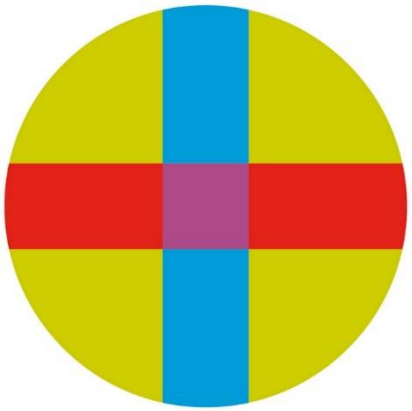
UNIVERSITY CEU - SAN PABLO
POLYTECHNIC SCHOOL
TELECOMMUNICATIONS ENGINEERING DEGREE



BACHELOR THESIS

Integration of tridimensional reconstruction and classification algorithms for single particle analysis

Author: Ana Sanmartin Domenech
Director: Carlos Óscar Sánchez Sorzano

September 2020

UNIVERSIDAD SAN PABLO-CEU

ESCUELA POLITÉCNICA SUPERIOR

División de Ingeniería

Datos del alumno

NOMBRE:

Datos del Trabajo

TÍTULO DEL PROYECTO:

Tribunal calificador

PRESIDENTE:	FDO.:

SECRETARIO:	FDO.:

VOCAL:	FDO.:

Reunido este tribunal el _____/_____/_____, acuerda otorgar al Trabajo Fin de Grado presentado por Don_____la calificación de _____.

ACKNOWLEDGMENTS

A mi familia, por su paciencia y aliento.

A Carlos Óscar por su incondicional paciencia y amabilidad. A los compañeros del Centro Nacional de Biotecnología por su energía y ayuda con cualquier problema.

A mis compañeros por sus consejos y a Alberto por sus figuras.

En especial, a mi padre, por confiar siempre en mí.

ABSTRACT

How macromolecular complexes control and achieve complicated tasks in living cells is a mystery that researchers have been trying to address and answer. Several techniques have been developed and improved in the last century. One of those is Cryogenic electron microscopy (Cryo-EM). Cryo-EM is a process for extraction of 2D projections. These 2D projections will then be processed to obtain the desired 3D reconstruction. Although a wide amount of research has been done and validated, there is yet no reference software framework. This affects analysis of the data at a computational level. Different software packages may have different requirements and procedures, thus, not allowing interoperability among them.

The main goals of this Bachelor's Thesis are: (1) to extract the pipeline used in a Cryo-EM processing application, and (2) integrate this application into a Cryo-EM image processing framework. We expect these goals to be beneficial towards the integration of protocols into a software framework called Scipion. Scipion is an open-source software that can be downloaded from http://scipion.cnb.csic.es/.

Keywords: Electron microscopy; Single particles; Software packages, Image processing; 3D reconstruction

RESUMEN

Cómo los complejos macromoleculares controlan y consiguen realizar tareas complejas es una cuestión que los investigadores han intentado responder. En el último siglo, varias técnicas han sido desarrolladas. Una de ellas es microscopía crioelectrónica. Se trata de un proceso de extracción de proyecciones 2D, las cuales, serán procesadas para obtener una reconstrucción 3D. Gran cantidad de investigación ha sido creada y validada, pero todavía no existe un programa que actúe de marco de referencia. Esta falta, afecta a la hora de análisis de datos, puesto que programas diferentes presentarán diferentes requisitos y procedimientos, por tanto, no posibilitando la interoperabilidad.

Los objetivos de este proyecto son: (1) extraer el flujo de trabajo de una aplicación de procesamiento de imágenes de microscopía crioelectrónica y (2) integrar la aplicación en un posible marco de referencia. Esperamos que estos dos objetivos sean beneficiosos a la hora de integrar nuevos protocolos en Scipion. Scipion es un software de código libre que puede ser descargado en http://scipion.cnb.csic.es/.

INDEX

FIGURE INDEX

TABLE INDEX

1 INTRODUCTION

1.1 Motivation

How macromolecular complexes control and achieve complicated tasks in living cells is a mystery that researchers have been trying to understand during decades. In order to determine their structure, function and interaction, different techniques have been developed and improved in the last century. One remarkable procedure is Cryogenic electron microscopy (Cryo-EM).

In cryo-EM procedure, samples are cooled to cryogenic temperatures. This samples are then secured in an aqueous condition and embedded in a grid-mesh where, 2D projections are obtained. These 2D projections will then be processed to obtain the desired 3D reconstruction. Although a wide amount of research has been done and validated, there is yet no reference software framework. This affects analysis of the data at a computational level. Different software packages have different required parameters, as well as they may not present compatible formats. This Project was created to tackle the need of a framework where Cryo-EM processing tools can be integrated into a program for researchers, scholars and students.

1.2 Objectives

There are two ways to enhance current research on Cryo-electron microscopy (Cryo-EM): improving projection quality by upgrading machinery or trying a wide variety of image processing tasks. Following the second path, these tasks, may unveil deeper, further knowledge of macromolecular structures and spatial disposition.

This Bachelor's Thesis aims to (1) extract the pipeline used in a Cryo-EM processing application, and (2) integrate this application into a Cryo-EM image processing framework. These general goals are translated into the following technical objectives:

1. Test and acquire the workflow of an existing C++ open source application used for Cryo-EM image processing, called CisTEM.
2. Design and create a wrapper method to access all CisTEM methods previously analyzed from Scipion-Xmipp image processing framework.

All processed and stored data in this project comes from the Electron Microscopy Public Image Archive (EMPIAR), which is open for browsing, updating and downloading for building a 3D structure.

1.3 Thesis' structure

Chapter 2 presents the state of the art on cryogenic electron microscopy as well as software to process the 3D reconstruction.

Chapter 3 describes the materials and methods used in this Thesis. It thoroughly details all the steps done. It also presents code and pseudocode needed for the automatization of the process.

Chapter 4 shows the automatization process for the CisTEM application.

Chapter 5 presents the discussion of the results.

Chapter 6 concludes this work and presents possible future research lines.

Finally, a detailed referenced literature is provided at the end of the document and the appendices.

2 State of the Art

2.1 Cryogenic electron microcopy (Cryo – EM)

How does macromolecular complexes achieve complicated tasks in living cells? This is a question that has raised the curiosity of researchers during decades. Through structural biology, researches can study the 3D disposition of molecules, their assembly, function and interaction [6]. In order to determine such complexes, some techniques have arisen, such as X-ray crystallography, Nuclear Magnetic Resonance and Cryo-Electron Microscopy, although in this bachelor thesis, only one process (Cryo-EM) will be explained thoroughly.

During the first half of the 20th century, the analysis of electron microscope (EM) samples started to take off thanks to the creation of the EM by the Ruska brothers [53]. The first problem EM encountered was the degradation of the structural integrity of the sample since macromolecules are susceptible to radiation (chemical bonds may break). Usage of dehydrated samples introduced artifacts, specifically negative staining [25] which added heavy-metal salt to replace water. Although not having too good results, DeRosier and Klug [20] made use of negative staining and presented a method to calculate 3D structures from 2D projections in different directions.

The Cryo-EM method known nowadays was developed thanks to Dubochet and earlier attempts [36, 70, 26], it is a method to preserve the model by freezing them in a thin layer of non-crystalline vitreous ice and its name is given since the vitreous ice is preserved at liquid nitrogen temperature.

The approach to make a 3D reconstruction from multiple 2D projections of macromolecular particles is called single-particle analysis, and the main problem is to determine the orientation in which the 2D image has been extracted. This happens because when extracting the 3D reconstruction of a specimen the image is the superposition of the 3D dimension object in a single 2D image and in an unknown orientation. Therefore, it is easier remodeling for helical arrays and symmetrical structures.

The first reconstruction was done on the hepatitis B virus core (1997) [19], later, amino acid backbone could be traced for epsilon-15 virus, polyhedrosis virus and rotavirus inner capsid partible (2009) [15], aqua reovirus and adenovirus were later reconstructed (2010) [46]. For asymmetric structures, the reconstruction started using negatively stained complexes such as ribosomes. All these discoveries led to a gradual progress on resolution, revolving on 40A during 1990s to 7-9A nowadays.

We can observe in Figure 1 how the progress of single-particle analysis has evolved leading to be called *revolutionary* due to its speed. Two important factors that have helped the speed is the evolution of electron detectors and improved processing procedures.

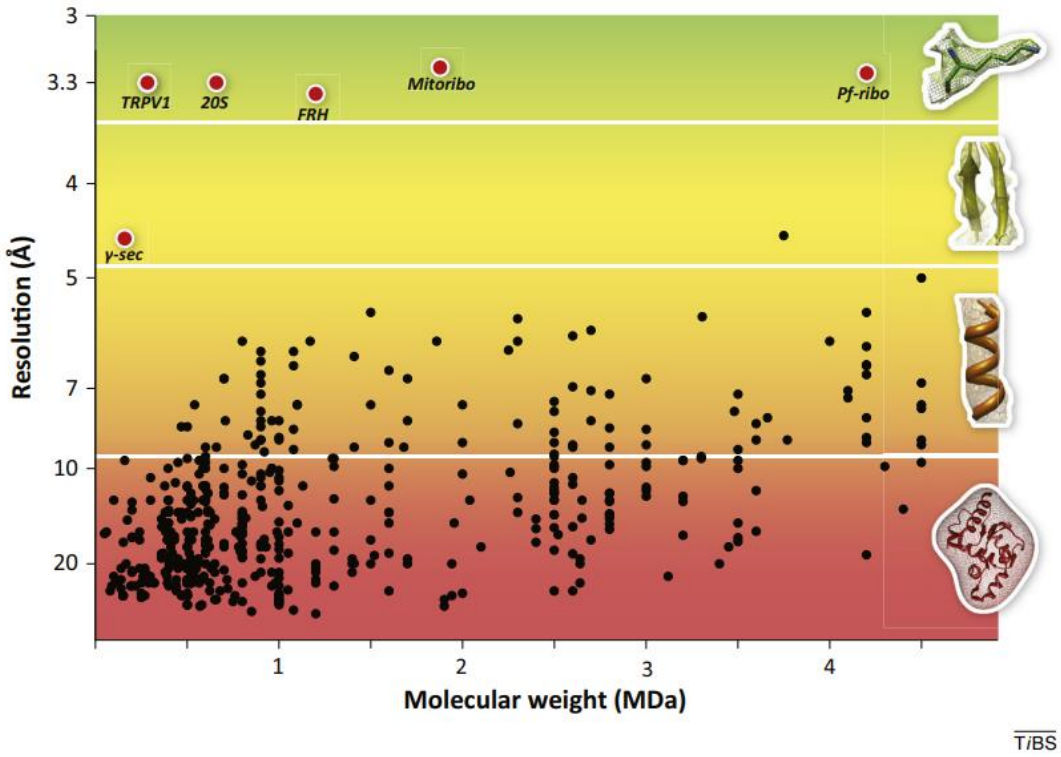


Figure 1: Progress in cryo-EM single particle analysis. Black dots are structures that were released from the EMDB (Electron Microscopy Data Bank) in 2000-2012. Read dots are recent structures reconstructed, we can observe a clear difference in the resolution. g-secretase (g-sec), the transient receptor potential cation channel subfamily V member 1 (TRPV1), the 20S proteasome (20S), F420-reducing [NiFe] hydrogenase (FRH), the large subunit of the yeast mitochondrial ribosome (mitoribo), and the cytoplasmic ribosome of Plasmodium falciparum in complex with emetine (Pf-ribo). Adapted from Bai X *et al*. [6]

2.1.1 Cryo-EM Processing techniques

One of the roots of computer image processing in electron microscopy is David DeRosier (1968) [20] by using Fourier techniques, along with Linda Amos and Aaron Klug (1972) [4] by using Fourier filtering techniques. In the other hand, in 1982 a new system for correction of very high-resolution micrographs (0.1 - 0.2 nm) emerged [35].

According to Carragher (1996) [2] image processing tools for microscopy can be catalogued in 4 functional groups:

- Image restoration
- Image enhancement
- Structure reconstruction
- Structure visualization

Image restoration manages the quality of the initial image, for example the usage of the CTF for removal of aberrations introduced by the used device. *Image enhancement* includes any technique that process the projection of the sample, so the representation of its features is clearer, for example, Signal-to-noise improvement. *Structure reconstruction* are the procedures needed to process 2D images to generate the 3D structure. *Structure visualization* allows to exhibit the 3D arrangements so it can be analyzed thoroughly.

Computational tools try to include implementation in all those groups to assert a particular problem. The first programs were in 1968 by Arron Klug's laboratory [41], in 1982 by Walter Hoppe's laboratory [35] and SPIDER in 1981 [28], which was one complete standalone package for cryo-EM processing. After the drop of workstations price in the 90s, new applications were created, such as SUPRIM (1996) [57], IMOD (1996) [43], solid contour (1996) [37], MDPP (1996) [62], PIC (1996) [71], IMAGIC (1996) [72], EM (1996) [34] and MRC package (1996) [21]. Also, some packages such as SPIDER (1996) [28] implemented a graphical user interface (GUI) – called SPIDER's Web tool - since it was an innovative introduction for novice users.

According to Smith [61], new developments in the last decade have influenced the growth of software tools. Those three changes are:

- Growth in hardware performance
- Dominance of the Internet
- The open source movement

Growth in hardware performance has allowed the increase of techniques since computational power plays a critical role for calculating the conformation of macromolecules. Limitation of CPU capacity can be diminished by parallelization of the large datasets - this procedure lifts a toll on wall-clock time as shown by Yang et al. (2007) [81] on SPIDER software package -, use of CPU with a higher number of cores – as shown by Tang et al (2007) [69] -, or employ GPU's or quantum computing [24].

The dominance of the Internet has discerned how software packages are distributed. Going from email address to appearing on the Wikipedia page, the global interconnection has greatly simplified the dissemination of data between computers.

The open source movement has provided a rich amount of free software, code, applications and libraries for the development and increase of capabilities of many projects which are supported by large communities of developers.

The pipeline used to resolve the 3D reconstruction in single particle cryo-EM can be separated in four different steps as seen in Figure 2 [23]. We can observe four different steps: preprocess micrographs, particle picking, 2D classification and 3D reconstruction. All four steps will be explained thoroughly below.

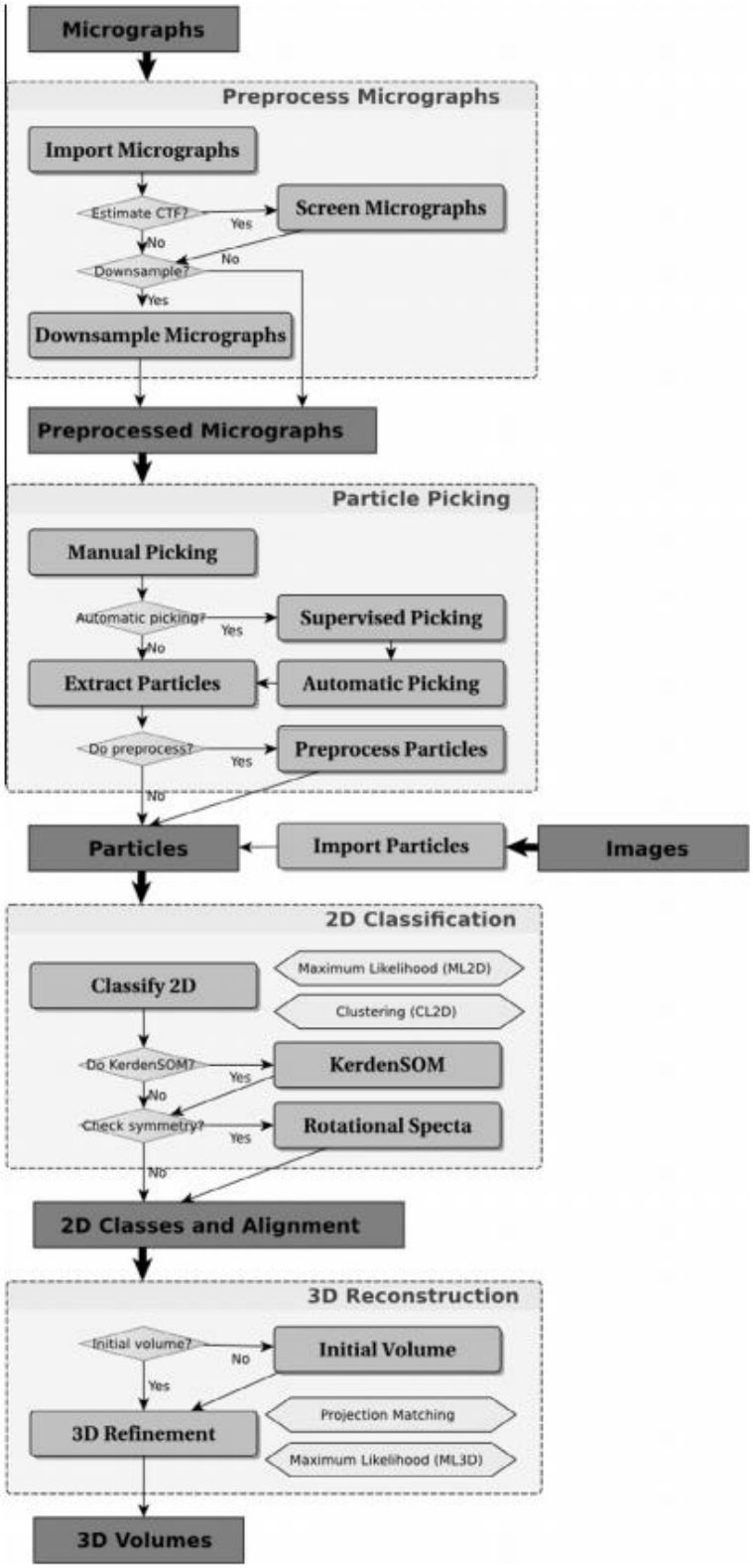


Figure 2: Example of a typical image processing pipeline. The pipeline consists of four different steps. Adapted from De la Rosa-Trevin *et al*. [23]

Preprocess Micrographs

The input for the preprocess is the micrographs, since lens that obtained the 2D representation may have introduced aberrations, the image needs to be processed. The most used file format for micrographs is MRC, it is a binary file format is widely used for storing image and volume data in 3D electron microscopy [16]. This format is used in software packages such as CCP4 [17], MRC library [16], IMOD [43], UCSFTomo [67], EMAN2 [68], Appion [44] and XMIPP. [65].

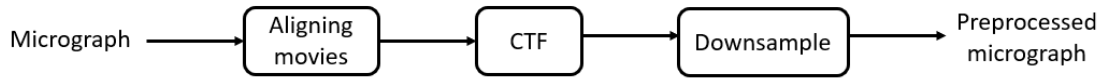


Figure 3: Preprocessing steps, aligning movies and CTF is used in CisTEM, the other step, Downsample is used in Xmipp 3.0 [28, 23].

Micrographs may be drifted physically or as a result of induced motion [10, 11, 45]. For resolving this problem, CisTEM [30], as seen in Figure 3, has implemented an *Aligning movies* step. This step is composed of *unblur algorithm*[1] described by Grant and Grigorieff (2015) [29] and a *SNR maximization* (see Equation 1). This equation attempts to maximize the signal-to-noise ratio by considering damage in the sample and is explained below. It represents the SNR after the exposure of N e^-/A.

$$SNR(k,N) = SNR(k,0)e^{-\frac{N}{Ne(k)}}$$

Equation 1: Sound to Noise ratio after the exposure to a certain amount of electrons.

k is the spatial frequency, **N** is the accumulated exposure over time, $\mathbf{N_e}$ is the resolution dependent critical exposure and **SNR(k,0)** is the signal to noise ratio when the

[1] Based on the iterative alignment of each raw frame to the current best total sum of all other frames, except itself. For more information see http://grigoriefflab.janelia.org/unblur

accumulated exposure is in its initial value (t = 0). The representation of Equation 1 is shown in Figure 4 [7]. It shows that the SNR will increase as the exposure to electrons is larger than the critical exposure, and according to Grant and Grigorieff (2015) [29], we can conclude that overexposing the specimen is slightly better than underexposing it.

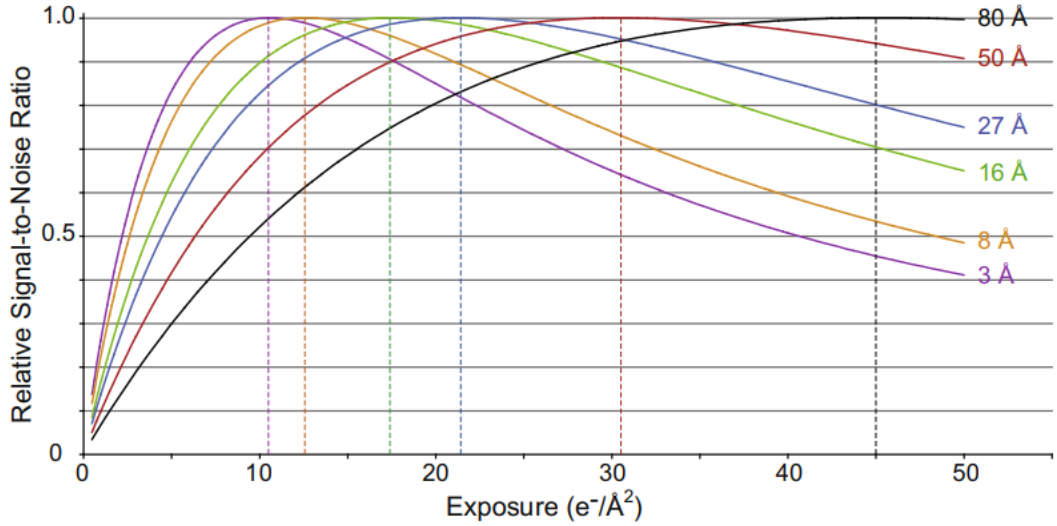


Figure 4: Relative signal-to-noise ratio as a function of exposure at 200kV. SNR is scaled [0,1] and different resolutions are shown (each colored line). In order to select the best exposure, compromise between maximize SNR at lower resolutions and maintain sufficient SNR at high resolution is necessary. Adapted from Baker [7]

The other preprocessing step is the *Contrast Transfer Function* (CTF). The CTF is a mathematical description of the imaging process in the Fourier space [18]. Ideally, images represent the 2D projections of the specimen, although, in reality these images are distorted and with high levels of noise. CTF is the function that allow us to correct all these effects by analyzing the 1D and 2D power spectrum, it filters both high and low frequencies or at certain frequencies [64]. For example, as shown in Figure 5 drift is characterized by directional fall-off in the Thon rings and astigmatism is confirmed by elliptical rings (when they should be circular).

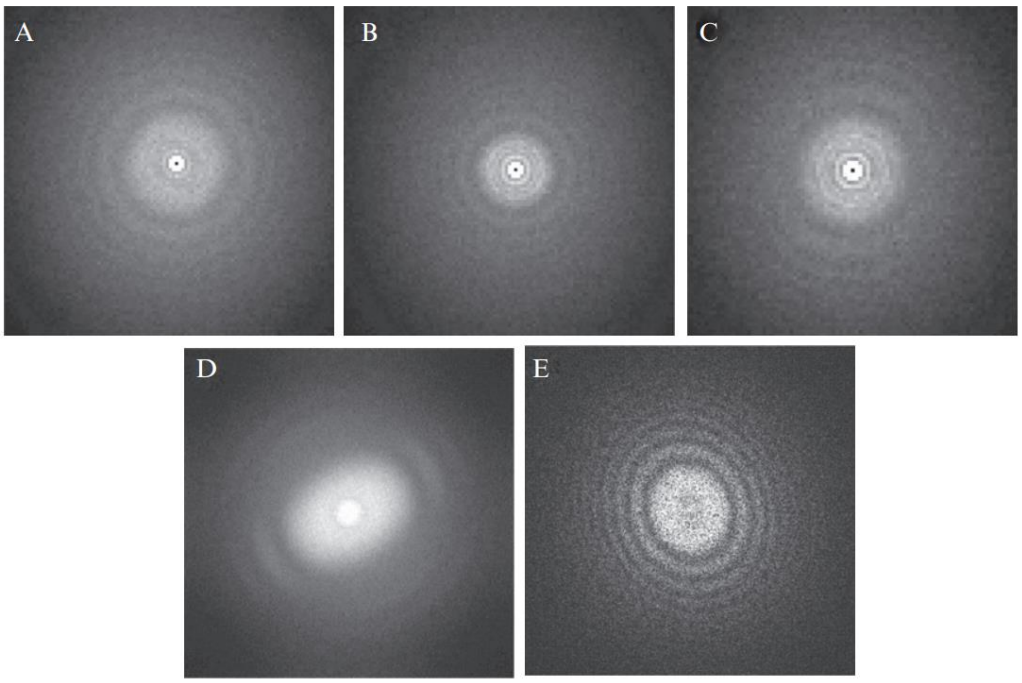


Figure 5:. 2D power spectra of typical good particle images close to focus (A), and far from focus (B), and images with drift (C), vibration (D), and astigmatism (E). (Adapted from Cong 2010 [18]

Particle Picking

The particle picking problem is to detect and locate particles randomly allocated in an image [60]. The input for the particle picking step is the preprocessed micrographs. This step can be done with hours and hours of manual picking, however, thanks to modern automatized techniques, automatic picking has been widely implemented and used. In this step, we will present a fully automated process such as the particle picking implemented in FREALIGN software, and a implementation of automatic particle based on a "matched filter" or "correlation detector" – called *ab initio* - presented by Sigworth (2004) [60] and implemented in cisTEM. Although there are many more methods, some of them will be only named since the material is too extense.

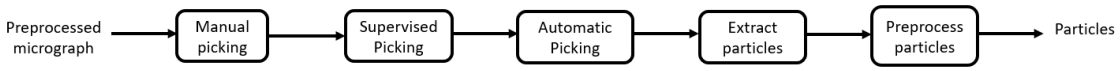


Figure 6: Particle picking steps for the fully automated process.

The first type of particle picking that will be presented is the fully automated process. Although it is called "fully automated" it is composed of two different parts: The unsupervised process of training to learn how to classify particles and the single particle picking using supervised deep learning. We can observe in Figure 6 a diagram of the general workflow and in Figure 7 a flowchart with the deep learning scheme.

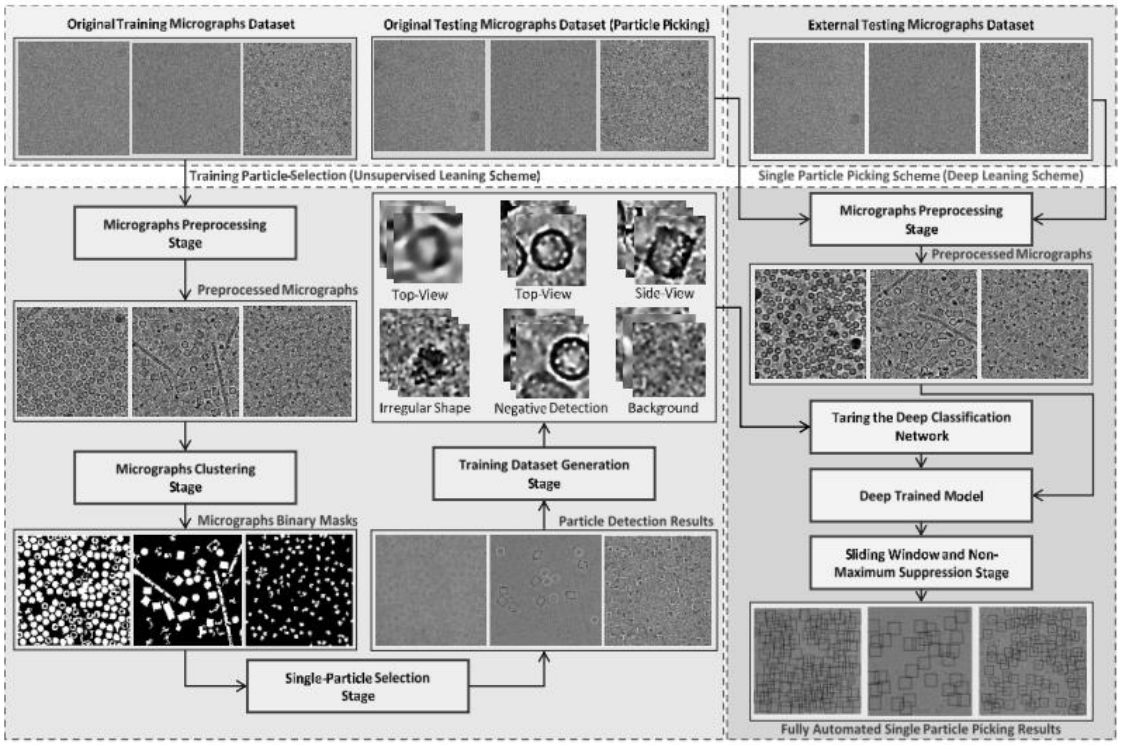


Figure 7: Diagram of a flowchart with the deep learning scheme. Image adapted from Al-Azzawi *et al* **[3]**

The unsupervised process of training goes under two different sections:

- Section 1 is composed of two different steps. First step is preprocessing of the micrograph images and second step is clustering of cryo-EM images through two different unsupervised learning clustering algorithms and then clean, detect and isolate each particle.
- Section 2 is to automatically evaluate each particle as "good" or "bad" for the training sample.

The fully automated single particle picking also has two different sections:

- Section 1 is the design and train of the deep convolutional neural network using the dataset created in Section 2 of the unsupervised process.
- Section 2 is using the trained model to test micrographs after the pre-processing.

This will allow us to extract the location of the particles and they can go through a preprocess method if the image is not satisfactory to the user.

The other method presented by Sigworth (2004) [60] - called *ab-initio* - is composed of two different steps. The full process is shown in Figure 8.

Firstly, we consider that the noise is independent and identically distributed, this way, noise will present a flat power spectrum. Since this is not true, we must force the noise power spectrum as seen in Figure 8.A to be white with the help of a pre-whitening filter.

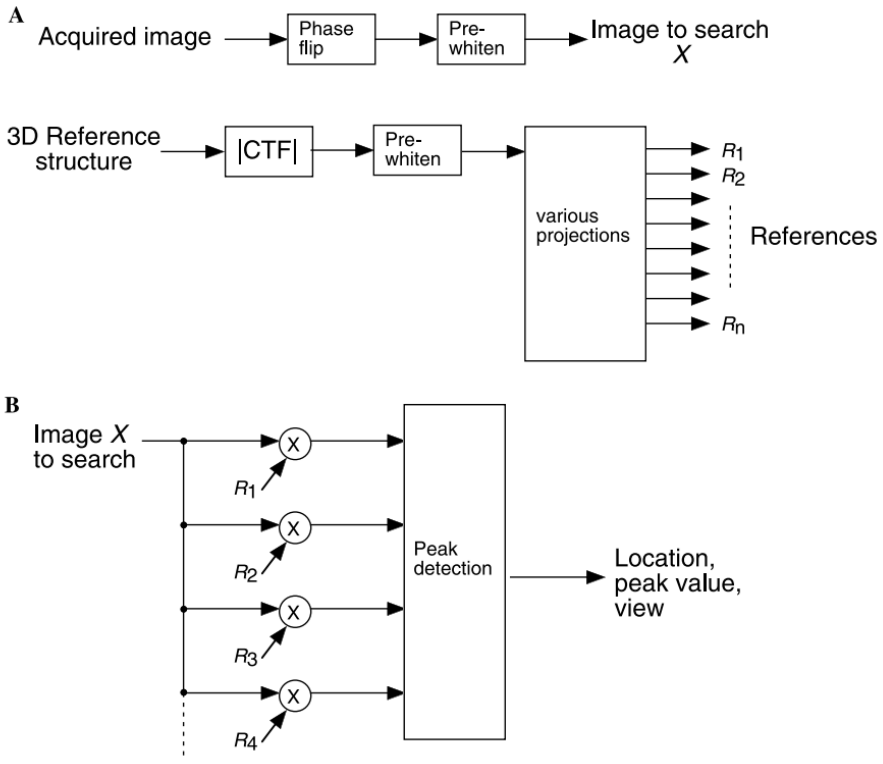


Figure 8: Particle detection process done with a matching filter. (A) The micrograph can be processed by phase flipping (this reduces the dispersion) and by a pre-whitening filter. Meanwhile, the 3D reference is filtered by the CTF and the pre-whitening filter to create references. (B) The processed image is cross correlated with each reference to obtain the location of the particles. Image adapted from **[60]**

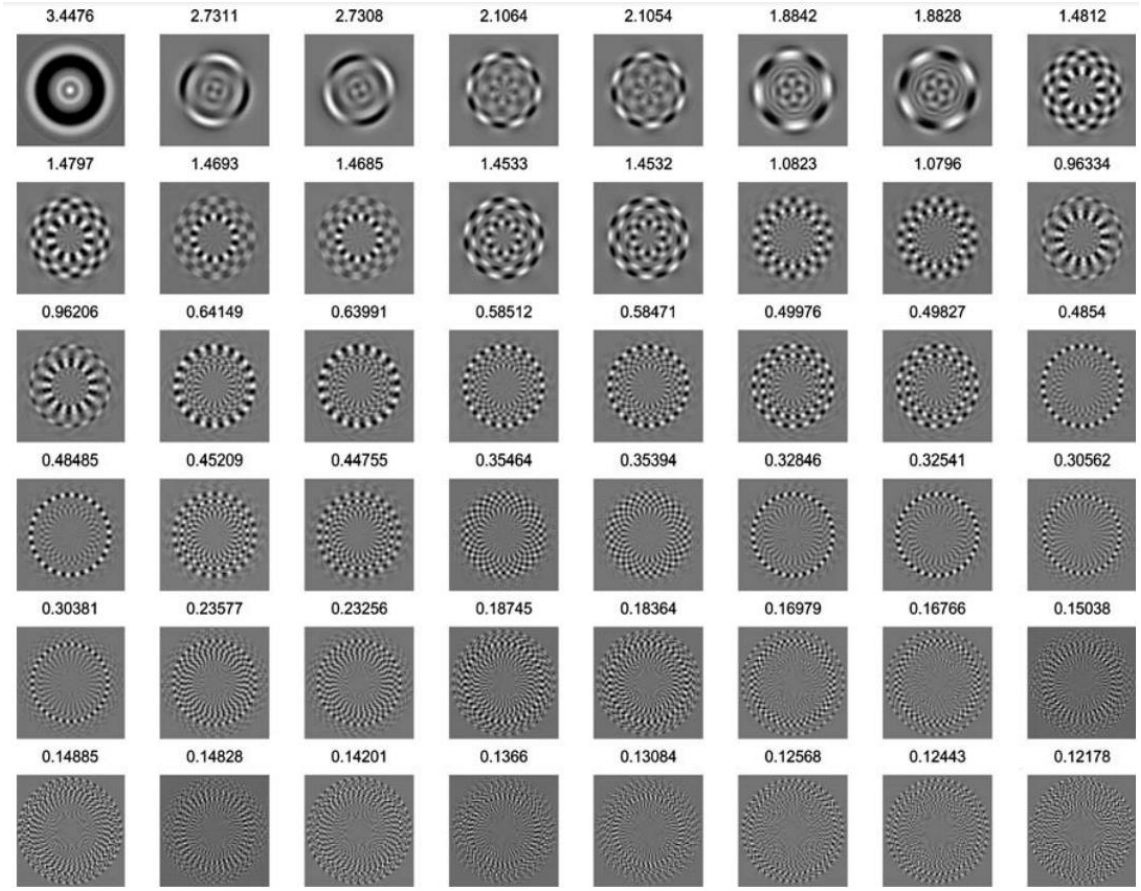


Figure 9: Eigenimages from each principal component analysis of the references. Above each eigenimage, the singular value is written, which is proportional to the RMS (root mean square) value of its coefficients. Image adapted from **[60]**

Secondly, a representative set of reference projections must be created. For higher efficiency, eigenimages from the set will be created (we can observe an example of eigenimages in Figure 9). Finally, cross-correlations will be done between the data images with the pre-whitening filter and the eigenimages. This last process will show what kind of potential particles does surpass a certain threshold and will be classified ad particles. The correlation detector does not discriminate well between particles and objects since it only classifies images as particles if the inner product surpasses a determined threshold.

Let us present now some particle picking algorithms for the reader's knowledge: cyYOLO based on deep-learning object detection based on YOLO (You Only Look Once) – used in SPHIRE-CrYOLO software - [77], with a Laplacian-of-Gaussian (LoG)

filter which is a reference free particle picking – used in RELION3 software – [85], Difference in Gaussian (DoG) filter – used in DoGpicker – [76], APPLE picker – used in ASPIRE software – [33, 58], local Cross-correlation – used in SPIDER software – [52, 28], Circular Hough Transform algorithm which detects the shape and center of each particle – used in AutoCryoPicker software – [3], Deep Convolutional Neural Network – used in DeepCryoPicker software, DeepEM, DeepPicker – [3, 79, 84].

2D classification

The 2D classification can be used to remove figures that may not be particles or may not be from the desired class [30]. The process starts with classes calculated as averages or randomly sampled; these classes will be refined while iterating through all the particles. We can observe the refine process made on CisTEM application in Figure 10. A number of classes is selected and by different algorithms (presented below), each particle is "compared" to each class average. Each particle will be assigned to a class regarding their similarity and a new average will be computed including the new particle. This cycle will continue until no particles are left.

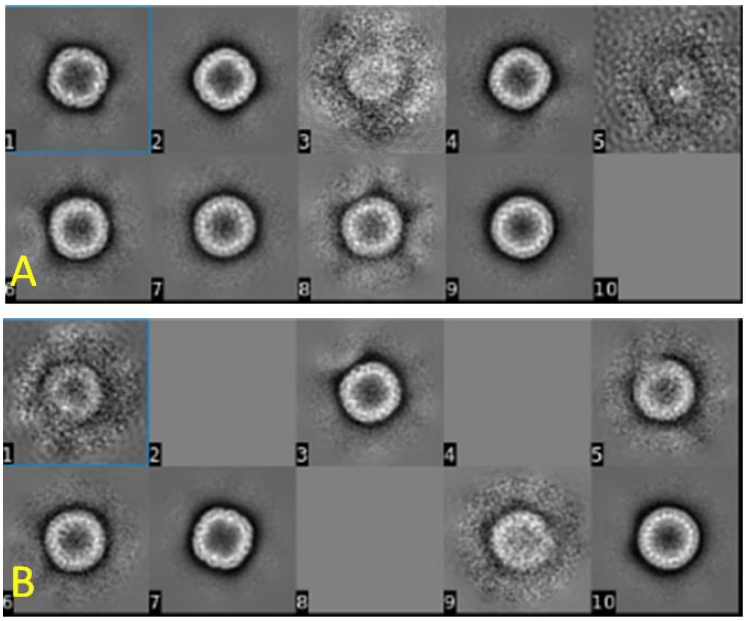


Figure 10: 2D classification process on CisTEM GUI application. We can observe how the desired classes are refined from A to B. Image from CisTEM software **[30].**

In the following iterations, class averages are refined using different algorithms such as: Maximum Likelihood Algorithm – used in IMAGIC and cisTEM software - [59, 27, 56], K-means clustering (implemented in ISAC algorithm) – used in SPIDER and SPHIRE software – [80], KerDenSOM (Kernel Density Estimator Self Organizing Map) algorithm – used in Xmipp software – [42, 65], Bayesian Polishing – used in RELION3 software – [85], FuzzySOM (Fuzzy Self Organizing Map) algorithm – used in Xmipp software – [65], PCA (Principal Component Analysis) – used in IMAGIC, SPIDER and Xmipp software – [65, 27, 72], Steerable PCA, – used in ASPIRE software – [58], Sammon mapping – used in Xmipp software – [65, 54], Fuzzy C-Means – used in Xmipp software – [65, 12], Kernel C-Means – used in Xmipp software – [65, 22, 50], Fuzzy Kohonen Clustering Network [65, 49], Deep Classification Network (can be seen in Figure X) – used in DeepCryoPicker software – [3], Deep learning class based tool called Cinderella – used in SPHIRE-crYOLO software – [78], fast randomized SVD – used in ASPIRE software – [58], fast randomized nearest neighbor – used in ASPIRE software – [58].

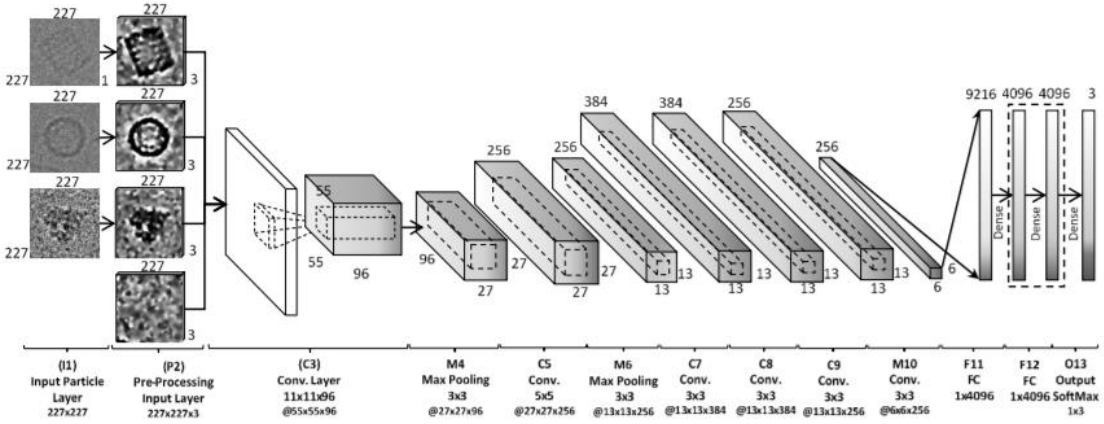


Figure 11: Deep Classification Network of 13 layers. Layers are input layer, pre-processing layer, convolutional layers, sub-sampling layers, two fully connected layers, and one output layer. Image adapted from Al-Azzawi *et al* [3]

3D Reconstruction

To recreate the 3D structure of a macromolecule, it is necessary to have thousands of projections of the sample [1]. This happens due to the low contrast and image deformations of 2D representations if a high-resolution reconstruction is wanted. The procedure done by Xmipp will be explained in the following paragraphs, this analysis is composed by a CTF correction, angle assignation and the 3D reconstruction.

If high-resolution must be achieved, the CTF must be corrected. Some ways to correct the CFT may be to use the IDR (Iterative Data Refinement) algorithm – which is implemented in Xmipp - [64, 66], to incorporate the PSF (Point Spread Function) into the reconstruction equations [66].

The next step would be to assign the angle in which the particle has been projected, in other words, the direction of the sample when the photo was taken. One approach could be to compare each image to some other reference particles that are similar to our sample study, but the assignation error may rise to 98% in some cases [66] due to high level noise [65]. Angular assignation applied in Xmipp software computes a correlation between the images and the reference library, if the correlation is high, further comparison will be done with a higher resolution. This has been shown to be more robust when compared to the standard method of angular assignment [66].

For the three-dimensional reconstruction, Xmipp implement variants of iterative 3D reconstruction algorithms. They are: ART (Algebraic reconstruction technique) [38, 66], SIRT (Simultaneous Iterative Reconstruction Technique) [5, 66], CAV (Component AVeraging) [14, 66], BICAV (Block-Iterative Component AVeraging) [14, 66] and Averaging Strings [13, 66]. Although voxels are popularly used in 3D reconstruction models, Xmipp opted for a Blob approach since they are smooth in real and Fourier space thus suitable for noisy projection images.

2.2 Software to process a 3D Reconstruction

2.2.1 ASPIRE

Algorithms for Single Particle Reconstruction (ASPIRE) is an open-source project that combines different tools focused on computational methods other than 3D iterative refinement.

This software is developed in Matlab and Python, although the official version is the Python one. The algorithms provided are 3D ab-initio modeling, 3D heterogeneity analysis and particle picking [58], 3D structural variability analysis [48, 40], 2D classification and averaging (steerable PCA) [82,83] and image restoration [9] and Angular reconstitution from common lines [73, 19, 87].

At present, ASPIRE does not have CisTEM integrated

2.2.2 cryoSPARC

CryoSPARC [51] is a backend and frontend software system developed in 2016 for single particle cryo-EM data and image analysis. It works using both user interface and command line tools.

Functionalities provided by cryoSPARC are import data (either movies, micrographs, particle stack, 3D volumes or templates), motion correction, CTF estimation, exposure curation, particle picking, particle curation (2D classification and selection of the classes), 3D reconstruction (Ab-initio), refinement, CTF refinement, local refinement and post-processing.

This software allows the user to process the data in a tree-like diagram. The Project is the high-level container of one or various Workspaces and the Jobs associated to those Workspaces. We can observe in Figure 12 the representation of a single particle cryo-EM processing process.

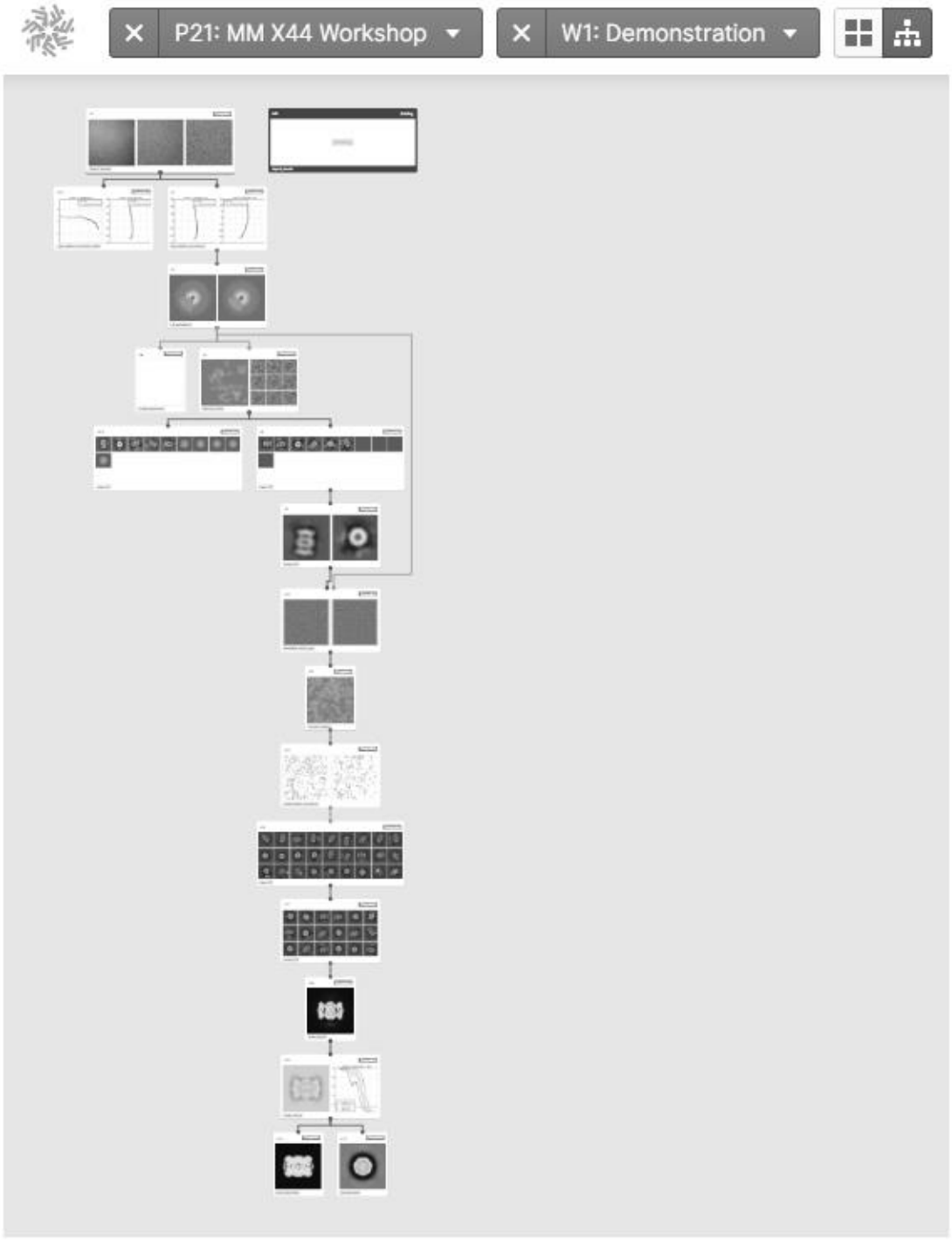


Figure 12: Screenshot of cryoSPARC interface. We can observe the Tree View, where it is shown how all jobs (processing of micrographies) are connected inside one Workspace.

At present, Scipion has integrated the following methods:

- Initial model: Generates a 3D initial model from 2D particles.

- 2D classification: Classifies particles into multiple 2D classes.
- 3D non-uniform refinement: Achieves higher resolution and map quality.
- 3D homogeneous refinement: Protocol to refine a 3D map.

2.2.3 RELION

REgularized LIkelihood OptimizatioN (RELION) is an open-source computer program developed in 2012 by the MRC Laboratory of Molecular Biology (Cambridge, UK) [55]. It refines macromolecular structures of cryo-EM data by applying a Bayesian approach, which results in high quality reconstructions and estimations with decent computational requirements.

In 2018 RELION-3 [85] was released with the addition of GPU support and new developments were added: CTF refinement, beam-tilt estimation and correction, Ewald sphere correction, CPU vector acceleration, Python scripts for automated processing, implementation of a Gaussian Process regression algorithm in Bayesian particle polishing and the addition of an iterative approach for multi-body refinement.

At present, Scipion has integrated the following methods:

- Motion correction: Implementation of motioncor algorithm.
- 3D initial model: Creates a 3D initial model from 2D particles using SGD algorithm.
- Autopicking LoG: Implementation of Relion's Laplacian of Gaussian option.
- Auto-picking: Protocol to pick particles from micrographs using 2D averages o 3D volumes.
- 2D classification: Relion's 2D classification.
- 3D auto-refine: Refines a 3D map employing an empirical Bayesian approach.
- Local resolution: Post processing operations for local resolution estimation.

2.2.4 CisTEM

Computational imaging system for Transmission Electron Microcopy (CisTEM) is an open-source software developed on 2018 by the Howard Hughes Medical Institute (United States) [30] for the processing of data for high-resolution electron cryo-microcopy and single-particle averaging.

The software is written in c++ and it consists of a well-defined pipeline divided into the following steps: movie processing, image defocus determination, automatic particle picking, 2D classification, abínitio 3D map generation from random parameters, 3D classification and high-resolution refinement and reconstruction.

One of the most interesting characteristics of CisTEM is that all functionalities are developed for both command line and user interface. We can observe in Figure 13 how the main menu of the interface is presented.

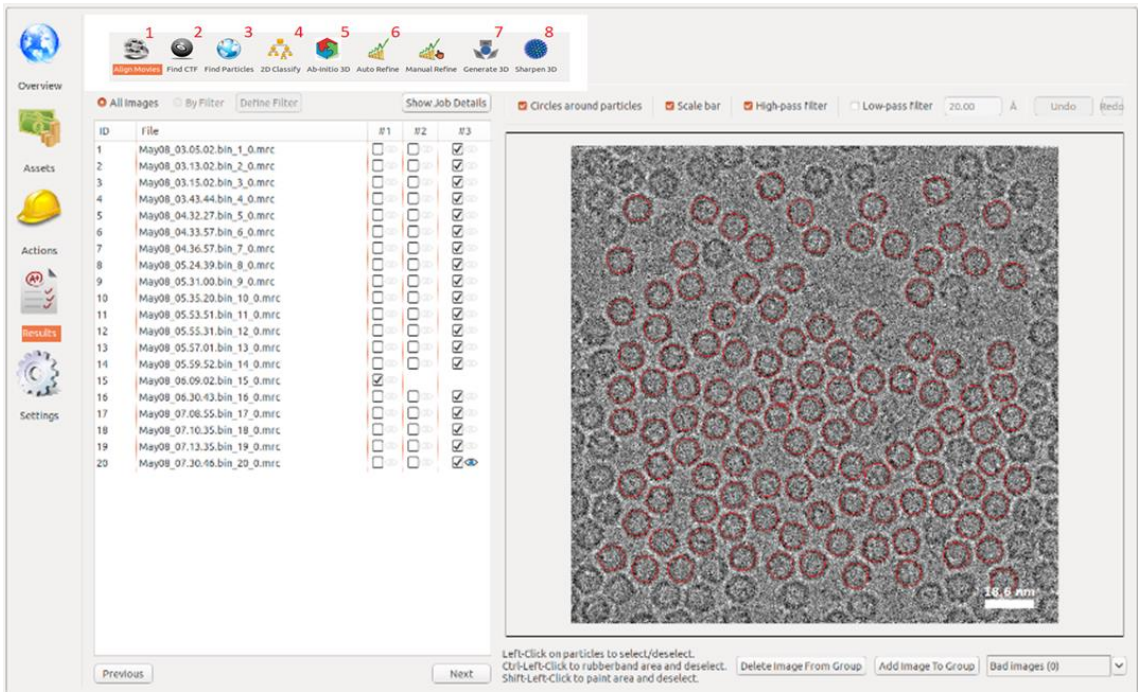


Figure 13: Session on CisTEM running under Linux. User throws the toolbox through command line (./CisTEM) which starts a session. We can observe how al functionalities are presented in a user-friendly interface. Functionalities are movie processing (1), image defocus determination (2), automatic particle picking (3), 2D classification (4), abínitio 3D map generation from random parameters (5), 3D classification (6) and high-resolution refinement and reconstruction (7) [30].

CisTEM does not support computation on graphical processing units (GPUs) and it is focused on being optimized for CPUs. The processing of datasets (typically 2000 micrographs with 200.000 to 300.000 particles) is typically half a day or less.

At present, Scipion does not have CisTEM integrated.

2.2.5 SPIDER

System for Processing Imagen Data from Electron microscopy and Related fields (SPIDER) [28] is an open source project developed in 1978 by Joachim Frank (shared the 2017 Nobel Prize in Chemistry). The latest version is supported for Linux on Intel & AMD (July 31st, 2020) in which they updated functionalities for the 64 Linux distribution.

This software has two different distributions (as separate modules): SPIDER – written in Fortran and used for mathematical manipulation of images- and the Web module – written in C++ and Java and used for visual display and interaction with the images created by the SPIDER module-.

SPIDER presents more than 200 different operations (including simple tasks such as adding images or arithmetic operation). The user can create a processing pipeline using 2D techniques (CTF correction, particle picking, particle alignment, supervised and unsupervised alignment) followed by 3D techniques (SPR, CTF correction, and more) for the reconstruction of the molecule.

SPIDER also makes use of Spire [8] and python tools (for analyzing or preparing data). We can observe in Figure 14 the graphical user interface that Spire provides to SPIDER, with this functionality the output files generated by SPIDER may be better organized and managed.

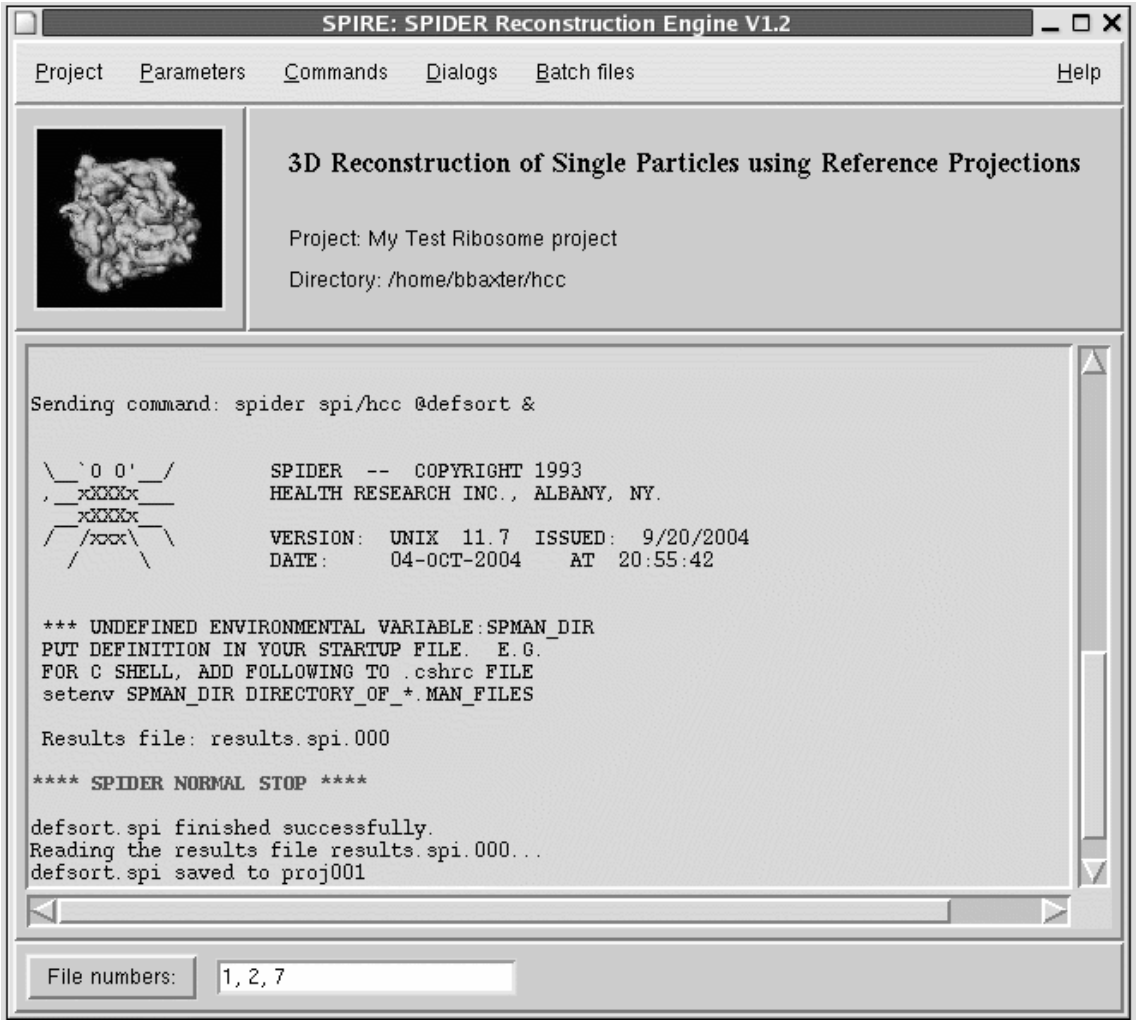


Figure 14: Screenshot of SPIRE run on SPIDER. Image obtained from https://spider.wadsworth.org/spider_doc/spider/spire/spire-docs/spire.html

Unfortunately, since little number of active contributors maintain the software, SPIDER will have its final release in 2020.

At present, Scipion has integrated the following methods:

- Classify ward: Finds clusters of images/elements using Diday's method.
- Classify K-means: Automatic k-means clustering and 2d classification.
- Classify diday: Automatic clustering using Diday's method and 2d classification.
- Refine 3D: Iterative refinement which improves the determination of orientations.

2.2.6 SPARX

Single Particle Analysis for Resolution Extension (SPARX) [39] is a Transmission Electron Microscopy (TEM) oriented image processing software developed on 2006 in the United States of America. It is based on a broad Python library (which perform the TEM computational tasks) and a C++ core library for image processing functions which contains scripts from SPARX developers and from the open-source PHENIX project (EMAN2 library and cctbx) [68, 69].

As we can observe from Figure 15, the software allows interaction with the users in three different ways: a graphical user interface (no need for an appropriate programming background), use of pre-written scripts from a command shell or through a text based customized python interpreter.

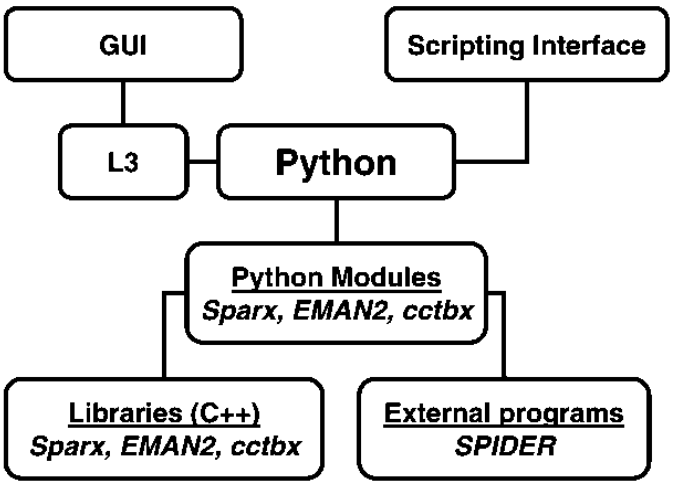


Figure 15: Diagram of the design of SPARX. We can observe thar SPARX is an independent Python module, as well as EMAN2 and *cctbx*, this first module is built on top of the other toolkits. We can also observe the different interaction user-software. Image adapted from **[39]**

From the disposition of the modules in Figure 15, it is shown that SPARX can be accessed or obtain data from different External programs, making it a user-friendly wrapper for Python. The benefit of the disposition chosen for SPARX is the great advantage to

incorporate new projects and packages into itself making use of new possible functionalities.

At present, Scipion has integrated the following methods:

- SPARX gaussian picker: Automatic particle picker for SPA.

2.2.7 IMAGIC

IMAGe analysis In the Computer (IMAGIC) [72] is an open-source software package developed in 1981 in The Netherlands. It allows the user to make complicated image analysis procedures. IMAGIC is written in FORTRAN 77 (ANSI standard X3.9-1978) and its first version consists of over 30 thousand program lines (40% are comments). After major revisions [74] and porting to the VMS and UNIX operating systems, adaptation to X11 standard and changes in user interactions lines grew up to 500 thousand. Later, the version [75] will be expanded to Linux, Windows XP/Vista/7 and Mac OS-X.

The first software has an organization divided into 4 programming levels:

- Level 0 are subroutines that perform operations on one-dimensional arrays. These high-speed machine language subroutines are used by level 1 routines.

- Level 1 are organized in different libraries, which englobes buffering techniques and I/O routines. Theses subroutines are used by the level 2 programs.

- Level 2 are the interactive FORTRAN 77 programs (image manipulations). Programs in this level were written according to a structured development tree, for example from programs called SAMPLE1 and SAMPLE2 new level 2 programs were developed increasing complexity in organization.

- Level 3 is the highest level. It consists of a supervising program and a command library. The software allows the creation of new commands as the combination of existing commands, thus, nested in a pipeline-like workflow.

Although IMAGIC is an old software. It has been continuously adapted and is one of the firsts software's to give well suited results for image analysis (specially for the electron microcopy field) and many procedures were pioneered in IMAGIC.

At present, Scipion has integrated the following methods:

- Msa-classify: Enhanced ward-type algorithm (variance on variance-oriented hierarchical ascendant classification program).

2.2.8 XMIPP

X-Windows-based Microscopy Image Processing Package (XMIPP) is a set [23, 65] and designed primarily for the analysis of single particles, being able to help the reconstruction of biological macromolecules. The software is developed in ANSI-C and X11 library for graphical output although bindings were also added for Python and Java. This last addition allows code written in Python and Java to work as protocols that can access in a fast and easy was to XMIPP functionalities [23].

The software is composed of different steps that can be called as single standalone programs, allowing the smooth integration of new modules. The user can either start (by offering their own set of images) or stop the workflow at any point of the processing pipeline, allowing XMIPP to easily combine different software packages. Figure 16 shows us, how steps are shown on XMIPP interface as a tree like diagram. All the Project information is stored in a Sqlite3 database, which stores basic information such as the runs and the individual steps of each run but does not store results.

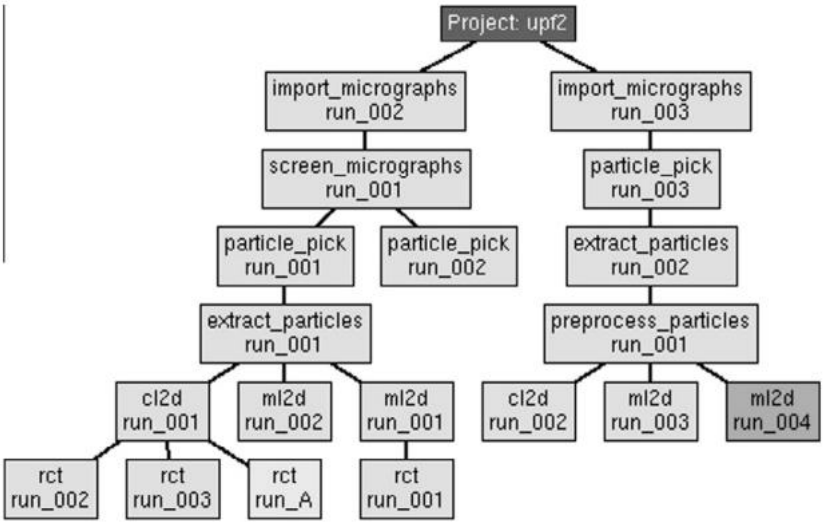


Figure 16: Tree-like schematic representing a typical XMIPP project. As we can observe, a Project can have two different set of micrographs, and from one set of images, we can run two different processes (as seen in particle_pickrun_001 and particle_pickrun_002). Image adapted from **[23]**

It was firstly designed for UNIX workstation with 32 MB of memory, later, it was portable to machines with GNU C++ compiler and Qt graphical libraries and in version 3.0 it is portable to all machines with UNIX operating system.

At present, Scipion has integrated the following methods:

- Optical alignment: Protocol for Movie Alignment by optical flow.
- Movie average: Protocol to average movies.
- FlexAlign: Protocol for Movie Alignment by cross-correlation.
- CTF estimation: Protocol to estimate CTF.
- Estimate local defocus: Protocol to refine the CTF.
- Reconstruct significant: Initial model problem with Weighted Least Squares framework.
- Ransac: Computes an initial 3d from projections using RANSAC algorithm.
- Auto-picking (step 2): Protocol to pick particles using previous training.
- Tilt pairs particle picking: Pick particles in untitled micrographs.
- Manual-picking (step 1): Pick particles manually or in supervised mode.
- Cl2d: Classification of images with a clustering algorithm
- Gl2d: 2D alignment using Xmipp GPU correlation.

- Gl2d streaming: 2D alignment in full streaming using Xmipp GPU correlation.
- Projection matching: 3D reconstruction and classification using multireference projection matching.
- Localdeblur sharpening: Obtain a sharpened map from a resolution map.
- Highres: 3D refinement protocol with volume and particles as input.
- Local MonoRes: Protocol assigns local resolutions given a map.

2.2.9 EMAN2

Electron Micrograph ANalysis (EMAN) [47] is a software package developed in 1999 and EMAN2 [69] is its successor. It is a image processing package focused on the reconstruction of particles extracted from Transmission Electron Microscopy (TEM).

We can observe in Figure 17 the design of EMAN2 consists of a C++ core library with image processing routines, Python bindings for accessing the C++ core library, command line programs written in Python, GUI Widgets for image display and manipulation and GUI Programs written in Python, which makes them easier to customize for certain applications.

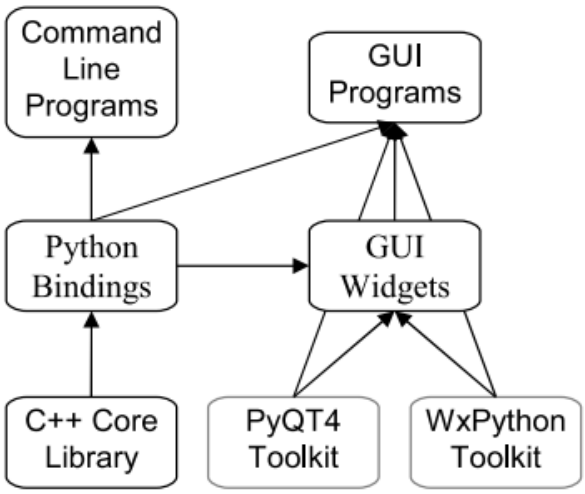


Figure 17: Diagram of the design of EMAN2 software. We can observe how the C++ Core library is accesses by the Python Bindings, which are used by either by command line programs, GUI programs or GUI Widgets (the three of them are written in Python). Image adapted from **[69]**

EMAN2 fully supports Linux, OSX and Windows.

At present, Scipion has integrated the following methods:

- Initial model: Takes a set of class averages to build a ser of 3D models.
- Boxer auto: Automated picker for SPA.
- Boxer: Semi-automated picker for SPA.
- SPARX gaussian picker: Automated picker for SPA.
- Refine easy: EMAN2's single particle refinement program.

2.2.10 FREALIGN

Fourier REconstruction and ALIGNment (FREALIGN) [31] is a free software presented in 2006 although the development began in 1996 at the MRC Laboratory of Molecular Biology (Cambridge, UK) [32]. Its main purpose is for either to determine initial alignment parameters or to refine the structure of a single particle by starting from an initial structure.

The main difference between FREALIGN and SPIDER, IMAGIC or EMAN, is that FREALIGN is focused on refinement and 3D reconstruction, however, the other software packages covers all processes needed to obtain a refined 3D structure from single particle images.

FRALIGN allows the distribution of N processes (specified by the user) across many nodes to speed up the refinement. When all parameters have been refined, the output parameters files can be combined and used in a final run to calculate a new 3D reconstruction. The software is supported for Linux and Mac OS. The run is done by command lines inside a terminal and is written in Fortran 77.

At present, Scipion does not have FREALIGN integrated.

2.2.11 SIMPLE

Single-particle IMage Processing Linux Engine (SIMPLE) [27] is a free software under the GNU General Public License. Although SIMPLE 1.0 is not a standalone for single-particle reconstruction since it focuses on *ab-initio* 3D reconstruction, heterogeneity analysis and refinement, SIMPLE 3.0 includes new features such as a full pipeline using minimal CPU resources. The software is supported for Linux (Ubuntu 16.04 and above) and MaxOSX (10.10 and above).

At present, Scipion does not have SIMPLE integrated.

2.2.12 DeepCryoPicker

DeepCryoPicker is a fully automated approach for particle picking [3] (therefore this means that the user does not need to manually select particles) based on deep learning. It is composed of two different parts: The unsupervised process of training to learn how to classify particles and the single particle picking using supervised deep learning. We can observe in Figure 19 a diagram of the general workflow of the deep learning scheme.

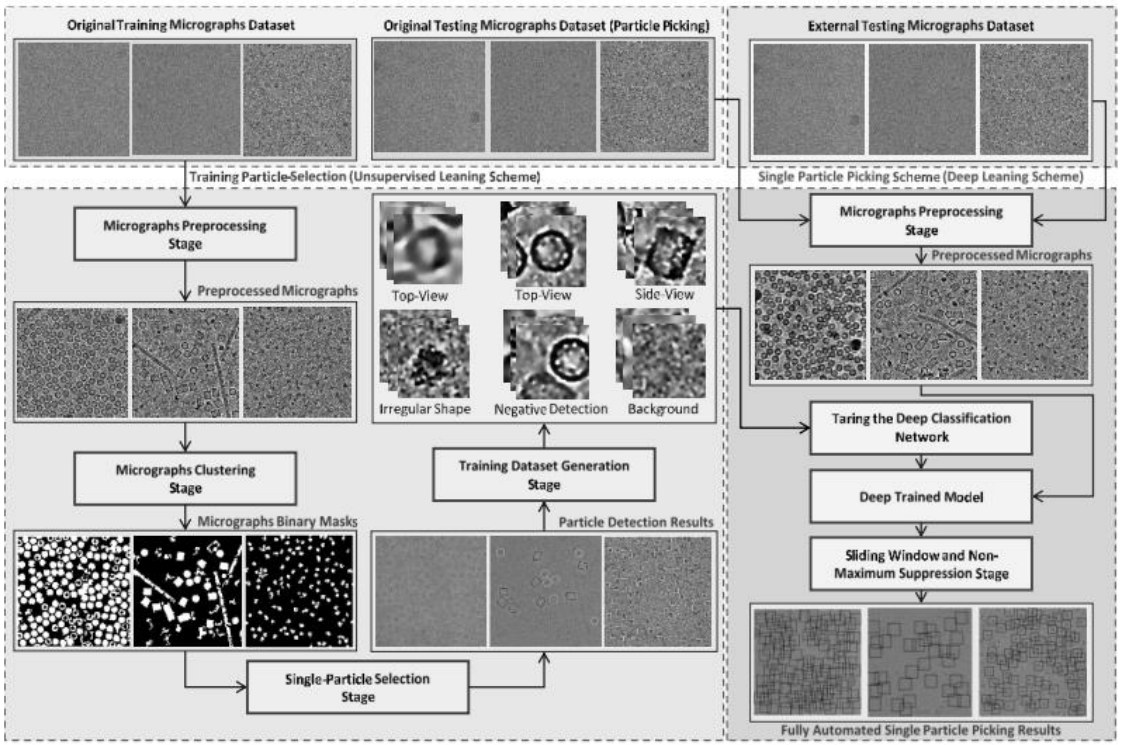


Figure 18: Flow chart of DeepCryoPicker. Image adapted from Al-Azzawi [3**]**

The unsupervised process of training goes under two different sections:

- First step is preprocessing of the micrograph images and second step is clustering of cryo-EM images.
- Automatically evaluate each particle as “good” or “bad” for the training sample.

The fully automated single particle picking also has two different sections:

- Section 1 is the design and train of the deep convolutional neural network.
- Section 2 is using the trained model to test micrographs after the pre-processing.

At present, Scipion does not have DeepCryoPicker integrated.

3 MATERIALS AND METHODS

Let us remember three objectives that we want to achieve in this Bachelor's Thesis:

1. Test and acquire the workflow of an existing C++ open source application used for Cryo-EM image processing, called CisTEM.
2. Design and create a wrapper method to access all CisTEM methods (previously analyzed in our first objective).

In the following sections, point 3.1 will cover objective 1 by analyzing CisTEM internal structure – what are the input and output arguments needed for the processing method. Additionally, it explains briefly the code needed for the extraction of the input parameters. Section 3.2 will cover the first part of objective 2, which is to automatize CisTEM's workflow by external scripts written in python as well as code explaining how it is done. Finally, section 3.3 will cover objective 2 and contains a detailed explanation on how the wrapper method was performed.

3.1 CisTEM

CisTEM is a C++ open source toolbox that can work as a standalone for the image processing of Cryo-EM extracted micrographs. The package is distributed with a GUI interface and can be downloaded in the following webpage: https://cistem.org/. There are two different distributions, one with the program modules stored as executable files and one with the program modules stored as text files for possible code modification. The toolbox follows a simple file organization. We can observe in Figure 20 and in Figure 21 the folder distribution. As explained before, in Figure 20 only executable files are present (green) – *apoferritinTutorial* and *betagal* were added for testing purposes, they are not present when downloading from the official webpage -. In Figure 21 we can observe how the file organization differs from Figure 20 and new modules such as *Makefile* and the source code can be accessed (*src* > *programs*).

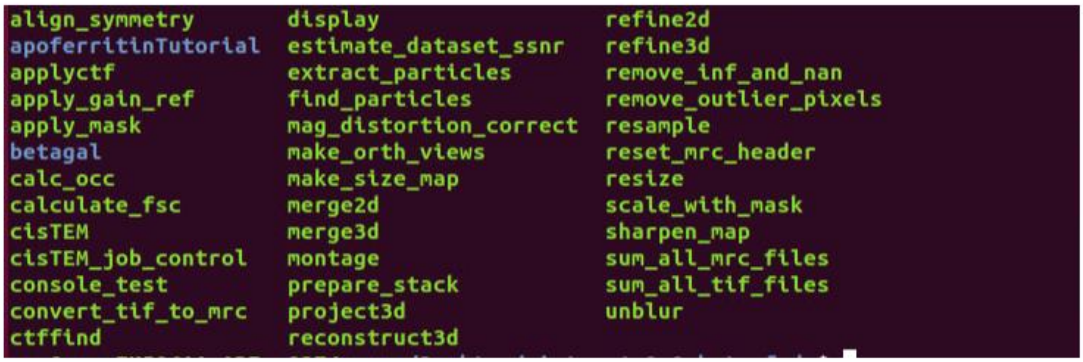


Figure 19: File view on CisTEM toolbox. We can observe that in this version all files are already executables (green). It is not possible to access the code; thus, this version will not be used in our testing of CisTEM workflow.

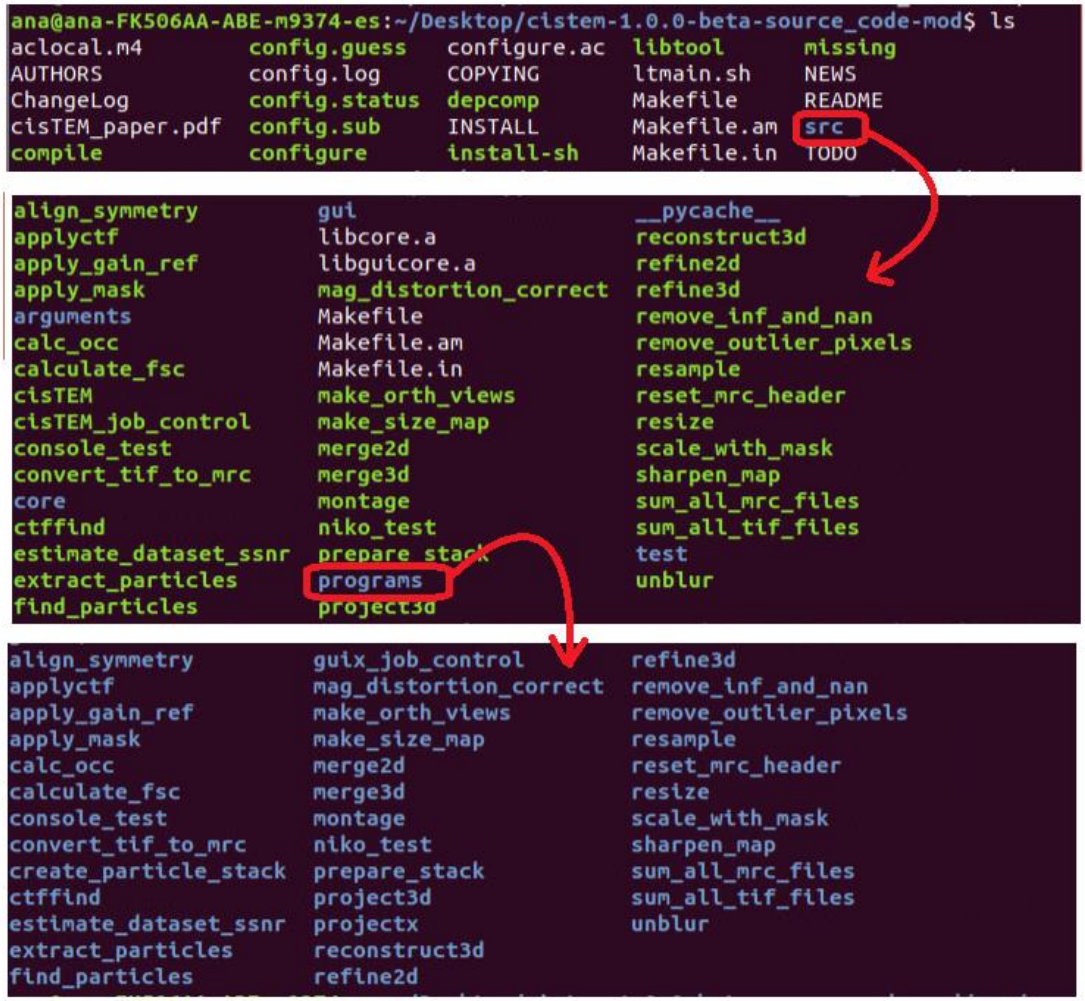


Figure 20: File view on CisTEM-source code toolbox. We can observe that in this version files are executables (green) and C++ scripts (inside each blue folder). Since code modification will be needed, this version will be used in our testing of CisTEM workflow.

Getting started

CisTEM is an opensource toolbox freely downloadable from https://cistem.org/. The tutorial to be followed is available at https://cistem.org/documentation#tab-1-1 and the Apoferritin tutorial dataset for Cistem is EMPIAR-10146 located in https://www.ebi.ac.uk/pdbe/emdb/empiar/entry/10146/.

CisTEM's workflow

CisTEM is composed of different executable files. Let us remember that these executable files can be accessed either by the GUI or by command line. When accessed by command line, the program will ask the user to input the value of each parameter, one by one. The main problem is which script is accessed, when, how many times and with what argument values when doing a task. To resolve this problem, the source code will be modified so that it stores in a .log file all the necessary information to recreate the process.

CisTEM stores all information in a *my_current_job_arguments* string variable. We make use of that variable and extract all parameters every time the GUI calls each script and store it in a .txt called *cisTEMpipeline*. Let us note that the order for storing data in our file is not the order in which the arguments are stored in *my_current_job*_arguments, but the order in which they are needed when the executable file asks for the parameters. We can observe a full example on Figure 22, although for the reader's simplicity, the following lines will represent the code used for storing data in *myfile*:

```
ofstream myfile("cisTEMpipeline.txt",ios::app);
myfile << when_using_strings + "\n";
myfile << BoolToString(when_using_boolean) + "\n";
myfile << std::to_string(when_using_numerical_value) + "\n";
myfile << "\n\n\n"<< endl;
myfile.close();
```

In addition to the code presented above, a file called *checker* was also created to know the times each script was accessed. We can see the code below and in Figure 22.

ofstream fout("checker.txt",ios::app);

fout << "calling *name_of_script*" << endl;

fout.close();

```
//FOR TESTING PURPOSES

wxPrintf("\n\nWe are in unblur, Parameters will be printed below \n");
my_current_job.PrintAllArguments();

wxPrintf("\n\nCreating unblur argument file, please wait... \n");
ofstream myfile("cisTEMpipeline.txt",ios::app);         //CHANGE NAME TO .PAR
myfile << "unblur parameters\n";
myfile << input_filename + "\n";
myfile << output_filename + "\n";

myfile << std::to_string(original_pixel_size) + "\n";
myfile << std::to_string(output_binning_factor) + "\n";
myfile << BoolToString(should_dose_filter) + "\n";
if(should_dose_filter){
        myfile << std::to_string(acceleration_voltage) + "\n";
        myfile << std::to_string(exposure_per_frame) + "\n";
        myfile << std::to_string(pre_exposure_amount) + "\n";
}
myfile << BoolToString(set_expert_options) + "\n";
if(set_expert_options){
        myfile << std::to_string(minumum_shift_in_angstroms) + "\n";
        myfile << std::to_string(maximum_shift_in_angstroms) + "\n";
        myfile << std::to_string(bfactor_in_angstoms) + "\n";
        myfile << std::to_string(vertical_mask_size) + "\n";
        myfile << std::to_string(horizontal_mask_size) + "\n";
        myfile << std::to_string(termination_threshold_in_angstoms) + "\n";
        myfile << std::to_string(max_iterations) + "\n";
        if(should_dose_filter){
                myfile << BoolToString(should_restore_power) + "\n";
        }
        myfile << BoolToString(movie_is_gain_corrected) + "\n";
        if(!movie_is_gain_corrected){
                myfile << gain_filename + "\n";
        }
        myfile << std::to_string(first_frame) + "\n";
        myfile << std::to_string(last_frame) + "\n";
}
myfile << BoolToString(correct_mag_distortion) + "\n";
if(correct_mag_distortion){
        myfile << std::to_string(mag_distortion_angle) + "\n";
        myfile << std::to_string(mag_distortion_major_scale) + "\n";
        myfile << std::to_string(mag_distortion_minor_scale) + "\n";
}
myfile << "\n\n\n"<< endl;
myfile.close();
ofstream fout("checker.txt",ios::app);
fout << "calling unblur" << endl;
fout.close();
wxPrintf("\n\n unblur argument file successfully created! :D \n");

//END OF TESTING PURPOSES
```

Figure 21: Example of code inserted in Unblur.cpp. The order in which all parameters are extracted into our file called cisTEMpipeline is the order in which they would be called if the processing

was done manually (on command line). This process of inserting code manually into our beta-source code files was done in all 37 scripts.

After modifying the beta-source code from the 37 scripts that compose cisTEM's core, the whole project was compiled (note that when we are modifying the source code, executable files are not modified and only the last ones are accessed when running the software). To compile the project, as explained in the INSTALLATION document provided by CisTEM, we must use the command *make* or *make install*.

CisTEM's tutorial

The next step is to obtain the input and output parameters for each script. For this, the Apoferritin tutorial in https://cistem.org/documentation#tab-1-1 was recreated. Figure 23 shows the functionalities in the tutorial and its order. They will be explained thoroughly in the following points.

Figure 22: Screencapture of CisTEM's functionalities. The numbers will be used as a guide for the explanation below.

Some of the needed parameters will be presented below (for Align Movies, Find CTF, Find Particles, 2D classify and Sharpen 3D). Parameters which are not present were not shown for being too extensive (more than 50 parameters). The order in which they are presented in the tables, is the order in which they will be asked by the manual usage of CisTEM. Please refer to the documentation we attached to CisTEM for the parameters needed in the rest of the processes.

1: Align Movies

Align movies is applied to each micrograph individually, it is called as many times as micrographs are going to be processed.

Table 1: Parameters needed for Align Movies. This function calls for Unblur.cpp.

Variable	Type of data	Brief explanation
input_filename	String	Input stack name. It is a mrc file.
output_filename	String	Output stack name. It is a mrc file.
original_pixel_size	Integer	Pixel size of images in Angstroms.
output_binning_factor	Float	Output images will be down sampled by this factor.
should_dose_filter	Boolean	Apply exposure filter before adding the frames.
acceleration_voltage	Float	Acceleration voltage in kV. Available if *should_dose_filter* = 1
exposure_per_frame	Float	Exposure of the frame in e/A^2 Available if *should_dose_filter* = 1
pre_exposure_amount	Float	Pre-exposure prior to the first frame in e/A^2 Available if *should_dose_filter* = 1.
set_expert_options	Boolean	Following data can be inserted manually or by approximations.
minumum_shift_in_angstroms	Float	Minimum shift will be limited by this value in Angstroms. Available if *set_expert_options* = 1.
maximum_shift_in_angstroms	Float	Maximum shift will be limited by this value in Angstroms.

		Available if *set_expert_options* = 1
bfactor_in_angstoms	Float	Factor for filtering the reference prior to alignment. Available if *set_expert_options* = 1
vertical_mask_size	Float	Half-width of vertical Fourier mask. Available if *set_expert_options* = 1
horizontal_mask_size	Float	Half-width of horizontal Fourier mask. Available if *set_expert_options* = 1
termination_threshold_in_angstoms	Float	Algorithm will iterate until maximum shift is below this value. Available if *set_expert_options* = 1.
max_iterations	Integer	Algorithm will stop when this is reached, even if *termination_threshold_in_angstoms* is not reached Available if *set_expert_options* = 1
should_restore_power	Boolean	Restore the noise power. Available if *set_expert_options* = 1 && *should_dose_filter* =1
movie_is_gain_corrected	Boolean	Input frames are already gain-corrected.
gain_filename	String	Filename of the gain-correction. Available if *movie_is_gain_corrected* =! 1
first_frame	Integer	First frame to use for sum. Available if *set_expert_options* = 1
last_frame	Integer	Last frame to use for sum. Available if *set_expert_options* = 1
correct_mag_distortion	Boolean	If correct a magnification distortion is needed.
mag_distortion_angle	Float	Distortion angle in degrees. Available if *correct_mag_distortion* = 1
mag_distortion_major_scale	Float	Major axis scale factor.

		Available if *correct_mag_distortion* = 1
mag_distortion_minor_scale	Float	Minor axis scale factor. Available if *correct_mag_distortion* = 1

2: Find CTF

Find CTF is also applied to each micrograph individually, it is called as many times as micrographs are going to be processed.

Table 2: Parameters needed for find CTF function,

Variable	Type of data	Brief explanation
input_filename	String	Input image name. It is a mrc file.
input_is_a_movie	Boolean	Input is a stack of frames.
number_of_frames_to_average	Integer	Number of frames to average Available if *input_is_a_movie* = 1
output_diagnostic_filename	String	Output image file. Contains Power Spectrum and CTF fit.
pixel_size_of_input_image	Float	Pixel size in Angstroms.
acceleration_voltage	Float	Acceleration voltage in kV.
spherical_aberration	Float	Spherical aberration in mm.
amplitude_contrast	Float	Fraction of amplitude contrast.
box_size	Integer	Size of the amplitude spectrum in pixels.
minimum_resolution	Float	Lowest resolution for fitting CTF in Angstroms.
maximum_resolution	Float	Highest resolution for fitting CTF in Angstroms.
minimum_defocus	Float	Lowest value to search over in Angstroms.
maximum_defocus	Float	Highest value to search over in Angstroms.

defocus_search_step	Float	Step size for defocus search in Angstroms.
astigmatism_is_known	Boolean	Is astigmatism present.
slower_search	Boolean	More exhaustive search against 2D spectra. Available if *astigmatism_is_known* = 1
known_astigmatism	Float	Astigmatism in Angstroms. Available if *astigmatism_is_known* = 1
known_astigmatism_angle	Float	Astigmatism in degrees. Available if *astigmatism_is_known* = 1
slower_search	Boolean	More exhaustive search against 2D spectra. Yes, if a high astigmatism is expected. Available if *astigmatism_is_known* = 0
should_restrain_astigmatism	Boolean	CTF parameter will penalize large astigmatism. Available if *astigmatism_is_known* = 0
astigmatism_tolerance	Float	Expected or tolerated astigmatism. Available if *astigmatism_is_known* = 0 && *should_restrain_astigmatism* = 1
find_additional_phase_shift	Boolean	If input micrograph was recorded using a phase plate with variable phase shift.
minimum_additional_phase_shift	Float	Lower bound of the search for additional phase shift. Available if *find_additional_phase_shift* = 1
maximum_additional_phase_shift	Float	Higher bound of the search for additional phase shift. Available if *find_additional_phase_shift* = 1
additional_phase_shift_search_step	Float	Step size for phase shift search. Available if *find_additional_phase_shift* = 1
give_expert_options	Boolean	Add more options to the processing.
resample_if_pixel_too_small	Boolean	Resample micrographs so we don't have suboptimal fitting. Available if *give_expert_options* = 1
movie_is_gain_corrected	Boolean	If the movie gain corrected or not.

		Available if *give_expert_options* = 1 && *input_is_a_movie* = 1
gain_filename	String	Filename of the gain reference. Available if *give_expert_options* = 1 && *input_is_a_movie* = 1 && *movie_is_gain_corrected* = 0
correct_movie_mag_distortion	Boolean	Magnitude distortion can be specified. Available if *give_expert_options* = 1 && *input_is_a_movie* = 1
movie_mag_distortion_angle	Float	The angle of the distortion. Available if *give_expert_options* = 1 && *input_is_a_movie* = 1 && *correct_movie_mag_distortion* = 1
movie_mag_distortion_major_scale	Float	The scale factor on the major axis. Available if *give_expert_options* = 1 && *input_is_a_movie* = 1 && *correct_movie_mag_distortion* = 1
movie_mag_distortion_minor_scale	Float	The scale factor on the major axis. Available if *give_expert_options* = 1 && *input_is_a_movie* = 1 && *correct_movie_mag_distortion* = 1

3: Find Particles

Find Particles is applied to each micrograph individually. It calls the method each time a micrograph is going to be processed.

Table 3: Parameters needed for Find Particles function

Variable	Type of data	Brief explanation

micrograph_filename	String	Input micrograph. It is a mrc file.
original_micrograph_pixel_size	Float	Micrograph pixel size in Angstroms.
acceleration_voltage_in_keV	Float	Acceleration voltage in kV.
spherical_aberration_in_mm	Float	Spherical aberration in mm.
amplitude_contrast	Float	Amplitude contrast as a percentage over 1.
additional_phase_shift_in_radians	Float	Additional phase shift in radians
defocus_1_in_angstroms	Float	Micrograph defocus 1 in Angstroms.
defocus_2_in_angstroms	Float	Micrograph defocus 2 in Angstroms.
astigmatism_angle_in_degrees	Float	Micrograph astigmatism angle in degrees.
already_have_templates	Boolean	Supply or not templates.
templates_filename	Float	Templates filename. Available if *already_have_templates* = 1
average_templates_radially	Boolean	Should the templates be radially averaged? Available if *already_have_templates* = 1
rotate_templates	Boolean	Search for rotated versions of the templates. Available if *already_have_templates* = 1 && *average_templates_radially* = 0
number_of_template_rotations	Integer	Number of in-plane rotations. Available if *already_have_templates* = 1 && *average_templates_radially* = 0 && *rotate_templates* = 0
typical_radius_in_angstroms	Float	Estimate of the average radius.
maximum_radius_in_angstroms	Float	Maximum radius of the templates in angstroms.
highest_resolution_to_use	Float	Highest resolution for picking in angstroms.

output_stack_filename	String	Filename for output stack of candidate particles.
output_stack_box_size	Integer	Box size for candidate particles in pixels.
minimum_distance_from_edges_in_pixels	Integer	Minimum distance from edge in pixels.
minimum_peak_height_for_candidate_particles	Float	Minimum peak height for candidate particles.
avoid_high_variance_areas	Boolean	Avoid areas with abnormal high local variance.
avoid_high_low_mean_areas	Boolean	Avoid areas with abnormal local mean.
algorithm_to_find_background	Integer	Find background areas.
number_of_background_boxes	Integer	Number of boxed to be extracted from the micrographs.
particles_are_white	Boolean	Particles are white on a dark background.

4: 2D Classify

Align movies is applied to each micrograph individually, it is called as many times as micrographs are going to be processed.

Table 4: Parameters needed for 2D classify function

Variable	**Type of data**	**Brief explanation**
input_particle_images	String	Input image stack with the particles. It is a mrc file.
input_parameter_file	String	Input parameter file with alignment parameters. It is a par file.
input_class_averages	String	2D references with the best estimates. It is a mrc file.
ouput_parameter_file	String	Output parameter file, to be used in the next cycle. It is a par file.

ouput_class_averages	String	Output 2D references with the best estimates, to be used in the next cycle. It is a mrc file.
number_of_classes	Integer	Number of classes to be refined. If 0, the number is determined by the stack of input averages.
first_particle	Integer	Number of the first particle to be processed.
last_particle	Integer	Number of the last particle to be processed.
percent_used	Float	Percentage of the particles chosen to be processed. Percentage out of 1.
pixel_size	Float	Pixel size of the image in Angstroms.
voltage_kV	Float	Energy of the electron beam in kV.
spherical_aberration_mm	Float	Spherical aberration of the lens in mm.
amplitude_contrast	Float	Assumed amplitude contrast.
mask_radius	Float	Radius of a circular mask applied to input class averages in Angstroms.
low_resolution_limit	Float	Low resolution limit for data in Angstroms.
high_resolution_limit	Float	High resolution limit for data in Angstroms.
angular_step	Float	Angular step size for global grid search.
max_search_x	Float	Maximum global peak search distance along X
max_search_y	Float	Maximum global peak search distance along Y
smoothing_factor	Float	Factor for likelihood-weighting
padding_factor	Integer	Factor for improving the interpolation for image rotation
normalize_particles	Boolean	Input images must be normalized unless they are pre-processed
invert_contrast	Boolean	Particle images contrast should be inverted or not
exclude_blank_edges	Boolean	Particles with blank edges should be excluded from processing or not
dump_arrays	Boolean	Intermediate 2D sums will be dumped to a temporal file for later merging
dump_file	String	Name of the dump file with intermediate 2D sums

Table 5: Parameters needed for merge2d function

Variable	Type of data	Brief explanation
ouput_class_averages	String	Output class averages. It is a mrc file.
dump_file_seed	String	Seed for the dump files with intermediate 2D class sums. It is a dat file.
number_of_dump_files	Integer	Number of dump files to be merged

8: Sharpen 3D

Align movies is applied to each micrograph individually, it is called as many times as micrographs are going to be processed.

Table 6: Parameters needed for Sharpen 3D function

Variable	Type of data	Brief explanation
input_volume	String	Input volume file name
output_volume	String	Output volume file name
input_mask	String	Mask file name.
res_statistics	String	Input reconstruction statistics.
use_statistics	Boolean	Use statistics or not.
pixel_size	Float	Pixel size in Anstroms.
inner_mask_radius	Float	Inner radius of mask to be applied.
outer_mask_radius	Float	Outer radius of mask to be applied.
bfactor_low	Float	B-factor to be applied to the non-flattened part.
bfactor_high	Float	B-factor to be applied to the flattened part.
bfactor_res_limit	Float	Resolution at which spectral flattening starts.

resolution_limit	Float	Resolution of low-pass filter in Angstroms.
filter_edge	Float	Cosine edge used with low-pass filter in Angstroms.
fudge_SSNR	Float	Scale the Part_SSNR.
use_mask	Boolean	Should a 3D mask be used before sharpening.
invert_hand	Boolean	Should the map headedness be inverted.

3.2 Automatization of CisTEM's workflow

By modifying the code and doing all the work presented in Section 3.1, *cisTEMpipeline.txt* and *checker.txt* have all the necessary information needed for the recreation of a full process made by CisTEM.

We can see in Figure 23 the structure that was done and the one that we followed.

- At first, each micrograph was processed once by unblur.cpp
- Each micrograph is processed by ctffind.cpp
- All micrographs are processed by findparticles.cpp
- 2D classify will separate images in four batches, compute each batch separately and merge those four batches with merge2d. This process will be done for the number of cycles the user has chosen.
- Ab initio also separates images in four batches, computes reconstruct3d for each one, then refine3d and finally merge3d of the four batches. This process goes on until a maximum number of cycles or a threshold is reached.
- Autorefine and generate3d then processes and computes making use of libraries.
- Sharpen3d calls the method with the same name by taking a volume as the input.

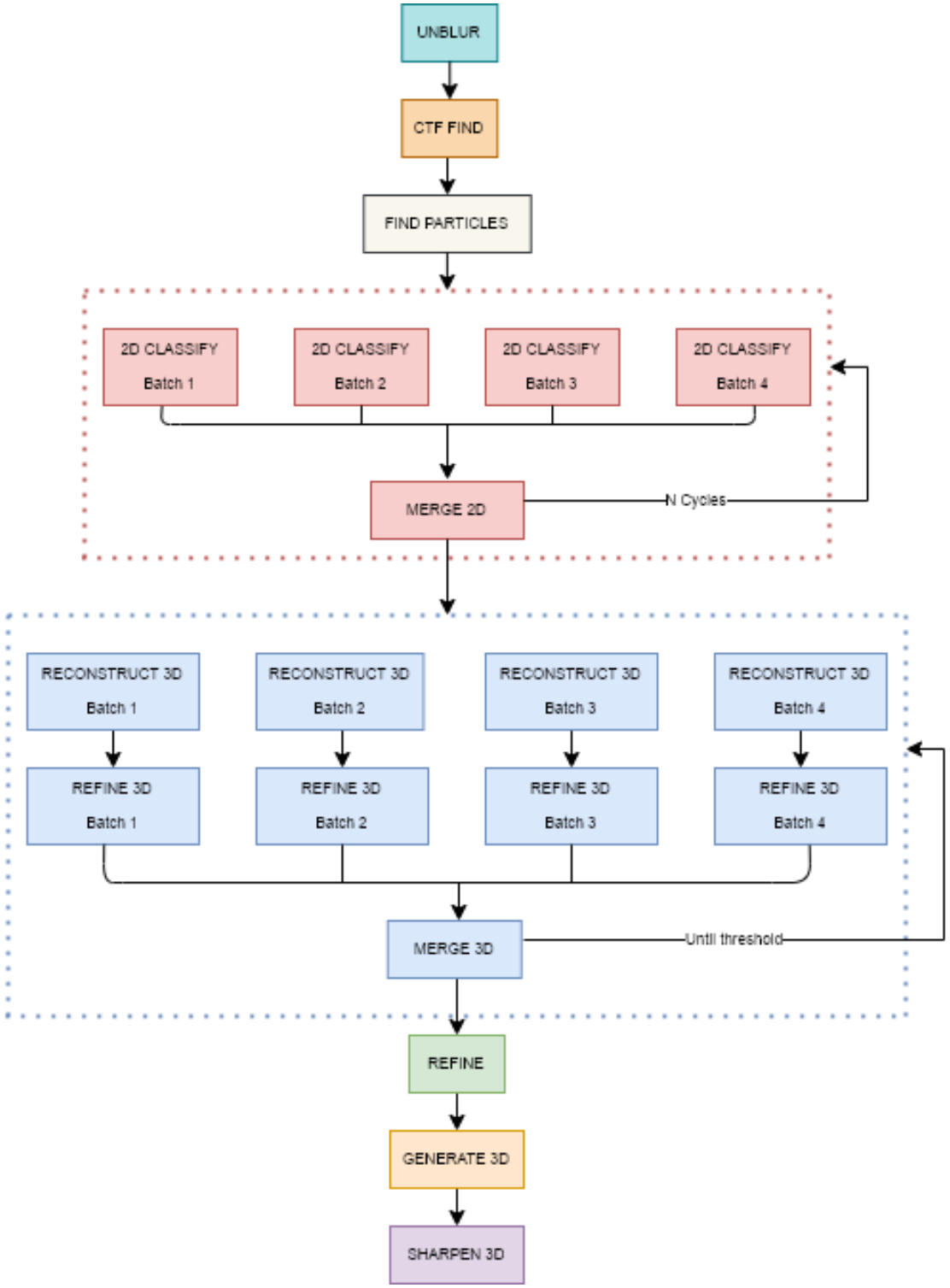


Figure 23: CisTEM's workflow.

The main problem when using CisTEM manually, is the tedious work when calling all methods and having to input parameters one by one. Some of the parameters may be the

same as the ones used in the previous cycle, but some may change because of the processing.

We can observe in Figure 24 and Figure 25 the code for one file. This file is the one in charge of finding particles in the micrograph. Figure 24 show us the piece of code that iterates given a directory all the micrographs present. This will call the method presented in Figure 25, which creates the arguments text file with all the parameters needed for processing. Notice that some of the parameters needed have been computed in a previous step (such as defocus 1, defocus 2 and astigmatism, which were computed in CTF find step). Finally, going back to Figure 24, the script will be executed from command line with the pertinent arguments and storing all the output into a log file for optional further analysis.

```
for movie_name in os.listdir(movie_path):
        #print(os.path.splitext(movie_name[]))
        #if (os.path.splitext(movie_name)[--1] == 'mrc'):
        if movie_name.endswith('.mrc'):
                if os.path.isfile(os.path.join(movie_path,movie_name)):
                        create_findparticles_arg(os.path.join(movie_path,movie_name),movie_name,arg_path,data_file,ctf_path)
                        with open("%sarg/%s_arg.txt" %(arg_path,movie_name), "r") as file:
                                values = file.readlines()
                                values_1 = (''.join(map(str,values)))

                        command = './find_particles << eot >> findparticles_apoferritin.log 2>&1 \n%seot\n' %(values_1)
                        os.system(command)
```

Figure 24: Script that iterates for each micrograph in the desired file. This part of the code mainly reads the argument file and throws the method with the arguments previously stored.

```
#We must take all data from a txt file in the CORRECT order and pass it to this method
def create_findparticles_arg(movie_path, movie_name, arg_path,data_file,ctf_path):
        a = []
        f = open("%sarg/%s_arg.txt" %(arg_path,movie_name), "w")
        f2 =open(data_file, "r")
        line = []
        i = 0

        #for name in os.listdir(movie_path):
                        #if os.path.isfile(os.path.join(movie_path,name)):
        ctf_data_path = ("%s%s_CTF_0.txt") %(ctf_path,os.path.splitext(movie_name)[0])
        df_txt = pd.read_csv(ctf_data_path,header = None)
        a = df_txt[0][5]
        ctf_data = a.split(" ")


        #defocus1
        #defocus2
        #astigmatism_angle
        #Let's try to make this faster and with less variables
        f.write("%s\n" %movie_path)

        for line in f2:


                #data from 6, 7 and 9th line come from ctf, we must go into the argument txt and obtain the columns 1, 2 and 3 respectively
                #data from 13nd line will be the same as in the template
                #print(read_data)
                read_data = f2.readlines()
                print(df_txt)
                read_data[5] = ctf_data [1] + "\n" #defocus 1
                read_data[6] = ctf_data [2] + "\n" #decofus 2
                read_data[7] = ctf_data [3] + "\n" #astigmatism
                read_data[12] = "%s%s_COOS_0%s\n" %(arg_path,os.path.splitext(movie_name)[0],os.path.splitext(movie_name)[1])
                data = (''.join(map(str,read_data)))
                f.write(data)
```

Figure 25: Script that creates the argument file that will be used in the next iteration**.**

3.3

3.3 Wrapper

Let us present our wrapper method. We can observe the structure in Figure 26.

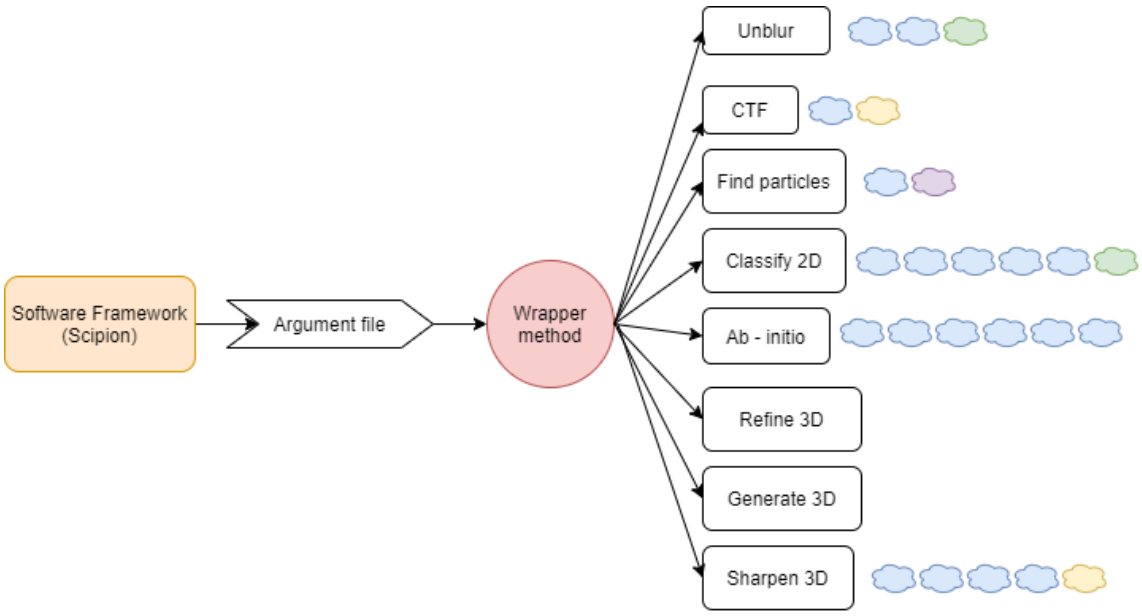


Figure 26: Schematic of the wrapper. The different cloud represents the data that come from the argument file. Blue clouds are Strings, green clouds are Integer, yellow clod are Booleans and purple clouds are Floats.

Following the fashion presented in Figure 26, the wrapper method was done making a simple *if* concatenation, the method reads the firsts variables from the argument file and decides which process it should start.

- String followed by a Boolean, CTF will be processed.
- String followed by a Float, find particles will be processed.
- Two Strings followed by an Integer, Unblur will be processed.
- Four Strings followed by a Boolean, Sharpen3D will be processed.
- Five Strings followed by an Integer, Classify 2D will be processed.
- Six Strings, Ab – initio will be processed.

The other two methods, Refine 3D and Generate 3D were not implemented this way since they use a different method.

4 RESULTS AND DISCUSSION

The achievement of those objectives is assessed in the next sections. Two examples employing the workflow created are thoroughly presented below.

Section 4.1 assesses the correct functioning of the workflow by comparing the results obtained to CisTEM's GUI. Some testing was done on other datasets such as beta-galactosidase (EMPIAR-10012) or Urease which presented satisfactory results presented by C-CINA (https://www.c-cina.org/fileadmin/files/events/Workshops/2018-cryo-EM/NCCRTransCure-CryoEMWorkshop-2018-08-23-5-Righetto-cisTEM_tutorial3.pdf). Although not all the parameters where known and processing was not satisfactory, so it was not included in this paper. Please refer to the attached code if visualization of all the results were necessary.

4.1 Apoferritin tutorial dataset for CisTEM (EMPIAR-10146)

The data in this example comes from the Electron Microscopy Public Image Archive (EMPIAR), which is publicly available at their webpage with the code 10146.

Data is presented in MRC format micrographs. 20 images are presented, with 50 frames per image. The pixel size is 32 bit-float with a 1.5 Angstroms of spacing. Further information such as beam energy (300 keV), spherical aberration (0.0), exposure rate (2 e/A^2), apoferritin molecular mass (440 kDa) and symmetry (O) is also specified.

Figures presented are extracted using either Chimera Visualization tool or CisTEM results window. Unfortunately, not all steps will be presented with visual information, since manual processing cannot be shown through CisTEM user interface and certain files were not supported by our chosen application.

4.1.1 Align Movies

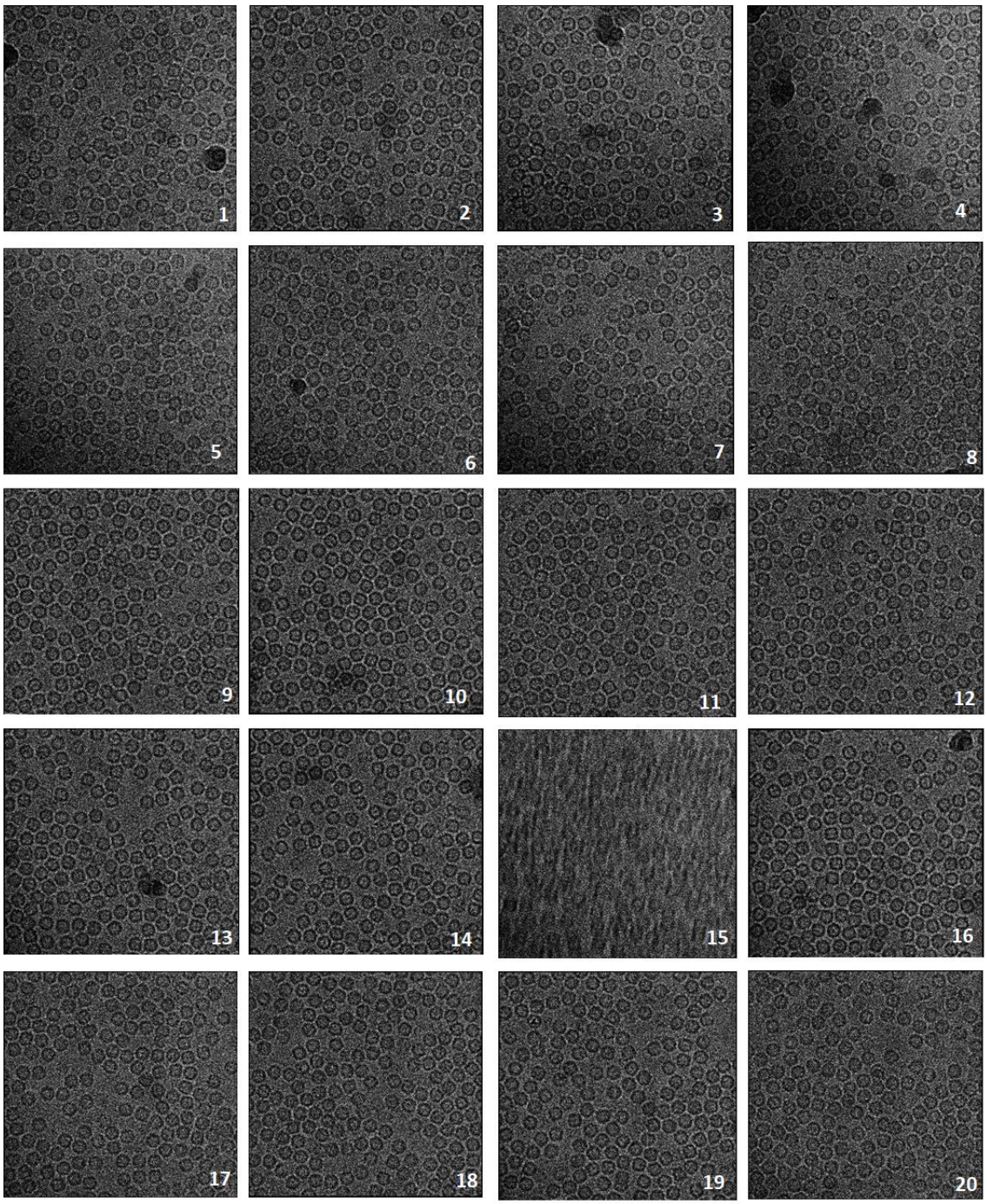


Figure 27: Ublur process on CisTEM user interface. We can observe how image have been processed. Micrograph 15 presents high blurring and will affect considerably our results, so it may be removed in the future steps. This figure was done by extraction each micrograph thanks to Chimera visualization tool.

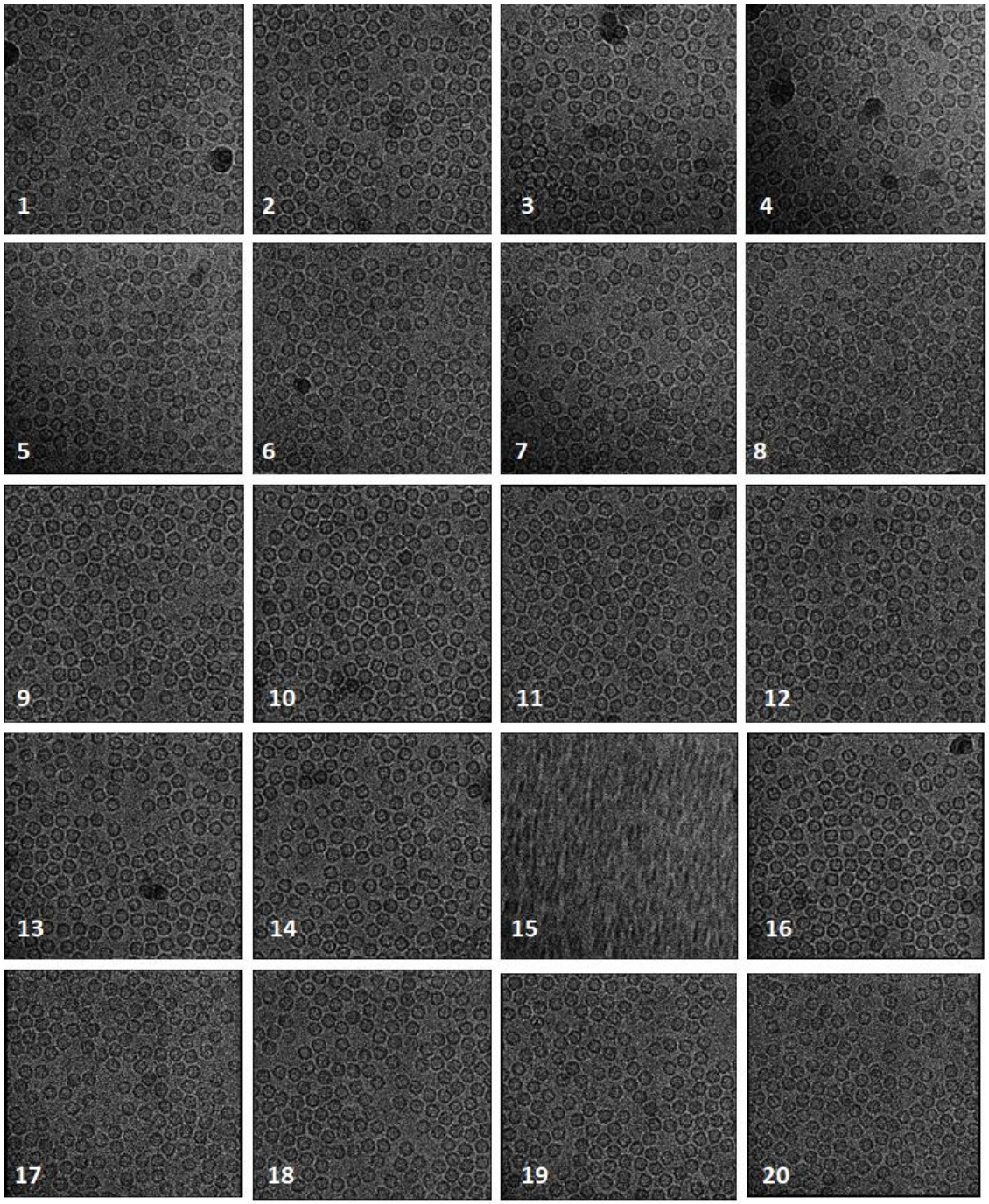


Figure 28: Ublur process on CisTEM manual pipeline. As well as in figure X we can observe how image have been processed and micrograph 15 as well presents high blurring. This figure was done by extraction each micrograph thanks to Chimera visualization tool.

4.1.2 Find CTF

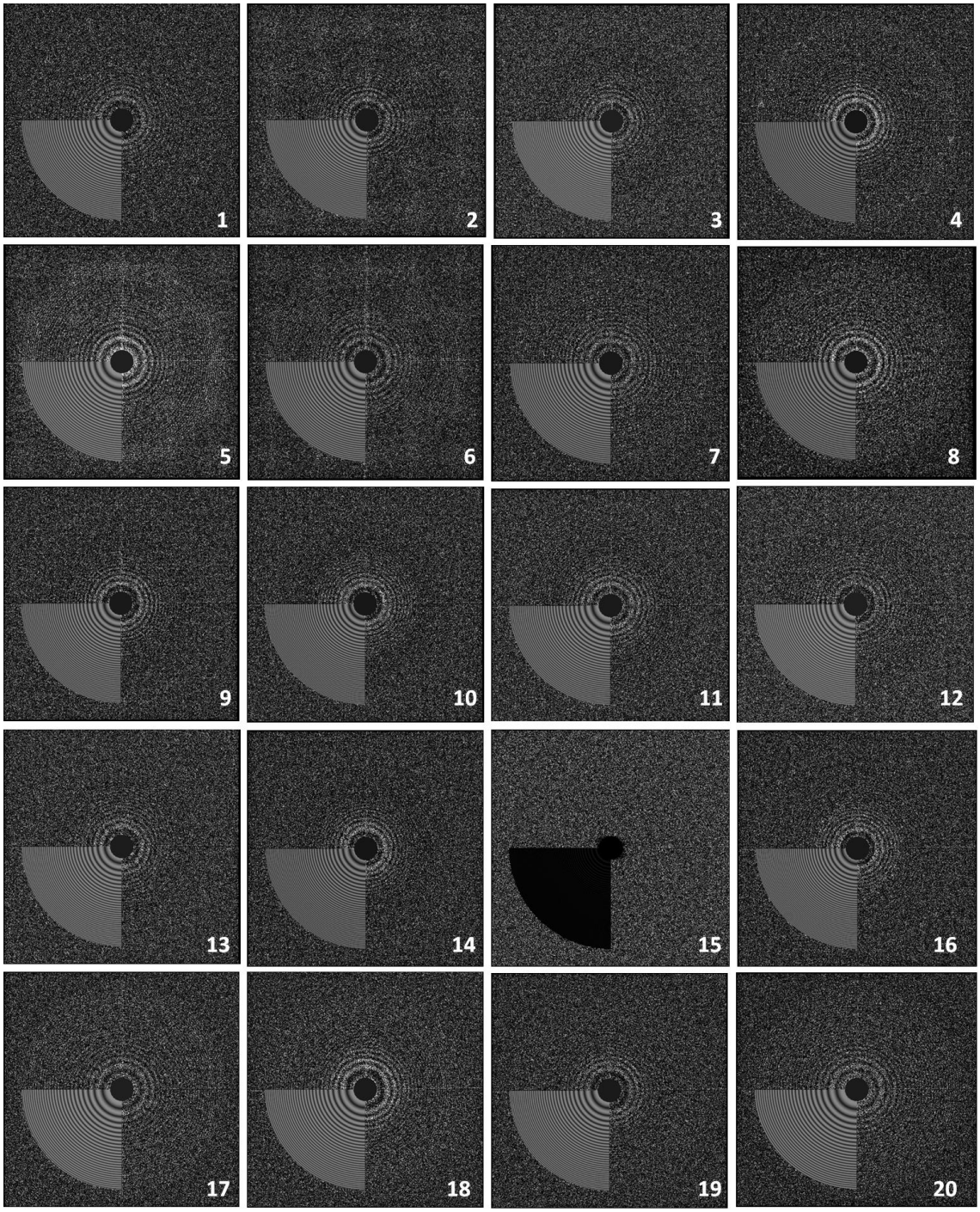


Figure 29: CTF process on CisTEM user interface. We can observe how image have been processed. As expected from the last step, micrograph 15 also presents some information in the Fourier space that will affect considerably our results. It was deleted after this step. This figure was done by extraction each micrograph thanks to Chimera visualization tool.

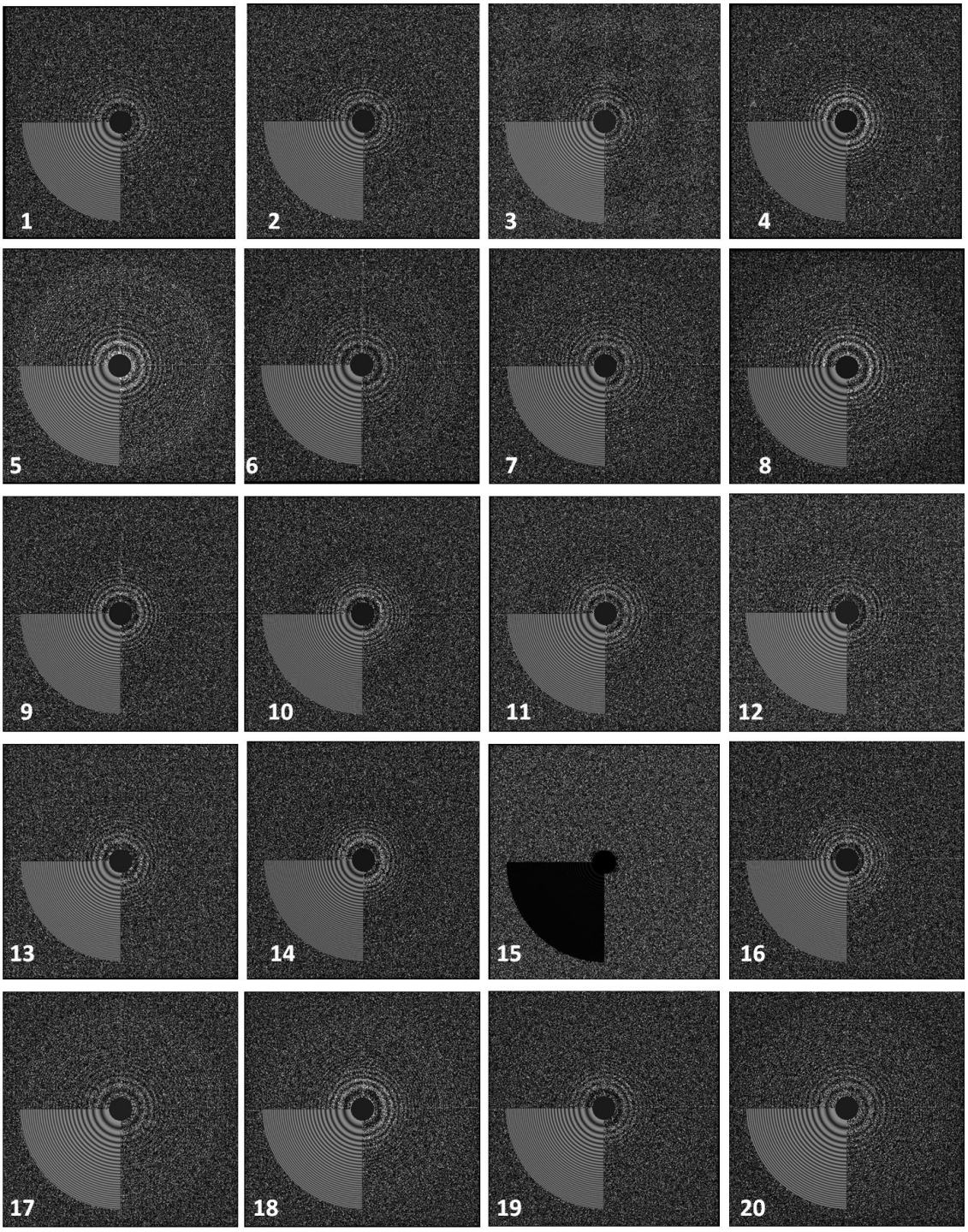


Figure 30: CTF process on CisTEM manual pipeline. Confirming our information from figure X, micrograph 15 does not present beneficial information in the Fourier space. It was deleted after this step. This figure was done by extraction each micrograph thanks to Chimera visualization tool.

4.1.3 Find Particles

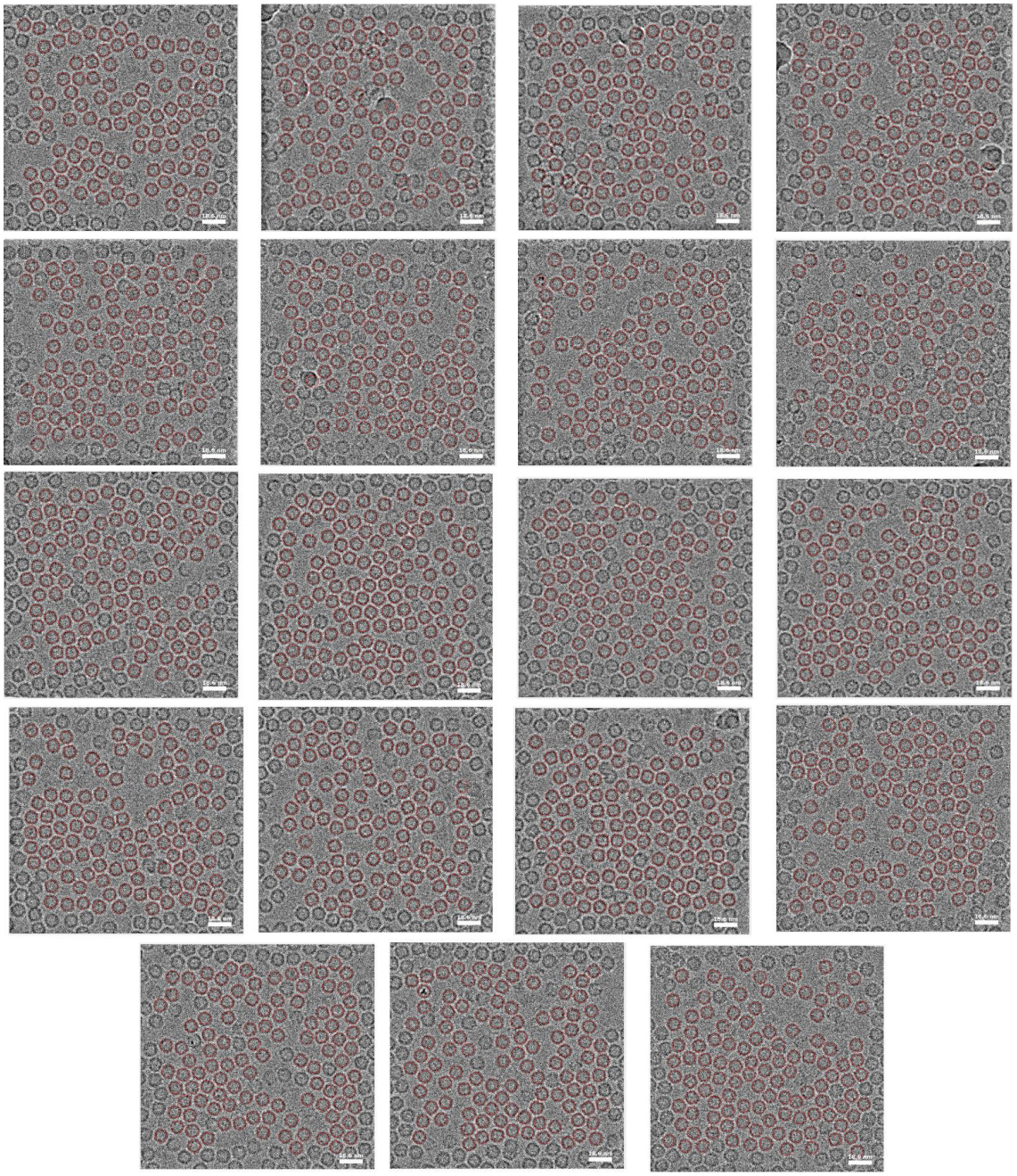


Figure 31: Find particles process on CisTEM user interface. CisTEM stores particles location in the database, thus making it hard to extract and print them from the pipeline. Micrograph 15 was removed from the batch because of bad results. This figure was obtained with cisTEM visualization tool.

4.1.4 2D Classify

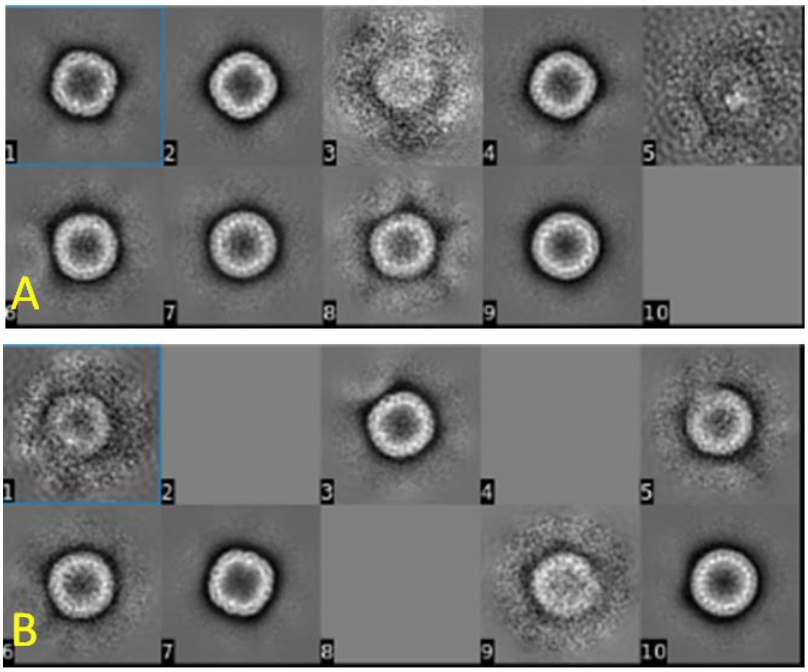


Figure 32: 2D classification comparison. Image A is the classification obtained from CisTEM user interface. Image A is the classification obtained from CisTEM manual pipeline. This figure was obtained with cisTEM visualization tool.

4.1.5 Ab – initio 3D

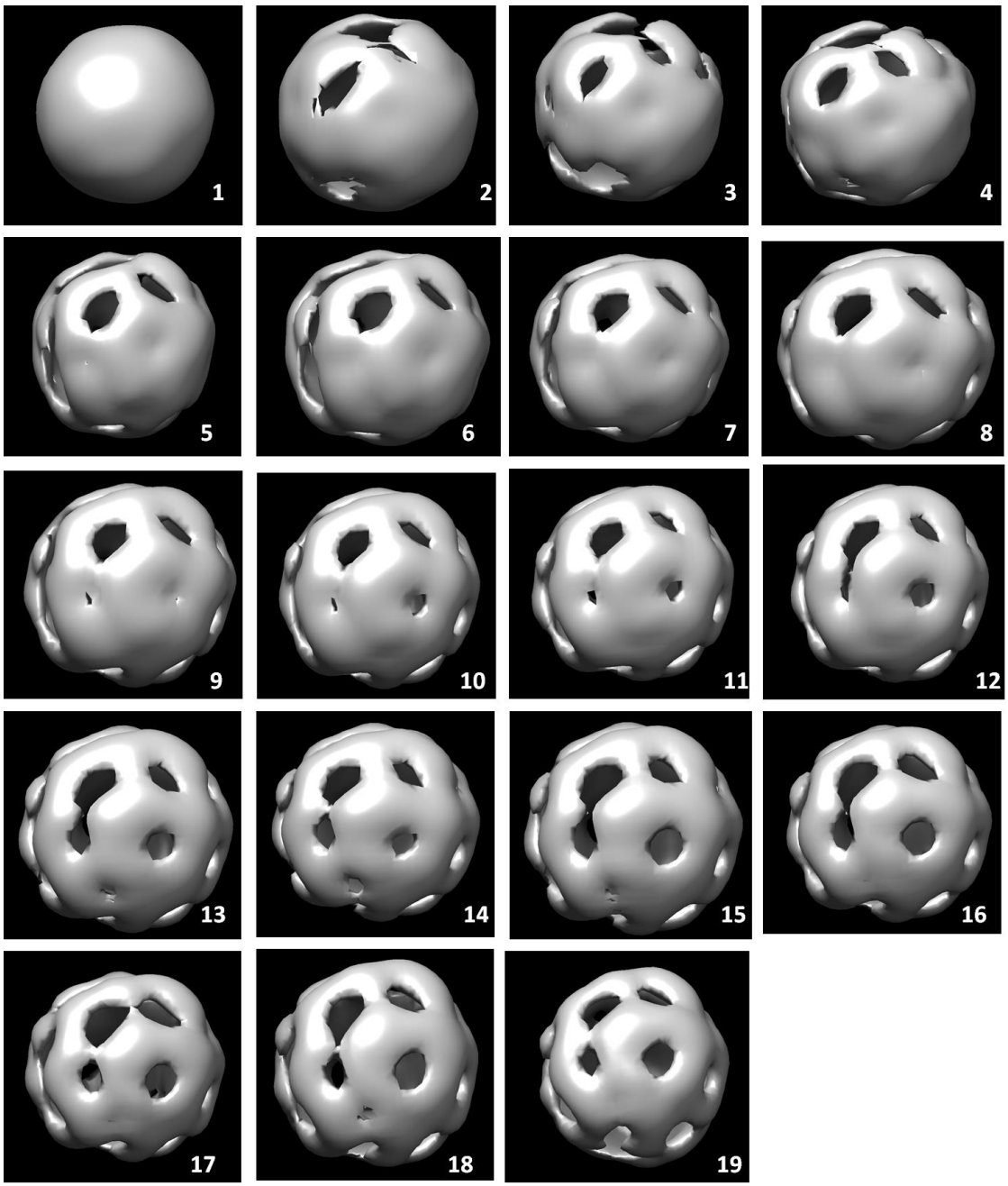


Figure 33: Ab-initio 3D process on CisTEM manual pipeline. This figure was done by extraction each micrograph thanks to Chimera visualization tool.

4.1.6 3D refinement

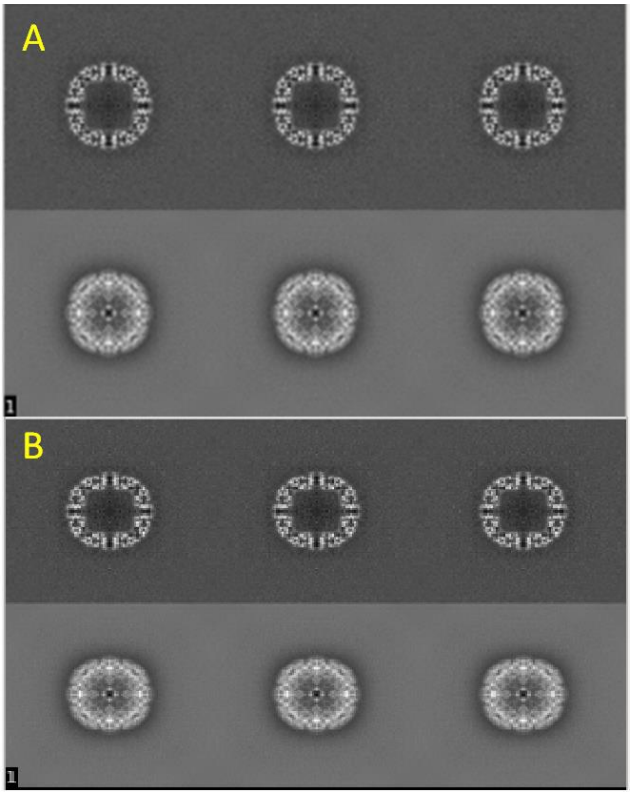


Figure 34: 3D Automatic refinement comparison. Image A is the refinement obtained from CisTEM user interface. Image A is the refinement obtained from CisTEM manual pipeline. This figure was obtained with Chimera visualization tool.

4.1.7 Generate 3D

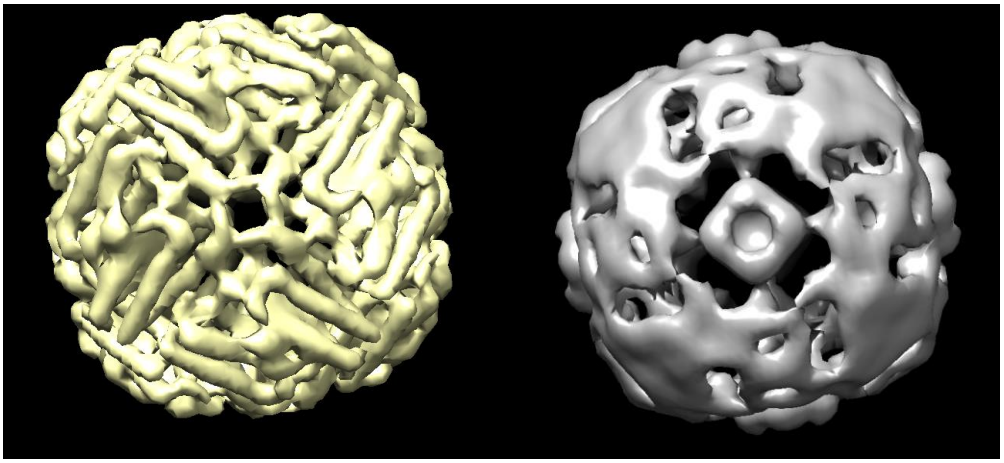
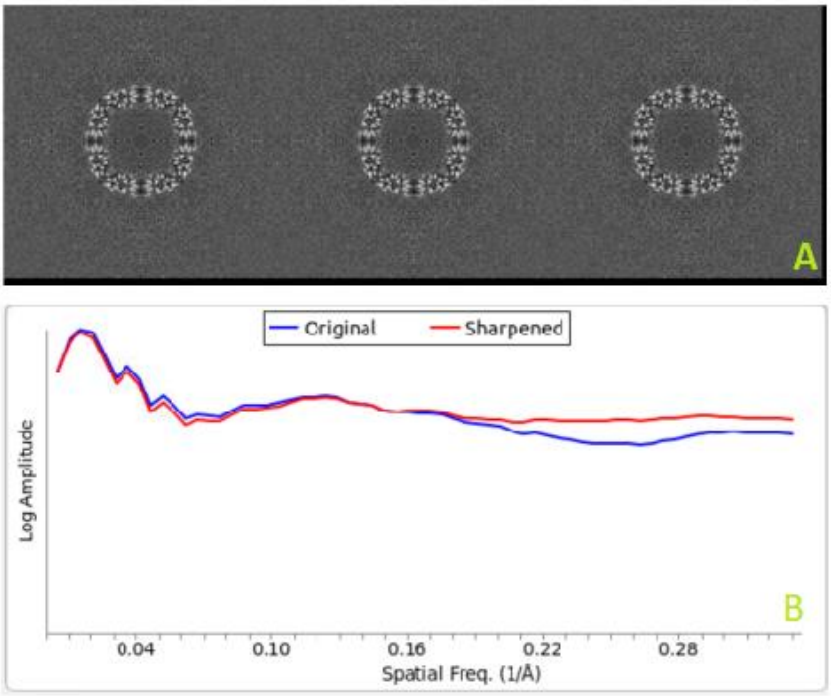


Figure 35 Generation of 3D structure with 2 startup volumes. Both representations come are obtained from CisTEM manual pipeline. This figure was obtained with Chimera visualization tool.

4.1.8 Sharpen 3D



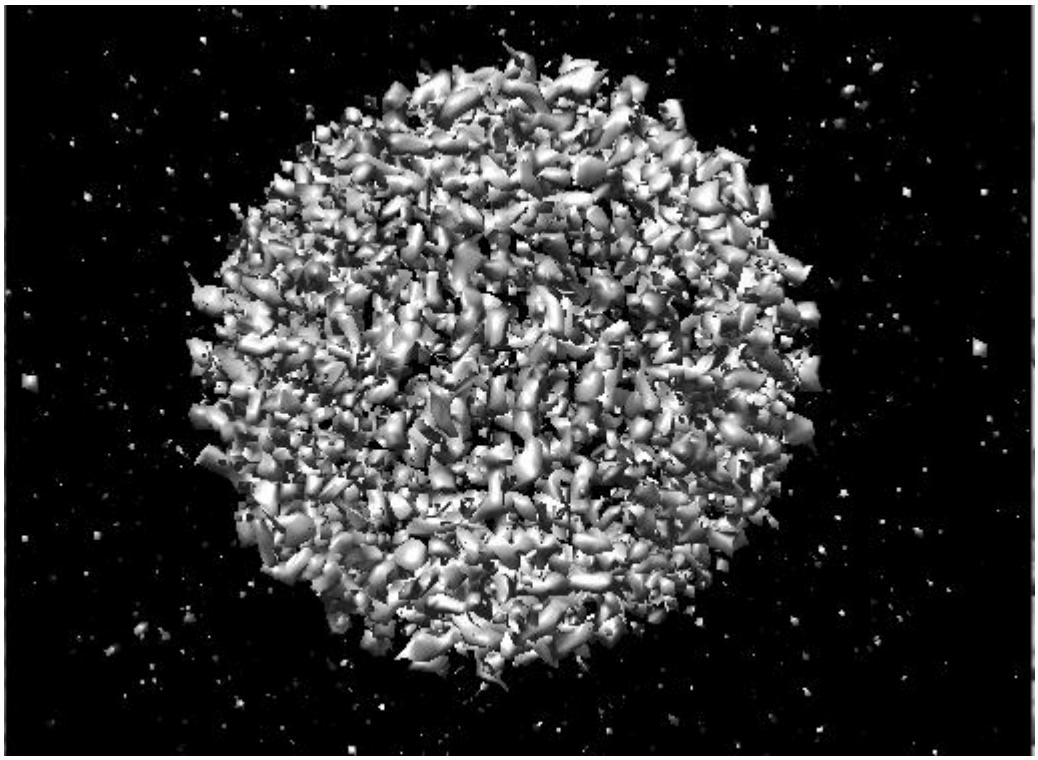


Figure 36. Reconstructed Apoferritin. Representation was obtained from CisTEM manual pipeline. This figure was obtained with Chimera visualization tool.

5 CONCLUSIONS

This Bachelor's Thesis has created a wrapper and automatized the process of manual processing using CisTEM image processing package.

We can observe in the previous chapter that functionalities work correctly, and results obtained from the GUI processing are almost the same as the ones obtained through the scripts recreating that process.

Finally, all developed code has been commented and documented for future use by researchers of students who want to continue this line of research.

5.1 Limitation of the current work

Main limitations of this work are related with the data loaded. Each processing step needs a lot of input parameters, which must be stored in a text file in the correct order. Also, not all the required parameters are available for the researcher unless they personally ask the laboratory where the samples have been taken. This makes the reconstruction process almost impossible if the researcher needs to constantly test 30 different parameters to obtain satisfactory results.

Another important limitation was the selection of "good" and "bad" data. The processing and testing of this project was done with two known datasets with no micrographs to remove due to lack of quality. If an unknown dataset is to be analyzed, the user must revise manually all micrographs in each step for verification since "bad" data must be removed by hand.

5.2 Future work

This Bachelor's Thesis has contributed towards the integration of image processing tools into a software framework. Let us take notice that although this wrapper method has been

developed and successfully works locally, it has not yet been implemented into the desired software framework, some revisions must be done, in which it is included:

- First of all, file paths must be improved. Scripts need to receive the absolute path for all the arguments location. This makes the command too long for manual insert in command line.
- Parameters from some processes are obtained from previous steps. This creates a dependency between processes which makes hard the interconnection of processes from different sources.
- Integration of the software package to Scipion

6 REFERENCES

[1] Abrishami, S., Naghibzadeh, M., & Epema, D. H. (2013). Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Computer Systems*, *29*(1), 158-169.

[2] Aebi, U., Carragher, B., & Smith, P. R. (1996). Advances in computational image processing for microscopy. *Journal of Structural Biology*, *116*(1), 1-1.

[3] Al-Azzawi, A., Ouadou, A., Tanner, J. J., & Cheng, J. (2019). DeepCryoPicker: fully automated deep neural network for single protein particle picking in cryo-EM. *bioRxiv*, 763839.

[4] Amos, L. A., & Klug, A. (1974). Arrangement of subunits in flagellar microtubules. *Journal of cell science*, *14*(3), 523-549.

[5] Andersen, A. H., & Kak, A. C. (1984). Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm. *Ultrasonic imaging*, *6*(1), 81-94.

[6] Bai, X. C., McMullan, G., & Scheres, S. H. (2015). How cryo-EM is revolutionizing structural biology. *Trends in biochemical sciences*, *40*(1), 49-57.

[7] Baker, L. A., & Rubinstein, J. L. (2010). Radiation damage in electron cryomicroscopy. In *Methods in enzymology* (Vol. 481, pp. 371-388). Academic Press.

[8] Baxter, W. T., Leith, A., & Frank, J. (2007). SPIRE: the SPIDER reconstruction engine. *Journal of structural biology*, *157*(1), 56-63.

[9] Bhamre, T., Zhang, T., & Singer, A. (2016). Denoising and covariance estimation of single particle cryo-EM images. *Journal of structural biology*, *195*(1), 72-81.

[10] Brilot, A. F., Chen, J. Z., Cheng, A., Pan, J., Harrison, S. C., Potter, C. S., ... & Grigorieff, N. (2012). Beam-induced motion of vitrified specimen on holey carbon film. *Journal of structural biology*, *177*(3), 630-637.

[11] Campbell, G. S., & Norman, J. (2012). *An introduction to environmental biophysics*. Springer Science & Business Media.

[12] 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. *Journal of microscopy*, *157*(2), 187-203.

[13] Censor, Y., Elfving, T., & Herman, G. T. (2001). Averaging strings of sequential iterations for convex feasibility problems. In *Studies in Computational Mathematics* (Vol. 8, pp. 101-113). Elsevier.

[14] Censor, Y., Gordon, D., & Gordon, R. (2001). Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems. *Parallel computing*, *27*(6), 777-808.

[15] Chen, J. Z., Settembre, E. C., Aoki, S. T., Zhang, X., Bellamy, A. R., Dormitzer, P. R., ... & Grigorieff, N. (2009). Molecular interactions in rotavirus assembly and uncoating seen by high-resolution cryo-EM. *Proceedings of the National Academy of Sciences*, *106*(26), 10644-10648.

[16] Cheng, A., Henderson, R., Mastronarde, D., Ludtke, S. J., Schoenmakers, R. H., Short, J., ... & Winn, M. (2015). MRC2014: Extensions to the MRC format header for electron cryo-microscopy and tomography. *Journal of structural biology*, *192*(2), 146-150

[17] Collaborative, C. P. (1994). The CCP4 suite: programs for protein crystallography. *Acta crystallographica. Section D, Biological crystallography*, *50*(Pt 5), 760.

[18] Cong, Y., & Ludtke, S. J. (2010). Single particle analysis at high resolution. In *Methods in enzymology* (Vol. 482, pp. 211-235). Academic Press.

[19] Conway, J. F., Cheng, N., Zlotnick, A., Wingfield, P. T., Stahl, S. J., & Steven, A. C. (1997). Visualization of a 4-helix bundle in the hepatitis B virus capsid by cryo-electron microscopy. *Nature*, *386*(6620), 91-94.

[20] Crowther, R. A., DeRosier, D. J., & Klug, A. (1970). The reconstruction of a three-dimensional structure from projections and its application to electron microscopy. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, *317*(1530), 319-340.

[21] Crowther, R. A., Henderson, R., & Smith, J. M. (1996). MRC image processing programs. *Journal of structural biology*, *116*(1), 9-16.

[22] De Alarcón, P. A., Pascual-Montano, A. D., & Carazo, J. M. (2002, July). Spin images and neural networks for efficient content-based retrieval in 3D object databases. In *International Conference on Image and Video Retrieval* (pp. 225-234). Springer, Berlin, Heidelberg

[23] De la Rosa-Trevín, J. M., Otón, J., Marabini, R., Zaldivar, A., Vargas, J., Carazo, J. M., & Sorzano, C. O. S. (2013). Xmipp 3.0: an improved software suite for image processing in electron microscopy. *Journal of structural biology*, *184*(2), 321-328.

[24] Díez, D. C., Mueller, H., & Frangakis, A. S. (2007). Implementation and performance evaluation of reconstruction algorithms on graphics processors. *Journal of Structural Biology*, *157*(1), 288-295.

[25] Dorner, W. C. (1930). The negative staining of bacteria. *Stain Technology*, *5*(1), 25-27.

[26] Dubochet, J., Adrian, M., Lepault, J., & McDowall, A. W. (1985). Emerging techniques: Cryo-electron microscopy of vitrified biological specimens. *Trends in Biochemical Sciences*, *10*(4), 143-146.

[27] Elmlund, D., & Elmlund, H. (2012). SIMPLE: Software for ab initio reconstruction of heterogeneous single-particles. *Journal of structural biology*, *180*(3), 420-427.

[28] Frank, J., Shimkin, B., & Dowse, H. (1981). SPIDER—a modular software system for electron image processing. *Ultramicroscopy*, *6*(4), 343-357.

[29] 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.

[30] Grant, T., Rohou, A., & Grigorieff, N. (2018). cisTEM, user-friendly software for single-particle image processing. *elife*, *7*, e35383.

[31] Grigorieff, N. (2007). FREALIGN: high-resolution refinement of single particle structures. *Journal of structural biology*, *157*(1), 117-125.

[32] Grigorieff, N. (2016). Frealign: an exploratory tool for single-particle cryo-EM. In *Methods in enzymology* (Vol. 579, pp. 191-226). Academic Press.

[33] Heimowitz, A., Andén, J., & Singer, A. (2018). Apple picker: Automatic particle picking, a low-effort cryo-EM framework. *Journal of structural biology*, *204*(2), 215-227.

[34] Hegerl, R. (1996). The EM program package: a platform for image processing in biological electron microscopy. *Journal of structural biology*, *116*(1), 30-34.

[35] Hegerl, R., & Altbauer, A. (1982). The "EM" program system. *Ultramicroscopy*, *9*(1-2), 109-115.

[36] Henderson, R., & Unwin, P. N. T. (1975). Three-dimensional model of purple membrane obtained by electron microscopy. *Nature*, *257*(5521), 28-32.

[37] Henn, C., Teschner, M., Engel, A., & Aebi, U. (1996). Real-time isocontouring and texture mapping meet new challenges in interactive molecular graphics applications. *Journal of structural biology*, *116*(1), 86-92.

[38] Herman, G. T., & Levkowitz, H. (1988). Initial performance of block-iterative reconstruction algorithms. In *Mathematics and computer science in medical imaging* (pp. 305-317). Springer, Berlin, Heidelberg

[39] Hohn, M., Tang, G., Goodyear, G., Baldwin, P. R., Huang, Z., Penczek, P. A., ... & Ludtke, S. J. (2007). SPARX, a new environment for Cryo-EM image processing. *Journal of structural biology*, *157*(1), 47-55.

[40] Katsevich, E., Katsevich, A., & Singer, A. (2015). Covariance matrix estimation for the cryo-EM heterogeneity problem. *SIAM journal on imaging sciences*, *8*(1), 126-185.

[41] Klug, A., & De Rosier, D. J. (1966). Optical filtering of electron micrographs: reconstruction of one-sided images. *Nature*, *212*(5057), 29-32.

[42] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics*, *43*(1), 59-69.

[43] Kremer, J. R., Mastronarde, D. N., & McIntosh, J. R. (1996). Computer visualization of three-dimensional image data using IMOD. *Journal of structural biology*, *116*(1), 71-76.

[44] Lander, G. C., Stagg, S. M., Voss, N. R., Cheng, A., Fellmann, D., Pulokas, J., ... & Lyumkis, D. (2009). Appion: an integrated, database-driven pipeline to facilitate EM image processing. *Journal of structural biology*, *166*(1), 95-102.

[45] Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997*.

[46] Liu, H., Jin, L., Koh, S. B. S., Atanasov, I., Schein, S., Wu, L., & Zhou, Z. H. (2010). Atomic structure of human adenovirus by cryo-EM reveals interactions among protein networks. *Science*, *329*(5995), 1038-1043.

[47] Ludtke, S. J., Baldwin, P. R., & Chiu, W. (1999). EMAN: semiautomated software for high-resolution single-particle reconstructions. *Journal of structural biology*, *128*(1), 82-97.

[48] Moscovich, A., Halevi, A., Andén, J., & Singer, A. (2020). Cryo-EM reconstruction of continuous heterogeneity by Laplacian spectral volumes. *Inverse Problems*, *36*(2), 024003.

[49] Pascual, A., Bárcena, M., Merelo, J. J., & Carazo, J. M. (2000). Mapping and fuzzy classification of macromolecular images using self-organizing neural networks. *Ultramicroscopy*, *84*(1-2), 85-99.

[50] Pascual-Montano, A., Donate, L. E., Valle, M., Barcena, M., Pascual-Marqui, R. D., & Carazo, J. M. (2001). A novel neural network technique for analysis and classification of EM single-particle images. *Journal of structural biology*, *133*(2-3), 233-245.

[51] Punjani, A., Rubinstein, J. L., Fleet, D. J., & Brubaker, M. A. (2017). cryoSPARC: algorithms for rapid unsupervised cryo-EM structure determination. *Nature methods*, *14*(3), 290-296.

[52] Rath, B. K., & Frank, J. (2004). Fast automatic particle picking from cryo-electron micrographs using a locally normalized cross-correlation function: a case study. *Journal of structural biology*, *145*(1-2), 84-90.

[53] Ruska, E. (1987). The development of the electron microscope and of electron microscopy. *Bioscience reports*, *7*(8), 607-629.

[54] Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, *100*(5), 401-409.

[55] Scheres, S. H. (2012). RELION: implementation of a Bayesian approach to cryo-EM structure determination. *Journal of structural biology*, *180*(3), 519-530.

[56] Scheres, S. H., Valle, M., Nuñez, R., Sorzano, C. O., Marabini, R., Herman, G. T., & Carazo, J. M. (2005). Maximum-likelihood multi-reference refinement for electron microscopy images. *Journal of molecular biology*, *348*(1), 139-149.

[57] Schroeter, J. P., & Bretaudiere, J. P. (1996). SUPRIM: easily modified image processing software. *Journal of structural biology*, *116*(1), 131-137.

[58] Singer, A., & Shkolnisky, Y. (2019). ASPIRE Algorithms for Single Particle Reconstruction.

[59] Sigworth, F. J. (1998). A maximum-likelihood approach to single-particle image refinement. *Journal of structural biology*, *122*(3), 328-339.

[60] Sigworth, F. J. (2004). Classical detection theory and the cryo-EM particle selection problem. *Journal of structural biology*, *145*(1-2), 111-122.

[61] Smith, R., & Carragher, B. (2008). Software tools for molecular microscopy. *Journal of structural biology*, *163*(3), 224-228.

[62] Smith, P. R., & Gottesman, S. M. (1996). The micrograph data processing program. *Journal of structural biology*, *116*(1), 35-40.

[63] Sorzano, C. O. S., De La Fraga, L. G., Clackdoyle, R., & Carazo, J. M. (2004). Normalizing projection images: a study of image normalizing procedures for single particle three-dimensional electron microscopy. *Ultramicroscopy*, *101*(2-4), 129-138.

[64] Sorzano, C. O. S., Marabini, R., Herman, G. T., Censor, Y., & Carazo, J. M. (2004). Transfer function restoration in 3D electron microscopy via iterative data refinement. *Physics in Medicine & Biology*, *49*(4), 509.

[65] Sorzano, C. O. S., Marabini, R., Velázquez-Muriel, J., Bilbao-Castro, J. R., Scheres, S. H., Carazo, J. M., & Pascual-Montano, A. (2004). XMIPP: a new generation of an open-source image processing package for electron microscopy. *Journal of structural biology*, *148*(2), 194-204.

[66] Sorzano, C. O. S., Jonić, S., El-Bez, C., Carazo, J. M., De Carlo, S., Thévenaz, P., & Unser, M. (2004). A multiresolution approach to orientation assignment in 3D electron microscopy of single particles. *Journal of structural biology*, *146*(3), 381-392.

[67] Suloway, C., Shi, J., Cheng, A., Pulokas, J., Carragher, B., Potter, C. S., ... & Jensen, G. J. (2009). Fully automated, sequential tilt-series acquisition with Leginon. *Journal of structural biology*, *167*(1), 11-18.

[68] Tang, G., Peng, L., Mann, D., Yang, C., Penczek, P. A., Goodyear, G., ... & Ludtke, S. J. (2006). EMAN2: Software for Image Analysis and Single Particle Reconstruction. *Microscopy and Microanalysis*, *12*(S02), 388-389.

[69] 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. *Journal of structural biology*, *157*(1), 38-46.

[70] Taylor, K. A., & Glaeser, R. M. (1976). Electron microscopy of frozen hydrated biological specimens. *Journal of ultrastructure research*, *55*(3), 448-456.

[71] Trus, B. L., Kocsis, E., Conway, J. F., & Steven, A. C. (1996). Digital image processing of electron micrographs: the PIC system-III. *Journal of structural biology*, *116*(1), 61-67.

[72] van Heel, M., & Keegstra, W. (1981). IMAGIC: a fast, flexible and friendly image analysis software system. *Ultramicroscopy*, *7*(2), 113-129

[73] Van Heel, M. (1987). Angular reconstitution: a posteriori assignment of projection directions for 3D reconstruction. *Ultramicroscopy*, *21*(2), 111-123.

[74] van Heel, M., Harauz, G., Orlova, E. V., Schmidt, R., & Schatz, M. (1996). A new generation of the IMAGIC image processing system. *Journal of structural biology*, *116*(1), 17-24.

[75] van Heel, M., Portugal, R., Rohou, A., Linnemayr, C., Bebeacua, C., Schmidt, R., ... & Schatz, M. (2006). Four-dimensional cryo-electron microscopy at quasi-atomic resolution: IMAGIC 4D. *International tables for crystallography*, 624-628.

[76] 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. *Journal of structural biology*, *166*(2), 205-213.

[77] Wagner, T., Merino, F., Stabrin, M., Moriya, T., Antoni, C., Apelbaum, A., ... & Quentin, D. (2019). SPHIRE-crYOLO is a fast and accurate fully automated particle picker for cryo-EM. *Communications biology*, *2*(1), 1-13.

[78] Wagner, T., & Raunser, S. (2020). The evolution of SPHIRE-crYOLO particle picking and its application in automated cryo-EM processing workflows. *Communications Biology*, *3*(1), 1-5.

[79] Wang, F., Gong, H., Liu, G., Li, M., Yan, C., Xia, T., ... & Zeng, J. (2016). DeepPicker: A deep learning approach for fully automated particle picking in cryo-EM. *Journal of structural biology*, *195*(3), 325-336.

[80] Yang, Z., Fang, J., Chittuluru, J., Asturias, F. J., & Penczek, P. A. (2012). Iterative stable alignment and clustering of 2D transmission electron microscope images. *Structure*, *20*(2), 237-247.

[81] Yang, Y. J., & Liu, C. T. (2006). *U.S. Patent No. 7,010,704*. Washington, DC: U.S. Patent and Trademark Office.

[82] Zhao, Z., Shkolnisky, Y., & Singer, A. (2016). Fast steerable principal component analysis. *IEEE transactions on computational imaging*, *2*(1), 1-12.

[83] Zhao, Z., & Singer, A. (2013). Fourier–Bessel rotational invariant eigenimages. *JOSA A*, *30*(5), 871-877.

[84] Zhu, Y., Ouyang, Q., & Mao, Y. (2017). A deep convolutional neural network approach to single-particle recognition in cryo-electron microscopy. *BMC bioinformatics*, *18*(1), 1-10.

[85] Zivanov, J., Nakane, T., Forsberg, B. O., Kimanius, D., Hagen, W. J., Lindahl, E., & Scheres, S. H. (2018). New tools for automated high-resolution cryo-EM structure determination in RELION-3. *Elife*, *7*, e42166.