Contents lists available at ScienceDirect

Journal of Structural Biology

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

Xmipp 3.0: An improved software suite for image processing in electron microscopy

J.M. de la Rosa-Trevín [a,b,*], J. Otón [a], R. Marabini [b], A. Zaldívar [a], J. Vargas [a], J.M. Carazo [a], C.O.S. Sorzano [a,c]

[a] Biocomputing Unit, National Center for Biotechnology (CSIC), c/Darwin, 3, Campus Universidad Autónoma, 28049 Cantoblanco, Madrid, Spain
[b] Escuela Politécnica Superior, Universidad Autónoma de Madrid, Campus Universidad Autónoma, 28049 Cantoblanco, Madrid, Spain
[c] Bioengineering Lab., Universidad CEU San Pablo, Campus Urb. Montepríncipe s/n, 28668 Boadilla del Monte, Madrid, Spain

ARTICLE INFO

Article history:
Received 4 April 2013
Received in revised form 10 September 2013
Accepted 18 September 2013
Available online 26 September 2013

Keywords:
Electron microscopy
Single particles analysis
Image processing
Software package

ABSTRACT

Xmipp is a specialized software package for image processing in electron microscopy, and that is mainly focused on 3D reconstruction of macromolecules through single-particles analysis. In this article we present Xmipp 3.0, a major release which introduces several improvements and new developments over the previous version. A central improvement is the concept of a project that stores the entire processing workflow from data import to final results. It is now possible to monitor, reproduce and restart all computing tasks as well as graphically explore the complete set of interrelated tasks associated to a given project. Other graphical tools have also been improved such as data visualization, particle picking and parameter "wizards" that allow the visual selection of some key parameters. Many standard image formats are transparently supported for input/output from all programs. Additionally, results have been standardized, facilitating the interoperation between different Xmipp programs. Finally, as a result of a large code refactoring, the underlying C++ libraries are better suited for future developments and all code has been optimized. Xmipp is an open-source package that is freely available for download from: http://xmipp.cnb.csic.es.

1. Introduction

From the beginning, image processing and software development have been fundamental parts of electron microscopy (EM) studies of macromolecular structures. In 1996, the Journal of Structural Biology dedicated a special issue to software tools in the field (Carragher and Smith, 1996). Some of the papers published in that issue are among the most cited works of the journal (Smith and Carragher, 2008). In this issue, several software packages used by the EM community were reviewed, such as SPIDER (Frank et al., 1996), IMAGIC (van Heel et al., 1996), MRC (Crowther et al., 1996), EM (Hegerl, 1996), PFT (Baker and Cheng, 1996), Suprim (Schroeter and Bretaudiere, 1996), MDPP (Smith and Gottesman, 1996) and Xmipp (X-Windows-based microscopy image processing package) (Marabini et al., 1996). Most of these packages are still in use today, and have been improved over the years. Some new packages have also been developed during the last decades, including EMAN (Ludtke et al., 1999; Tang et al., 2007), IPLT (Philippsen et al., 2003) and Bsoft (Heymann and Belnap, 2007).

Initially, Xmipp was a set of individual programs written in ANSI-C that included some visualization tools based on the X11 graphics library. It was designed originally for single-particles analysis, but it also included some tools for working with 2D crystals. Simplicity and portability were the main features of the package. While relatively small, Xmipp provided a broad variety of methods for classification, ranging from neural networks (Marabini and Carazo, 1994) to fuzzy multivariate statistical analysis (Carazo et al., 1990). Several reconstruction algorithms, such as weighted back-projection (WBP) and arithmetic methods (ART with blobs (Marabini et al., 1998)) were implemented as part of the package. The file formats used were compatible with SPIDER, facilitating interaction with this well known package. Xmipp was conceived as a development framework that could easily incorporate new methodological advances developed by other groups. Except for some tasks involving a graphical interface, the processing at that time was originally performed by running individual programs for each task and then manually connecting the results.

In 2004, the second major release of Xmipp (Xmipp 2.0) was described by Sorzano et al. (2004). The package was completely rewritten in C++, and included improved data structures and functions, as well as new algorithms and methods. The main processing workflow (from image acquisition to 3D reconstruction) still required the execution of individual programs. However, the use of

* Corresponding author at: Biocomputing Unit, National Center for Biotechnology (CSIC), c/Darwin, 3, Campus Universidad Autónoma, 28049 Cantoblanco, Madrid, Spain. Fax: +34 913720112.
E-mail address: jmdelarosa@cnb.csic.es (J.M. de la Rosa-Trevín).

1047-8477/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jsb.2013.09.015

Qt, a higher level graphics library (Eng, 1996), improved visualization tools such as particle picking or image rendering. Additionally, the computational overhead needed to run many time-consuming algorithms was reduced through parallelization with Message Passing Interface (MPI) library.

Another milestone in Xmipp development was the release of version 2.4, which was distinguished mainly by the introduction of a set of computational protocols (Scheres et al., 2008) and several new methods, such as the ability to classify based on maximum likelihood (Scheres et al., 2005, 2007b) and to estimate the contrast transfer function (CTF) (Velázquez-Muriel et al., 2003). The protocols, which summarized the expertise of the Xmipp community, were implemented as Python scripts containing input parameters and code needed for program execution. Another interesting and useful feature of the protocols was the ability to automate the creation of a graphical user interface (GUI) for launching batch processes.

Although Xmipp 2.4 was a major advance, Xmipp still contained several limitations that would require a deep reorganization of the code to resolve them. We began to address these limitations 2 years ago when we began work on a new version of Xmipp. The culmination of that work was the release of Xmipp 3.0. The primary improvements in this version are:

- *A Project management tool* was created to improve the data processing workflow by offering traceability and reproducibility through the management of protocol runs and individual processing steps stored in a project database.
- *Graphical interfaces* are now more powerful and flexible. Project and protocol GUIs facilitate the monitoring and launching of tasks. Some "wizards" have been included for the selection of important parameters. Data visualization and particle picking have been significantly improved.
- *Image formats support* has been extended to cover most of the formats used in the field, such as: spider, mrc, imagic, tiff, jpeg, dm3, ser, spe, em and pif. Internal implementation of the data model is independent of the image format used.
- *Standardization of results* allows for better interaction between different Xmipp programs as well as communication with external applications. A metadata structure was implemented to handle different types of information using a SQL engine and input/output from/to text files.

2. Project management

In Xmipp 2.4 there was no clear concept of a project. While there was a folder with a set of Python scripts (protocols) for each type of execution, there was no formal relationship among the scripts. In Xmipp 3.0 we have organized the processing workflow into projects (as other packages in the field do), composed of protocol runs. A run is an execution of a protocol, with specific values as input parameters. All workflow information is stored in a project database. The project database is contained in a single binary file thanks to the use of an embedded SQL library that does not require any server setup. This database only contains information about protocol executions and its size does not increase significantly as the project gets larger, since images are not part of the database.

Previously, the execution was protocol-oriented: the user launched Python scripts to perform the required operations. Communication between different protocols was achieved manually, which required that the user know what output files were needed as input for the next protocol. There was also no way to track workflows (that is, which protocols where executed and in which order). In Xmipp 3.0, protocols are still implemented in Python, but the individual processing steps (e.g., functions and program execution) are saved in a database. This approach allows the user to easily monitor the progress of a run, restart it from a specific point and validate that the expected results were obtained; if expected results were not obtained, the entire execution can be labeled as "failed". To save computing time, if a protocol is restarted with different parameters, only those steps affected by the new parameters will be executed.

Moreover, results have been standardized to allow some input parameters to refer to the output of another protocol; in this way, the user does not need to know what files are required from one protocol to another. Additionally, the protocol results are better defined, facilitating the intercommunication between protocols. For example, all protocols that produce 2D classes should generate two metadata files (`classes.xmd` and `images.xmd`) that contain the assignment parameters of each image to each class. A clear advantage of this approach is that other protocols that need a classification result as input can use these files without knowledge of which protocol produced them. As long as new classification protocols produce files with the specified structure, they can be integrated into the general workflow with no other changes.

Projects can be easily ported to other computers by simply copying the project folder. After a project is copied into a new computer, all unfinished runs can be restarted. This can be particularly useful for running interactive protocols on your local computer and then moving the project to a computing cluster for more computationally intensive protocols, such as classification or reconstruction. The ability to easily move projects also facilitates sharing with collaborators. Xmipp is the only additional requirement for running a project on a target computer.

2.1. General processing workflow

While the general processing workflow of Xmipp 3.0 is very similar to the one implemented in Xmipp 2.4 and described in detail in Scheres et al. (2008), the processing has been improved with knowledge gained during the last several years. The Xmipp 3.0 workflow starts by importing the micrographs, together with some information about imaging conditions (see Fig. 1). After import, a screening is performed to check that micrographs do not exhibit astigmatism or drift. Micrographs can also be downsampled to accelerate further calculations.

Particle coordinates are picked from micrographs and later used for particle extraction. Particles selection should be done first in manual-supervised mode and then completely automated. Some operations can be applied during extraction, such as filtering and contrast inversion. After extraction, particle images are stored in a gallery. Particles are sorted according to a *Z*-score (Scheres, 2010; Sorzano et al., 2013) to identify possible outliers such as wrongly picked particles. The user may also directly import a set of particle images and skip all steps preceding particles extraction.

The extracted images can be used as input for 2D classification algorithms to detect possible heterogeneities from contamination or conformational changes. Images can be discarded and not considered in subsequent steps. If micrographs were taken as tilt-pairs, an initial volume can be generated using the random conical tilt method (RCT) (Radermacher et al., 1987). The initial volume (obtained by RCT or provided by the user) can be refined by projection matching. Finally, 3D heterogeneity can be handled by ML3D, MLF3D (Scheres et al., 2007a,b) or a multireference projection matching approach.

The main differences between this workflow and the one presented in Scheres et al. (2008) are:

- After CTF computation, a GUI helps users to decide which micrographs should be kept. Several criteria are provided for

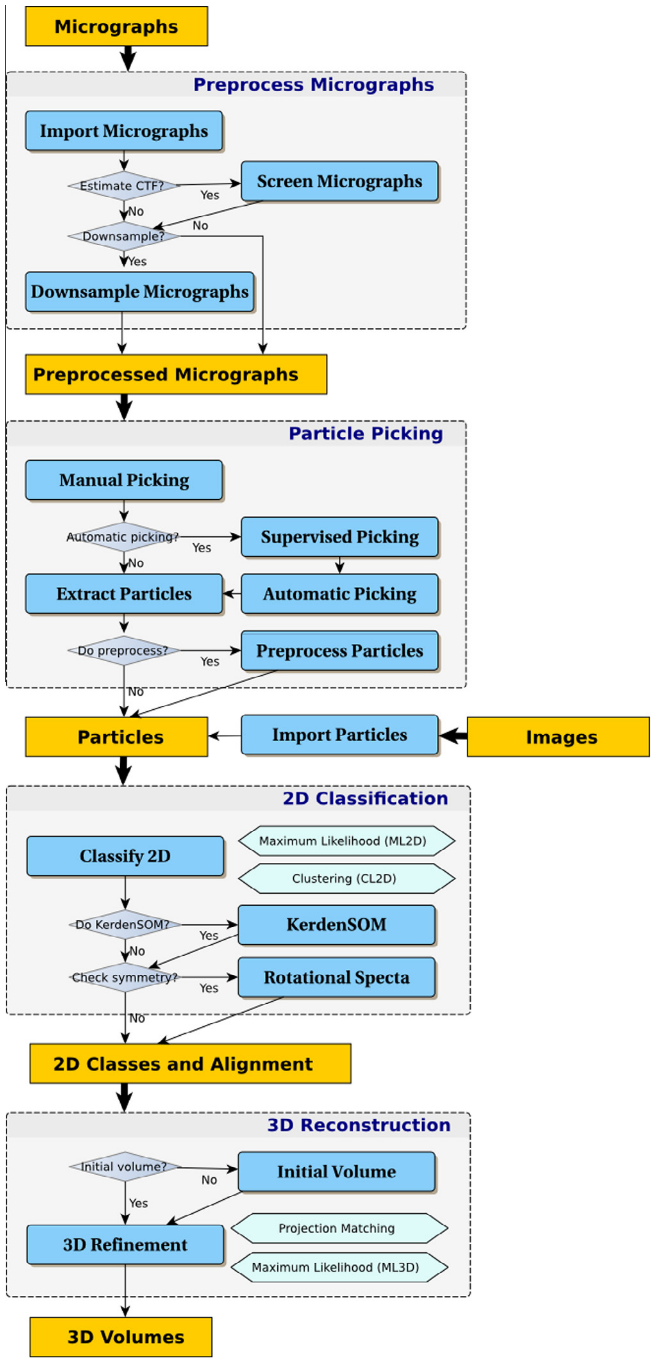


Fig. 1. A generalized Xmipp 3.0 processing workflow. The protocols developed may be divided into (1) micrograph acquisition and preprocessing, (2) particle picking and extraction, (3) 2D alignment and classification and (4) 3D alignment and reconstruction. Input and output data are shown in yellow, while operations are in blue.

micrograph selection such as: astigmatism, resolution, fitting to the CTF model, etc.

- Data maybe be processed at different resolution in different steps of the workflow.
- Automatic particle picking is available in the main workflow.
- After particle extraction, users can discard bad images using a GUI based on similarity criteria.
- A new 2D classification algorithm (CL2D) has been incorporated.
- A new 3D classification algorithm has been added (multireference projection matching).

2.2. Project GUI

The main project GUI was designed to easily visualize project information and simplify workflow management (see Fig. 2A). It is divided into three main panels: (1) the left panel, which contains buttons to open each group of protocols, (2) the right-top panel labeled “History”, which displays different runs and their name, status and modification time, and (3) the right-bottom panel labeled “Details”, which contains summary information for the selected run. There are two buttons between panels 2 and 3: `Analyze Results` and `Output Files`. The first launches some visualization tools to inspect results of the selected protocol run. The second opens the three standard logs files (out, error and log) associated with the run. These files are useful for detailed monitoring of the underlying task executions because they contain the command line program calls and their output. In short, while the intent of the project GUI is to hide all non-essential information from the user, we also provide direct access to this information. Moreover, all processing tasks started from the GUI can also be launched from command line.

The list of protocol runs displayed in the “History” panel can also be managed as an interactive graph (see Fig. 3). In this view the user can access the same functionality as in the list view (i.e., create, duplicate, edit and delete runs) but with a better visual representation of the project workflow. Boxes represent protocol runs (their colors indicate different run states) and the connecting lines indicate dependencies between the runs.

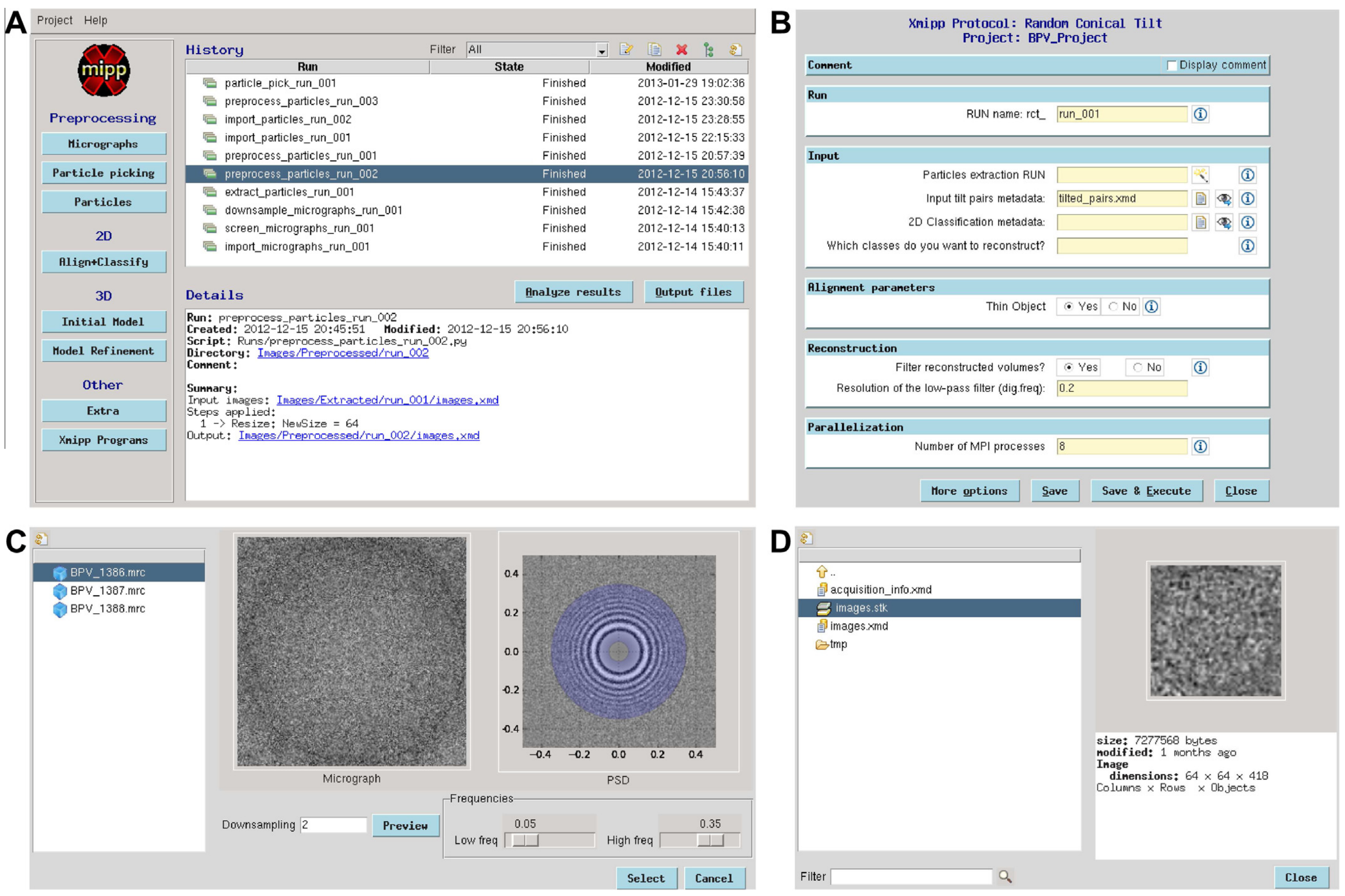
Each protocol contains a header with input parameters that is parsed for automatic GUI generation (see Fig. 2B). All protocol GUIs contain a comment box, where annotations about the run can be made. The parallelization section is also common to all protocols and can be used to set the number of threads or MPI processes. In this section, the user can also choose to launch the process in a queue system or execute the process immediately. Different queue systems can be configured easily in Xmipp 3.0. This need only be configured once and, after that, users will no longer need to create submission scripts.

An important tool related to project and protocols is a specialized file browser (see Fig. 2D), that provides more functionality than a standard file browser, such as recognition of EM specific file formats, object previews and integration with external tools. The specialized browser has been used as a base for the development of other GUIs called “wizards”, which are integrated in protocol forms. Some critical parameters can produce poor results if wrongly selected, however, wasting user and computation time. The basic idea behind the wizards is to provide an additional GUI to assists the user with critical parameter selection. For example, the wizard shown in Fig. 2C helps the user select high and low frequencies for CTF estimation.

The *Getting Started* section of Xmipp’s website (http://xmipp.cnb.csic.es/twiki/bin/view/Xmipp/GettingStarted) describes some of the graphical interfaces in more detail. Video tutorials demonstrating how to use the different wizards are provided at http://xmipp.cnb.csic.es/twiki/bin/view/Xmipp/Tutorials#Video_Demos.

2.3. Visualization tools

A key program used for data visualization is `xmipp_show`, which displays metadata files (in a table view), individual images, volumes or stacks. In Fig. 4A a volume is displayed slice by slice in the *Z*-direction. This program is highly configurable and, users can choose what columns to display, preview an image in a column or perform many other operations on metadata, such as adding, removing or searching. It is also possible to open external



History Filter All

Run	State	Modified
particle_pick_run_001	Finished	2013-01-29 19:02:36
preprocess_particles_run_003	Finished	2012-12-15 23:30:58
import_particles_run_002	Finished	2012-12-15 23:28:55
import_particles_run_001	Finished	2012-12-15 22:15:33
preprocess_particles_run_001	Finished	2012-12-15 20:57:39
preprocess_particles_run_002	Finished	2012-12-15 20:56:10
extract_particles_run_001	Finished	2012-12-14 15:43:37
downsample_micrographs_run_001	Finished	2012-12-14 15:42:38
screen_micrographs_run_001	Finished	2012-12-14 15:40:13
import_micrographs_run_001	Finished	2012-12-14 15:40:11

Details

```
Run: preprocess_particles_run_002
Created: 2012-12-15 20:45:51   Modified: 2012-12-15 20:56:10
Script: Runs/preprocess_particles_run_002.py
Directory: Images/Preprocessed/run_002
Comment:

Summary:
Input images: Images/Extracted/run_001/images.xmd
Steps applied:
  1 -> Resize: NewSize = 64
Output: Images/Preprocessed/run_002/images.xmd
```

Xmipp Protocol: Random Conical Tilt
Project: BPV_Project

Comment ☐ Display comment

Run
RUN name: rct_ run_001

Input
Particles extraction RUN
Input tilt pairs metadata: tilted_pairs.xmd
2D Classification metadata:
Which classes do you want to reconstruct?

Alignment parameters
Thin Object [✓ Yes] No

Reconstruction
Filter reconstructed volumes? [✓ Yes] No
Resolution of the low-pass filter (dig.freq): 0.2

Parallelization
Number of MPI processes 8

Downsampling 2
Low freq 0.05
High freq 0.35

```
size: 7277568 bytes
modified: 1 months ago
Image
  dimensions: 64 x 64 x 418
Columns x Rows  x Objects
```

Fig. 2. Xmipp project graphical interfaces. (A) The main project GUI; left panel contains buttons to open protocol groups as well as parameter form; right-top panel contains a list of all runs, their state and modification date; right-bottom panel contains a brief summary of the selected run. (B) Example of an automatically generated protocol GUI. (C) Helper wizard to select low and high frequencies for CTF estimation. (D) Specialized file browser recognizing EM file formats and displaying an object preview.

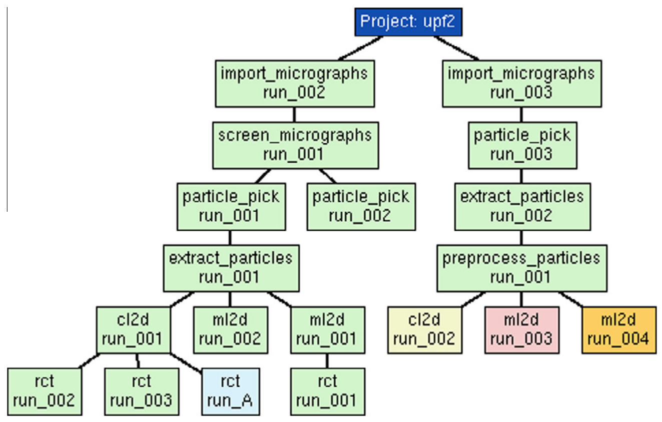


Fig. 3. Detailed view of the interactive workflow graph. This alternative representation can be selected from the main project GUI. Each box represents a protocol run. Colors are used to differentiate runs status in the following way: green for successfully finished, red for failed, light blue for saved, light yellow for aborted and yellow for currently running. The lines indicate the dependencies between runs. In this example, the user imported two sets of micrographs and used different parameters for some steps. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

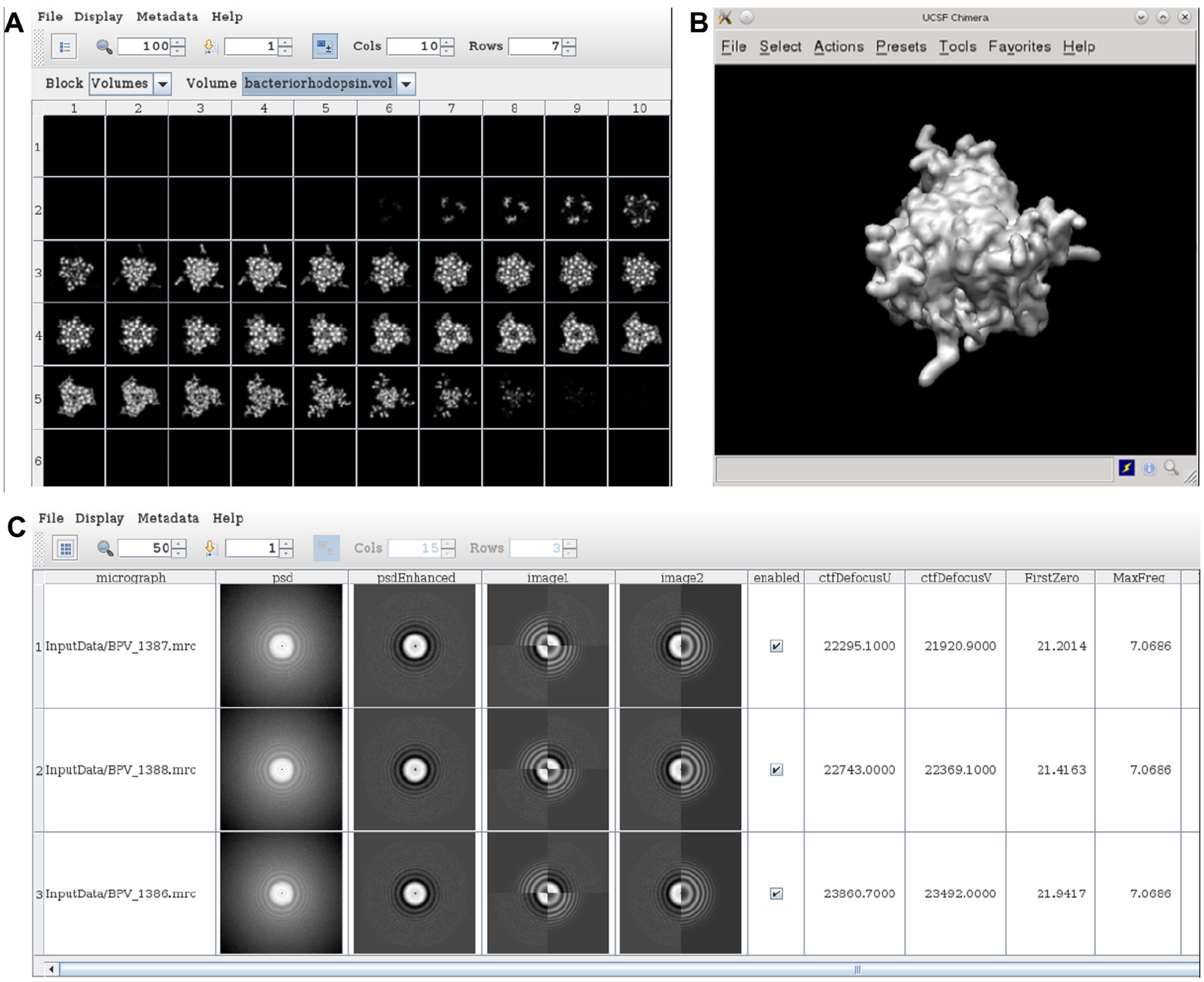
visualization tools such as ImageJ (Abramoff et al., 2004) or Chimera (Pettersen et al., 2004). For example, Fig. 4B displays the same volume as Fig. 4A but opened in Chimera through `xmipp_show`. In Fig. 4C a customized view of a metadata file (after micrograph screening) is displayed. In this example, some columns are displayed with a preview and some headers have been renamed.

Some tasks required the development of very specific GUIs to enhance the final user experience. For example, the particle picking interface was developed from scratch, it is now better organized and easier to use. The general interface for particle picking is displayed in Fig. 5. Fig. 5A displays a selected micrograph with particle markers, while Fig. 5B displays the list of micrographs in the main control window. Several filters can be applied from the main menu to improve micrograph visualization and to help with particle selection (see Fig. 5C).

2.4. Programs

As in previous versions, Xmipp 3.0 contains several command-line utility programs, that are designed to perform specific tasks. We have carefully rewritten all of these programs using a common library that enables sharing of important functionality. There are a number of advantages to this design, such as allowing each program to define its own input parameters, that can be validated automatically and used to generate a GUI.

By default, if a user types a program name without arguments, a usage message is printed in the console. Program `xmipp_apropos` allows for querying the database to search for desired programs by name or keywords related to their function (e.g., Fourier filter, reconstruction, sampling). There are some common parameters for all programs such as `—more`, where advanced options are


A
File Display Metadata Help
100
1
Cols 10 Rows 7
Block Volumes Volume bacteriorhodopsin.vol
B
UCSF Chimera
File Select Actions Presets Tools Favorites Help
C
File Display Metadata Help
50
1
Cols 15 Rows 3

	micrograph	psd	psdEnhanced	image1	image2	enabled	ctfDefocusU	ctfDefocusV	FirstZero	MaxFreq
1	InputData/BPV_1387.mrc					✓	22295.1000	21920.9000	21.2014	7.0686
2	InputData/BPV_1388.mrc					✓	22743.0000	22369.1000	21.4163	7.0686
3	InputData/BPV_1386.mrc					✓	23860.7000	23492.0000	21.9417	7.0686

Fig. 4. Data visualization tools. (A) xmipp_show tool rendering a volume slice by slice in the *Z*-direction. (B) The same volume shown in A opened from `xmipp_show` with Chimera as an external tool. (C) Another configuration of `xmipp_show` for displaying data in table view with some image previews.

described, and –`gui`, which launches the program GUI. Programs have been organized into groups and named so that program's function can be deduced from its name.

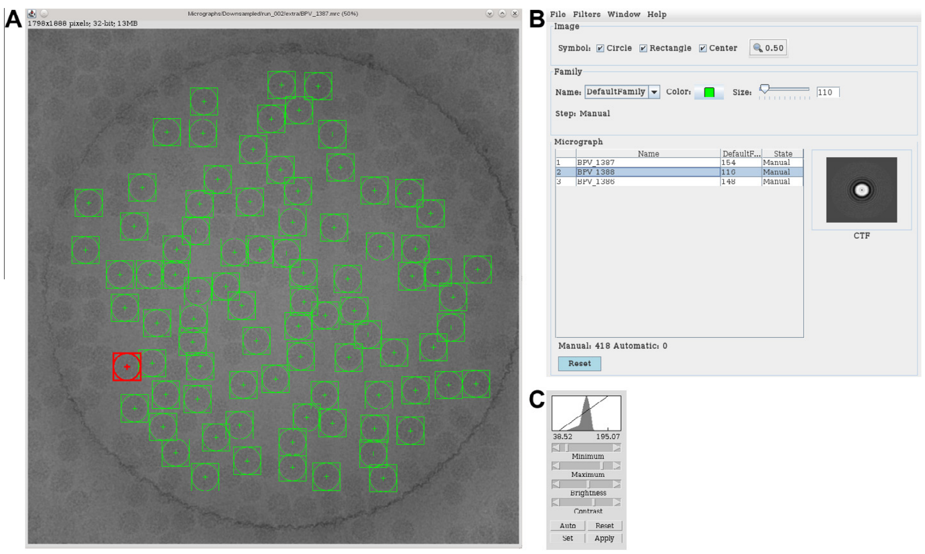
3. Software

From a computer engineering point of view, Xmipp has gone through a major transformation that includes the following: (1) the level of abstraction has increased, allowing for a clear separation between file formats and the data model, (2) the protocols have changed from spawning new processes (Xmipp programs) to using, in many cases, direct calls to C++ routines through a binding layer, (3) the introduction of projects provides traceability and reproducibility, and (4) the implementation of threads and MPI parallelization as a centralized library promotes code reuse and leverages established design patterns.

3.1. Data model

There are two common types of data in EM: binary (micrographs, images, volumes, stacks) and meta-information (coordinates, class assignments, alignment parameters, etc.). In Xmipp 2.4 a tight relationship existed between the data model (how data are represented in memory and handled by programs) and the file format (how data are stored on the hard disk). A consequence of this design was support of only one file format. This approach was problematic for users working with other EM formats, because it required extra work to convert data at the beginning and end of Xmipp processing. Additionally, disk space was wasted because users usually needed several copies of the same image data set in different formats.

To address this problem, we completely redesigned the data model and implemented new input/output (I/O) functions. Binary data is handled through the Image class, which contains a four dimensional array (gray scaled 2D and 3D objects in stacks) and a set of vectors (one per object) of label-value pairs (as a generalized header). This new design made it possible to implement several I/O routines that read data from different formats (see Table 1). Another enhancement is the ability to handle very large files mapped to disk instead of loading them into memory. When reading images, three options are available: (1) load them into memory, (2) map them to disk and, (3) map them to disk if they do not fit in physical memory. It is up to the developer to choose which option to use for each specific program.



Micrographs/Downsampled/run_002/extra/BPV_1387.mrc (50%)
1798x1888 pixels; 32-bit; 13MB

Symbol: ☑ Circle ☑ Rectangle ☑ Center 0.50

Name: DefaultFamily Color: 🟩 Size: 110

Step: Manual

	Name	DefaultF...	State
1	BPV_1387	154	Manual
2	BPV_1388	116	Manual
3	BPV_1386	148	Manual

Manual: 418 Automatic: 0

38.52 195.07

Fig. 5. Particle picker graphical interface. (A) Micrograph display with marks for all selected particles. (B) Management window, containing a list of all micrographs and some other options. (C) Example of a contrast adjustment that can be made to improve micrograph visualization.

Table 1
Image formats supported by Xmipp 3.0.

Extension	Filetype	Read/write	Image	Volume	Stack	Data types
raw#	Raw files without header info	R	x	x		All
spi,xmp,vol,stk	Spider	R/W	x	x	x	float, cfloat
mrc,map,mrcs	Medical Research Council	R/W	x	x	x	uint8, int16, float, cfloat
hed/img	Imagic	R/W	x		x	uint8, int16, float
inf/raw	Xmipp raw format	R/W	x	x		(u)int8, (u)int16, float
tiff	TIFF	R/W	x		x	uint8, uint16, uint32, float
jpg	Jpeg	R/W	x			uint8
dm3	Digital Micrograph	R	x		x	int8, (u)int16, (u)int32, float, double
ser	TIA (Tecnai imaging and analysis)	R	x		x	(u)int8, (u)int16, (u)int32, float, double, cfloat, cdouble
spe	Princeton Instrument CCD camera	R	x			uint16
em	EM Software Package	R	x	x		int8, int16, int32, float, cfloat, cdouble
pif	Portable Image Format for EM	R	x	x	x	int8, int16, int32, float, cfloat

Important changes have also been made with respect to meta-information. Conceptually, Xmipp 2.4 metadata were tables of numbers (implemented as arrays of doubles) that were stored in disk as SPIDER document files. This approach was very rigid since it only allowed numeric values while the meaning of the values strongly relied on the order of columns.

On the other hand, Xmipp 3.0 metadata are tables in which different data types (number, string, or vector) may be assigned to each column. In this way, all columns are defined and used consistently by all Xmipp programs. Furthermore, the MetaData class provides functions to operate with big datasets while hiding to developers the internal implementation. Indeed, the implementation uses a SQL embedded library (Sqlite3) that allows to perform faster searches and complex queries. One advantage of this embedded library is that there is no need to install or maintain a SQL server. From the internal representation we have implemented functions to read/write metadata as STAR text files, Sqlite3 binary files or XML files. The STAR format is used by default for program outputs within the project workflow. This decision was based on two main reasons: first, users normally operate with text files and view/edit results with text editors instead of dealing with databases; second, a text format is a simple way to achieve integration with external programs.

Examples of how to use the Image and MetaData classes are available at: http://xmipp.cnb.csic.es/twiki/bin/view/Xmipp/Image and http://xmipp.cnb.csic.es/twiki/bin/view/Xmipp/MetaData, respectively.

3.2. Project implementation

Project information is maintained in a simple Sqlite3 database composed of a small number of tables. The most important tables are *runs* and *steps*, which store the project's protocol runs and the individual steps of each run, respectively. This database is stored in a Sqlite3 file inside the project folder and does not contains any result files from program executions.

Before a protocol can run, it first defines a list of steps (Python functions that are usually wrapping program calls) that are stored in the database. These steps are compared with previous execution (if it exists), to determine at which step the protocol should restart (due to the use of different input parameters or result files), if necessary. This approach saves computing time by skipping all unchanged steps and executing needed steps only. Because of this level of granularity, the progress of each protocol run can be easily monitored. The individual steps can also be defined as "parallel" and executed concurrently, allowing for parallelization both at the program and protocol level.

The input of some protocols may be another protocol run, or individual files produced previously. Because we keep track of all these dependencies, the project workflow graph can be dynamically generated. This graph allows users to check on performed operations and the relationships between them (see Fig. 3).

3.3. Python and Java bindings

We have added bindings for Python and Java, which are widely used languages in scientific computing. These bindings allow access to a large number of functions, classes and programs from these two languages without compromising performance. Bindings were designed as light interface layers that contain only the code needed to perform the communication between both sides. This approach avoids code duplication, which can be more difficult to maintain. We also maintained similar syntax whenever possible.

These layers allows fast prototyping and access to Xmipp 3.0 core functionality for code written in Python or Java. We used these bindings in the development of some components of the Xmipp 3.0 package. For example, the protocols and main project GUI were implemented in Python and make extensive use of Python bindings. Other visualization tools were written in Java with the ImageJ library and rely on the Java binding for image manipulation.

3.4. Parallelization

Image processing operations in Single Particle Analysis (EM-SPA) involves many tens or hundreds of thousands of images and requires a large amount of computing resources. Therefore, parallel computing is often a necessary tool in EM-SPA for reasons of computational overhead. In general, parallel programs splits the work to be done by data or by instructions. A simple parallelization approach (and most likely the most common in EM-SPA) is to split the data across many parallel processes. In this case, some operations are performed simultaneously on several subsets of images.

Parallelization can be performed at two levels: (1) using threads, which are lightweight processes that share memory, and (2) by running different processes, possibly on different computers, that communicate over a network. We have developed several tools to facilitate the use of both threads and MPI processes. For example, the `ParallelTaskDistributor` class handles dynamic distribution of tasks. This class has no knowledge of the specific task performed (processing an image, operating in a volume slice or any other operation). So, in many contexts, where a task distribution process is needed, `ParallelTaskDistributor` can be used to avoid repeating the logic of distributing tasks. On the other hand, creation of POSIX threads usually requires writing similar lines of setup code. The `ThreadManager` class make the use of threads easy; it also has the ability to reuse threads for computing several functions without the need to destroy and create them every time.

A detailed explanation of the parallel API implementation and philosophy is available at http://xmipp.cnb.csic.es/twiki/bin/view/Xmipp/ParallelProgramming.

3.5. New methods

Xmipp is the framework where our new methods were tested and are now available to users. In the following section we discuss some of the new algorithms included in the package since the publication of Scheres et al. (2008).

In Sorzano et al. (2009b) it was proposed an automatic particle picking algorithm based on some particle features and supervised learning. This approach was improved in Abrishami et al. (2013) with the use of two support vector machine (SVM) classifiers in combination with several rotational invariant features and statistical properties for particle selection. The algorithm is now faster and provides better results than in the previous implementation.

Another important addition is a 2D classification method based on hierarchical clustering (Sorzano et al., 2010). This algorithm uses both correntropy and correlation as similarity measures; it also includes criteria for the definition of the clusters appropriate for classes with small differences in their signal-to-noise ratio (SNR).

Moreover, a fast and automated algorithm for CTF estimation developed by Vargas et al. (2013) has been implemented in Xmipp 3.0. This method is based on the Spiral Phase Transform and it does not require an initial defocus or amplitude contrast estimate. This new approach is approximately an order of magnitude faster than existing solutions (Mindell and Grigorieff, 2003; Sorzano et al., 2007) in cases where a large defocus searching range of $[0.5, 10]$ μm is used. At the same time, it provides defocus and astigmatism estimates at accuracies comparable to well established methods (Mindell and Grigorieff, 2003; Sorzano et al., 2007).

Despite Xmipp focus on SPA, it also includes tools that are relevant to other EM applications. Some Xmipp algorithms such as ART, WBP or align_tilt_series (Sorzano et al., 2009a) have been ported to Java in TomoJ (Messaoudi et al., 2007) for EM tomography. Xmipp also includes a maximum likelihood algorithm for subtomogram averaging described in Scheres et al. (2009). For Soft X-ray tomography we introduced an accurate image formation model (Oton et al., 2012) for the generation of X-ray pseudo-projections. This new model provides a better understanding of the X-ray projection process and will, in the near future, be the base of development of specialized reconstruction algorithms.

3.6. Installation

Xmipp 3.0 is distributed as free software and can be downloaded from: http://xmipp.cnb.csic.es/twiki/bin/view/Xmipp/New-Download. While a GUI has been created for Xmipp 3.0 compilation and installation, it can also be compiled and installed from the command line. In this way, the package may be used in environments that lack a graphical interface.

The software can be installed from binaries or compiled from source code. It was tested with several Linux distributions and the MacOS. A complete list of supported platforms and installation instructions can be found here: http://xmipp.cnb.csic.es/twiki/bin/view/Xmipp/HowToInstall.

Xmipp 3.0 smoothly integrates with cluster environments through the configuration fileXMIPP_DIR/protocols/config_launch.py. Once this file is setup, users do not need to directly interact with system queues because jobs can be launched from the GUI. At present, Xmipp can only launch jobs in the same system where the project is stored. We are working to allow remote execution in future releases. Instructions on how to configure this file are available at http://xmipp.cnb.csic.es/twiki/bin/view/Xmipp/HowToInstall#Make_a_job_submission_template_s.

4. Conclusions

The development of Xmipp 3.0 was based on over 5 years of accumulated user experience since the previous major release. During this time, Xmipp users have solved and deposited more than one hundred new structures in the Electron Microscopy Data Bank (Lawson et al., 2011).

This new release constitutes a complete package re-engineering, that enhances the user experience while providing detailed tracking of the image processing workflow. Better interfaces and graphical tools enable navigation through the different steps of the process and provide direct control of key parameters. Users no longer need to be concerned with different file formats, because Xmipp 3.0 transparently handles many of the formats used in the

EM field. To improve information management, the concept of a "project" was developed and given a central role in the natural management of all processing associated with a given data set. The complete history of all actions performed can be tracked and visualized in an interactive graph.

The last decade has witnessed the development of key approaches for handling large data sets in the quest for higher resolution reconstructions. Most of these new methods represent important computing challenges. As a consequence, it is necessary to pay close attention to the management of computational resources. To this end, Xmipp 3.0 provides detailed job monitoring and permits the use of batch queues. Furthermore, the libraries contain several tools for quick development of parallel algorithms, using both multithreading and multiprocessors. In short, the package is now better prepared to manage complex workflows and computational resources. It is also a more compact and robust software platform, capable of adapting to future developments of increasingly demanding approaches.

Acknowledgments

The authors would like to acknowledge all former Xmipp developers who made contributions to the package during the past several years. We also acknowledge financial support from the Comunidad de Madrid through grant CAM (S2010/BMD- 2305), the NSF through Grant 1114901; the Spanish Ministry of Economy and Competitiveness through Grants AIC-A-2011-0638 and BIO2010-16566. "Juan de la Cierva" postdoctoral grants with references JCI-2011-10185 and JCI-2010- 07594, and predoctoral grants with references BES-2011-044096 and JAEPre_09_01717. C.O.S. Sorzano is a recipient of a Ramón y Cajal fellowship. This work was partially funded by Instruct, which is part of the European Strategy Forum on Research Infrastructures (ESFRI) and supported by national member subscriptions.

References

Abramoff, M.D., Magelhaes, P.J., Ram, S.J., 2004. Image processing with ImageJ. Biophotonics Int. 11, 36–42.
Abrishami, V., Zaldívar-Peraza, A., de la Rosa-Trevín, J.M., Vargas, J., Otón, J., et al., 2013. A pattern matching approach to the automatic selection of particles from low-contrast electron micrographs. Bioinformatics 29, 2460–2468.
Baker, T.S., Cheng, R.H., 1996. A model-based approach for determining orientations of biological macromolecules imaged by cryoelectron microscopy. J. Struct. Biol. 116, 120–130.
Carazo, J.M., Rivera, F.F., Zapata, E.L., Radermacher, M., Frank, J., 1990. Fuzzy sets-based classification of electron microscopy images of biological macromolecules with an application to ribosomal particles. J. Microsc. 157, 187–203.
Carragher, B., Smith, P.R., 1996. Advances in computational image processing for microscopy. J. Struct. Biol. 116, 2–8.
Crowther, R.A., Henderson, R., Smith, J.M., 1996. MRC image processing programs. J. Struct. Biol. 116, 9–16.
Eng, E., 1996. Qt GUI Toolkit: porting graphics to multiple platforms using a GUI toolkit. Linux J. 1996. Article No. 2. URL: http://dl.acm.org/citation.cfm?id=326464&CFID=250746326&CFTOKEN=69585403.
Frank, J., Radermacher, M., Penczek, P., Zhu, J., Li, Y., et al., 1996. SPIDER and WEB: processing and visualization of images in 3D electron microscopy and related fields. J. Struct. Biol. 116, 190–199.
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, 17–24.
Hegerl, R., 1996. The EM program package: a platform for image processing in biological electron microscopy. J. Struct. Biol. 116, 30–34.
Heymann, B., Belnap, D., 2007. Bsoft: image processing and molecular modeling for electron microscopy. J. Struct. Biol. 157, 3–18.
Lawson, C.L., Baker, M.L., Best, C., Bi, C., Dougherty, M., et al., 2011. Emdatabank.org: unified data resource for cryoem. Nucleic Acids Res. 39, D456–D464.
Ludtke, S.J., Baldwin, P.R., Chiu, W., 1999. EMAN: semiautomated software for high-resolution single-particle reconstructions. J. Struct. Biol. 128, 82–97.
Marabini, R., Carazo, J.M., 1994. Pattern recognition and classification of images of biological macromolecules using artificial neural networks. Biophys. J. 66, 1804–1814.
Marabini, R., Herman, G.T., Carazo, J.M., 1998. 3D reconstruction in electron microscopy using ART with smooth spherically symmetric volume elements (blobs). Ultramicroscopy 72, 53–65.
Marabini, R., Masegosa, I.M., San Martín, M.C., Marco, S., Fernández, J.J., et al., 1996. Xmipp: an image processing package for electron microscopy. J. Struct. Biol. 116, 237–240.
Messaoudi, C., Boudier, T., Sorzano, C.O.S., Marco, S., 2007. Tomoj: software for multiple-axis tomography. BMC Bioinf. 8, 288.
Mindell, J.A., Grigorieff, N., 2003. Accurate determination of local defocus and specimen tilt in electron microscopy. J. Struct. Biol. 142, 334–347.
Oton, J., Sorzano, C.O.S., Pereiro, E., Cuenca-Alba, J., Navarro, R., et al., 2012. Image formation in cellular x-ray microscopy. J. Struct. Biol. 178, 29–37.
Pettersen, E.F., Goddard, T.D., Huang, C.C., Couch, G.S., Greenblatt, D.M., et al., 2004. UCSF chimera – a visualization system for exploratory research and analysis. J. Comput. Chem. 25, 1605–1612.
Philippsen, A., Schenk, A.D., Stahlberg, H., Engel, A., 2003. IPLT – image processing library and toolkit for the electron microscopy community. J. Struct. Biol. 144, 4–12.
Radermacher, M., Wagenknecht, T., Verschoor, A., Frank, J., 1987. Three-Dimensional reconstruction from a single-exposure, random conical tilt series applied to the 50s ribosomal subunit of *Escherichia coli*. J. Microsc. 146, 113–136.
Scheres, S., Núñez-Ramírez, R., Gómez-Llorente, Y., San Martín, C., Eggermont, P.P.B., et al., 2007a. Modeling experimental image formation for likelihood-based classification of electron microscopy data. Structure 15, 1167–1177.
Scheres, S.H., Melero, R., Valle, M., Carazo, J.M., 2009. Averaging of electron subtomograms and random conical tilt reconstructions through likelihood optimization. Structure 17, 1563–1572.
Scheres, S.H.W., 2010. Classification of structural heterogeneity by maximum-likelihood methods. Methods Enzymol. 482, 295–320.
Scheres, S.H.W., Gao, H., Valle, M., Herman, G.T., Eggermont, P.P.B., et al., 2007b. Disentangling conformational states of macromolecules in 3D-EM through likelihood optimization. Nat. Methods 4, 27–29.
Scheres, S.H.W., Núñez-Ramírez, R., Sorzano, C.O.S., Carazo, J.M., Marabini, R., 2008. Image processing for electron microscopy single-particle analysis using Xmipp. Nat. Protoc. 3, 977–990.
Scheres, S.H.W., Valle, M., Núñez, R., Sorzano, C.O.S., Marabini, R., et al., 2005. Maximum-likelihood multi-reference refinement for electron microscopy images. J. Mol. Biol. 348, 139–149.
Schroeter, J.P., Bretaudiere, J.P., 1996. Suprim: easily modified image processing software. J. Struct. Biol. 116, 131–137.
Smith, P.R., Gottesman, S.M., 1996. The micrograph data processing program. J. Struct. Biol. 116, 35–40, URL: http://dx.doi.org/10.1006/jsbi.1996.0007, doi:10.1006/jsbi.1996.0007.
Smith, R., Carragher, B., 2008. Software tools for molecular microscopy. J. Struct. Biol. 163, 224–228.
Sorzano, C.O.S., Bilbao-Castro, J.R., Shkolnisky, Y., Alcorlo, M., Melero, R., et al., 2010. A clustering approach to multireference alignment of single-particle projections in electron microscopy. J. Struct. Biol. 171, 197–206.
Sorzano, C.O.S., de la Rosa Trevín, J.M., Otón, J., Vega, J.J., Cuenca, J., et al., 2013. Semiautomatic, high-throughput, high-resolution protocol for three-dimensional reconstruction of single particles in electron microscopy. Methods Mol. Biol. 950, 171–193.
Sorzano, C.O.S., 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, 249–262.
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.S., Messaoudi, C., Eibauer, M., Bilbao-Castro, J.R., Hegerl, R., Nickell, S., Marco, S., Carazo, J.M., 2009a. Marker-free image registration of electron tomography tilt-series. BMC Bioinf. 10, 124.
Sorzano, C.O.S., Recarte, E., Alcorlo, M., Bilbao-Castro, J.R., San-Martín, C., et al., 2009b. Automatic particle selection from electron micrographs using machine learning techniques. J. Struct. Biol. 167, 252–260.
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, 38–46, URL: http://dx.doi.org/10.1016/j.jsb.2006.05.009.
Vargas, J., Otón, J., Marabini, R., Jonic, S., de la Rosa-Trevín, J.M., et al., 2013. FASTDEF: fast defocus and astigmatism estimation for high-throughput transmission electron microscopy. J. Struct. Biol. 181, 136–148.
Velázquez-Muriel, J.A., Sorzano, C.O.S., Fernández, J.J., Carazo, J.M., 2003. A method for estimating the CTF in electron microscopy based on ARMA models and parameter adjusting. Ultramicroscopy 96, 17–35.