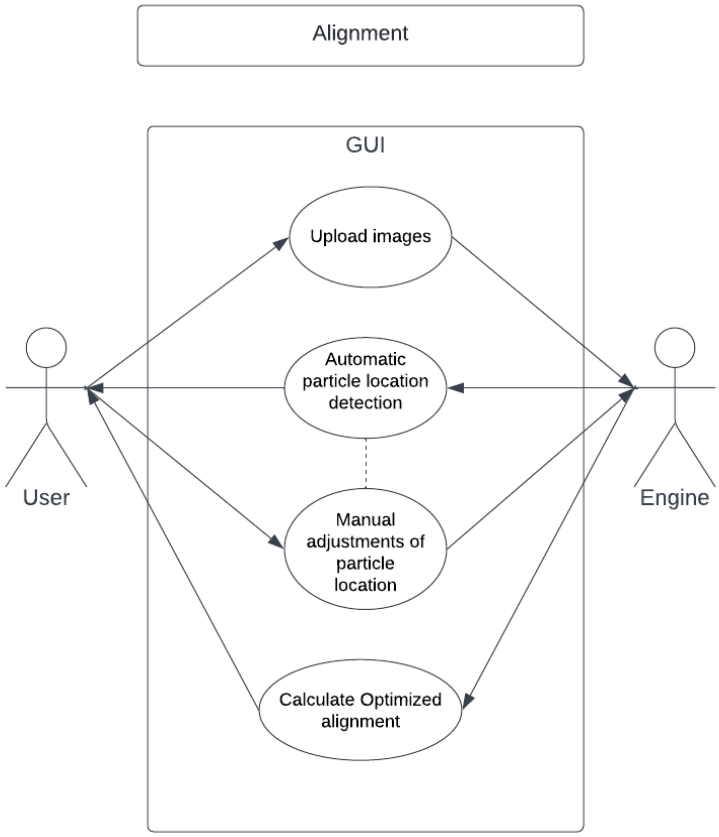
Description	Align nanoparticle images.
Flow of Events:	1. The user opens the software. 2. The user uploads nanoparticle images. 3. The user selects the location of the particle on the image. 4. The user requests aligned sequence images from software.
Pre-conditions:	● The user must have the software 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

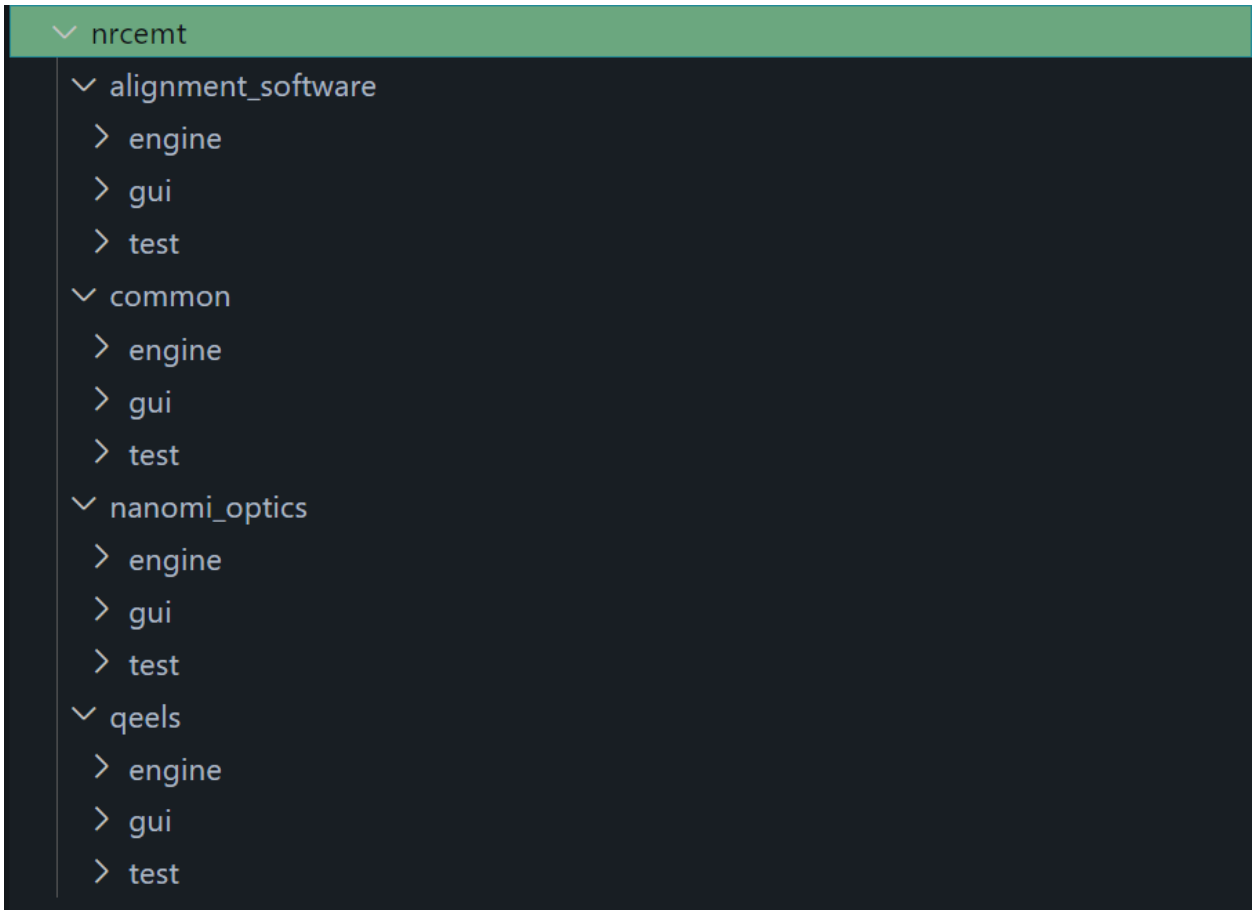



Nanomi Optics
GUI
Insert desired parameters
Calculate optimized settings
User
Engine


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

3. System Architecture

Our system will be broken up into python packages as follows:



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.

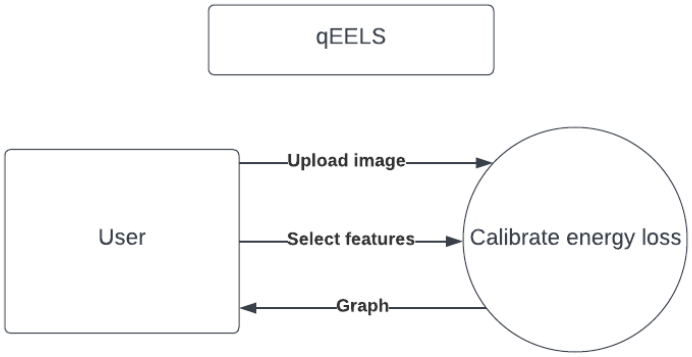
Each *gui* package should not include any data processing and should instead delegate all of that in calls to its respective engine. Each type of call made from a *gui* to an *engine* package should be tested. The *gui* should focus only on handling user interaction.

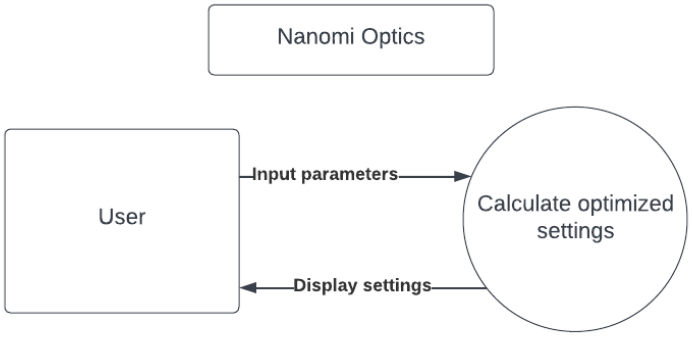
4. Diagrams

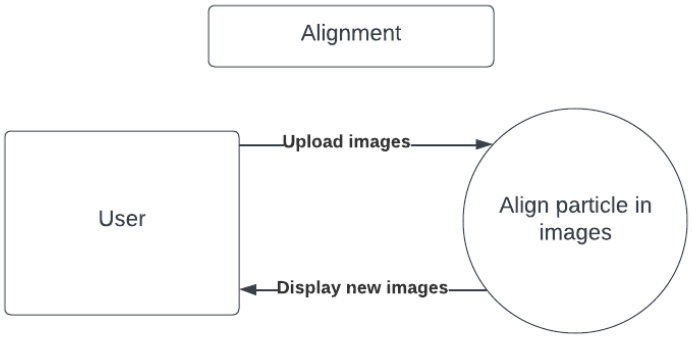
4.1. Data Flow Diagrams

Level 0 (High-Level)

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



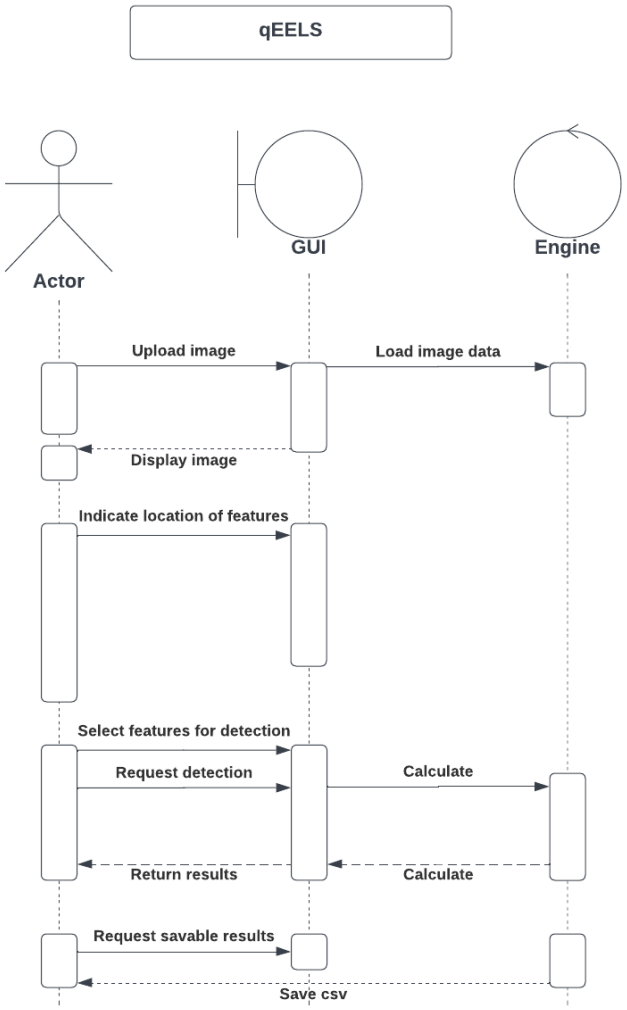




4.2. Dynamic Models

4.2.1. Sequence Diagrams

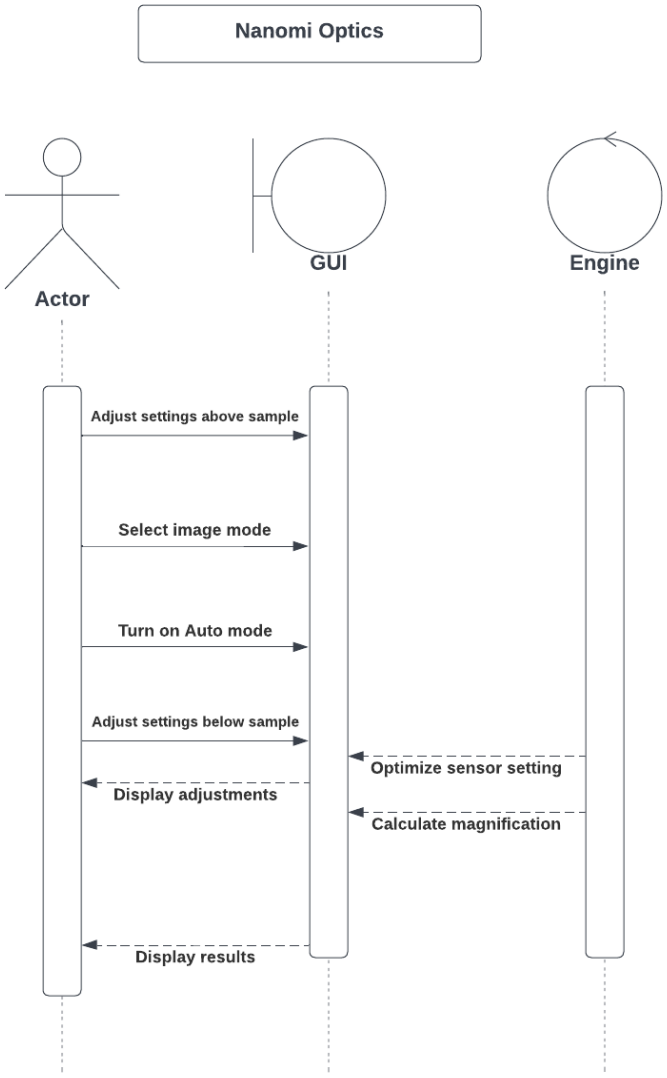
Sequence diagram of the user interface and the operations in the qEELS software:



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

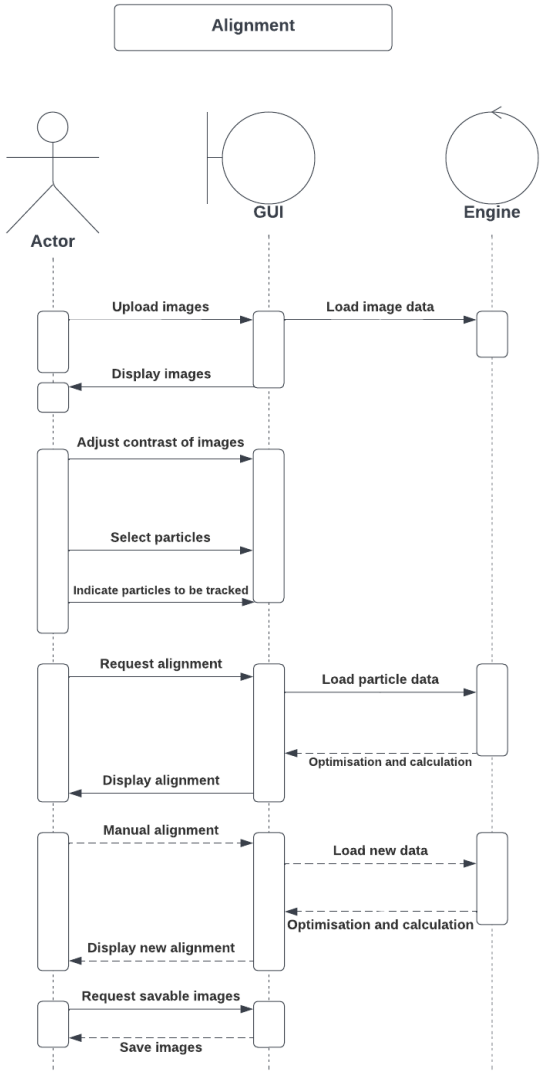
Sequence diagram of the user interface and the operations in the Nanomi Optics software. Note that the steps in the Nanomi Optics software can be done in any preferred order by the user.



There are several steps in calculating optimised 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 optimised 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 optimised 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:

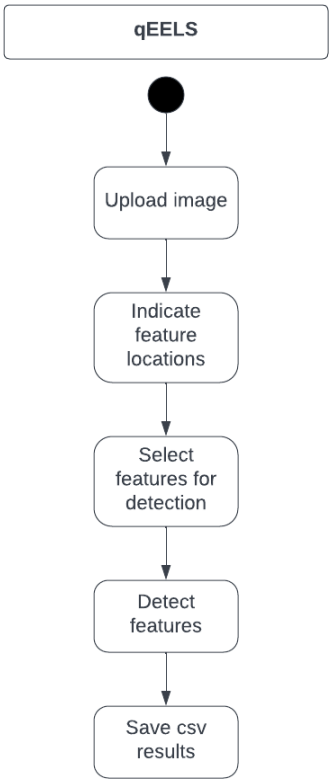


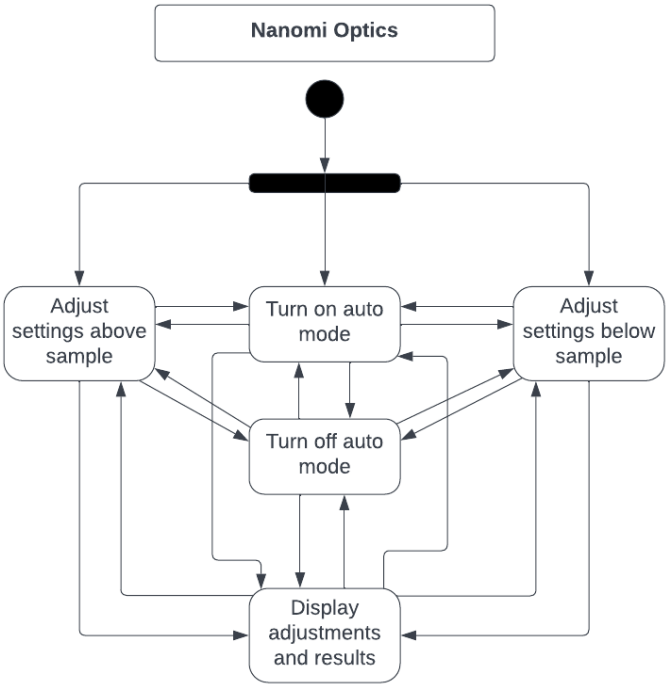
There are several steps in aligning nanoparticle images, which include:

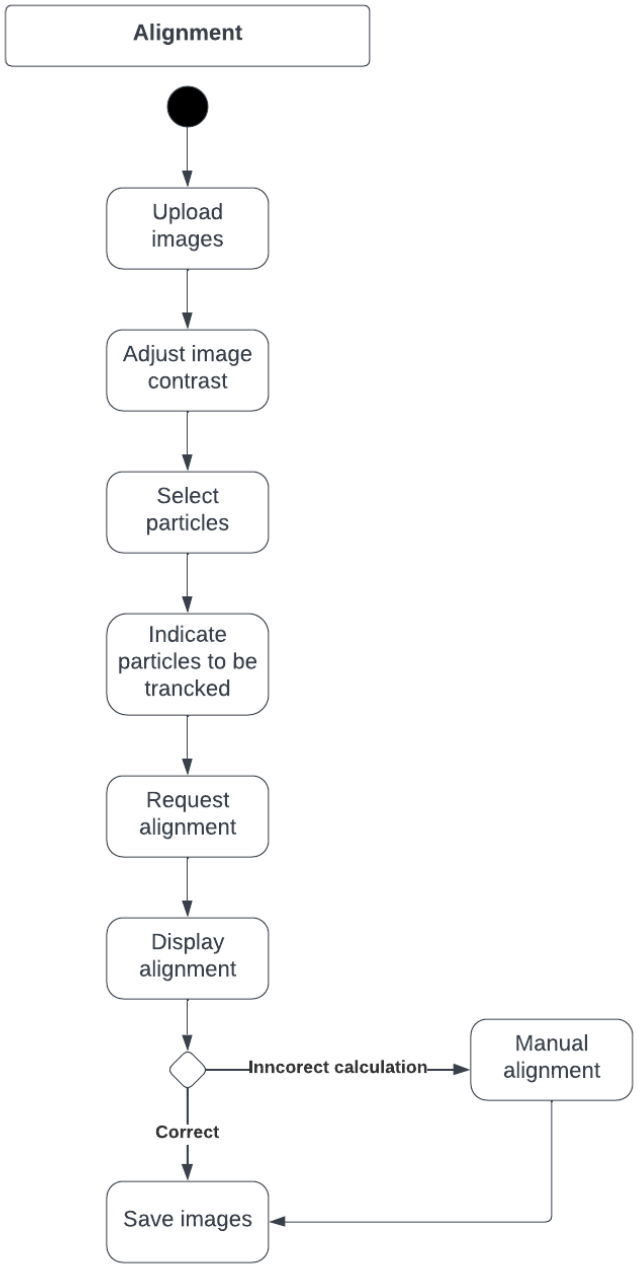
1. **Upload images**: uploading the images of the nanoparticles to the software
    a. **Display Images:** the images will be displayed on the GUI
2. **Adjust contrast of images**: adjust the contrast of the images for better viewing, and later optimisation
3. **Select particles**: the user selects all 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 optimised alignment
    a. **Display alignment**: the optimised aligned images are displayed to the user
6. **Manual alignment**: accepts manual adjustments of the location of particles
    a. **Display alignment**: the optimised 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:



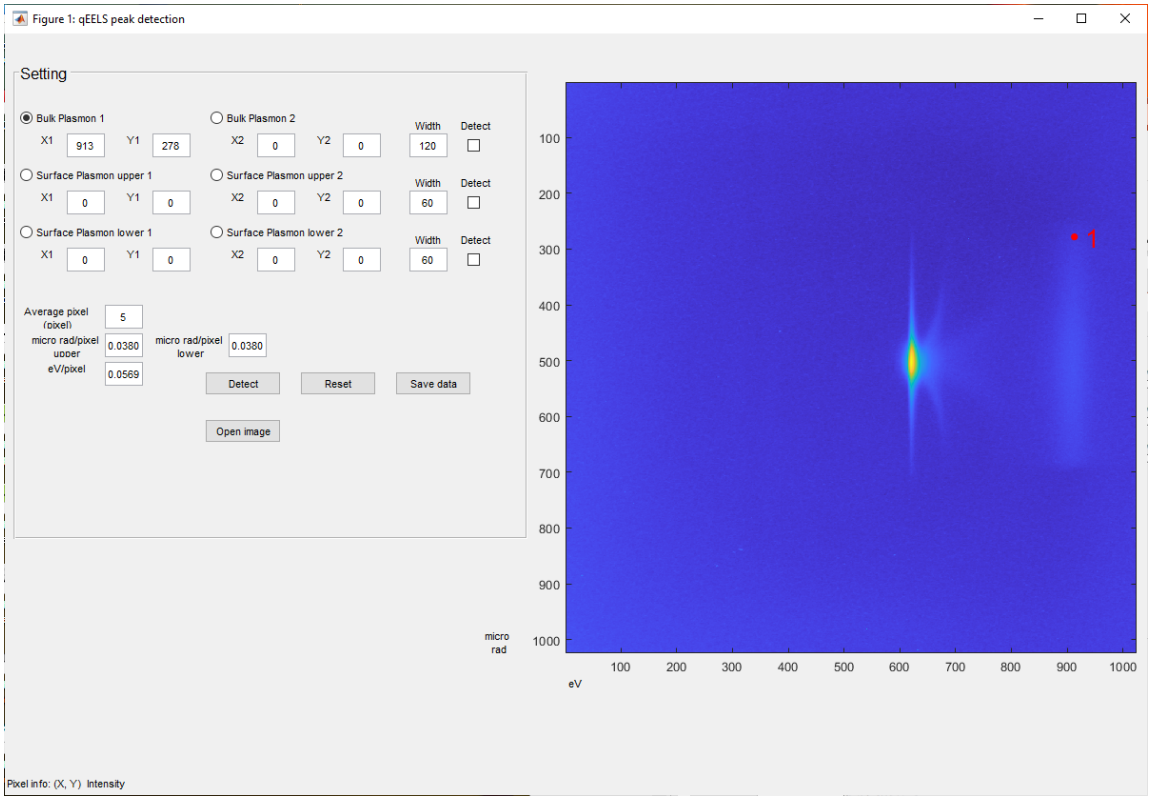

Nanomi Optics
Adjust settings above sample
Turn on auto mode
Adjust settings below sample
Turn off auto mode
Display adjustments and results



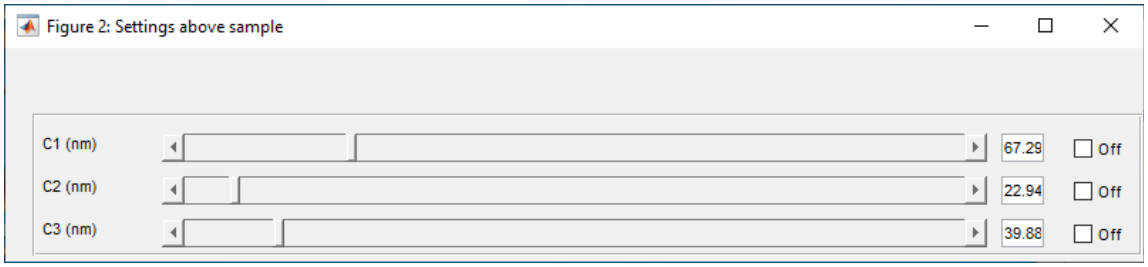
5. UI Mockups

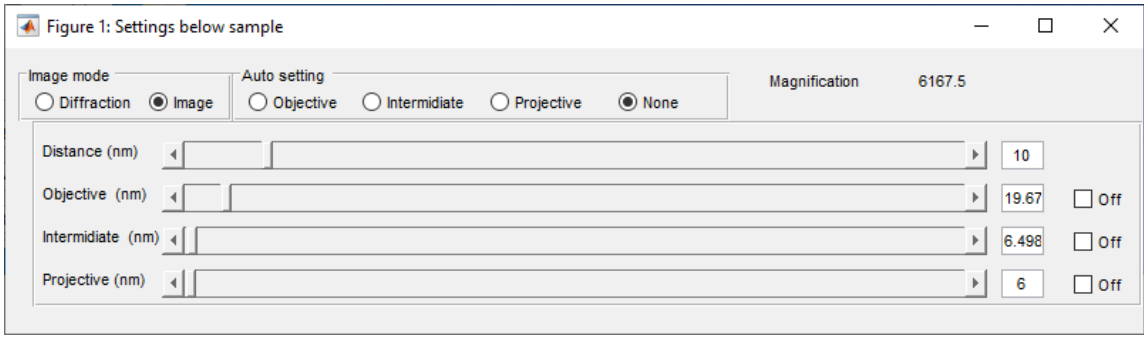
The new software GUI will be closely based on the legacy MATLAB scripts. Some adjustments will likely be made for better usability and design, but we will maintain familiarity with the users.

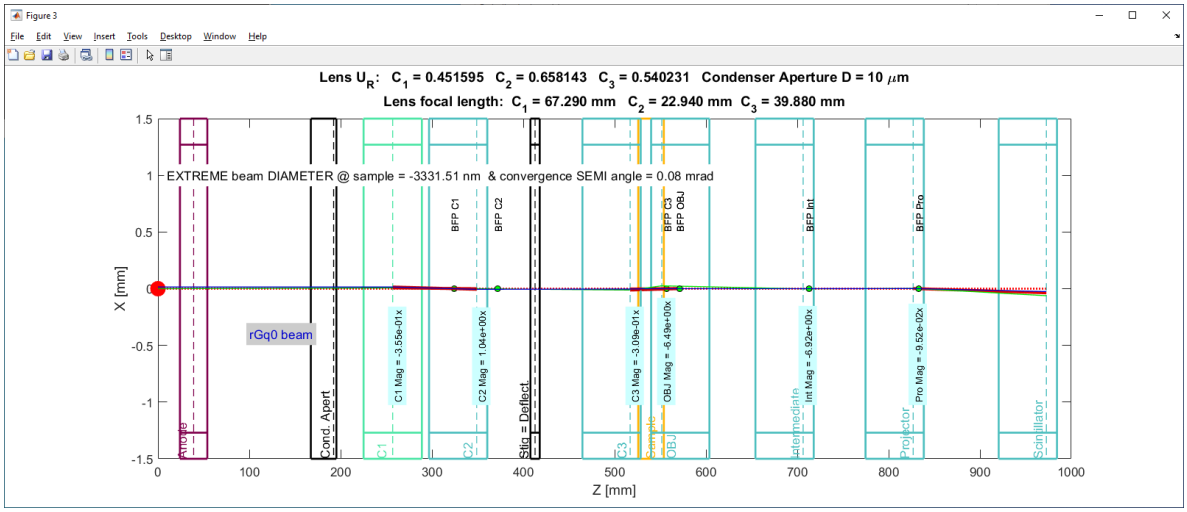
qEELS:



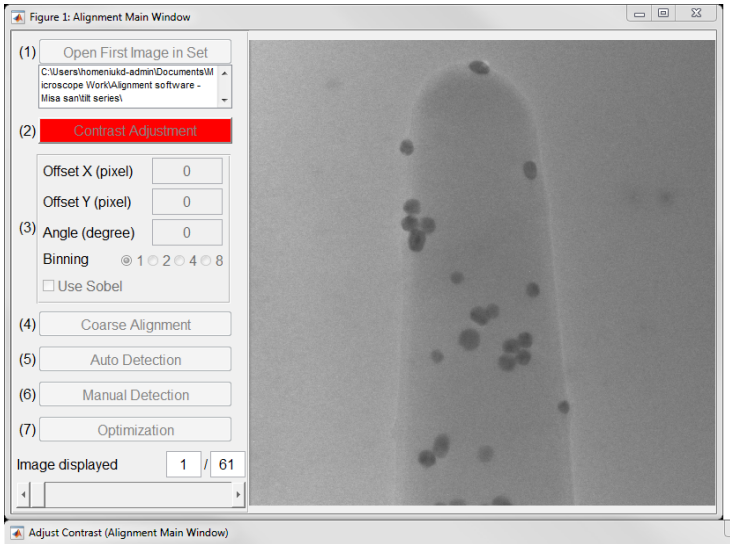
Nanomi optics:

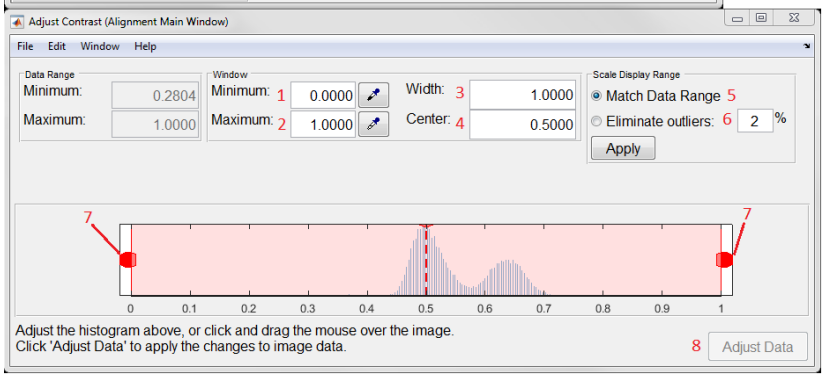






Alignment Software:





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

To ensure code quality we will set up Github Actions workflows to run all tests automatically when code changes are proposed. Pushing to the main branch directly will be blocked. Merging pull requests without all tests passing first will be blocked. All pull-requests will require an approving review form from at least one other team member. Pull-requests should not be approved unless they include tests which demonstrate the functionality they implement. Pytest will be used for writing unit tests. In addition to traditional unit tests, all code will be linted with Flake8 and if any warnings or errors are detected, merging will be blocked.

In-order to make our project testable, we must structure our project with testing in mind. The project's GUI will be difficult to test without excessive mocking, as there aren't any well-supported frameworks for testing tkinter GUIs. Because of this, anything tightly integrated with the GUI will also be difficult to test. For this reason, we must isolate backend logic from the GUI in-order to make sure that it can be tested thoroughly. The backend and GUI portions of each program will be clearly into 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. Especially for Nanomi Optics and the Alignment Software who use optimization algorithms which may vary from the output of the original software. Additionally, optimization may produce results that are not exactly the same run-to-run or machine-to-machine. For these scenarios, we will need to use some threshold tests to test the underlying conditions that the software is optimizing for. For example, the Alignment Software makes transformations to images to minimise the inter-frame displacement of particles. Rather than testing for a specific output transformed image, we would instead test whether the particle's movement was successfully minimised within an acceptable threshold in the output image within a threshold.