doi:10.1017/S1431927613005801
Microsc. Microanal. 19 (Suppl 2), 2013

A New Environment for Modular Image Reconstruction and Data Analysis

Michael Radermacher

Department of Molecular Physiology and Biophysics, University of Vermont, Burlington, VT, USA

We have developed a new image processing system that fills an important niche among the many image processing packages available. It is an environment for modular image reconstruction and data analysis and allows the easy addition of any program that is executable on the command line, written in any programing language and without requirements to adhere to specific standards. The system provides simple to use wrapper functions for these programs, which then become accessible to a user interface with simple syntax either for interactive use or for use in scripts that can be run in batch mode.

There are a wide variety of image processing packages available within the 3D electron microscopy field [1] [2]. While many image analysis programs exist in all packages, more specialized ones may not be commonly available. Efforts have been made to access several of these systems under a common user interface, examples being EMIP [3] and Appion [4]. If a specialized program is missing, one frequently used solution is writing a standalone program to accomplish the task, which, however, interrupts the workflow provided by many packages. The alternative, implementation of a new program into a larger package, however, requires the understanding of the internal infrastructure of the package, an expertise lacking in many laboratories not specialized in software development. Keeping the code in working condition can become time consuming, since modification of the package may require changes of the added code, without any immediate gain in functionality. Therefore we developed a system where programs that perform image or data processing tasks are independent from the user interface, which has an easy to understand syntax.

The ideas behind the new system are: 1. Separate the actual working program from the user interface, such that further interface development is independent from the running code. 2. Pose no restrictions on the programming language used. This allows each researcher to write code in the language she/he is most familiar with. 3. Keep the calls to the wrapper modules stable, such that improvements to wrapper functions do not require changes to wrappers. 4. Keep the user interface syntax simple such that it can easily be learned within 1 hour. 5. Keep changes to the user interface syntax backwards compatible.

We designed a system that follows the above principles. The design is not restricted to image processing programs but can be used for any combination of command-line programs independent of purpose. The system is written in Python 2.7, avoiding as much as possible older constructs that may complicate future conversion to Python3.x. Functions are provided for efficient wrapping of programs such that only minimal knowledge of Python is required for successfully wrapping new programs. The user interface syntax was designed with two objectives, one was to keep it as simple as possible, and the second was to design it such that it can easily take advantage of the power of built-in Python features, which keeps the code small. The syntax is inspired by the syntax of SPIDER 5.0 since it is easy to teach even to researchers with little computing experience. It offers the possibilities of loops and logical if-statement structures. The system includes variables that can be used in arithmetic expressions and as program inputs. Program outputs are either files that are written by the individual programs or specific values that can be retrieved into variables from standard output within the “run” function of the system. Other wrapper functions include a function to assemble the command line arguments and functions to ask for numbers, strings and filenames. Number substitution in filenames by either numbers or variables is provided transparently within the wrapper functions.

The wrapped modules are called through a plugin mechanism, thus no change to the main program is needed when modules are added. Any user can add a directory path for additional plugins through an environmental variable. Commands are recognized by the module name. For example the command "fourier" looks for a file named "em_fourier.py"

The current version of the system has only a limited number of operations wrapped and more will be added. They are the 2D and 3D Radon transform based programs for refinement and alignment of projections and volumes, basic operations like "copy", "fourier", "shift", "rotate". It will contain a complete set of programs for carrying out a random conical reconstruction, and 3D reference based 3D reconstructions.

[1] See e.g. Special Issue on Software Resources, in J. Struct. Biol.116 (1996).
[2] Hohn, M., et al., J Struct Biol, 157(2007), p. 47.
[3] Diaz, R., W.J. Rice, and D.L. Stokes, Methods Enzymol, 482(2010), p. 131.
[4] Lander, G.C., et al., J Struct Biol, 166(2009), p. 95.
[5] The author acknowledges funding from NIH, Grant number 1RO1 GM078202

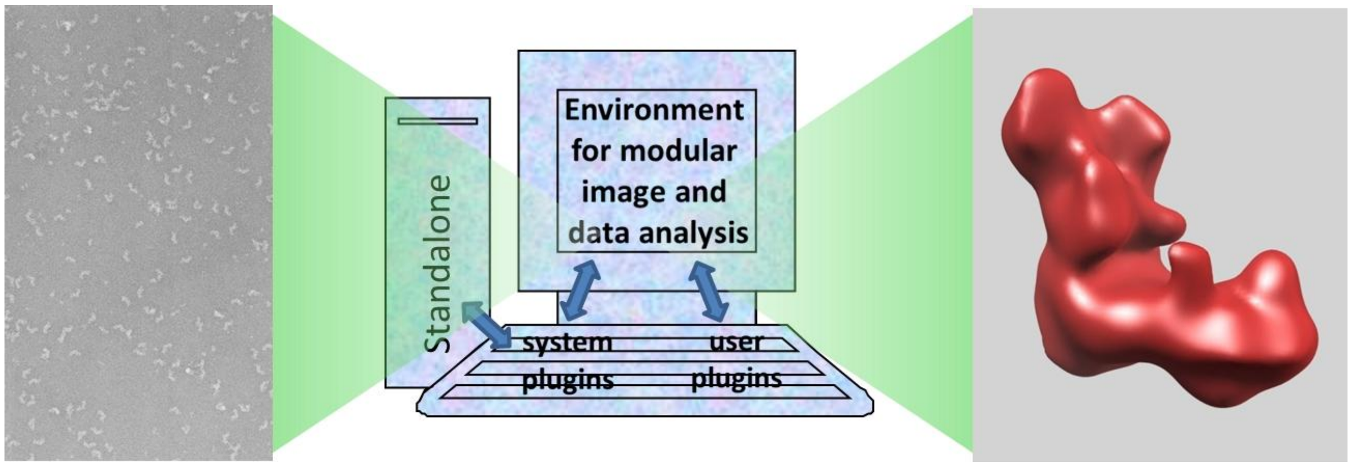

Figure 1. Principle of the image processing system. The analysis of digitized image data is carried out by standalone programs which are controlled by the user interface of the image processing environment. The system communicates with standalone programs through simple plugins either as part of the system or user contributed. The user environment allows for interactive use of each program and the assembly of modules into scripts to create complex workflows to carry out alignments of image series, 3D reconstructions, 3D alignments and classifications.