Available online at www.sciencedirect.com
ScienceDirect
Journal of Structural Biology 157 (2007) 56–63
Journal of Structural Biology
www.elsevier.com/locate/yjsbi

SPIRE: The SPIDER Reconstruction Engine

William T. Baxter [a], ArDean Leith [a], Joachim Frank [a,b,*]

[a] Wadsworth Center, Empire State Plaza, Albany, NY 12201-0509, USA
[b] Howard Hughes Medical Institute, Health Research Inc., Empire State Plaza, Albany, NY 12201-0509, USA

Received 31 March 2006; received in revised form 11 July 2006; accepted 29 July 2006
Available online 25 August 2006

Abstract

SPIRE is a Python program written to modernize the user interaction with SPIDER, the image processing system for electron microscopical reconstruction projects. SPIRE provides a graphical user interface (GUI) to SPIDER for executing batch files of SPIDER commands. It also lets users quickly view the status of a project by showing the last batch files that were run, as well as the data files that were generated. SPIRE handles the flexibility of the SPIDER programming environment through configuration files: XML-tagged documents that describe the batch files, directory trees, and presentation of the GUI for a given type of reconstruction project. It also provides the capability to connect to a laboratory database, for downloading parameters required by batch files at the start of a project, and uploading reconstruction results at the end of a project.

Keywords: Cryo-electron microscopy; Single-particle reconstruction; Graphical user interface

1. Introduction

The SPIDER software package, an image processing system developed by Joachim Frank and his colleagues, has been in use by the electron microscopy community for nearly three decades (Frank and Shimkin, 1978; Frank et al., 1981). SPIDER has operations for 3D reconstruction of cryo-EM macromolecules, multivariate statistical classification, and electron tomography (Frank et al., 1996; Frank, 2006). These operations may be entered individually in a command line interface, or executed collectively in scripts (known as procedures or batch files). In 2005, SPIDER became freely available under the terms of the GNU General Public License. The most current documentation is at the SPIDER home page: http://www.wadsworth.org/spider_doc/spider/docs/spider.html.

1.1. SPIDER commands

SPIDER has over 450 commands (www.wadsworth.org/spider_doc/spider/docs/operations_doc.html), which include standard image processing operations, many of which have been extended to work on 3D data, as well as many operations specialized for cryo-electron microscopy (cryoEM), such as alignment and reconstruction commands for single-particle reconstruction and tomography. Many of the specialized EM commands are actually sets of commands—for example, alignment operations have the 'AP' prefix, while reconstruction commands have the 'BP' prefix. These implement a variety of algorithms for each process, giving users the ability to try different alignment and reconstruction approaches with their data. In addition, new algorithms are constantly being incorporated into SPIDER.

Commands may be entered one line at a time at the SPIDER prompt. This lets users interactively try out individual operations, and provides maximum flexibility at the lowest level. However, complex tasks may use dozens or hundreds of SPIDER operations in sequence, requiring

* Corresponding author. Fax: +1 518 486 2191.
E-mail address: joachim@wadsworth.org (J. Frank).

1047-8477/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jsb.2006.07.019

them to be run in batch mode. SPIDER commands may be typed into a text file just as they are entered at the SPIDER prompt, and passed to SPIDER on the operating system command line. Such SPIDER scripts are called batch files, or procedures—the terms are often used interchangeably (although technically, a batch file is the top-level script, while a procedure is a called subroutine).

1.2. SPIDER batch files

SPIDER's procedural language is a full-fledged programming language with control structures (such as conditionals and iteration), variable assignments, and subroutine calls with controllable scoping rules. Batch files typically encapsulate an identifiable step in single-particle or tomographic processing, such as calculation of power spectra from micrographs, or an automated particle picking procedure. Hundreds of operations are often "chunked" together in a single file.

A typical SPIDER reconstruction project involves dozens of scripts and anywhere from hundreds to hundreds of thousands of data files. For example, tomographic reconstruction requires approximately 400 SPIDER commands in 12 batch files, while single-particle reconstruction currently uses over 1600 commands in about 50 scripts (see www.wadsworth.org/spider_doc/spider/docs/techs/recon/mr.html).

Over the years, SPIDER batch files for accomplishing various reconstruction tasks have been written by the cryo-EM community, encapsulating a tremendous amount of accumulated algorithmic knowledge. The sum of SPIDER, therefore, is not just the over 400 operations that constitute its command-level interface, but also the collection of batch files in laboratories using SPIDER around the world, particularly the collection at the Wadsworth Center in Albany, NY (see a listing of techniques at www.wadsworth.org/spider_doc/spider/docs/techniques.html). SPIDER thus gives scientists access to many specialized tasks for cryo-EM; it is also a programming environment for users who wish to try variations on the existing themes, or to write their own procedures from scratch.

This tremendous flexibility comes at the price of complexity, in terms of the number of operations required and the interconnectedness of procedures. Some users who are new to SPIDER have found it difficult to master the old-fashioned interface. Because SPIDER is run from the command prompt, some familiarity with the operating system is required to understand the layout of the files and directories that are generated by the batch files during a reconstruction project. In contrast, more recent systems for single-particle reconstruction, such as EMAN (Ludtke et al., 1999), offer simpler interfaces with programs that perform fewer, but higher-level, tasks. Students and new users are often more comfortable with this latter type of interface. Once scientists gain a deeper understanding of the processes occurring during reconstruction, however, they often wish to 'get their hands dirty' and make significant changes to the operations being applied, or even to write new ones. However, the programs behind the simpler interfaces are essentially 'black boxes', that present a significant hurdle to this level of user participation: one must either write code in a low-level programming language, or contact the authors of the software. For that matter, SPIDER operations are also 'black boxes'.

2. SPIRE

2.1. The concept of the "Reconstruction Engine"

To facilitate the execution of batch files, and to provide a graphical user interface (GUI) for executing SPIDER batch files and other programs, a "Reconstruction Engine" was envisaged. Such a utility would simplify carrying out reconstruction projects in SPIDER, while still allowing more advanced users to write their own scripts. This new interface would not require any radical redesign of the SPIDER software (which comprises some 180,000 lines of Fortran), but would act as a higher-level "shell" around SPIDER, keeping the SPIDER command structure and syntax in place.

SPIRE (the SPIDER Reconstruction Engine) is designed to simplify running projects that consist of numerous SPIDER batch files, as well as to manage and organize the many output files created during a reconstruction project. Users who are new to SPIDER can quickly start processing electron micrographs without having to learn the command-line interface. SPIRE provides a convenient environment for running, testing, and debugging batch files. SPIRE was primarily designed to be used with batch files written for a specific problem, such as single-particle reconstruction. These are typically a well-debugged set of batch files used on a regular basis in a laboratory, although SPIRE permits users to tailor the procedures to suit the particular needs of their project.

SPIRE is written in Python, and uses the Tkinter graphics package (www.python.org). The SPIRE distribution also includes numerous other graphical tools. It requires the following versions (or higher): SPIDER 13, Python 2.3, Python megawidgets 1.1 (Pmw), and Tcl/Tk 8.4. More detailed documentation, including a tutorial, may be found at the SPIRE home page: www.wadsworth.org/spider_doc/spider/docs/spire/index.html

2.2. The graphical interface

SPIRE presents a graphical environment for executing SPIDER batch files (Fig. 1). The interface is organized around dialogs (Fig. 2), which list a set of conceptually related batch files, such as procedures for alignment or particle picking. Each batch file is associated with an execution button, an Edit button, and a brief descriptive label (since it is sometimes difficult to tell a batch file's function from its name). After running a batch file, SPIRE checks which outputs have been generated, and adds them to an internal

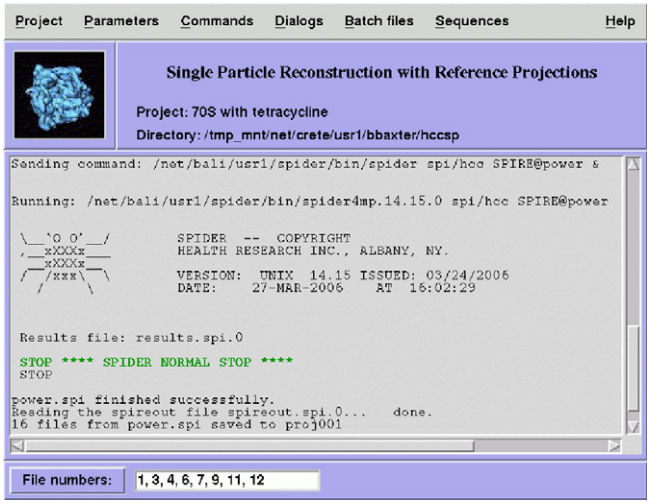

Fig. 1. The main window of SPIRE. The configuration shown is for a single-particle reconstruction project. When a SPIDER batch file is executed, its outputs are displayed in the window. File numbers are specified in the white entry box at the bottom.

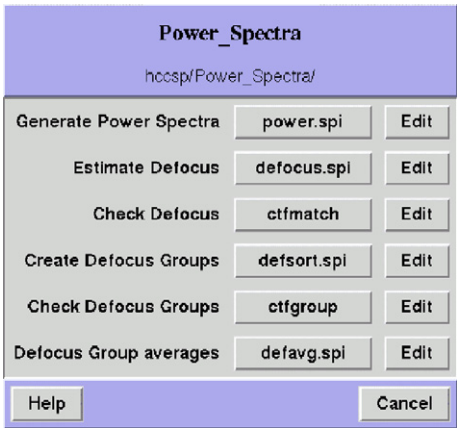

Fig. 2. A dialog in SPIRE, listing a set of SPIDER batch files. Each batch file has a brief descriptive label, an execution button with the batch file name (such as power.spi), and an Edit button that opens up a batch file edit form (see Fig. 6). Other programs can also be launched from the interface (such as Web or CTFmatch).

database of the project. The current state of the project can be viewed in the Project Viewer, which displays all batch files that have completed successfully, as well as output files generated by each one (Fig. 3). Output files can be easily reviewed: with a click of the mouse, contents of text files are displayed in a text viewer, while images are displayed in an image viewer. In short, SPIRE lets the user edit and run his or her SPIDER batch files, keep track of the current state of the SPIDER project, and easily view SPIDER data files, all in a simple graphical interface. Additional features include a set of graphical tools for data analysis, links to local documentation web pages, and interaction with a laboratory database for downloading parameters pertinent to the project and uploading reconstruction results.

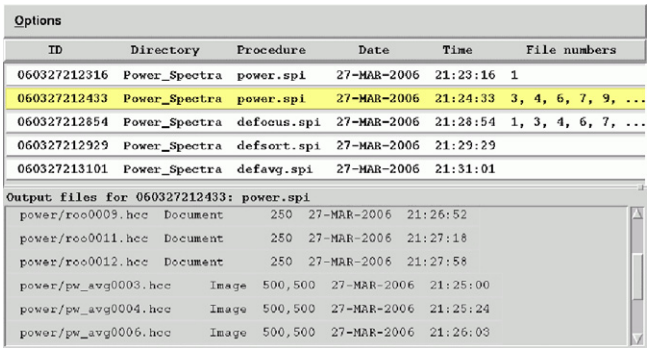

Fig. 3. The Project Viewer. Individual batch runs (see text) are displayed in the upper panel. Selecting a batch run (shown in yellow) displays its output files in the lower panel. Clicking an output file name displays that file; image files are shown in a viewer such as JWeb, while text files are shown in a text viewer or editor (the specific viewers and editors are specified in the Options section).

2.3. Management of projects in SPIRE

When a new project is started, the Project Dialog window opens (Fig. 4). The user must fill in required information, including a title specific to this project, the data extension, the location of project data files (e.g., micrographs), and a configuration file (see below). This information is saved to the project file; it may be edited at a later date. Once some batch files have been run and their data files generated, the project file also contains lists of executed batch files and their outputs. The project file is a Python *shelve* object, an external file that stores objects in a Unix-style database (see the Python documentation).

Typically, a laboratory using SPIDER has a core set of batch files in a central repository that it relies on for typical reconstruction projects. When a user starts a new project, these core batch files are copied to the user's local data

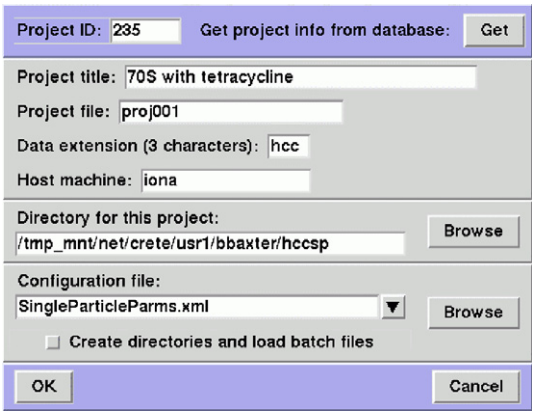

Fig. 4. The Project Form, for starting up new projects or editing previous projects. The user must enter project information such as a title, data extension, directory, etc. This information may be entered manually or retrieved from an external laboratory database. The configuration is selected at this point, which specifies batch files for tomography or single-particle work or the user's own batch files. When the user clicks 'OK', project directories are created and batch files are copied into the appropriate locations.

directory. Users can then make changes to their local copies without affecting the core set. SPIRE relies on this idea of a core set of batch files, specified in a configuration, which users may tailor to suit their specific needs.

2.4. Configuration files

A project in SPIRE requires the core set of batch files used for a specific reconstruction project, and the directory organization required for such a project. These are specified in project templates called *configurations*. Configurations are text files with XML tags that describe where to find the core set of batch files, and the target directories they should be copied to.

There is additional information that controls the presentation of the graphical interface, and any documentation URLs. For example, part of a configuration file is shown in Fig. 5. The tags in this section describe how the top two buttons in the dialog in Fig. 2 should appear. Configurations thus provide a way to describe standard, frequently used projects. For example, at Albany, there are configurations for single-particle reconstruction using reference projections, single-particle reconstruction using random conical data collection, and single- and double-tilt tomography, each with its own batch files and layout of files and directories. Variants on these approaches can be created by editing the XML files or by using the SPIRE's Configuration Editor. Users can create their own configurations to suit their processing needs. Configurations enable SPIRE to provide solutions for many different reconstruction strategies, and to manage the complexity of the SPIDER programming environment.

2.5. Running SPIDER batch files in SPIRE

A batch file can be launched interactively in several ways within SPIRE:

(1) by clicking its button in a dialog window (Fig. 2), or
(2) by selecting it under the Batch files menu, or
(3) by clicking the "Execute" button in the batch form.

```
<dialog>
    <title>Power_Spectra</title>
    <dir>Power_Spectra</dir>

    <button>
        <label>Generate Power Spectra</label>
        <buttontext>power.spi</buttontext>
        <proc>power.spi</proc>
    </button>

    <button>
        <label>Estimate Defocus</label>
        <buttontext>defocus.spi</buttontext>
        <proc>defocus.spi</proc>
    </button>
```

Fig. 5. Part of a SPIRE configuration file, with XML tags for 2 batch file buttons (see Fig. 2). The <proc> tag specifies the batch file to be run by SPIDER.

SPIRE executes SPIDER in an operating system subshell. The executed command appears in SPIRE's main window, along with standard output and any error messages from SPIDER. A small window opens with some information about the process, indicating whether it is still active, and allowing the user to check the progress of the process by displaying the last few lines of the Results file. If necessary, the user can elect to terminate the process from this window.

When the batch file has finished successfully, SPIRE determines which new files were created, and adds their file names to the project file. If there is an error, usually the first place to look when trying to diagnose a batch file problem is the SPIDER Results file, which is conveniently printed in the main window. If a batch file is run outside of SPIRE, at the operating system prompt, then the output files cannot automatically be added to the project file. However, even in this circumstance, SPIRE does provide a manual method to add data files to the project.

Finally, a series of batch files can be saved in a list, called a *sequence*. Execution of a sequence automatically runs each batch file in the list, one after another, unless an error halts the process. All activities are noted in a log file; all generated outputs are saved to the project file.

2.6. Editing batch files

SPIRE can present a batch file in a graphical form, with entry boxes for the user to enter or verify values (Fig. 6B). However, this only works if the batch file has a proper batch file header (Fig. 6A). After the user makes the desired changes to the form, and hits the Save button, the new values are written into the batch file.

SPIRE can run any SPIDER batch file. But in order to present a batch file in the graphical batch form, the file must have a header following certain rules:

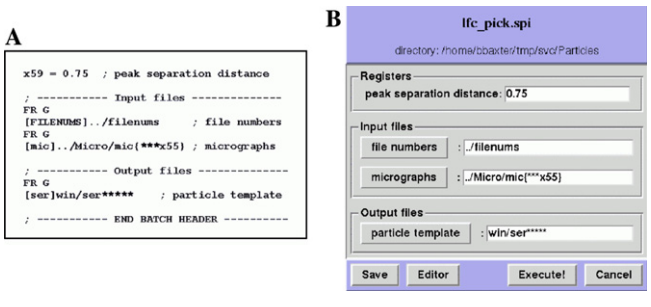

Fig. 6. (A) A header from a SPIDER batch file. Filenames are set using SPIDER's FR commands. (B) How SPIRE interprets the header and displays the batch file to user. Comments in the header appear as labels in the batch file form, or are used to delineate sections of the form ('Input files', 'Output files'). The user may edit the filenames in the white entry boxes, which are then written to the batch file.

- All register and symbolic filename assignments should be in the header.
- These assignments should have a one-line comment on the same line.
- Registers, input files, and output files should be enclosed in sections with section headings.
- Each section should start with a comment of the form "-- input files --". That is, there should be at least 2 hyphens before and after the title.
- The header ends with a final comment line containing the text "END BATCH HEADER".

If the batch file does not have a properly written header, SPIRE will inform the user about the errors it encountered. A header conforming to these rules is shown in Fig. 6A; the resulting graphical form is shown in Fig. 6B. Registers may be assigned using the new named format available starting with SPIDER version 14, or using the older "x11" format. Filename variables ("symbolic parameters" in SPIDER) must be created via the operations 'FR G' and 'FR L' (see the SPIDER documentation). In the resulting window, the labels to the left of the filename entries also act as "Browse" buttons, allowing the user to search the file system for the desired inputs (Fig. 6B).

While the use of a header does impose a constraint, it makes perfectly logical programming sense, whether or not one uses SPIRE. A header makes explicit the common organizing principle of having a set of user-changeable variables at the top of a program, followed by code that the user does not normally alter (and is in fact discouraged from doing so). Any set of commonly used batch files should have header structures to make them more readable and maintainable. If the user does wish to make alterations to the batch code following the header, then the batch file can be opened in a text editor via the "Editor" button, and SPIRE will incorporate any changes thus made. The particular editor is specified in the Options section.

2.7. SPIDER project files

Hundreds of thousands of data files can be generated in the course of a modern cryoEM single-particle reconstruction project, usually organized by some sort of directory structure. Dozens of batch files are executed in a typical SPIDER project; some of them fail due to errors, and need to be fixed and run again. Without some overall system of organization, it is often difficult to determine the current state of the project, in terms of which procedures have been run, which data files were produced, and, consequently, what needs to be done next. SPIRE's Project Viewer (Fig. 3) is a graphical management tool that displays the batch files that have been run, along with their location on disk, the date and time of execution, and which file numbers were included (see File Numbers, below).

A batch run refers to a successfully completed SPIDER batch file execution whose results are added to the internal project data base. Each batch run is given a unique Run ID (a 12-digit number corresponding to the current date and time). Thus, the same batch file can be executed repeatedly, with different input files—each execution is considered a separate batch run. For example, in Fig. 3, the procedure *power.spi* was initially run with a single input file (first row, file number 1). Satisfied with the correctness of the output, the user then ran *power.spi* again with a larger set of input files (2nd row, with file numbers 3,4,6,7,9,11,12).

Data files can be listed in the lower panel by clicking on the batch file that generated them. SPIDER generally creates two types of data files: *document files*, which are text files with columns of numbers, and *binary files*, which include 2D images, 3D volumes, and their Fourier-transformed counterparts. The data file display includes the file name, file type, creation time, and an indication of the file size: the number of lines for document files, or the dimensions for binary files. If the list of files is very large, SPIRE offers to display a subset. Clicking on a data file in the lower panel will bring up that file in some kind of display, depending on the type of file. Text files are displayed in a text viewer (usually an editor), while binary files are sent to an image/volume viewer. The viewing programs themselves are outside of SPIRE, but the Project Viewer is associated with text and image viewers in SPIRE's Options section. The default image/volume viewer is JWeb; clicking on an image or volume data file will bring up the appropriate display in JWeb.

A distinction should be made between data files listed in a SPIRE project and those residing on the physical disk. The Project Viewer lists data files that are in its internal project database—there is no consistency checking between the internal list of files and the physical files, but this feature may be added in later versions. SPIRE's file browser does allow one to see files on disk. Therefore, if a user deletes files using the operating system command line, SPIRE's internal database will not reflect this. Similarly, if a batch file is executed outside of SPIRE, its outputs will not be included in the project file. Therefore, once a project is started, it behooves the user to manage the file system through SPIRE, so that the project remains consistent and up-to-date. However, even if this stipulation is not rigorously followed, SPIRE does provide the ability to manually add or delete files from the internal project database.

Finally, SPIRE lets the user write the project file to disk in HTML format, so that the user (and others) can use a web browser to find out which batch files were run and which data files were generated in the course of a project. This is best done at the end of a project, since the HTML files may not reflect the most recent processing. Due to space considerations, only a relatively few results from a typical project are saved to the laboratory database, and the remainder of the intermediate data files are deleted at the end of a project. The project HTML files thus represent a convenient review of the project's history (Fig. 7).

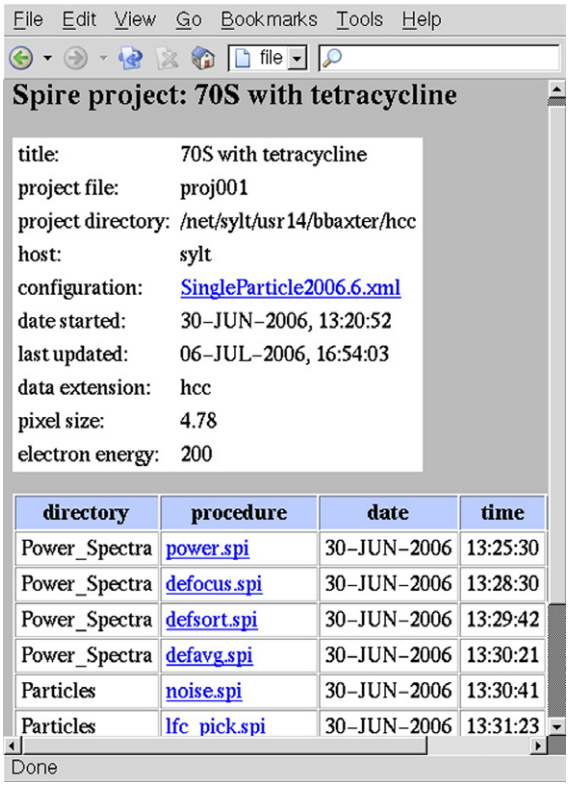

Fig. 7. Main HTML page of a SPIRE project viewed in a web browser (partial view). Batch files are listed in the order they were run—each provides a link to a page with information specific to that particular batch file.

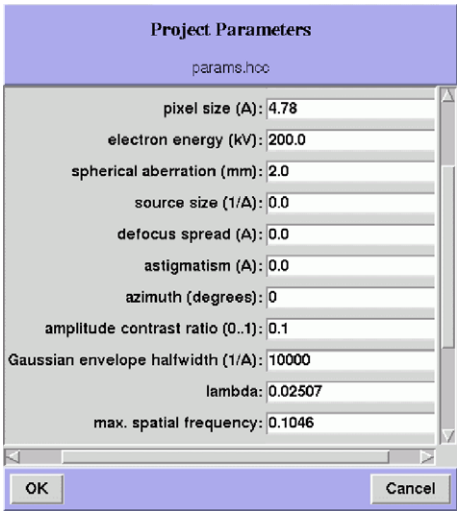

Fig. 8. The parameter file form. The parameter file is simply a SPIDER document file with comments for each line. The comments appear as labels in the form. Parameter values may be changed by editing the values and saving to the parameter file.

2.8. SPIDER–SPIRE interaction

How does SPIRE obtain the list of output files generated by a batch file? This brings up the question of how SPIDER and SPIRE are interfaced. SPIRE was designed as a very "light" wrapper around SPIDER, requiring a minimum of changes to the SPIDER code. When a batch file is executed from within SPIRE, SPIDER is called with the syntax (spider bat/dat SPIRE@procfile) to produce a special output file for SPIRE, called *spireout*, which lists the newly created data files. SPIDER runs as an independent subprocess, while SPIRE monitors this process. Upon completion of the SPIDER process, SPIRE reads the spireout file to obtain the list of output files.

2.9. Parameters and file numbers: special SPIDER document files

Some special document files have been incorporated into SPIRE: the *parameter file* and the *file numbers file*. They are not inherent to any SPIDER project or set of batch files, but rather represent an additional feature permitting authors to write batch files that are simpler to use and less error-prone. A parameter file makes the use of variables across batch files more consistent and thereby reduces errors. A file numbers file circumvents the problem of using nonconsecutive file numbers in a SPIDER DO loop. SPIRE expects the parameter and file numbers files to have specific names (usually *params.[ext]* and *filenums.[ext]*, respectively). The default name of these files, and the means to change them are in the Options section of SPIRE.

2.9.1. The parameters file

Some parameters are used by many batch files. For example, if a register containing the pixel size occurs in several procedures, the user must remember to set this register to the correct value in every file. It makes more sense to collect such global variables in a central location where all batch files may access them. The parameter file is simply a single-column SPIDER document file, which SPIRE presents in a graphical interface (Fig. 8). Descriptive labels must be supplied in the form of document file comment lines. Alternatively, the Parameters section of the configuration file can specify labels as well as default values. Use of parameters within batch files is not enforced by SPIRE—it is entirely up to the batch file authors to use this construct. SPIRE simply offers default parameters in the configuration, a graphical form to conveniently change parameters, and the opportunity to fetch parameters from an external database.

2.9.2. The file numbers file

The iteration construct in SPIDER's procedure language consists of the DO command:

```
DO LB1 x11 = 1,10
:
LB1
```

Since data files are usually numbered, it is convenient to use the file number as the loop variable when processing many files. However, in the course of a project, many candidate files are rejected, leaving one with a set of nonconsecutively numbered files. The file number cannot then be used as the loop variable, as the DO command requires regular increments in this variable. The answer is to use a file numbers file: a single-column document file with consecutive keys and file numbers in the first data column. As with the parameter file, use of such files is not inherent to the batch language, but is simply a sensible, commonly used construct.

The SPIRE interface to the file numbers file is simply an entry box on the lower left corner of the Main window (Fig. 1). When the user types file numbers into this box, and hits the "Enter" key, these values are written to the file numbers file. Consecutive numbers may be represented by hyphens: typing "3-8" will write the values 3,4,5,6,7,8 to a doc file (with keys numbers 1-6). This is very convenient for interactively testing batch files with one or a small number of inputs. Alternatively, the File Numbers label in the main window acts as a browse button, which will read the selected file and load its contents into the file numbers entry box.

SPIRE searches the batch file header for a symbolic variable called [FILENUMS]. To be recognized, it must be capitalized, and inside square brackets. If such a variable is found, SPIRE will replace the contents of the designated document file ("../`filenums`" in Fig. 6B) with the numbers in the File Numbers entry box.

2.10. Connecting to an external database

If a laboratory already has a database of reconstruction projects, SPIRE can connect to this external database to obtain or upload project information. This generally occurs at the beginning and end of a reconstruction project. At the start of a project, SPIRE can download some project parameters (e.g., voltage, magnification, and scanning resolution) from the database. At the end of a project, results (such as resolution curves, volumes, etc.) can be uploaded into the database.

SPIRE does not know which database is being used, nor anything about the internal organization of that database. Instead, it provides an application interface, which specifies the types of methods it uses to communicate with an external database. This interface specifies generic functions such as testing if the database connection is present, sending SQL queries, and fetching information. Briefly, setting up SPIRE to connect to an external database involves (1) installing the Python library to let Python communicate with that database, and (2) writing some Python code conforming to the application interface, enabling SPIRE to access the database. The SPIRE distribution includes Python code for setting up a tiny MySQL database as well as connecting SPIRE to that database. The database section of the Python website (www.python.org/topics/database/modules.html) has modules for connecting to Informix, Ingres, MySQL, ODBC, Oracle, Postgres, Sybase, and many other database systems. SPIRE's ability to connect to a laboratory database means it can become part of that laboratory's information management system.

2.11. Options and preferences

The Options window provides the means to adjust the behavior of SPIRE, various default parameter settings, and database connectivity. User preferences are saved to an external file, usually named *.spire*, in the user's home directory. This file is read whenever SPIRE starts up. User preferences include items such as whether to save various log files, how many lines or files should appear in various displays, and the names of text and image viewers that are called from the Project Viewer. System preferences include the default names of the parameter file, file numbers file, and configuration file, as well as the system command used to execute SPIDER. A database section contains buttons and entries for testing the database connection, writing and sending SQL queries, and uploading data.

3. Discussion

Rather than embarking on a radical rewriting of the SPIDER software, we have decided to utilize Python's touted ability to serve as a 'glue' for connecting disparate software systems (van Rossum, 1998). It has served well in this role, allowing us to create interfaces to display and plotting systems, database systems, and most importantly, an environment for executing SPIDER batch files. SPIRE is not designed to solve any particular reconstruction research problem. It is up to the batch file authors to produce a set of SPIDER procedures that correctly carry out a series of processing tasks. Nor does it address issues of program correctness, except in the case of the batch header syntax. If a batch run crashes, an error message is presented that will hopefully aid the user in debugging. The correct use of file numbers and parameters are not enforced in SPIRE—again, that is up to the logic of the batch files.

SPIRE is designed to provide new users with a friendlier interface to SPIDER, and to give all users a more organized overall view of their ongoing reconstruction project, while preserving SPIDER's powerful programming environment. The configuration file specifies the core set of batch files for a particular reconstruction strategy, the directory layout of a reconstruction project, how the batch files are presented to the user, and the order in which they should be executed. It uses XML tags to denote and describe each of these sections. Thus, the configuration represents a template for reconstruction projects, which users may use again and again. A different configuration is required for a different type of project; for example, single-particle vs. tomographic reconstruction. Thus SPIRE is most useful for running frequently used batch

files that are part of a laboratory's routine data analysis. It is not designed for "single-use" batch files that SPIDER users sometimes write to do "quick and dirty" jobs. However, since each project is slightly different, SPIRE does permit users to change their local copies of the batch files, tailoring them to meet their specific needs. New batch files can even be added to a project.

The graphical interface makes use of dialogs, as a means of grouping together batch files that form a conceptual unit. Given that some reconstruction projects require dozens of batch files, the configuration aids users' understanding of a project's organization by clustering together functionally related procedures. The graphical batch forms provide a form of "data hiding", in which users have immediate access only to those parameters they should be allowed to change. More extensive editing of the batch file logic is permitted, but this requires a few extra steps. SPIRE does not check for correctness of parameters, or whether specified data files are appropriate, or even exist. These checks are dependent on the batch file logic. One can imagine a system that does these things, but since "correctness" is highly dependent on the particulars of the task at hand, such a system would no longer be a general interface for running SPIDER batch files.

Connectivity to a laboratory database was incorporated into the design of SPIRE in its earliest phases. It requires a reconstruction project to exist in a database before any batch files have been run. This feature encourages users to enter into the database the various microscope parameters that SPIRE will request at the very start of a project, rather than to add their results belatedly to the laboratory archive after the completion of a project. Automatic downloading of parameters from a database also ensures their correctness (or at least it gives users fewer chances to enter them incorrectly). Similarly, at the end of a project, automated uploading of reconstruction results makes it more likely that database entries will be filled correctly. The problem with connecting to an external database is that SPIRE does not have any knowledge about the structure of the database or how to go about asking for information contained therein. Instead, a programming interface is provided, which requires some Python programming on the part of the laboratory using SPIRE; if necessary, with some assistance from the Wadsworth/SPIDER team. A MySQL interface comes with the SPIRE distribution as an example. There are many Python database libraries available; hopefully, as more people use SPIRE, more database interfaces will be written and included in the SPIRE distribution.

Thus, SPIRE is a program that provides a graphical interface for users of the SPIDER software, while allowing users to quickly view images via a convenient interface to JWeb and other display programs. It can also execute batch files that utilize PubSub (http://www.wadsworth.org/spider_doc/spider/pubsub/pubsub.html), a publish and subscribe system, written in Perl, that runs SPIDER procedures in parallel on a distributed cluster of computers or within a single cluster. A number of future improvements to SPIRE are envisaged. These include consistency checking between the internal project database and the files on disk, a greater selection of external database interfaces (in addition to the currently included MySQL), greater integration between SPIRE and display programs such as JWeb, enabling users to run a batch file and immediately view the resulting images, and the ability to write SPIDER batch files in the Python language. The latter will incorporate a Python wrapper for Spider commands, and the Spider Python Library (www.wadsworth.org/spider_doc/spider/docs/python/spipylib/index.html); both are currently under development.

SPIRE was written from a desire to provide functionality while preserving flexibility for SPIDER users, giving them access to display programs, parallel distributed processing, and interaction with the local laboratory database. It is hoped that the interface makes new SPIDER users more productive more quickly, and that it assists laboratory information systems to maintain and monitor reconstruction projects more efficiently.

Acknowledgments

This work was supported by NIH Grants P01 GM064692 and P41 RR01219 from the National Center for Reseach Resources (NCRR/NIH).

References

Frank, J., 2006. Three-Dimensional Electron Microscopy of Macromolecular Assemblies. Oxford University Press, New York.

Frank, J., Radermacher, M., Penczek, P., Zhu, J., Li, Y., Ladjadj, M., Leith, A., 1996. SPIDER and WEB: processing and visualization of images in 3D elctron microscopy and related fields. J. Struct. Biol. 116, 190–199.

Frank, J., Shimkin, B., Dowse, H., 1981. SPIDER—a modular software system for electron image processing. Ultramicroscopy 6, 343–358.

Frank, J., Shimkin, B., 1978. A new image processing software system for structural analysis and contrast enhancement. In: Sturgess, J.M. (Ed.), Proceedings of the 9th International Conference on Electron Microscopy, Toronto, Ontario, vol. I, p. 210.

Ludtke, S.J., Baldwin, P.R., Chui, W., 1999. EMAN: semiautomated software for high resolution single particle reconstruction. J. Struct. Biol. 128, 82–97.

van Rossum, G., 1998. Glue it all together with Python. OMG-DARPA-MCC Workshop on Compositional Software Architecture. Monterey, California, January 6–8.