Journal of Structural Biology xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Journal of Structural Biology

journal homepage: www.elsevier.com/locate/yjsbi

Scipion: A software framework toward integration, reproducibility and validation in 3D electron microscopy

J.M. de la Rosa-Trevín [a,b,*], A. Quintana [c], L. del Cano [a], A. Zaldívar [a], I. Foche [d], J. Gutiérrez [a], J. Gómez-Blanco [a], J. Burguet-Castell [a], J. Cuenca-Alba [a], V. Abrishami [a], J. Vargas [a], J. Otón [a], G. Sharov [e], J.L. Vilas [a], J. Navas [a], P. Conesa [a], M. Kazemi [a], R. Marabini [b], C.O.S. Sorzano [a,f], J.M. Carazo [a]

[a] Biocomputing Unit, National Center for Biotechnology (CSIC), Darwin 3, Campus Universidad Autónoma de Madrid, 28049 Cantoblanco, Madrid, Spain
[b] Escuela Politécnica Superior, Universidad Autónoma de Madrid, Campus Universidad Autónoma, 28049 Cantoblanco, Madrid, Spain
[c] Neuroscience, Physiology and Pharmacology Dept., Univ. College London, Gower Street, London WC1E 6BT, United Kingdom
[d] EyeSeeTea Ltd., Delta Place, 27 Bath Road, Cheltenham, Gloucestershire GL53 7TH, United Kingdom
[e] L'Institut de Génétique et de Biologie Moléculaire et Cellulaire, 1 Rue Laurent Fries, 67400 Illkirch-Graffenstaden, France
[f] Bioengineering Lab., Universidad CEU San Pablo, Campus Urb. Montepríncipe s/n, 28668 Boadilla del Monte, Madrid, Spain

ARTICLE INFO

Article history:
Received 19 March 2016
Received in revised form 18 April 2016
Accepted 19 April 2016
Available online xxxx

Keywords:
Electron microscopy
Single particle analysis
Image processing
Software package
Workflows
Reproducibility

ABSTRACT

In the past few years, 3D electron microscopy (3DEM) has undergone a revolution in instrumentation and methodology. One of the central players in this wide-reaching change is the continuous development of image processing software. Here we present Scipion, a software framework for integrating several 3DEM software packages through a workflow-based approach. Scipion allows the execution of reusable, standardized, traceable and reproducible image-processing protocols. These protocols incorporate tools from different programs while providing full interoperability among them. Scipion is an open-source project that can be downloaded from http://scipion.cnb.csic.es.

1. Introduction

In the last few years, we have witnessed a revolution in the 3DEM field due mainly to extraordinary improvements in equipment, computing power and software tools (Kuhlbrandt, 2014). The introduction of direct detection devices (DDD) has made a fundamental difference in image acquisition quality, as they enhance the resolution achieved by earlier image-recording media such as photographic film or charge-coupled devices. Moreover, computer power has increased notably through the use of multi-core machines, clusters, graphics cards and even cloud computing (Schmeisser et al., 2009; Cianfrocco and Leschziner, 2015). These combined developments have allowed more computationally intensive methods and larger datasets, which permit more challenging biological questions to be posed.

A variety of software tools are available to the 3DEM community, ranging from command line programs to complete software suites. A non-exhaustive list of these general packages includes Appion (Lander et al., 2009), Bsoft (Heymann and Belnap, 2007), CTFFIND (Mindell and Grigorieff, 2003; Rohou and Grigorieff, 2015), EMAN (Ludtke et al., 1999; Tang et al., 2007), FREALIGN (Grigorieff, 2007), IMAGIC (van Heel et al., 1996), 2dx (Gipson et al., 2007), RELION (Scheres, 2012), SIMPLE (Elmlund and Elmlund, 2012), SPARX (Hohn et al., 2007), SPIDER (Frank et al., 1996), and Xmipp (de la Rosa-Trevín et al., 2013; Sorzano et al., 2004) (for an exhaustive list, see https://en.wikibooks.org/wiki/Software_Tools_For_Molecular_Microscopy). Each of these packages has its strengths and weaknesses, and no single package provides the best solution in all situations. In many projects, researchers combine tools from various software packages to create a processing pipeline. Differences in file formats, conventions for Euler angles, contrast transfer function (CTF), and other metadata nonetheless make movement between packages time-consuming, error-prone and difficult to document.

* Corresponding author at: Biocomputing Unit, National Center for Biotechnology (CSIC), Darwin 3, Campus Universidad Autónoma de Madrid, 28049 Cantoblanco, Madrid, Spain.
E-mail address: jmdelarosa@cnb.csic.es (J.M. de la Rosa-Trevín).

http://dx.doi.org/10.1016/j.jsb.2016.04.010
1047-8477/

Previous approaches to package integration in the field include:

- *IPLT* is a software framework that provides an open-source comprehensive library for the EM community. It was implemented using the programming languages C++ and Python. In principle, the IPLT Python layer would allow it to call external programs, although to the best of our knowledge, this possibility has not been exploited fully (Philippsen et al., 2007).
- *2dx* is a software package that wraps the MRC software for electron crystallography. It has a friendly graphical user interface (GUI) in which final and intermediate outcomes can be reviewed using incorporated visualization tools (Gipson et al., 2007).
- *SPIRE* is a framework that provides a GUI to process SPIDER modules. It also has a database with some level of traceability to the processing workflow. SPIRE has a configuration file that, in principle, enables the user to execute any external program within its environment, although it has not been used extensively for this task (Baxter et al., 2007).
- *SPARX* is a Python framework and a core library of fundamental C++ image processing functions that includes a user interface built around EMAN2. It also introduces a distinct data/process-flow support infrastructure (Hohn et al., 2007).
- *Appion* is the only platform in the 3DEM field that allows real integration of different software packages (Lander et al., 2009). It is Python-based and tightly integrated with a relational SQL database and with Leginon (Suloway et al., 2005), a system designed for automated collection of images from transmission electron microscopes. Appion is a web-based pipeline with registered input and output data that provides user guidance throughout the reconstruction process.

In addition to the integration approaches enumerated above, there is a trend in most suites to incorporate tools from others, mainly through conversion scripts that translate from one format/convention to another. This approach, which is difficult to maintain and extend, places considerable responsibility on the final user.

Another important problem that affects the cryo-EM community (and the scientific community in general) is the difficulty in reproducing published research studies. Most of the time, the precise reproducibility of the final reconstruction cannot be guaranteed because the traceability of the process relies entirely on laboratory notebooks and good practices. Although the image processing is described to some extent in Materials and Methods sections, important steps or details can be missing, so that it might not be a trivial matter to reproduce a given result, even if access to the original raw data is provided.

It is in this scientific context that Scipion has been developed, to address the issues of integration and interoperability in 3DEM while providing full tracking of the entire image-processing workflow. Scipion also provides an intuitive GUI for both desktop and web, to launch jobs and to analyze results. Scipion was also designed to be extended easily, with rapid incorporation of new algorithms and a reduced learning curve for potential contributors. Indeed, the growth of the cryo-EM field is attracting new users from many other disciplines, which generates the need for intuitive, integrative and traceable frameworks for image processing.

2. Processing with Scipion

2.1. Integration and interoperability

The workflows proposed by different EM image processing packages are conceptually similar, and at first glance it would therefore seem an easy task to mix algorithms from the various software packages in new ways. Relatively small differences between format and the conventions followed by each package nonetheless heavily penalize software interoperability. Scipion aims to integrate algorithms from the main 3DEM software packages and to provide full interoperability among them.

An example can clarify Scipion integration capabilities. Scipion currently offers four methods for DDD movie alignment: *motioncorr* (Li et al., 2013), *correlation* (an unpublished CPU version of the *motioncorr* method developed in Xmipp), *unblur* (Grant and Grigorieff, 2015) and *optical flow* (Abrishami et al., 2015). The first three approaches perform global alignment, while the fourth is local. Within Scipion, movie frames can be aligned using any one of these approaches or any combination.

Following the standard processing workflow, the resulting micrographs are screened to eliminate those with excessive astigmatism or drift. Scipion provides three CTF estimation methods: *CTFFIND* (both versions 3 and 4) (Mindell and Grigorieff, 2003; Rohou and Grigorieff, 2015), *gCTF* (Zhang, 2016) and *Xmipp CTF estimation* (Sorzano et al., 2007; Vargas et al., 2013). The user can compute the CTFs using several algorithms and compare them with the protocol *ctf discrepancy*, which analyzes for agreement among distinct CTF estimations for the same micrograph. The protocol assumes that two CTFs are consistent if their phase values (wave aberration function (Frank, 1996)) are closer than 90 degrees. This algorithm reports an expected resolution, i.e., the resolution at which the CTF phases differ by 90 degrees.

Another important step is to select (pick) particle coordinates from the micrographs produced by any of the above-mentioned movie alignment algorithms. Scipion has eight options for this task: EMAN2 *boxer* and *sparx gaussian picker*, Xmipp *picker* (Abrishami et al., 2015), Relion *autopicker* (Scheres, 2015), Bsoft *particle picker*, *gEMpicker* (Hoang et al., 2013), *Gautomatch*(unpublished) and Appion *dogpicker* (Voss et al., 2009). Users can select any of these methods or several of them, followed by the *consensus picking* protocol, which estimates the agreement between different particle-picking algorithms. The *consensus picking* protocol takes several sets of coordinates calculated by different programs and/or parameter settings. A coordinate is considered to be correct if M pickers have selected the same particle within a radius of N pixels (both M and N are user-defined parameters).

The possibility of choosing among several algorithms as exemplified above is provided by Scipion at all levels of the workflow, in many cases adding the possibility of comparing the results of these methods to identify some form of self consistency. Fig. 1 shows a non-exhaustive summary of the algorithms offered by Scipion for the processing steps. A complete list with the current integrated Packages and Protocols (more than 100) is available at https://github.com/I2PC/scipion/wiki/Integrated-Protocols.

To achieve this level of interoperability, we developed a model for representing data and algorithms used in single particle image processing. Instead of dealing with files (and their formats) across the entire framework, we store data (and their relationships) as an abstract representation. Examples of data objects are *Particle*, *Volume*, *Mask*, *SetOfParticles*, *SetOfCTFs*, among others. We similarly model algorithms as protocol objects that perform a specific task and have well-defined inputs and outputs (to be chosen from Scipion data objects). Protocols serve as wrapper scripts that internally call one or more programs. The Scipion data model sets up the basic building blocks for the whole framework and handles all data and metadata exchange between algorithms.

2.2. Tracking and reproducibility

Traceability and reproducibility are an important part of scientific data analysis. Whereas a high-level summary of experimental results is typically recorded and published, a detailed description is

Please cite this article in press as: de la Rosa-Trevín, J.M., et al. Scipion: A software framework toward integration, reproducibility and validation in 3D electron microscopy. J. Struct. Biol. (2016), http://dx.doi.org/10.1016/j.jsb.2016.04.010

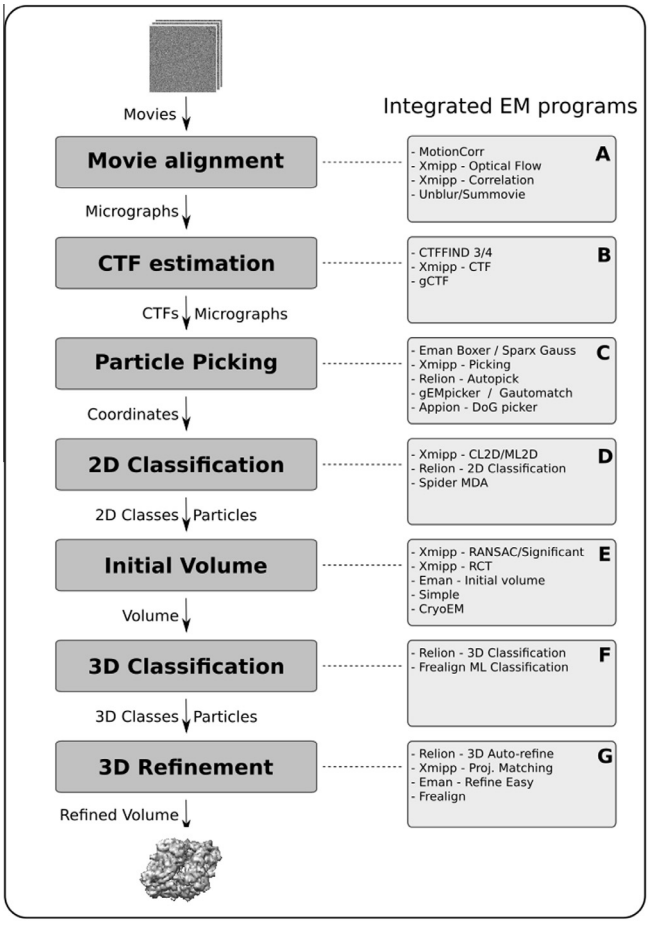

Fig. 1. General image processing workflow for single particle analysis. For each step in the workflow (left), we show some of the available protocols implemented in Scipion (right).

seldom available, which may make it difficult or impossible to reproduce the reported results.

One of Scipion's main goals is to provide the entire track of the processing pipeline. Each new protocol execution is stored, together with the parameters selected. All output logs are also stored, allowing the expert user to inspect more deeply the underlying commands and their results.

For a given project, Scipion displays each protocol execution (called *run* in Scipion terminology) as a list or as a tree in the top-right panel of the project window (Fig. 2). When using the tree view, two runs will be interconnected if the output of one is used as the input of the other. It is straightforward to "copy" a run and re-execute it with a different input or set of parameters. It is even possible to copy several runs and repeat a whole branch of the processing tree.

All runs (or only those selected by the user) can be exported as a workflow template. This template is simply a text file (in.json format) that contains all the necessary information (i.e., run parameters, execution order and user annotations) to reproduce the processing steps. Nevertheless, due the stochastic nature of some algorithms, the exact results are not guaranteed even if the program is executed with the same input data and parameters in the same machine. In addition, some steps of the workflow that require user interaction (such as manual particle picking) cannot be repeated automatically. When workflow templates are imported into another project, all the run boxes are recreated with the same parameters and the same workflow tree. This capability is very useful for reproducing steps, for re-using the pipeline with a similar dataset, or for testing slightly different parameters.

Since the workflow templates are simply text files, they can easily be shared (e.g., via email) with others, such as collaborators or colleagues, and can even be integrated into online data repositories. One can imagine a future in which the specific workflow that leads to a given structure is deposited together with the resulting structure. This workflow could be visualized over the web and downloaded by other researchers who wish to apply it to a new dataset or to repeat the processing from the original raw data. A Scipion workflow repository is currently being developed that will allow users to submit their own workflows or download existing ones. Such a repository could be a resource that allows the entire EM community to share best practices for image processing and 3D reconstruction.

2.3. User graphical interfaces

Users interact with Scipion through a collection of GUIs that provides a uniform interface for a plethora of heterogeneous EM programs. The main GUI is a workflow editor (Fig. 2). The left panel contains a protocol menu that can be customized. The top-right panel shows the project workflow, and the bottom-right panel offers information about the selected run such as input, output, summary and program log.

By default, flowcharts are used to represent workflows. In this representation, protocols are shown as boxes connected by lines when the output of one protocol is the input of another (Fig. 2). By clicking on these protocol boxes, a form is displayed (Fig. 3A) that allows users to provide parameters to the underlying programs, as well as to consult bibliographic references (Fig. 3B). Scipion registers all the objects created, including their associated type. The forms take advantage of this information and whenever possible suggest an appropriate subset of stored objects as input (Fig. 3C). This approach avoids direct manipulation and selection of files and reduces the possibility of choosing incorrect input. Finally, the protocol developer can specify validation rules for some parameters that could prevent common mistakes even before launching the job.

Associated with the action of choosing some of the key parameters for a specific method, Scipion has "wizards", special interfaces that allow the selection of parameter values while showing their effects in real time. Fig. 3D shows a wizard for a SPIDER protocol, in which the user can observe the filtered image obtained with the parameters selected before applying it to the whole set of images.

3. Architecture

As the size of software systems increases, the organization of the overall system -the software architecture- constitutes a major design challenge. In this section, we describe some of the fundamental design choices for Scipion.

Scipion uses Python as its main language, with which it glues together different software components. For performance-critical parts, Scipion relies on underlying C++ functions. Scipion is divided into modular components that interact with one another (Fig. 4). One of the lower level modules is the Mapper, which transparently stores objects and retrieves them from databases. The current implementation of the Mapper uses SQLite (http://www.sqlite.org), a self-contained, serverless SQL database engine. SQLite allows use of all the power of SQL with no need to configure and maintain a dedicated server.

The desktop interface for the project window, the forms and the wizards were developed with Tkinter, which is the Python binding

Please cite this article in press as: de la Rosa-Trevín, J.M., et al. Scipion: A software framework toward integration, reproducibility and validation in 3D electron microscopy. J. Struct. Biol. (2016), http://dx.doi.org/10.1016/j.jsb.2016.04.010

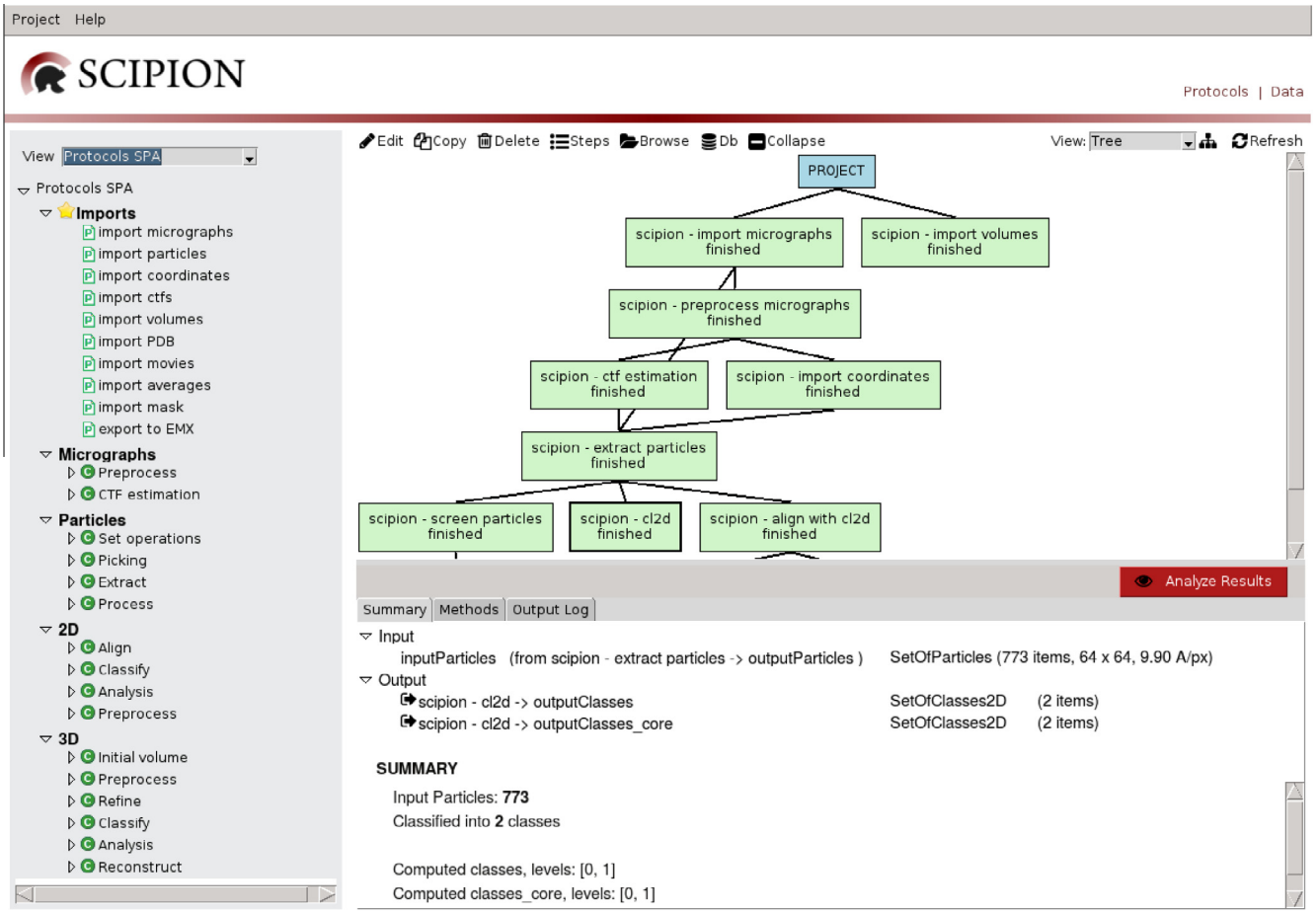

Fig. 2. Scipion main GUI. The left panel shows a customizable menu with the available protocols; top-right, the sequence of protocols executed and their status (running, finished, aborted). The bottom-right panel includes information for the selected run, such as inputs and outputs, execution logs or bibliography, and also provides the "Analyze Results' button for visualizing the run outputs.

to the cross-platform widget library Tcl/Tk (http://tkinter.unpythonic.net/wiki/). In parallel, a web interface was implemented using the Python Django framework (https://djangoproject.com). For desktop and web interfaces, the parameters form is generated automatically from the protocol definition; if a new protocol is added, there is thus no need to develop a new form for it.

Although some of the current Scipion features are derived from our experience developing Xmipp, Scipion was designed from the outset with interoperability and extensibility in mind, to serve as the front-end of any underlying EM software package. Xmipp has now been modified to contain only the basic core C++ libraries, algorithms and command line programs, whereas Scipion takes care of all project management and GUIs. All Xmipp 3.X protocols have been ported into Scipion, which allows combination with programs from other packages.

3.1. Data management and conventions

In Scipion, the basic management unit is the project. All data related to a given project are contained in a single folder. Projects can thus be moved easily from one computer to another simply by copying this folder. The execution of other jobs can continue in the new computer and, when done, the results can be copied back to the original location. In the current version, if two "copies" of a project have diverged with different runs, there is not yet a means to merge them; this feature will be added in a future release. Another improvement under study is automatic handling of data transfer between different execution hosts, with the advantage of transferring only those files needed for a given job, rather than the whole project.

Each project has a main database that stores all related information except the binary files. Scipion protocols tend to produce only metadata files, which are negligible in size compared to binary data. Due to format restrictions, the binary data must sometimes be converted for a given program; in these cases, the converted files are placed in a temporary directory that will be removed at the end of protocol execution.

One of Scipion's main goals is to provide a framework that allows simple, unambiguous information interchange among 3DEM image processing packages. To transfer metadata between two packages, the information from the first package is converted to Scipion internal representation and then converted again to the format of the second package. Scipion internal representation follows the standard proposed by the Electron Microscopy Exchange (EMX, (Marabini et al., 2016)). The only difference between Scipion and the EMX standard is the transformation matrix used to define projection directions; the Scipion matrix is the inverse of that used by EMX.

3.2. Distributed computing

Parallel processing is ubiquitous in 3DEM image processing, as it allows faster execution. At the protocol level, Scipion has a simple mechanism to achieve parallelism by running independent

Please cite this article in press as: de la Rosa-Trevín, J.M., et al. Scipion: A software framework toward integration, reproducibility and validation in 3D electron microscopy. J. Struct. Biol. (2016), http://dx.doi.org/10.1016/j.jsb.2016.04.010

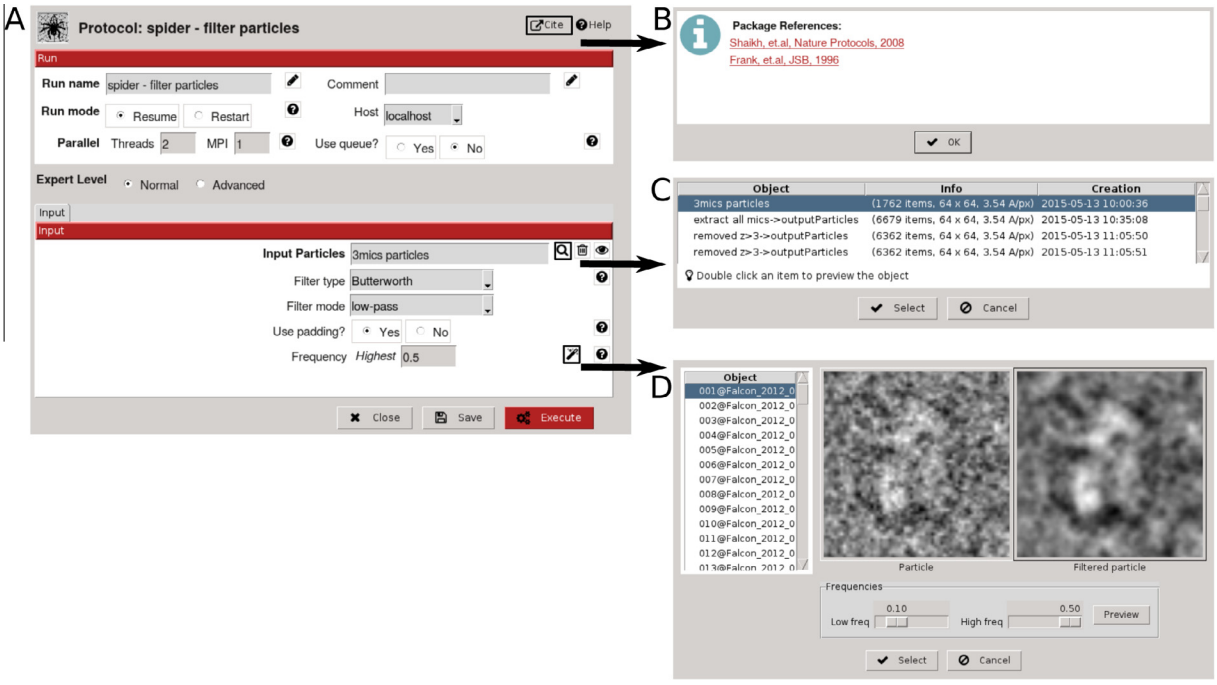

Fig. 3. Different GUIs related to the protocol *spider - filter particles*. (A) Form used to provide parameters to the underlying programs executed by the protocol. (B) Reference window with bibliography related to the protocol (pop-up window activated by clicking the cite icon). (C) Available sets of particles that can be used as input by the protocol (pop-up window activated by clicking the magnifying glass icon). (D) Wizard showing the filtering result for a given image, which helps to tune the appropriate values for low and high frequencies before executing the entire job.

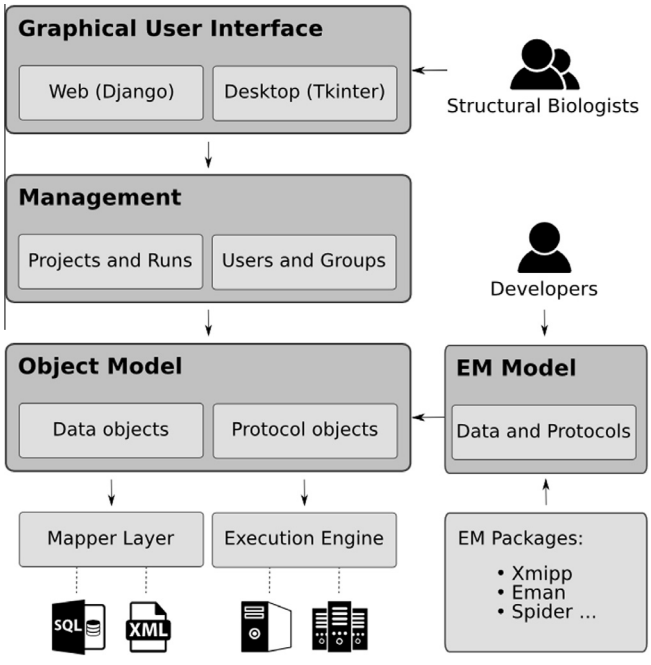

Fig. 4. Design diagram showing the main components of Scipion. Users interact with the system through a high level module that contains the graphical interfaces. Each request is handled by the Management layer, which organizes all protocol runs into projects. Data objects are automatically stored/retrieved by the Mapper layer, while protocols provide the mechanisms for task execution. General data and protocol objects provide the basis for modeling the EM domain and develop wrappers around the different software packages.

steps simultaneously. Scipion uses this parallelization strategy in protocols such as movie alignment or CTF estimation, where several movies/micrographs can be processed concurrently.

For those algorithms that provide their own parallel implementation, the configuration of the computing host (libraries, Message Passing Interface (MPI) flavor, queue system, etc.) has been isolated in a centralized location that can be accessed by all protocols. The GUI uses this configuration file, which is set up once at installation, to generate the appropriate script to launch the jobs.

Scipion is currently being extended for remote computing, to transparently run processes remotely on one system while the execution is displayed and controlled on a separate client device.

4. Getting started

Scipion is provided freely as open source software. Online documentation describing Scipion download and installation is available at http://scipion.cnb.csic.es/m/download_form/ and https://github.com/I2PC/scipion/wiki/How-to-Install, respectively.

Documentation pages for users can be found at https://github.com/I2PC/scipion/wiki/User-Documentation, including a description of most Scipion GUIs and a set of tutorials. Good starting points are the Introduction to Scipion and Mix-and-match in Scipion tutorials. The first provides a brief introduction to Scipion capabilities, while the second demonstrates the combination of different EM packages within a single workflow.

Some of the protocols incorporated in Scipion are available online, which allows users to test Scipion without actually installing it. Three workflows are currently available at http://scipion.cnb.csic.es/m/services/: (1) Initial volume estimation using EMAN2, Xmipp-RANSAC (Vargas et al., 2014) and Xmipp-Significant (Sorzano et al., 2015), (2) Movie alignment using *motioncorr* (Li et al., 2013) and Xmipp *optical flow* (Abrishami et al., 2015) and (3) Local resolution estimation with ResMap (Kucukelbir et al., 2014).

Documentation pages for developers are provided at https://github.com/I2PC/scipion/wiki/Developers-Page. Scipion architecture is described in detail and developers will find information

Please cite this article in press as: de la Rosa-Trevín, J.M., et al. Scipion: A software framework toward integration, reproducibility and validation in 3D electron microscopy. J. Struct. Biol. (2016), http://dx.doi.org/10.1016/j.jsb.2016.04.010

on how to extend Scipion as well as on development tools used in our daily work. The project is hosted at GitHub (https://github.com/I2PC/scipion), where all code is available. Any GitHub-registered developer can make a fork of the project, add new features and request that they be merged into the Scipion mainstream.

5. Discussion and conclusions

Scipion allows the transparent integration of various 3DEM software packages and offers a unified interface for experimentalists and developers. One of its key features is the underlying object-oriented model of 3DEM image processing, facilitating the writing of conversion routines among a variety of programs. The logic concepts of the model are separated from the interface, allowing GUIs to be built automatically for each protocol in both desktop and web environments.

Scipion also keeps track of all tasks performed by the user during the processing workflow, including all parameters selected and logs that can be reviewed at any time. The ability to export/import workflow templates is also of great value for teaching and training new researchers coming to the field. The storage mechanism relies on the modular Mapper layer, encapsulating the procedures of storing and retrieving any type of object, which is a crucial feature to allow extension of Scipion with new models and protocols without recurring to database details.

As mentioned before, Appion is the only other platform that allows real integration among software packages. Both frameworks are developed in Python and provide a database storage mechanism for all parameters and processing steps. Appion is tightly connected with a previous project by the same group, Leginon. This is possibly one of the main implementational differences with Scipion, in which data model and storage are less rigidly coupled. As a result, adding a new algorithm in Scipion does not require any modification of the underlying database. Both Appion and Scipion provide a web interface, while Scipion also has a desktop GUI.

In Scipion there is no relationship between two projects, while Appion maintains a unified database for all projects of a lab. This approach has pros and cons. In Scipion, it is a straightforward task to execute part of a project on one computer and then transfer it to another. On the other hand, Appion allows a more integrated view of the processing information. We opted for the lesser degree of integration, since it might be better suited to future developments in the area of remote and cloud computing.

At the software architectural level, perhaps the greatest difference is that since Scipion was developed later, special attention was paid to defining abstraction layers to simplify maintenance and extensibility. For example, to create a new protocol in Scipion, only a Python script needs to be developed. From this script, the system will discover the new protocol and will automatically build a form and store the protocol-related information in the database. In Appion, however, the developer must develop the script, the corresponding form, and an extra table in the database; Appion developers might thus need to know the underlying framework in more detail.

It is difficult to predict how EM software will evolve in the future. Our view is that software developers will continue to add algorithms to the different EM packages, but that the burden of many operations will be shifted from packages to frameworks. Bookkeeping will require special attention to provide real tracking and reproducibility. Workflows will have a key role when exploring processing alternatives. Most algorithms will need to use distributed computing through clusters or the Cloud. Scipion is our first step in implementing an integrative framework to address important problems in the field simply and effectively.

Acknowledgments

The authors want to thank Tanvir Shaikh for his contribution to the integration of SPIDER protocols of the MDA workflow, Slavica Jonic for help with the Normal Modes Analysis protocols, Alp Kucukelbir and Hemant Tagare for support with ResMap. We also thank Sjors Scheres for useful comments about RELION protocols. We are also grateful to Yaser Hasem, David Belnap and Amy Jobe for extensive testing and insightful feedback, and Catherine Mark for editorial assistance.

Javier Vargas is the recipient of a "Comfuturo" grant funded by the Fundación General CSIC, C.O.S. Sorzano is the recipient of a Ramón y Cajal fellowship from the Spanish Ministry of Economy and Competitiveness (MINECO). The authors acknowledge financial support from the MINECO through Grants AIC-A-2011-0638, BFU2013-41249-P and BIO2013-44647-R, the Comunidad de Madrid through Grant CAM (S2010/BMD-2305), the European Union (EU) and Horizon 2020 through Grants EINFRA-2014-2: EGI Engage (Proposal: 654142) and EINFRA-2015-1: West-Life (Proposal: 675858).

This work was partially funded by Instruct, a component of the European Strategy Forum on Research Infrastructures (ESFRI), and supported by national member subscriptions.

References

Abrishami, V., Vargas, J., Li, X., Cheng, Y., Marabini, R., Sorzano, C., Carazo, J., 2015. Alignment of direct detection device micrographs using a robust optical flow approach. J. Struct. Biol. 189 (3), 163–176.

Baxter, W.T., Leith, A., Frank, J., 2007. SPIRE: the SPIDER reconstruction engine. J. Struct. Biol. 157 (1), 56–63.

Cianfrocco, M.A., Leschziner, A.E., 2015. Low cost, high performance processing of single particle cryo-electron microscopy data in the cloud. Elife 4.

de la Rosa-Trevín, J., Otón, J., Marabini, R., Zaldívar, A., Vargas, J., Carazo, J., Sorzano, C., 2013. Xmipp 3.0: an improved software suite for image processing in electron microscopy. J. Struct. Biol. 184 (2), 321–328.

Elmlund, D., Elmlund, H., 2012. Simple: software for ab initio reconstruction of heterogeneous single-particles. J. Struct. Biol. 180 (3), 420–427.

Frank, J., 1996. Three-Dimensional Electron Microscopy of Macromolecular Assemblies. Academic Press, Burlington.

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 electron microscopy and related fields. J. Struct. Biol. 116 (1), 190–199.

Gipson, B., Zeng, X., Zhang, Z.Y., Stahlberg, H., 2007. 2DX–user-friendly image processing for 2D crystals. J. Struct. Biol. 157 (1), 64–72.

Grant, T., Grigorieff, N., 2015. Measuring the optimal exposure for single particle cryo-EM using a 2.6 Å reconstruction of rotavirus VP6. Elife 4, e06980.

Grigorieff, N., 2007. FREALIGN: high-resolution refinement of single particle structures. J. Struct. Biol. 157 (1), 117–125.

Heymann, J.B., Belnap, D.M., 2007. Bsoft: image processing and molecular modeling for electron microscopy. J. Struct. Biol. 157 (1), 3–18.

Hoang, T.V., Cavin, X., Schultz, P., Ritchie, D.W., 2013. gEMpicker: a highly parallel gpu-accelerated particle picking tool for cryo-electron microscopy. BMC Struct. Biol. 13 (1), 1–10.

Hohn, M., Tang, G., Goodyear, G., Baldwin, P.R., Huang, Z., Penczek, P.A., Yang, C., Glaeser, R.M., Adams, P.D., Ludtke, S.J., 2007. Sparx, a new environment for Cryo-EM image processing. J. Struc. Biol. 157 (1), 47–55.

Kucukelbir, A., Sigworth, F.J., Tagare, H.D., 2014. Quantifying the local resolution of cryo-em density maps. Nat. Meth. 11, 63–65.

Kuhlbrandt, W., 2014. Biochemistry. The resolution revolution. Science 343 (6178), 1443–1444.

Lander, G.C., Stagg, S.M., Voss, N.R., Cheng, A., Fellmann, D., Pulokas, J., Yoshioka, C., Irving, C., Mulder, A., Lau, P.W., Lyumkis, D., Potter, C.S., Carragher, B., 2009. Appion: an integrated, database-driven pipeline to facilitate EM image processing. J. Struct. Biol. 166 (1), 95–102.

Li, X., Mooney, P., Zheng, S., Booth, C.R., Braunfeld, M.B., Gubbens, S., Agard, D.A., Cheng, Y., 2013. Electron counting and beam-induced motion correction enable near-atomic-resolution single-particle cryo-em. Nat. Meth. 10 (6), 584–590.

Ludtke, S.J., Baldwin, P.R., Chiu, W., 1999. EMAN: semiautomated software for high-resolution single-particle reconstructions. J. Struct. Biol. 128 (1), 82–97.

Marabini, R., Ludtke, S.J., Murray, S.C., Chiu, W., de la Rosa-Trevin, J.M., Patwardhan, A., Bernard Heymann, J., Carazo, J.M., 2016. The electron microscopy exchange (EMX) initiative. J. Struct. Biol.

Mindell, J.A., Grigorieff, N., 2003. Accurate determination of local defocus and specimen tilt in electron microscopy. J. Struct. Biol. 142 (3), 334–347.

Philippsen, A., Schenk, A.D., Signorell, G.A., Mariani, V., Berneche, S., Engel, A., 2007. Collaborative EM image processing with the IPLT image processing library and toolbox. J. Struct. Biol. 157 (1), 28–37.

Rohou, A., Grigorieff, N., 2015. CTFFIND4: fast and accurate defocus estimation from electron micrographs. J. Struct. Biol. 192 (2), 216–221.
Scheres, S.H., 2012. Relion: implementation of a bayesian approach to cryo-em structure determination. J. Struct. Biol. 180 (3), 519–530.
Scheres, S.H., 2015. Semi-automated selection of cryo-EM particles in RELION-1.3. J. Struct. Biol. 189 (2), 114–122.
Schmeisser, M., Heisen, B.C., Luettich, M., Busche, B., Hauer, F., Koske, T., Knauber, K. H., Stark, H., 2009. Parallel, distributed and GPU computing technologies in single-particle electron microscopy. Acta Crystallogr. D Biol. Crystallogr. 65 (Pt 7), 659–671.
Sorzano, C.O.S., Marabini, R., Velázquez-Muriel, J., Bilbao-Castro, J.R., Scheres, S.H. W., et al., 2004. XMIPP: a new generation of an open-source image processing package for electron microscopy. J. Struct. Biol. 148, 194–204.
Sorzano, C.O., Jonic, S., Núñez Ramírez, R., Boisset, N., Carazo, J.M., 2007. Fast, robust, and accurate determination of transmission electron microscopy contrast transfer function. J. Struct. Biol. 160 (2), 249–262.
Sorzano, C., Vargas, J., de la Rosa-Trevin, J., Oton, J., Alvarez-Cabrera, A., Abrishami, V., Sesmero, E., Marabini, R., Carazo, J., 2015. A statistical approach to the initial volume problem in single particle analysis by electron microscopy. J. Struct. Biol. 189 (3), 213–219.
Suloway, C., Pulokas, J., Fellmann, D., Cheng, A., Guerra, F., Quispe, J., Stagg, S., Potter, C.S., Carragher, B., 2005. Automated molecular microscopy: the new Leginon system. J. Struct. Biol. 151 (1), 41–60.
Tang, G., Peng, L., Baldwin, P.R., Mann, D.S., Jiang, W., Rees, I., Ludtke, S.J., 2007. EMAN2: an extensible image processing suite for electron microscopy. J. Struct. Biol. 157 (1), 38–46.
van Heel, M., Harauz, G., Orlova, E.V., Schmidt, R., Schatz, M., 1996. A new generation of the IMAGIC image processing system. J. Struct. Biol. 116 (1), 17–24.
Vargas, J., Otón, J., Marabini, R., Jonic, S., de la Rosa-Trevín, J., Carazo, J., Sorzano, C., 2013. Fastdef: fast defocus and astigmatism estimation for high-throughput transmission electron microscopy. J. Struct. Biol. 181 (2), 136–148.
Vargas, J., Álvarez-Cabrera, A.-L., Marabini, R., Carazo, J.M., Sorzano, C.O.S., 2014. Efficient initial volume determination from electron microscopy images of single particles. Bioinformatics, btu404.
Voss, N.R., Yoshioka, C.K., Radermacher, M., Potter, C.S., Carragher, B., 2009. Dog picker and tiltpicker: software tools to facilitate particle selection in single particle electron microscopy. J. Struct. Biol. 166 (2), 205–213.
Zhang, K., 2016. Gctf: real-time CTF determination and correction. J. Struct. Biol. 193 (1), 1–12.

Please cite this article in press as: de la Rosa-Trevín, J.M., et al. Scipion: A software framework toward integration, reproducibility and validation in 3D electron microscopy. J. Struct. Biol. (2016), http://dx.doi.org/10.1016/j.jsb.2016.04.010