A SCRIPTING LANGUAGE APPROACH TO CONTROL SOFTWARE FOR CRYO-ELECTRON MICROSCOPY

William T. Baxter[2], ArDean Leith[2], Joachim Frank[1]
[1]Howard Hughes Medical Institute, [2]Health Research, Inc., Wadsworth Center, Albany NY

Abstract. **Cryo-electron microscopy (Cryo-EM) of single particles has developed into a widely used technique for determining the 3-dimensional structure of large molecules and molecular assemblies. The low signal-to-noise ratio of cryo-EM requires thousands of images of single molecules to be averaged together. The field has matured to the point where realization of high-resolution structures is limited primarily by computational constraints. These constraints are at the algorithmic level, as well as the control level, where dozens of complex procedures and thousands of mathematical operations are applied to the raw data. SPIDER is a highly modular and flexible software package for single particle reconstruction. A typical reconstruction project involves dozens of procedure files, which in turn draw on hundreds of available low-level operations. In the present system, it is virtually impossible to rerun the system from selected branching points in the process flow to determine the effects of various parameters values. A Reconstruction Engine (RE) is being developed as a high-level "shell" for controlling processes in the SPIDER software system. The RE allows backtracking, optimization of parameters, and automation of processing flow. The RE is implemented in a scripting language, Python, which provides an overall management capability at the global level of the project.**

INTRODUCTION

Over the past two decades, cryo-electron microscopy (cryo-EM) of single particles has developed into a tool for determining the 3-dimensional structure of macromolecules (above several hundred kDa) [1,2]. In single particle reconstruction, images represent 2-dimensional projections of the 3-dimensional molecule seen in different views. The structure can be recovered from projections by mathematical reconstruction techniques. Thus very large molecular assemblies not amenable to other approaches such as X-ray crystallography or NMR can be visualized. Since it is not limited by the packing constraints of crystallography, dynamic functional behavior can be studied with single-particle reconstruction, for example, the conformational changes induced by ligand binding. Rapid freezing (vitrification) allows molecules to be visualized in their native hydrated state. Typical doses (10-15 electrons/Å^2) are low enough to minimize the effects of radiation damage, when aiming at ~10Å resolution. However, the resulting micrographs have low signal-to-noise ratio, requiring tens of thousands of images to be included in the reconstruction.

Micrograph images are electron scattering density maps. Structural information comes from elastically scattered electrons, while inelastic scattering results in radiation damage and does not contribute to the image in a constructive way. Samples must be thin for the weak phase approximation to be applicable, the theory used to describe image formation in the electron microscope [1]. The resulting images have virtually no contrast at focus, so the microscope is defocused to provide phase contrast. Imaging characteristics are determined by the CTF (contrast transfer function), which is analogous to the transfer function in optical systems, and consists of alternating bands of positive and negative contrast whose zero crossings vary with defocus. In the vicinity of these zeroes, no contrast information is conveyed, so a range of differently defocused images are acquired to cover all frequencies. Micrographs contain many copies of randomly oriented 2D projections of the 3D molecule. These different views are combined using computational techniques of reconstruction from projections, originally devised for electron microscopy [3], and later extended to diverse fields such as medical tomography.

PROCESSING STEPS FOR ELECTRON MICROGRAPHS

After the specimen images are taken by the electron microscope, a wide variety of computational processes are applied to the images [4,5]. The CTF characteristics (in terms of parameters such as defocus, astigmatism, etc.) are determined for each micrograph. Individual particles from the micrograph images must be identified and selected, usually by a combination of automated and interactive techniques. This results in 20,000-80,000 small particle images forming the raw input to the reconstruction process. Particles are classified into orientation groups, a process referred to as 3D projection alignment [6], by correlating the particle images with projections from an existing 3D reference map. Once the viewing angles have been thus estimated, various reconstruction algorithms (back-projection, ART, SIRT) may be used to generate an initial reconstruction from the particle images. CTF correction is applied in this step. Lastly, in the process of orientation refinement, the view angles of every single particle are

0-7803-7584-X/02/$17.00 ©2002 IEEE

iteratively adjusted until a 3D structure is found that minimizes a least squares regularizing cost criterion [4].

Using these techniques, 10-17 Å resolution can be obtained in the final reconstruction, but the cryo-EM community is aiming to improve that resolution to 6-7 Å, and if possible, to the atomic range in selected applications. Some improvement in resolution will come from better data collection in the electron microscope and the development of better algorithms, but a major improvement will come from increasing the number of particles participating in the reconstruction by a factor of 10 to 50. This increase will require a much expanded use of computational resources, such as exploiting parallelism, optimizing parameters, and automating as far as possible the application of the mathematical procedures which currently involves a large amount of user interaction.

SOFTWARE FOR CRYO-EM

Numerous software packages have been developed for cryo-EM [7-10]. SPIDER, written in Albany, NY by Joachim Frank and his colleagues, is the most widely used system for single-particle reconstruction of cryo-EM images. SPIDER is powerful but complex, with over 400 image and volume processing operations. While operations may be executed in an interactive SPIDER session, more frequently multiple operations are collected into batch files, and run as a single unit. In addition to providing a fully programmable environment with loops, conditionals, subroutines, and symbolic variables, batch files also represent a compendium of knowledge and expertise. They embody elaborate algorithms for various reconstruction procedures, such as particle selection and alignment. The outputs of one batch file are usually used as inputs for other batch files. A single reconstruction project may use over 40 batch files. Although batch files are optimized individually, globally the system lacks an overall intelligence in the execution of a series of algorithms. This makes it extremely difficult to go back to a branching point and redo the processing with modifications.

To remedy this problem, a software layer termed the Reconstruction Engine is being developed, which will provide a high-level organizational framework for image processing and reconstruction to facilitate backtracking based on complete records of the data. Among the specifications: (1) the result should require minimal changes to the ~120,000 lines of Fortran code that comprise SPIDER; (2) the batch files represent a more flexible level, but any changes must preserve the image processing algorithms; (3) the Reconstruction Engine, acting as a "wrapper" around SPIDER processing, will be written in a scripting language that controls the various modules of SPIDER processing, managing the inputs and outputs of each batch file. Further, beginning users should be able to easily use SPIDER batch files, while experienced users should still have the capability to write their own batch files and add them to the system. Python was selected to provide a high level, rapid development environment. The components consist of individual SPIDER procedures, as well as a project-wide database system.

THE SCRIPTING LANGUAGE APPROACH

The objectives of the software control system were to include (1) bookkeeping of the many parameters and data objects in the history of a project, (2) providing ease of user interaction, while (3) maintaining overall processing efficiency. In order to automate a processing project centered on SPIDER batch files, a typical reconstruction project was analyzed to assess how SPIDER batch files are utilized. In particular, potential obstacles to bookkeeping and backtracking were noted, as well as places of heavy user interaction and common sources of user errors. Next, the system was decomposed into its functional objects, consisting of processes (batch files) and the data objects (text and binary files), as well as the information flow among objects. Finally, a list of functional requirements from the user's point of view was created.

Analysis

A project consists primarily of running a series of batch files (batches are run in serial, although actual process flow is not strictly sequential). The set of batch files is not fixed - users may add new ones. Batch files may call procedures, which are simply other batch files used as subroutines. At various points, outputs are examined to see if they meet certain criteria. Nearly all algorithms used in reconstruction projects have been collected into batch files. Currently, the user selects, edits, and runs batch files one at a time. Batch files read and write files to the hard drive. File names that are input to multiple SPIDER operations must be typed into the text at each point where they are used. Given a set of input micrographs, batch files parameters are adjusted until they produce desired outputs.

Each batch file is run within its own SPIDER session. There is no direct communication from one batch file to the next. Therefore, the user must keep track of parameters and file names across batch files. Procedure calls solve this problem, because the calling module and subroutine directly share all variables passed. However, when many procedures are collected under a single top-level batch file, the multiple nesting of procedures makes it difficult to understand and debug the code, or to adjust parameters. Thus the current reconstruction process has been broken down into many top-level batch files, with limited nesting of procedures.

The major difficulties identified in a SPIDER reconstruction project were:

(1) Within batch files, file names (both input and output) can occur at numerous locations. If a different filename is desired, that change must be made at all locations.

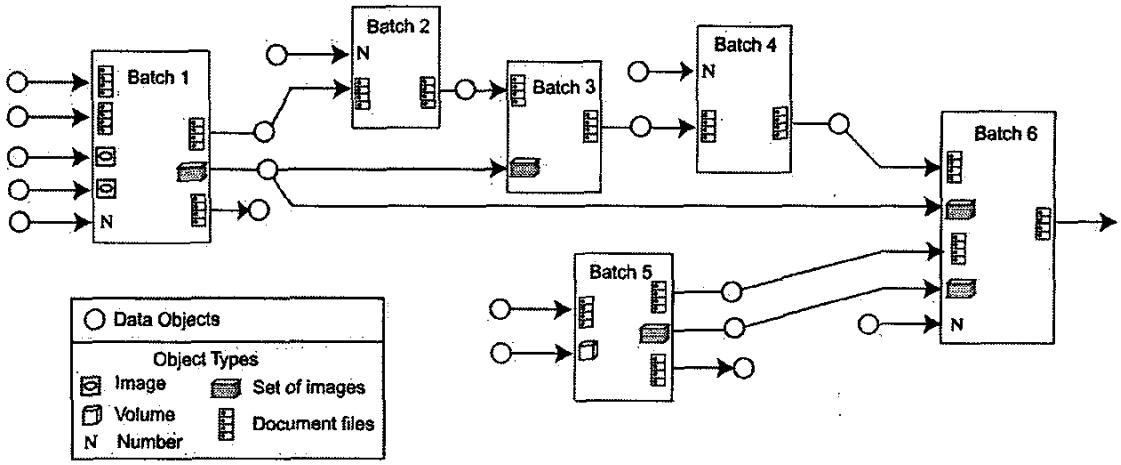

Figure 1. Example of processing flow during a single particle reconstruction project.

(2) Similarly, renamed batch files cause difficulties when they are output from one batch file to be used as input for the next. All name changes in both files must be checked.
(3) Certain numerical parameters, such as pixel size, are used by several algorithms throughout the reconstruction process. These also are set explicitly within the text of the batch files. If a parameter's value is altered, then care must be taken to change this number in every affected batch file.

The above observations accounted for the great majority of user errors during a reconstruction project. They all suggested the need for a global repository of information. The next step was system decomposition, to determine the functional units of the system.

Definition of Objects

The system was decomposed into processes and data objects.
(1) *Processes* consist of batch files, procedure files, and non-SPIDER operations (programs external to SPIDER). Because all of these can read/write SPIDER files, they may be considered functionally as batch files.
(2) *Data objects* may be textual (document files), binary (which are further subdivided into 2D images, 3D volumes, 2D Fourier Transforms, etc.), or numerical arguments.

Information Flow

Data objects act as inputs and outputs of processes. Data objects have one source (the batch file that generated them) and potentially many targets (batch files downstream that will use them as inputs). Numerical parameters do not have a source batch file. However, they all may be collected into a global parameter document file.

Batch files can have any number of inputs and outputs. Inputs may be data objects or numerical arguments. Outputs are more data objects. Figure 1 shows a set of batch files and their respective inputs and outputs. Data objects of all types are indicated as circles, although they are associated with a type. Batch file input and output "ports" also have associated types. File names and parameters are explicitly written in the text of batch files. Because file names and numerical parameters usually change across projects, starting a new project often requires considerable editing of the batch files. Also, the same file or numerical argument may be input to several batch files. If this input is changed in one batch file, there is no mechanism to ensure that other batches will refer to the same updated value.

Functional Requirements

The requirements list below describes how a user will interact with the new system. It was produced without any regard to the underlying implementation. The user should be able to: (1) Select, specify inputs, and run SPIDER batch files as above, but within a graphical user interface (GUI), (2) add the results to a "project database" when they are deemed correct, (3) later recall that project, change the inputs, then run the batch files again, (4) save the new results. A further requirement is an iterating mechanism whereby the outputs of a chain of processing may be used to generate a new set of input parameters. This provides the Reconstruction Engine with the capability to search for optimal parameter solutions for the given processing sequence.

Implementation

The implementation of the above requirements was carried out at two levels: (1) some rewriting of the common

SPIDER batch files and (2) writing the Python-based controller which reads the batch files and keeps track of their inputs and outputs. These required no changes to be made to SPIDER's underlying Fortran code. The batch files, contributed over the years by a number of users, were altered slightly to conform to a new system-wide standard of "well-formed batch files" amenable to automated management. This new standard was designed primarily to incorporate good programming practices, such as use of variables rather than explicit values, and collection of variables into central locations for efficient management. These simple changes circumvented most of the weaknesses noted in the above analysis. In addition, a consistency-checking function checks file names and numerical arguments both within and across batch files.

The Reconstruction Engine obtains information about batch file input and outputs in a form-based interface. The Python scripting language was utilized for text processing and interfacing with the project database (in this case, MySQL). The GUI was written using the Python interface to the Tk/Tcl toolkit. A file selection utility, with browsing capability, enables users to graphically select batch files. A batch file parser reads text batch files and presents a form to the user; the entered information is then written out to a standardized batch file, which is then run in SPIDER. As each batch file, or module, is run, is outputs are monitored for consistency and entered into the project database. The Reconstruction Engine shows promise as a fairly simple approach to modernizing scientific software, relieving users of much of the burden of interaction, and providing much needed tools for management.

REFERENCES

[1] J. Frank, *Three-Dimensional Electron Microscopy of Macromolecular Assemblies*. San Diego: Academic Press, 1996.

[2] H.R. Saibil, "Macromolecular structure determination by cryo-electron microscopy," *Acta Crystollographica*, vol. D56, pp. 1215-1222, 2000.

[3] D. DeRosier and A. Klug, "Reconstruction of 3-dimensional structures from electron micrographs," *Nature (London)*, vol. 217, pp. 130-134, 1968.

[4] P. Penczek, M. Rademacher, and J. Frank, "Three-dimensional reconstruction of single particles embedded in ice," *Ultramicroscopy*, vol. 40, pp. 33-53, 1992.

[5] J. Frank, P. Penczek, R.K. Agrawal, R.A. Grassucci, and A.B. Heagle, "Three-dimensional cryoelectron microscopy of ribosomes," *Methods in Enzymology*, vol. 317, pp. 276-291, 2000.

[6] P. Penczek, R.A. Grassucci, and J. Frank, "The ribosome at improved resolution: New techniques for merging and orientation refinement in 3D cryoelectron microscopy of biological particles," *Ultramicroscopy*, vol. 53, pp. 251-270, 1994.

[7] J. Frank et al. "SPIDER and WEB: Processing and visualization of images in 3D electron microscopy and related fields," *J. Struct Biol*, vol. 116, pp. 190-199, 1996.

[8] M. van Heel, G. Harauz, E.V. Orlova, R. Schmidt, and M. Schatz, "A new generation of the IMAGIC processing system," *J. Struct Biol*, vol. 116, pp. 17-24, 1996.

[9] S.J. Ludtke, P.R. Baldwin, and W. Chiu, "EMAN: semiautomated software for high-resolution single-particle reconstruction," *J. Struct Biol*, vol. 128, pp. 82-97, 1999.

[10] R.A. Crowther, R. Henderson, and J.M. Smith, "MRC image processing programs," *J. Struct Biol*, vol. 116, pp. 9-16, 1996.