Journal of
open research software

**SOFTWARE METAPAPER**

# PLEASE: The Python Low-energy Electron Analysis SuitE – Enabling Rapid Analysis of LEEM and LEED Data

Maxwell Grady, Zhongwei Dai and Karsten Pohl
University of New Hampshire Department of Physics and Materials Science Program, US
Corresponding author: Maxwell Grady, Lead (max.grady@gmail.com)

PLEASE, the Python Low-energy Electron Analysis SuitE, provides an open source and cross-platform graphical user interface (GUI) for rapid analysis and visualization of low energy electron microscopy (LEEM) data sets. LEEM and the associated technique, selected area micro-spot low energy electron diffraction ($\mu$-LEED), are powerful tools for analysis of the surface structure for many novel materials. Specifically, these tools are uniquely suited for the characterization of two-dimensional materials. PLEASE offers a user-friendly point-and-click method for extracting intensity-voltage curves from LEEM and LEED data sets. Analysis of these curves provides insight into the atomic structure of the target material surface with unparalleled resolution.

**Keywords:** Physics; Materials Science; Electron Microscopy; Electron Diffraction; Surface Science; Python; 2D Materials

## (1) Overview

### Introduction

Low energy electron microscopy, LEEM, and the associated technique of micro-spot low energy electron diffraction, $\mu$-LEED, provide a unique avenue for atomic level surface structure determination with sub-Å resolution [3], (1Å = $10^{-10}$m). These characterization tools are useful for the investigation of a wide gamut of novel materials such as graphene, black phosphorus, and other two-dimensional (2D) materials. The electronic structure of 2D materials is inherently linked to the surface structure as a result of reduced dimensionality, thus, in order to more fully understand the electronic properties of these materials, their surface structure must be determined with high resolution. Furthermore, detailed understanding of the surface structure will help to guide future applications of these materials.

The process of surface structure determination with sub-Å resolution for ultra thin or 2D materials remains a complicated task. There are a number of experimental techniques that are surface sensitive, however, many are not suited to the study of two-dimensional materials. Many 2D materials can currently only be created in small micron-sized flakes with one to few layer thickness. LEEM and $\mu$-LEED provide an excellent method for collecting experimental data from a wide variety of materials including 2D materials with small surface area, specifically due to the non-destructive nature of the technique, and its very small data acquisition area of order 1 $\mu$m. Surface structure information can be obtained from LEEM and $\mu$-LEED experiments through analysis of intensity versus voltage, *I(V)*, data sets. These data sets map the relationship between the intensity *(I)* of the reflected or diffracted electrons and the energy *(V)* of the incident electron beam. LEEM and LEED *I(V)* data sets are sensitive to the target material's atomic surface structure and composition in all three dimensions with sub-Å resolution [3, 8].

The general outline of surface structure determination using LEEM and LEED proceeds as follows. First, experimental data is acquired using either LEEM or LEED operation mode. *I(V)* data sets are collected by varying the energy of the incident electrons in fixed steps and recording each resulting image. Once the data has been collected, the next step is to extract *I(V)* curves from the data set from relevant parts of the images. The PLEASE software package aims to make this step of the surface structure determination easy and user friendly. *I(V)* curves extracted using PLEASE can be output to text for further processing. Next, the experimental data must be compared to computational models. This is done by performing dynamic electron multiple scattering calculations to simulate the *I(V)* relationship. Already, software packages exist for computational generation of LEED-*I(V)* curves based on multiple scattering theory [7, 9, 10]. There is also active research in using first-principles calculations to generate LEEM-*I(V)* curves for novel materials [2, 4, 5]. The computational process iterates by comparing the experimental data to the calculated data and adjusting the modeled surface structure until sufficient agreement is found between experiment and calculation. The final structure input in this procedure can then be considered the optimized surface structure and composition.

At the time of initial development for this project, there were no open source solutions for specifically designed for the analysis of LEEM and $\mu$-LEED data. The PLEASE software package fills this gap by providing an open source and cross platform graphical user interface for rapid visualization and analysis of LEEM and LEED data. While there are many steps to the overall process of surface structure determination, emphasis in this software is placed on ease of use in the extraction of *I(V)* curves from LEEM and LEED data sets and analysis of background signals. The software package adds to the already rich ecosystem of scientific software written in python, and provides a simple interface for interacting with LEEM data.

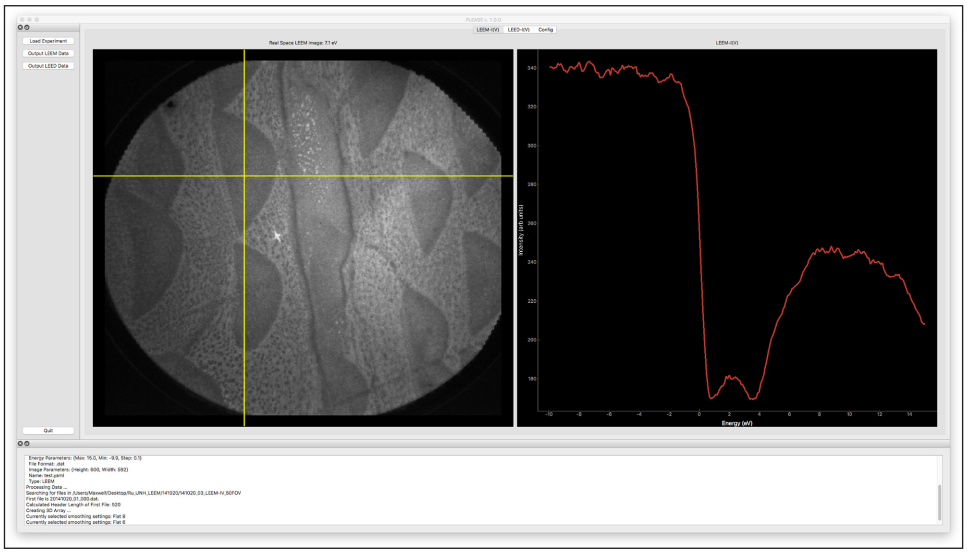
## Examples and Usage

The main data constructs in PLEASE are LEEM and LEED *I(V)* data sets, which are represented natively as three-dimensional arrays using the python NumPy library for fast and efficient storage. The first and second array axes represent the image axes. The third array axis represents the voltage of the incident electron, or rather, its kinetic energy. The *I(V)* data set, can thus be envisioned as a vertical stack of images with the vertical axis representing electron imaging energy. Each two-dimensional slice of the three-dimensional array constitutes an image of the sample in real space (LEEM) or reciprocal space (LEED) at a fixed incident electron energy as shown in **Figure 2**. **Figure 1** demonstrates an example of real space LEEM data analysis, whereas **Figure 3** demonstrates reciprocal space LEED analysis.

In a LEEM data set, an *I(V)* curve is generated by taking a vertical slice through the array and plotting the intensity of each pixel in the slice versus the incident electron energy from that slice (*I* as a function of *V*). **Figure 1** demonstrates this functionality. The left-hand image shows a single image from a LEEM *I(V)* set of graphene islands on a ruthenium substrate with a 50 $\mu$m field of view collected at a fixed incident electron energy of 7.1 eV. The right-hand side image displays the *I(V)* curve extracted from the pixel designated by the yellow cross-hair in the left image. The user interface tracks the mouse movement within a LEEM image and automatically extracts the electron *I(V)* curve and displays in the right hand plotting area in real-time. Optionally, the user can choose to apply a data smoothing method to the displayed *I(V)* curve. The smoothed *I(V)* curve is calculated via convolution of the input signal with a pre-defined and user selectable window function. While data smoothing is enabled, LEEM-*I(V)* curves are still plotted in real-time by tracking user mouse movement in the LEEM image plot.

The extraction of relevant *I(V)* curves from LEED data sets is slightly more complicated, however the user interface makes the process easy for the end user. In LEEM analysis, by default, the *I(V)* curves are extracted from a one pixel wide vertical slice of the main data array. For LEED images, the intensity of an entire electron beam spot must be recorded. Thus, the user selects one or more electron beams with an adjustable sized square window. The average intensity per image from the window region is plotted as a function of incident electron energy, as shown in **Figure 3**. Optionally, for LEEM-*I(V)* analysis the user can chose to extract an average *I(V)* from a rectangular window.

Finally, when analyzing LEED data sets, it is often useful to extract *I(V)* data not only from the reflected and diffracted electron beams but also from the surrounding regions representing a local background signal. PLEASE provides an automated way to extract the local background signal for a given user selection. The background *I(V)* curves can then be output to tab-delimited text alongside the user selected data. An example of the automated background selection process is shown in **Figure 4**.



```
  Energy Parameters: {Max: 15.0, Min: -9.9, Step: 0.1}
  File Format: .dat
  Image Parameters: {Height: 600, Width: 592}
  Name: test.yaml
  Type: LEEM
Processing Data ...
Searching for files in /Users/Maxwell/Desktop/Ru_UNH_LEEM/141020/141020_03_LEEM-IV_50FOV
First file is 20141020_01_000.dat.
Calculated Header Length of First File: 520
Creating 3D Array ...
Currently selected smoothing settings: Flat 8
Currently selected smoothing settings: Flat 6
```

**Figure 1:** 50 $\mu$m field of view bright field LEEM image of graphene islands (dark areas) atop a ruthenium single crystal substrate (light). Image collected with incident electron energy of 7.1 eV. The *I(V)* curve is extracted from an area on the graphene island marked by the yellow cross-hair.

**Implementation and architecture**

PLEASE is written purely in the interpreted programing language, python, with the user interface based on the Qt C++ application framework via the python bindings, PyQt and PyQtGraph. Python was chosen for this project for a number of reasons. First and foremost, python is cross-platform and open source, which is beneficial for reaching the largest audience in the scientific community. Second, python has been well accepted in the scientific community as an excellent resource for scientific computing due to its well established set of third-party libraries, which are often referred to as the "scientific-stack" [6]. Third, in general, python features lower development time compared to many other modern languages. Simply put, it was quicker to learn to write a full application in python rather than C++. Finally, the python language emphasizes readability, which is crucial for promoting reusability in scientific programing.

To elaborate on the second point, its worth noting that at its core, python provides a very high degree of extensibility and interoperability for usage of code from other languages, namely C/C++ and Fortran [6]. Thus, the success of the python "scientific stack" stems from the ability to create python wrappers around heavily optimized C, C++, and Fortran libraries. This allows the user to offload the heavy lifting in numeric code to other languages, while retaining the ability to write their code in a high-level readable language. For example, rather than reinventing the wheel, the python NumPy library will link to standard math libraries such as BLAS, Intel MKL, etc. The impact this has on writing scientific code in python is two fold. First, python features low development time for numeric code as a result of not needing to write in low-level languages. Finally, python can leverage the speed of optimized C or Fortran code while retaining a high-level and readable syntax. This is an important aspect of scientific software, where often reproducibility and reusability are required.

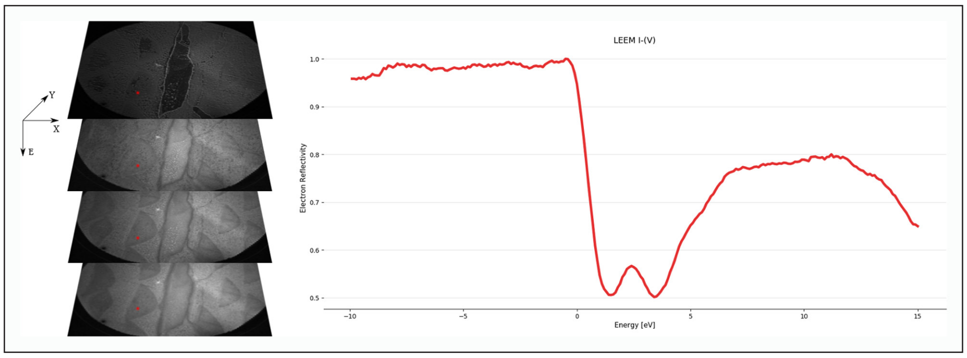
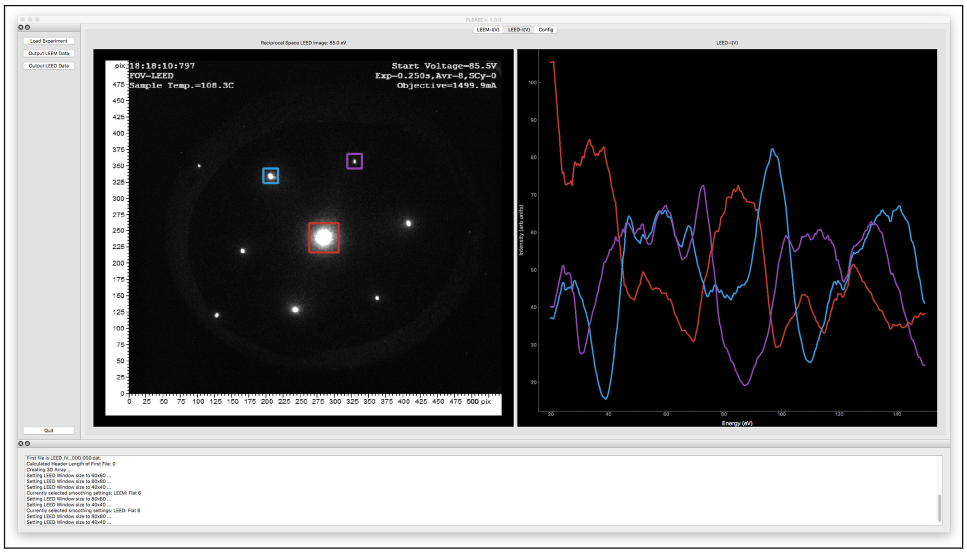
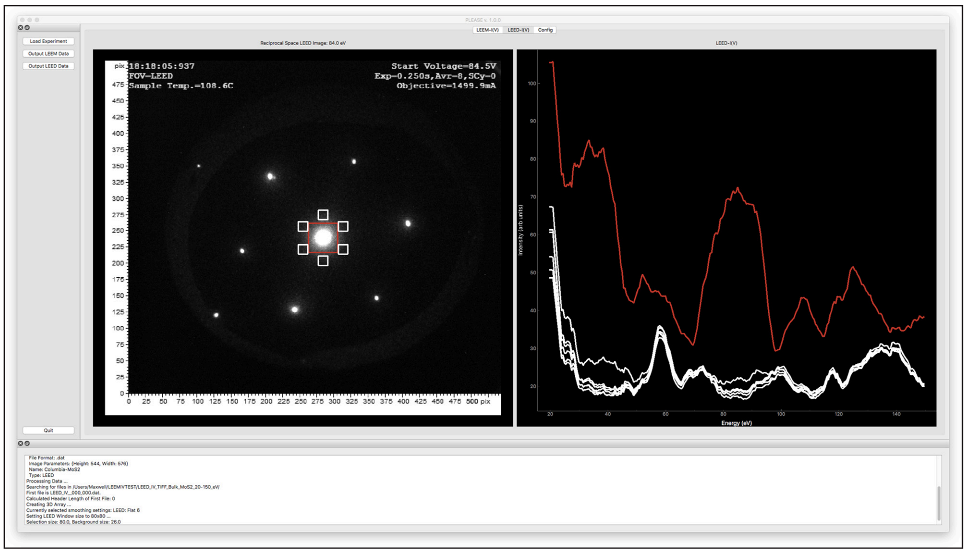


**Figure 2:** Intensity-Voltage data sets can be envisioned as a vertical stack of LEEM or LEED images acquired at varying incident electron energies. Shown here is an example of real space LEEM images acquired at multiple energies. The intensity of a given pixel varies from image to image as a function of electron energy; this relationship is shown to the right with the intensity extracted from the location labeled by the red dots. For clarity not all images from the data set are shown.



```
First file is LEED_IV__000_000.dat.
Calculated Header Length of First File: 0
Creating 3D Array ...
Setting LEED Window size to 60x60 ...
Setting LEED Window size to 80x80 ...
Setting LEED Window size to 40x40 ...
Currently selected smoothing settings: LEEM: Flat 6
Setting LEED Window size to 80x80 ...
Setting LEED Window size to 40x40 ...
Currently selected smoothing settings: LEED: Flat 6
Setting LEED Window size to 80x80 ...
Setting LEED Window size to 40x40 ...
```

**Figure 3:** LEED pattern obtained from a 5 $\mu$m diameter region on the 2H surface termination of $MoS_2$ with an incident electron energy of 85.0 eV [1]. *I(V)* curves are extracted from three user selected diffracted electron beams. The center beam intensity is extracted using an 80 × 80 pixel window, the first order diffracted beam intensities are extracted using a 40 × 40 pixel window.



```
 File Format: .dat
 Image Parameters: {Height: 544, Width: 576}
 Name: Columbia-MoS2
 Type: LEED
Processing Data ...
Searching for files in /Users/Maxwell/LEEMIVTEST/LEED_IV_TIFF_Bulk_MoS2_20-150_eV/
First file is LEED_IV__000_000.dat.
Calculated Header Length of First File: 0
Creating 3D Array ...
Currently selected smoothing settings: LEED: Flat 6
Setting LEED Window size to 80x80 ...
Selection size: 80.0, Background size: 26.0
```

**Figure 4:** For LEED analysis, the local background signal (white) can be automatically generated from a user selection (red). Here the *I(V)* data is plotted from the user selected electron beam and six background windows equally spaced around the selected beam are generated for background analysis.

The PLEASE software package depends on a number of python libraries that are all freely available and fairly straightforward to install:

- The NumPy library provides optimized and efficient methods for handling n-dimensional arrays and computations on arrays in python. This library forms the core of the python "scientific stack," and is used heavily by PLEASE for efficient representation and manipulation of LEEM/LEED data.
- SciPy extends NumPy and provides many methods for statistical analysis of array-like data.
- Pillow, the friendly fork of PIL, the Python Imaging Library, provides convenience methods for loading many image types into NumPy arrays for analysis and visualization. Pillow is used for converting images into single channel grayscale format.
- The PyQt library, from Riverbank Computing Ltd., acts as a python wrapper for the Qt C++ application framework, and serves as the main library for creating and managing the graphical user interface. The Qt event system is used to handle user interaction with the experimental data. Finally, the QThread system provided by PyQt is used to handle multithreading of all data I/O processes. This ensures that the main UI thread is not blocked while reading or writing data from disk.
- PyQtGraph is an additional library that provides convenient UI classes for plotting and scientific applications by leveraging the Qt QGraphicsView Framework for ultra-fast display. Here it is worth noting that early versions of PLEASE utilized Matplotlib as the library for plotting data and displaying images. While Matplotlib is likely the most popular and well-used python plotting library, its aim is centered on generating static publication quality plots. Matplotlib is not well suited to slideshow-like display of a large number of images, and thus was replaced by PyQtGraph to improve performance in real time plotting.
- PyYAML provides methods for reading and writing YAML documents. YAML, ("YAML ain't a markup language"), is a human readable data serialization language similar to JSON. This is used to store the configuration for each LEEM and LEED experiment in a simple, structured, and readable fashion. This greatly reduces the amount of time needed to begin analyzing data from new experiments.
- Pendulum provides an easy to use method of handling date/time data. This is only used in writing user configuration files to help with debugging. If there are problems during the usage of PLEASE, it may be beneficial to output as much information as possible about the user runtime environment to look for potential conflicts and problems. Thus, a convenience method to write this data to file in a structured format is provided.

The source for PLEASE is written in an object-oriented manner. While scientific programs are often written in a procedural manner for clarity, the object-oriented paradigm is used to facilitate ease of creating a robust graphical user interface. The PEP-8 python coding convention is followed, albeit sometimes loosely, with a few small changes. Notably, the maximum line length is allowed to be 120 instead of 80. This facilitates descriptive naming of variables and methods at the expense of lengthier lines of code.

PLEASE is designed to be executed as a GUI as opposed to a module imported for use with other python software. In the future it may be possible to refactor the source to provide a user facing API that does not depend on the GUI, however, for now the focus is primarily oriented on providing a GUI for ease of use during data analysis.

## Quality control

Since the PLEASE software package is intended to be used for exploratory data analysis, the majority of testing for the software has focused on functional testing using real experimental data sets. Continued exposure to LEEM and LEED experiments studying novel materials provided a plethora of data sets for continuous testing and development of this software. Differences in data formats between one experiment to the next provided a rigorous test of the I/O capabilities of the software. Using the YAML document format, a standardized and structured format was designed to store the experimental parameters PLEASE needs to open a data set. This streamlines the processing of many different experimental data sets stored with potentially many different file parameters. To further enhance ease of use, the main UI provides an option to automate the generation of YAML files. Thus, when a new data set is collected and ready to be analyzed with PLEASE, a user may select the action "Generate Experiment Config File" from the File menu. This will open a dialog box for the user to input the required parameters and then save the output to an appropriately formatted .yaml file.

In order to verify that the basic data extraction methods were functioning as required, a number of test data sets were created. These have no relation to actual experimentally collected data and were designed to provide a reference data set for testing user feature selection. The main code repository contains a directory for test data. This directory contains both actual experimental data as well as computer generated test data. The computer generated test data can be loaded into PLEASE to test that the data extraction in both LEEM and LEED is working properly. The only action required to load the data sets is editing the experiment configuration file (a .yaml file) provided with each data set. Only one property of the file needs to be set in order to load the data, this property is the local path to the data files. Instructions are provided for editing and creating new experimental configuration files. In order to facilitate ease of use, an additional python script as been included within the test data directory, which will attempt to automatically update the .yaml files with the appropriate paths to the test data. This script will only work properly if the test data directory is left as is upon installation of PLEASE and not modified or moved to an alternate location. Instructions for usage of this automated script have been included within the test data directory.

**Figure 5** demonstrates a test of the LEED user selection process. The test data set contains a mock LEED image repeated 100 times. Thus the *I(V)* curve from the simulated electron beam region should be constant. The *I(V)* curve from the background area should also be constant but everywhere equal to zero. The three simulated electron beam spots were created to have slightly different intensities. Data sets consisting of actual experimental data as well as computer generated data are provided for both LEEM and LEED analysis for testing purposes.

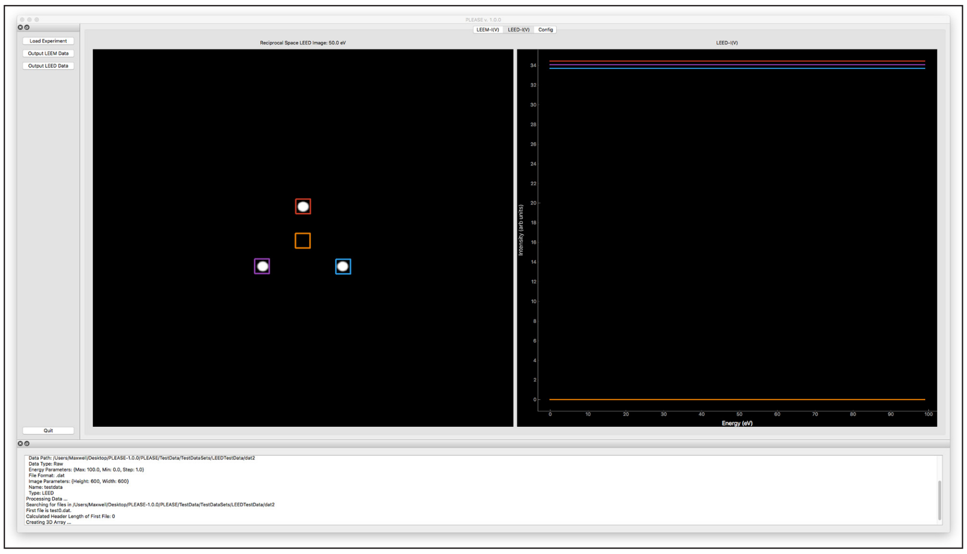
## (2) Availability

### Operating system

The PLEASE software package was tested on Mac OS X (version 10.8+), Windows 10, and Ubuntu 16.0.4. In principle, any system that supports python versions 2.7 and 3.5+ will be compatible with PLEASE. No support is provided or planned for mobile operating systems such as Android or iOS, and the software has not been tested on embedded-style Linux variants such as Raspbian for the Raspberry Pi.

### Programming language

Python versions 2.7 and 3.5+ are supported. All other python versions are not officially supported. The Anaconda Python distribution, provided free of charge by Continuum Analytics, is suggested for ease of use and community support. All python modules required by PLEASE are available via a combination of Anaconda's package manager, Conda, and the standard python package manager, Pip. All testing for PLEASE was performed using the CPython implementation of the python programing language. It may be possible to use another implementation however, given that PLEASE relies heavily on NumPy, it may be difficult to target another python implementation.



```
  Data Path: /Users/Maxwell/Desktop/PLEASE-1.0.0/PLEASE/TestData/TestDataSets/LEEDTestData/dat2
  Data Type: Raw
  Energy Parameters: {Max: 100.0, Min: 0.0, Step: 1.0}
  File Format: .dat
  Image Parameters: {Height: 600, Width: 600}
  Name: testdata
  Type: LEED
Processing Data ...
Searching for files in /Users/Maxwell/Desktop/PLEASE-1.0.0/PLEASE/TestData/TestDataSets/LEEDTestData/dat2
First file is test0.dat.
Calculated Header Length of First File: 0
Creating 3D Array ...
```

**Figure 5:** A test data set for LEED *I(V)* data is provided with the source. This data is created to simulate an electron diffraction *I(V)* data set and test the user data selection process.

### Additional system requirements

The system and hardware requirements to run PLEASE are relatively low, however, the amount of storage and memory capacity required will vary greatly due to differences in experimental data sets. The required memory to load a given data set will depend on the size and format of the data. Simultaneous visualization of LEEM and LEED data sets can easily require greater than 1 Gb of RAM. Thus, to prevent an out-of-memory exception from being raised by the python interpreter, it is suggested that PLEASE run only on machines with 6 Gb or more RAM. The most intensive tasks executed by the GUI are the reading and writing of data. These tasks are pushed to a separate thread so as to not block the main UI thread. These tasks are generally I/O bound as opposed to CPU bound. As a result, the CPU requirements for PLEASE are not high. Most modern computers with Intel or AMD CPU's should have no trouble running PLEASE. No support is provided for mobile devices using ARM or other mobile architecture processors.

Since PLEASE is used for exploratory data analysis in LEEM and LEED data sets, it is beneficial to run the software on a high resolution display. When run on screens with low resolution, the UI can limit the available area for displaying data. To help alleviate this problem, the main UI controls are split into dockable widgets. Thus, on small displays, it may be beneficial to pop-out and move the control widgets to make more room for the data display. It is, however, recommended that a display with sufficient size and resolution should be used where possible. PLEASE has been tested on monitors ranging in resolution from 1280 × 720 to 2560 × 1440 and is usable on all sizes in between.

An ideal PC to run PLEASE would be a CPU >= 2.5 GHz with 8+Gb RAM and a 20″ or larger display with 1080p resolution, however these are by no means minimum requirements.

### Dependencies

- NumPy >= 1.12.0
- SciPy >= 0.19.0
- Pillow >= 4.0.0
- PyQt >= 5.6.0*
- PyYAML >= 3.12.0
- Pendulum >= 1.1.0
- PyQtGraph >= 0.10.0**

*** Note:** Riverbank Computing Ltd. provides two versions of their python bindings for the Qt library, PyQt4 and PyQt5. The APIs provided by these bindings are not backwards compatible. As currently written, PLEASE will not work with PyQt4 and there are no plans to add this functionality in the future.

**** Note:** There are a few minor API changes between versions 0.9 and 0.10.0 for the PyQtGraph module, with the later version enabling support for PyQt5. PLEASE will not work with version 0.9.0, thus, whether installing via conda or pip, you must ensure version 0.10.0 or greater is used.

### List of contributors

1. Grady, Maxwell – Sole Developer
2. Dai, Zhongwei – Testing and Feature Suggestion
3. Pohl, Karsten – Testing and Feature Suggestion

### Software location

#### *Archive*

***Name:*** Figshare
***Persistent identifier:*** https://doi.org/10.6084/m9.figshare.4907369.v2
***Licence:*** GPLv3
***Publisher:*** Maxwell Grady
***Version published:*** 1.0.0
***Date published:*** 24/08/17

#### *Code repository*

***Name:*** GitHub
***Persistent identifier:*** https://www.github.com/mgrady3/PLEASE
***Licence:*** GPLv3
***Date published:*** June 21, 2017

### Language

English

## (3) Reuse potential

Development of the PLEASE software package was originally focused on providing a piece of software to aid in the analysis of one specific set of LEEM and LEED data. However, over the course of its development, the project grew to encompass many features beneficial in a wide array of material science experiments. This did not happen by chance, rather reusability became a prominent design goal of the project.

While usage of this software for atomic surface structure determination through analysis of $\mu$-LEED data has already been demonstrated [1], there are a number of related experimental techniques for which this software can provide data analysis. Photoemission electron microscopy (PEEM) is a complementary technique available to many LEEM systems. PEEM works by illuminating the target with a collimated beam of monochromatic photons and forming images from the photo-emitted electrons. these image sequences can contain embedded spectroscopic information such as elemental content, electronic structure, and magnetic structure [3]. The direct measurement of elemental composition is something LEEM and LEED are unable to do on their own, making PEEM a very useful and versatile technique. Since PEEM and LEEM share a common data type, the PLEASE software package can also be used to visualize and analyze PEEM data, providing spectroscopic data in a pixel by pixel fashion.

PLEASE uses a custom metadata format using the YAML data serialization language to provide descriptions of experimental data in order to properly load the data for visualization. This metadata format can be easily extended to include experimental configuration parameters for PEEM and other related techniques. While the current version of the software provides a minimal framework for analysis and visualization of PEEM data sets, future releases will extend this to include more useful features. An example of a recent extension of the metadata format is the addition of the ability to process LEEM/LEED data sets as time series data rather than *I(V)* data. This allows for analysis of *I(t)* data sets, which is useful for understanding dynamic processes imaged with LEEM such as epitaxial growth or surface structure phase transitions as seen with LEED.

At the time of initial development, open source solutions for analysis were limited. The two main options were GXSM [11] and ImageJ. GXSM is a software package for multi-dimensional image processing with an emphasis on scanning probe microscopy techniques; this software package supports most Linux variants as well as OS X, but not Windows. This package also includes a plugin for working with the UKSOFT LEEM/LEED image format. ImageJ is a public domain image processing software package including an extensive and extensible plugin, macro, and scripting system and will run natively on all major OS variants.

While ImageJ is capable of loading the raw LEEM data in the native UKSOFT image format via the "Import Raw" method, this requires the user to input the specifics of the image format each time data is loaded, such as image height, width, header length, bits per pixel, and byte order. PLEASE offers a streamlined method of loading data using the YAML metadata format. In this method, users enter the experimental configuration once and only once, then can load data at a later time from a .yaml file with ease. PLEASE also offers a UI dialog for generation of .yaml files to store experimental configurations. Thus there is no need for the user to worry about editing the .yaml files by hand in a text editor.

While both GXSM and ImageJ are capable of analyzing LEEM and LEED images, they are much more generically featured, and understanding how to use them for LEEM/LEED analysis "out of the box" may be difficult for a novice user. PLEASE was designed to create a simple and straightforward method for visualization of LEEM and LEED data with a "point-and-click" method for extracting data of interest as opposed to being a drop-in replacement to existing tools for analysis of arbitrary multidimensional data sets. As a result, PLEASE offers some features such as configurable data smoothing for extracted *I(V)* signals and automated extraction of local background signals in LEED data, which are not available in GXSM or ImageJ without the usage of external plugins or user defined macros.

For those users who may already have a well established data processing pipeline for analysis of LEEM/LEED data, PLEASE may not offer much as a drop-in replacement. However, PLEASE is well suited for those users who are new to the field of LEEM/LEED experiments. Given that many LEEM systems are hosted at national laboratory user facilities, assistance is provided for collecting the experimental data, however users are then left to their own devices to visualize and analyze the data. While PLEASE does not provide the robust ecosystem of plugins and macros, instead it offers a streamlined and intuitive interface for rapid analysis of I(V) data with a low learning curve.

In order to make the software available to the largest audience, PLEASE was designed to be fully cross platform and open source. Building PLEASE around the core python scientific libraries makes the software easily extensible. The software was written to provide flexibility with respect the data that it can load for visualization.

Currently, data can be loaded from TIFF and PNG image files, as well as raw binary (.dat) files output from the LEEM control software. The core data construct is a NumPy array, thus any type of image that can be converted to a 2D NumPy array using pillow (PIL) should be possible to read with PLEASE with minimal alteration of the source. In the future it may be beneficial to extend the PLEASE data I/O capabilities to other standardized scientific data formats such as HDF5 or NetCDF. There are pre-existing python libraries for handling these types of data formats, thus integration into PLEASE should be straightforward. Initial work has begun to facilitate storage of experimental data and experiment configuration settings in a hierarchical database using the HDF5 file format. This further streamlines the ability to load and analyze data from many different experiments while also providing a single repository for data and configuration information.

Finally, it should be noted that PLEASE is still an evolving piece of software. There are a wide variety of potential features, which are being tested for future releases, however the current edition of the software contains features that are stable and useful for the rapid analysis of LEEM and LEED data sets. The main GitHub repository will always contain the "master" branch, which contains the most up to date stable release of PLEASE, as well as a number of other branches labeled with the prefix "dev_". These branches are created to test potential features. For example, a branch exists for testing the usage of a HDF5 database for storing experimental data and parameters, whereas another branch exists for testing an alternate method of ROI selection using the pyqtgraph native ROI objects.

The primary method of communication about the project should be the main source repository, which is hosted on GitHub. The GitHub page for the project has an issue tracker system, which can be used for communication between users as well as the project maintainer. This communication channel is strongly encouraged not only for bug reporting but for all questions, comments, and suggestions for the project. The project will continue to be hosted in a public source repository on GitHub for the foreseeable future. GitHub was chosen not only to provide version control and a remote host, but also to facilitate conversation and contribution from outside users of the project. Pull-requests for the source are encouraged as a way to fix problems and suggest new features for the program. Finally, interested parties should feel free to fork the project and customize the software for their own needs.

**Acknowledgements**
Special thanks to Steven Arias (University of New Hampshire) for discussing ideas for the original groundwork of this software. Finally, gratitude is due to Taisuke Ohta and Bogdan Diaconescu from the Sandia National Laboratory

Center for Integrated NanoTechnology as well as Samuel Tenney and Jurek Sadowski from the Brookhaven National Laboratory Center for Functional Nanomaterials for providing a thorough and hands-on introduction to the techniques of LEEM and $\mu$-LEED.

**Funding Information**

Funding for this work was provided by the following grants and fellowships: National Science Foundation DMR-1006863 and the University of New Hampshire NASA Space Grant Graduate Research Fellowship.

This research used resources of the Center for Functional Nanomaterials, which is a U.S. DOE Office of Science Facility, at Brookhaven National Laboratory under Contract No. DE-SC0012704.

This work was performed, in part, at the Center for Integrated Nanotechnologies, an Office of Science User Facility operated for the U.S. Department of Energy (DOE) Office of Science by Los Alamos National Laboratory (Contract DE-AC52-06NA25396) and Sandia National Laboratories (Contract DE-NA-0003525).

**Competing Interests**

The authors have no competing interests to declare.

## References

1. **Dai, Z, Jin, W, Grady, M, Sadowski, J T, Dadap, J I, Osgood, R M, Jr.** and **Pohl, K** 2017 Surface Structure of bulk 2H-MoS$_2$: A selected area low energy electron diffraction study. *Surface Science*, 660. DOI: https://doi.org/10.1016/j.susc.2017.02.005
2. **Feenstra, R M, Srivastava, N, Qin Gao, M, Widom, M, Diaconescu, B, Ohta, T, Kellogg, J T R** and **Vlassiouk, I V** 2013 Low-energy electron reflectivity from graphene. *Physical Review*, B 87. DOI: https://doi.org/10.1103/PhysRevB.87.041406
3. **Hannon, J** and **Tromp, R** 2012 Low-energy electron microscopy for nanoscale characterization. In: *Handbook of Instrumentation and Techniques for Semiconductor Nanostructure Characterization, World Scientific*, 127–179. DOI: https://doi.org/10.1142/9789814322843_0004
4. **Hibino, H, Kageshima, H, Maeda, F, Nagase, M, Kobayashi, Y** and **Yamaguchi, H** 2008 Microscopic thickness determination of thin graphite films formed on SiC from quantized oscillation in reflectivity of low-energy electrons. *Physical Review*, B 77(7). DOI: https://doi.org/10.1103/PhysRevB.77.075413
5. **McClain, J** 2015 A supercell, Bloch wave method for calculating low-energy electron reflectivity with applications to free-standing graphene and molybdenum disulfide. Ph. D. Thesis. University of New Hampshire.
6. **Oliphant, T** 2007 Python for Scientific Computing. *Computing in Science & Engineering*, 9(3): 10–20. DOI: https://doi.org/10.1109/MCSE.2007.58
7. **Pendry, J** 1974 Low energy electron diffraction: the theory and its application to determination of surface structure. London: Academic Press.
8. **Sun, J, Hannon, J B, Kellogg, G L** and **Pohl, K** 2007 Local structural and compositional determination via electron scattering: Heterogeneous Cu (001)-Pd surface alloy. *Physical Review*, B 76. DOI: https://doi.org/10.1103/PhysRevB.76.205414
9. **Van Hove, M** and **Tong, S** 1979 Surface Crystallography by LEED. Berlin: Springer-Verlag. DOI: https://doi.org/10.1007/978-3-642-67195-1
10. **Van Hove, M, Weinberg, W** and **Chan, C** 1986 Low-energy electron diffraction. Berlin: Springer-Verlag. DOI: https://doi.org/10.1007/978-3-642-82721-1
11. **Zahl, P,** et al. GXSM software project homepage. http://gxsm.sourceforge.net 2000-today.

**How to cite this article:** Grady, M, Dai, Z and Pohl, K 2018 PLEASE: The Python Low-energy Electron Analysis SuitE – Enabling Rapid Analysis of LEEM and LEED Data. *Journal of Open Research Software,* 6: 7. DOI: https://doi.org/10.5334/jors.191

**Submitted:** 05 September 2017 **Accepted:** 16 January 2018 **Published:** 05 February 2018