Universidad Autónoma de Madrid

Escuela Politécnica Superior

Research and Innovation in Computational Intelligence and Interactive Systems

MASTER'S THESIS

IMPROVEMENT OF THE VALIDATION CAPABILITY FOR CRYOET SOI DETECTION WORKFLOWS INSIDE SCIPION

Mikel Iceta Tena
Tutors: Roberto Marabini Ruiz, Carlos Óscar Sorzano Sánchez

September 2023

IMPROVEMENT OF THE VALIDATION CAPABILITY FOR CRYOET SOI DETECTION WORKFLOWS INSIDE SCIPION

Author: Mikel Iceta Tena
Tutors: Roberto Marabini Ruiz, Carlos Óscar Sorzano Sánchez

BioComputing Unit CNB-CSIC
Computer Science and Engineering Department
Escuela Politécnica Superior
Universidad Autónoma de Madrid

September 2023

Acknowledgements

Moving to a new city far from everything you have ever known is difficult. Doing so while studying *and working full time* is even harder. That is why it is necessary to acknowledge the actions of some individuals and groups who have contributed to the realisation of this Master and (in turn) the Thesis you are reading right now.

To my friends for putting up with me and cheering me up: Txu, Leire, Andrea, Bubu, Álvaro, Alba, Oier, Irene, Dani, Carolina, Fede, Giedre, Ben, Dmitry...

To the Bio Computing Unit at the CNB-CSIC for giving me employment and a Thesis to work on, as well as a friendly home: José María Carazo, Carlos Óscar Sorzano, Blanca Benítez - Thank you for your support and trust. Marcos, Ana, Patri, Vilas, JJ Conesa, FJ Chichón, Jorge J., Jorge C. - you have been with me since I started working here, you helped me value my work, prioritise and never forget my goals. To my tutor, Roberto - Thank you for helping me keep my feet on the ground and be realist. And to everyone else who has been bearing with me, for supporting me, for helping me and for giving me space.

To my family for their unconditional support and push towards finishing this Master. To my aunts/uncles Amaya & Juan, Mari & Alberto, to my grandparents Antonia & Kepa, for making it possible for me to come live to Madrid. To my sister Ainhoa and brother Kepa, for the remote support. To my beloved *ama* Pilar and my *aita* Kepa, to whom I owe everything.

To you, reader, for spending time reading this work.

Finally, I would like to dedicate this work specially to Laura. A great team leader and guide. A smart and kind person. Always armed with patience and good advise. A role model and a friend both inside and outside of the walls of the CNB-CSIC.

"Para servirles a Dios y a Vds." Carlos Óscar Sorzano.

Abstract

Abstract — Cryo-electron tomography (CryoET) is an imaging technique designed to produce high-resolution (<1nm) three-dimensional reconstructions after imaging a sample in a series of tilt angles. This technique derived from transmission electron microscopy (TEM) has gained traction in the past few years in the field of macromolecular structure characterisation and analysis due to improvements in acquisition techniques, processing workflows/software and hardware itself. The creation of several pieces of software - each using different approaches for the same problem - provides a varied output for each set of data in all of the steps of the CryoET processing workflow.

Classic science quality standards make it clear as to the need of reproducibility and traceability in the experiments performed for a certain project. Recently, results validation has come into the spotlight. The Scipion framework provides such features in the CryoET field thanks to the *scipion-em-xmipptomo* extension that is being developed inside the BioComputing Unit at the CNB-CSIC. There are several steps of the typical processing workflow that have been covered by a validation protocol, but three-dimensional particle picking still needs to be worked out. Particle picking is the process in which the reconstructed 3D volume is examined, looking for regions of interest (e.g. concrete proteins) and annotating them by their coordinates.

The use of diverse machine learning techniques could be of use in the task of adding a validation layer in Cryo-ET picking. The idea is to generate a consensus between the different programs used to pick particles inside of three-dimensional volumes inside of the cryoET Scipion plugin by using neural networks. This brings several challenges, such as managing data in the varied formats from the different picking programs, as well as the internal data structures and processing of Scipion itself.

This Master's Thesis culminates with the development of a piece of code inside of Scipion 3. This code will allow the import of the result of different structure-of-interest finders (or "pickers") and combine them to feed a neural network, that in turn will provide the final user with enriched results thanks to the differences between each existing processing algorithm. Moreover, an up to now nonexistent capability of validating CryoET workflows at the picking stages is introduced, empowering the obtaining of better quality results.

Key words — CryoET, 3DEM, Scipion, Structural Biology, Validation, Neural Networks

Resumen

Resumen — La crio-tomografía electrónica (CryoET) es una técnica de adquisición diseñada para la generación de estructuras tridimensionales de alta resolución ($<$1nm) mediante la captura de imágenes de una muestra a lo largo de un rango de inclinaciones. Esta técnica derivada de la microscopía de transmisión de electrones (TEM) ha ganado peso en el campo del análisis de estructuras macromoleculares en los últimos años debido a los avances en técnicas de adquisición, software y flujos de procesamiento y el mismo hardware. La creación de distintas piezas de software - cada una utilizando distintas aproximaciones para un mismo problema - deriva en una serie de resultados que difieren entre sí para un mismo conjunto de datos de entrada, ocurriendo esto en cada uno de los pasos del flujo de trabajo típico de CryoET.

Los estándares de calidad clásicos en la ciencia son muy claros en cuanto a que los principios de reproducibilidad y trazabilidad de los experimentos de un proyecto son importantes. Se ha sumado a ellos la validación de los resultados recientemente, tomando el foco de atención. El software de procesamiento Scipion incluye todos estos principios para CryoET gracias a la extensión *scipion-em-xmipptomo* que está siendo desarrollada dentro de la Unidad de BioComputación del CNB-CSIC. Ya hay varios pasos del flujo de trabajo "tipo" que han sido cubiertos por medidas de validación, pero la parte referente a la detección de estructuras de interés tridimensional (llamado 3D "picking") aún necesita ser diseñada y programada. El "picking" es el proceso mediante el cual se examinan los volúmenes 3D reconstruidos en busca de estructuras de interés (p.ej. proteinas concretas) y se anotan en base a sus coordenadas.

El uso de distintas técnicas del campo del aprendizaje máquina pueden ser de utilidad a la hora de añadir una capa de validación en el picking de CryoET. La idea es generar un consenso entre los distintos programas que se utilizan para encontrar estructuras de interés en volúmenes tridimensionales generados por CryoET, todo ello dentro de la extensión de tomografía que se encuentra en el software Scipion y utilizando redes neuronales. Esto requiere atajar varios problemas, como el manejo de la salida en distintos formatos de los programas existentes, así como la estructura interna de los datos y métodos de procesamiento dentro del mismo software Scipion.

En el marco de este Trabajo Fin de Máster se desarrolla un código integrado en Scipion 3 que permitirá la importación de los resultados de distintos buscadores de estructuras de interés dentro de tomogramas (o "pickers") y utilizarlos para alimentar una red neuronal que proveerá al usuario final de unos resultados enriquecidos por las diferencias entre los distintos algoritmos existentes. Gracias a ello, además, se añade una mayor capacidad de validación al flujo de trabajo de CryoET inexistente hasta ahora en esta etapa concreta del proceso, mejorando la calidad de los resultados obtenibles.

Palabras clave — CryoET, 3DEM, Scipion, Biología Estructural, Validación, Redes Neuronales

Contents

List of Tables

List of Figures

1
Introduction

In the past few years, 3D Electron Microscopy (3DEM) has undergone a revolution in terms of instrumentation and methodology [1] [2]. One of the central players in this wide-reaching change is the continuous development of image-processing software, which has allowed very high resolutions to be reached [3]. By solving the structures of interest inside of well-known [4] as well as novel macromolecules [5], 3DEM and its variants have proved to be the way to go in the field of structural biology [6].

Figure 1.1: The Instruct Image Processing Center logo.

The Biocomputing Unit (BCU) @ CNB-CSIC is the group in which this Master's Thesis is going to be developed. The BCU is also the Instruct Image Processing Center (I2PC) (see http://i2pc.es). The I2PC was created with the objective of providing efficient support for European research projects which demand expertise in image processing in electron microscopy. In order to support a large collection of projects, the first tasks of the I2PC were oriented towards the design and implementation of the necessary software tools. Soon, the need for a platform armed with advanced - yet standardised, traceable and reproducible - image processing algorithms and a user-friendly GUI arose. This led to the development of Scipion, a software framework that integrates several 3DEM packages through a workflow-based approach [7] [8] [9].

The process of analysing a tomography volume with image processing techniques in order to find structures of interest (for example, a ribosome, a certain protein...) is called *tomogram picking*, *tomo picking* or just *picking*, while the programs that do the task are called *pickers*.

This allows structural biologists to isolate structures and further process the image data for research purposes. This Master's Thesis aims to improve the workflow validation capabilities in Cryo-Electron Tomography (CryoET) picking step inside of the Scipion framework as a part of the active task force with the aim of introducing validation in the whole workflow [10] [11] carried out inside of the I2PC.

Research done in the field of CryoET in the past years has brought several mechanisms and pieces of software [12] [13] [14] aimed to find structures of interest inside of 3D tomography images, but they introduce a problem. Not only the number of detected structures is different throughout the pickers, but also the exact coordinates vary and some of them might be wrongly picked. In order to enhance the results, the output of different pickers can be combined but the aforementioned problems have to be solved first.

The proposed solution is to preprocess all of the output data of the different pickers, to generate a set of coordinates and shapes that represent each of the real 3D structures of interest present in the images. Then, a neural network would be trained with the structures which have a full consensus as a positive input and random noise picked from the background as a negative input. This neural network would finally be used to score the representative structures against the model and provide an output based on a previously-set threshold.

In summary, the developed protocol should be able to combine the outputs of different picking mechanisms applied to the same 3D tomography image, combine and/or average the coordinates that apply to the same sub-tomogram and then train a neural network model to differentiate between correct and wrongly-picked structures of interest. At the end of the project, the following questions need to be answered by the developed solution:

- Can the output of different pickers be effectively combined?
- Does this combination yield better results?
- Is this approach useful for workflow automation?
- Does using neural networks introduce actual benefits over "simple" voting mechanisms?

The answer to all of these questions can be found in the results section.

1.1 Reach

The main tasks of the project can be summarized into the next list:

- Create a basic program inside of the scipion-em-xmipptomo plugin, exposing a protocol at GUI level that accepts picker program outputs as inputs.
- Add the code needed to manage and convert the inputs coming from the most commonly used pickers.
- Add the support for undirected structures (x, y, z) and define the internal data structures needed for the whole processing.
- Implement a neural network, defining initial parameters for it to accept the preprocessed data.

- (Optional) Add the support for directed structures (x, y, z, orientation).
- Perform hyperparameter optimisation for the neural network.

This project is difficult as the tasks do not only include the coding of a package and its integration into an almost 7 years old software suite, but also require knowledge on the Structural Biology field and electron microscopy. Thus, the reach of the project is designed to be flexible to account for any possible time deviation due to steep learning curves.

1.2 Structure of the document

This document is structured as to first gently and briefly acquaint the ICT-background reader with some concepts of the cryo-electron microscopy, tomography and structural biology fields. After that, it introduces the Scipion processing framework to then explain how it is internally organised in terms of code, with special emphasis on the CryoET workflows. After the background is presented and the problem is identified, the proposal, design and implementation of the solution is explained. Finally, some tests are run in order to be able to emit some conclusions about the solution, the project and the future work altogether. At the end of the document there are several appendices, which include project management information, deviation analysis, installation manuals...

2
State of the art

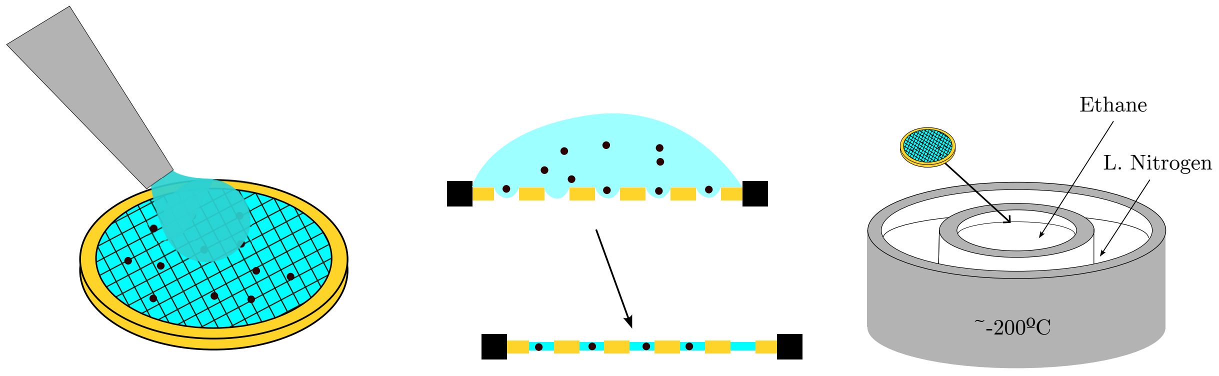
Before diving into the world of the actual CryoET workflow necessities inside of the Scipion Framework and working on new code, there are some concepts that need to be clear and some context that needs to be added. This chapter is devoted to introducing the reader to the field this thesis has been conceived in: isolated macromolecular electron microscopy imaging techniques inside of structural biology.

2.1 Structural Biology and CryoEM

Structural biology is a sub-field of molecular biology. It aims to elucidate the structure of the biological macromolecules found in living organisms in order to determine the relation between their conformation and the biological functions they perform. The classical ways of imaging (i.e optical microscopy) lack the resolution required for this level of detail [15], so more advanced techniques need to be used. This includes X-Ray crystallography, nuclear magnetic resonance (NMR) and cryogenic electron microscopy (CryoEM).

CryoEM uses electrons as the source of contrast for the images instead of visible light used in optical microscopy. There are several types of electron microscopy (EM), like Scanning EM (SEM) or Transmission EM (TEM). This work will revolve around TEM. An electron gun works at very high energy variations (normally up to 300keV) to generate high amounts of free electrons. They are then guided using electromagnetic lenses towards the specimen to be sampled. The electrons traverse the sample and suffer several transformations (changes in energy, trajectory...) and end up in a receptor creating a projection. The receptor can either be a fluorescent screen that is scanned afterwards or a direct detector, and converts the energy into information that can later be converted into images through reconstruction algorithms [16] [17].

The imaging is done under a high vacuum state, as the interaction between the free electrons and the airborne particles would make the detected image very noisy and useless. This introduces another problem: under vacuum, water evaporates and biological material disintegrates. Thus, the samples must be either fixed in resin (room temperature EM) or frozen (**cryo** EM). Water crystals could destroy the sample, so the sample must be vitrified in a special way using methods such as plunge (see Figure 2.1) or high-pressure (for thicker samples) freezing. This fast vitrification prevents the creation of crystals. At the same time, it makes the proteins and structures stay in a close-to-native state and protects them from the electron radiation.



1. The sample is transferred to the grid, leaving some excess liquid.
2. The residue is removed (blotting) using filter paper.
3. The grid is frozen (plunging) using liquid ethane and nitrogen.

Figure 2.1: Plunge freezing - initially proposed by Jacques Dubochet - helps preserve the living-state properties of a sample if correctly applied.

2.2 Cryogenic Electron Tomography

Transmission Electron Microscopy (TEM) is an imaging technique in which electrons are transmitted *through* an specimen and end up on an detector device (film, fluorescent screen or scintillators connected to electronic counting devices). Figure 2.2 from *Chegg.com*[1] illustrates how it works.

An interesting approach for 3D imaging inside of the TEM field is electron cryotomography (CryoET). This method is used to reconstruct sub-cellular macromolecular structures, as it can reach atomic levels [18]. Advances in the field have made CryoET gain popularity in the last years, gaining traction on the reference data bank for EM (EMDB), as it can be seen on Table 2.1.

	2013	**2014**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**	**2022**	**2023**
SPA	428	485	478	815	869	1207	1937	3360	3977	5260	3720
Tomo	94	81	103	198	169	418	367	312	326	602	471

Table 2.1: Evolution of EMDB entry number for CryoEM and CryoET per year. Data checked on: 2023/07/31. Single Particle Analysis is still the leader in the field of cryoEM imaging techniques.

Even though tomography techniques started gaining some traction at the beginning of the past decade of the 2010s, it is not until 2018 that they start to be a significant part of the yearly EMDB depositions. When compared to the most widely practised imaging technique - Single Particle Analysis (SPA) - tomography seems to be 3-5 years behind - at least in terms of deposition numbers.

[1]`https://www.chegg.com/learn/topic/transmission-electron-microscope`

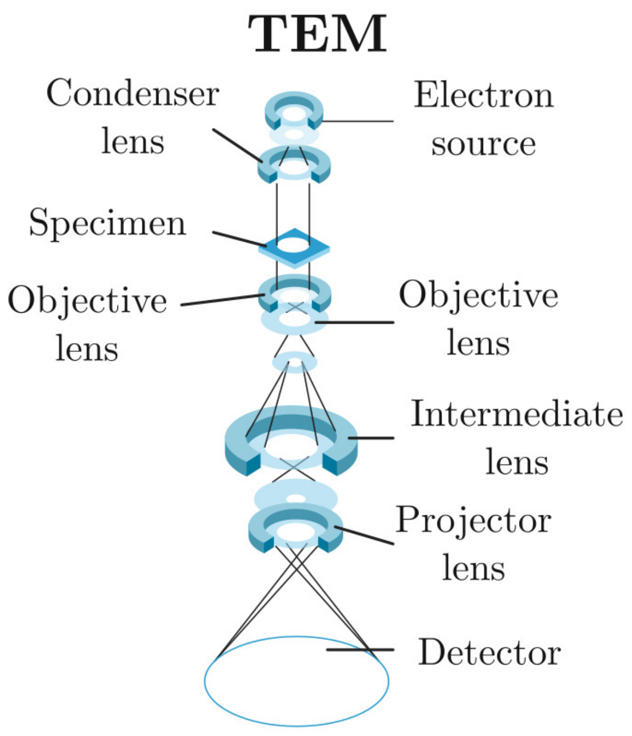


Figure 2.2: Simplified anatomy of a TEM, courtesy of Chegg.com

Similarly to what is done in other tomography techniques, the sample is tilted to different angles, usually in the range of ±60º and acquiring one or more images every 1 or 2 degrees. The resulting set of images - *called a tilt-series* - can be used to reconstruct a 3D volume using computerised algorithms, such as filtered backprojection, ART, SIRT, SAMV[2]...

As it can be seen in Figure 2.3, each of the tilt-series images contributes to the reconstruction of a 3D model. But this process is not trivial and really involves much more intricate steps, compiled in the following list:

- **Sample preparation -** the biological samples are either plunge (when thin) or pressure-frozen (when too thick for plunge freezing) to get them in a cryogenic state. This creates a *vitreous* environment (some literature says it's an amorphous non-crystalline state of the matter, whilst others state it's a very dense liquid) in which the state of the biological samples stays still.

- **Data acquisition -** This step involves the acquisition of all the images of the tilt-series, radiating electrons on the frozen sample to get projections in the form of electron counts and generate the initial images that are transferred to the processing machines. For a tilt range, a projection is done after every rotation, resulting in a series of projections of the biological sample.

- **Movie alignment -** At each of the tilt-angles, several images are acquired - this is called a *movie*. The radiation from the electron gun itself can cause movements or damage on

[2]`https://en.wikipedia.org/wiki/Iterative_reconstruction`

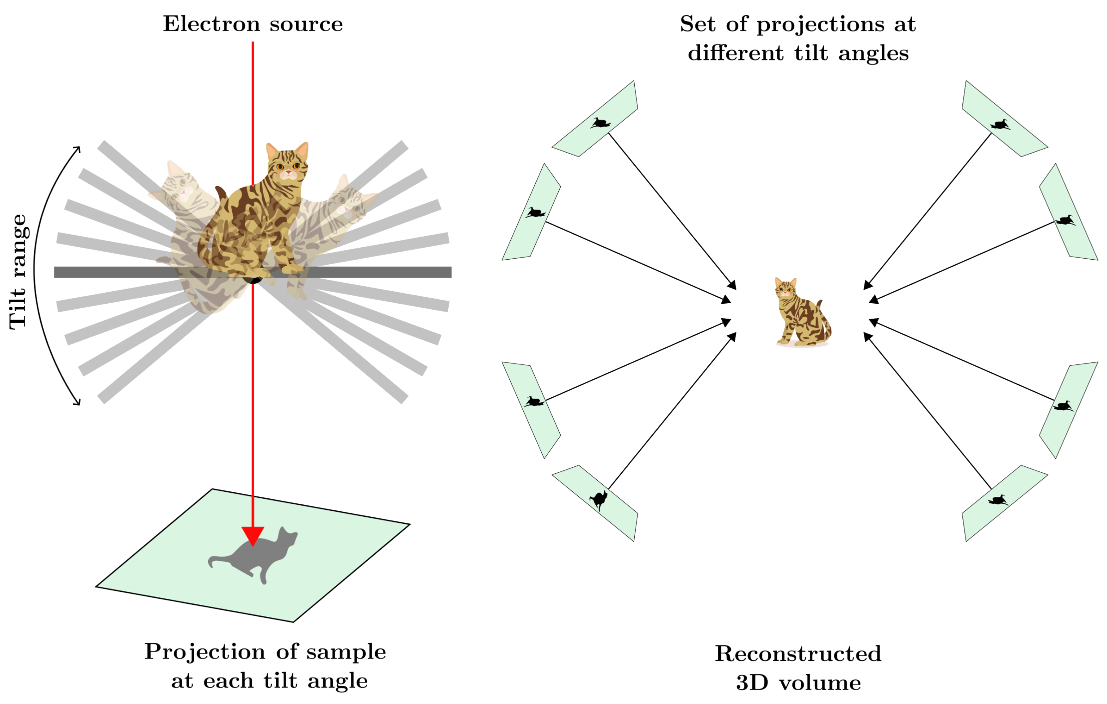

Electron source
Set of projections at different tilt angles
Tilt range
Projection of sample at each tilt angle
Reconstructed 3D volume

Figure 2.3: Reconstruction of a 3D volume using tomography.

the sample, so this step performs a relative movement analysis between all of the images from a single tilt angle.

- **Tilt-series alignment -** From any tilt angle to the next one, positional errors are introduced. The mechanical parts of the microscope responsible for the movement of the sample are not perfect, and they introduce unintended movements. It is very important to also correct these movements before carrying on with the reconstruction.
- **Contrast Transfer Function (CTF) -** Again, the microscope introduces some artefacts in the final captured image. These image artefacts can be corrected, as the CTF models the aberrations and imperfections of the microscope hardware.
- **Tomogram reconstruction -** Once the tilt-series is aligned and the aberrations and defocus have been corrected, it's time to reconstruct the three-dimensional figure from the set of projections either using back-projection or series expansion techniques.
- **Tomogram post-processing -** The original reconstructions tend to have a very poor signal-to-noise ratio (SNR), which could make interpreting the results very difficult. Several filters and techniques can be applied to increase the SNR by denoising and sharpening every slice of the 3D image.
- **Picking -** It consists on finding the proteins or structures of interest in the tomogram. As the SNR is usually low, this identification of proteins is not an easy task. This work revolves around this particular step of the workflow, and the exact approach is described in Chapter 3.
- **Subtomogram averaging -** Instances of proteins can be grouped by classifying them for their conformational state, and after determining the orientation of each of them they can be combined (averaged) to get a more detailed protein structure.

2.3 Processing software - The Scipion Framework and Xmipp

2.3.1 The big frame: Scipion

SCIPION

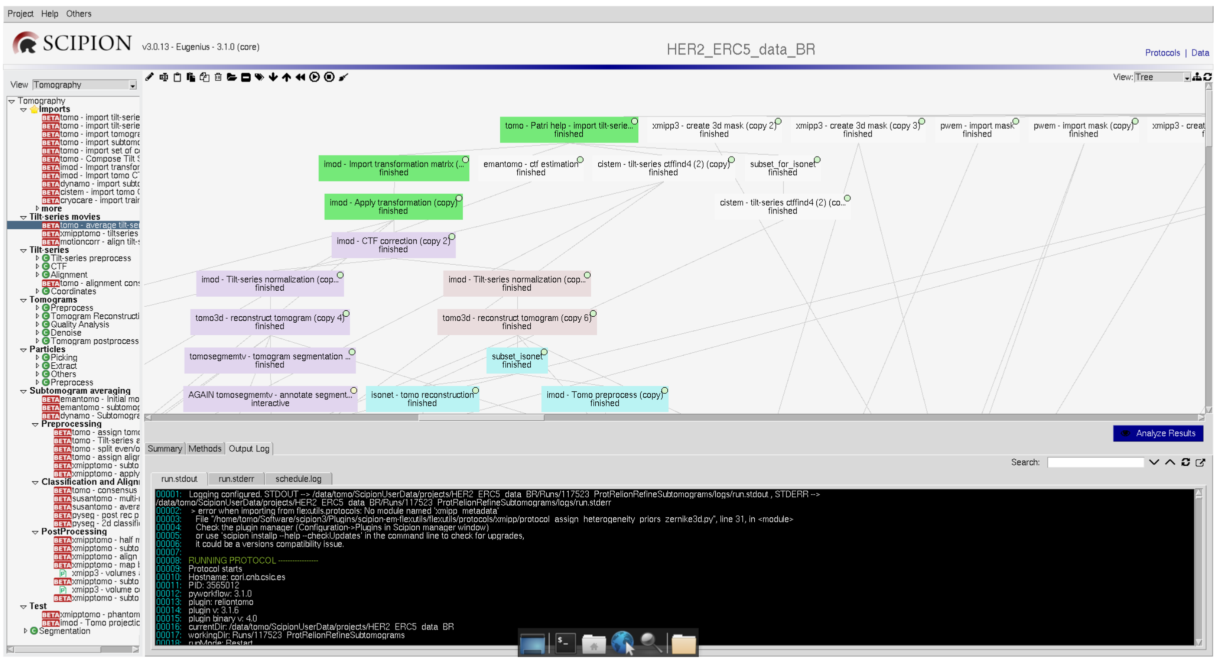
Being a fairly novel field, CryoET lacks a set of "decently" mature tools to go for in all of the pipeline. As it is mentioned several times throughout this work, it is important to strive for a higher level of accuracy by adding validation protocols to the electron cryotomography workflow [10] by using the results of many different programs for the same task (amongst other measures).

Using several programs for the same task introduces many problems. First, if the scientific rigour is to be maintained, one has to keep a thorough track of every execution, as well as the order and the parameters utilised. Given that a good track of the experiments is being kept, the next task is to make an enquiry on how to convert the different formats and conventions found in every program. Finally, an easy way to export and share the process and results is needed if one wants to do something useful with their work. In short, it is a huge amount of work apart from the experimentation itself.

The Biocomputing Unit (BCU) at the National Center for Biotechnology (CNB-CSIC) developed the Scipion framework back in 2016 [7] to integrate their image processing suite for electron microscopy [8] with other existing software. With its v3.0 release, Scipion was converted to a fully modular system in which any developer can integrate third-party programs. Some insights on the installation can be found in the Appendix A. As Figure 2.4 shows, the software allows for big projects with complex pipelines to be easily managed whilst maintaining an user-friendly interface, thanks to the tree system that allows custom colours and notes/comments.

As stated before, Scipion3 is a modular system that relies mainly on the internally called *three musketeers* **scipion-app, scipion-pyworkflow and scipion-em**. The first one is responsible for the installation of the remaining core packages, as well as plugins. The second one is the *true core*, providing the workflow management graphical user interface (GUI), as well as all of the needed definitions for the functioning of the plugins (SQLite3 DB management, data structures for projects, protocols, tests, launching mechanisms...). The third one provides the definition of the Electron Microscopy field domain (Scipion could accept other domains). Figure 2.5 shows the simplified structure of the Scipion framework.

The modularity of the whole system allows more domain models to be added (such as CryoEM [19] and Chemoinformatics [20] which are the two existing ones at the time of writing). Each of the domains can be expanded using **plugins** to add functionality in a specific area. These plugins can be identified as their name is always formed like: **scipion-*domain-plugin*** For instance, the *crYOLO* program for picking [12] is introduced into the CryoEM suite by means of the *sphire* plugin: *scipion-em-sphire*, and the basic programs for electron cryotomography and subtomogram averaging are introduced by *scipion-em-tomo*.


Project Help Others
SCIPION v3.0.13 - Eugenius - 3.1.0 (core)
HER2_ERC5_data_BR
Protocols | Data
View: Tomography
View: Tree
tomo - Patri help - import tilt-serie... finished
xmipp3 - create 3d mask (copy 2) finished
xmipp3 - create 3d mask (copy 3) finished
pwem - import mask finished
pwem - import mask (copy) finished
imod - Import transformation matrix (... finished
emantomo - ctf estimation finished
cistem - tilt-series ctffind4 (2) (copy) finished
subset_for_isonet finished
imod - Apply transformation (copy) finished
cistem - tilt-series ctffind4 (2) (co... finished
imod - CTF correction (copy 2) finished
imod - Tilt-series normalization (cop... finished
tomo3d - reconstruct tomogram (copy 4) finished
tomo3d - reconstruct tomogram (copy 6) finished
tomosegmemtv - tomogram segmentation ... finished
subset_isonet finished
AGAIN tomosegmemtv - annotate segment... interactive
isonet - tomo reconstruction finished
imod - Tomo preprocess (copy) finished
Analyze Results
Summary
Methods
Output Log
run.stdout
run.stderr
schedule.log
Search:

```
00001:  Logging configured. STDOUT --> /data/tomo/ScipionUserData/projects/HER2  ERC5  data  BR/Runs/117523  ProtRelionRefineSubtomograms/logs/run.stdout , STDERR -->
/data/tomo/ScipionUserData/projects/HER2  ERC5  data  BR/Runs/117523  ProtRelionRefineSubtomograms/logs/run.stderr
00002:    > error when importing from flexutils.protocols: No module named 'xmipp  metadata'
00003:     File "/home/tomo/Software/scipion3/Plugins/scipion-em-flexutils/flexutils/protocols/xmipp/protocol  assign  heterogeneity  priors  zernike3d.py", line 31, in <module>
00004:     Check the plugin manager (Configuration->Plugins in Scipion manager window)
00005:     or use 'scipion installp --help --checkUpdates' in the command line to check for upgrades,
00006:     it could be a versions compatibility issue.
00007:
00008:  RUNNING PROTOCOL -----------------
00009:  Protocol starts
00010:  Hostname: cori.cnb.csic.es
00011:  PID: 3565012
00012:  pyworkflow: 3.1.0
00013:  plugin: reliontomo
00014:  plugin v: 3.1.6
00015:  plugin binary v: 4.0
00016:  currentDir: /data/tomo/ScipionUserData/projects/HER2  ERC5  data  BR
00017:  workingDir: Runs/117523  ProtRelionRefineSubtomograms
00018:  runMode: Restart
```

Figure 2.4: A real-life project opened in the Scipion3 GUI. Image provided by Ana Cuervo and Patricia Losana.

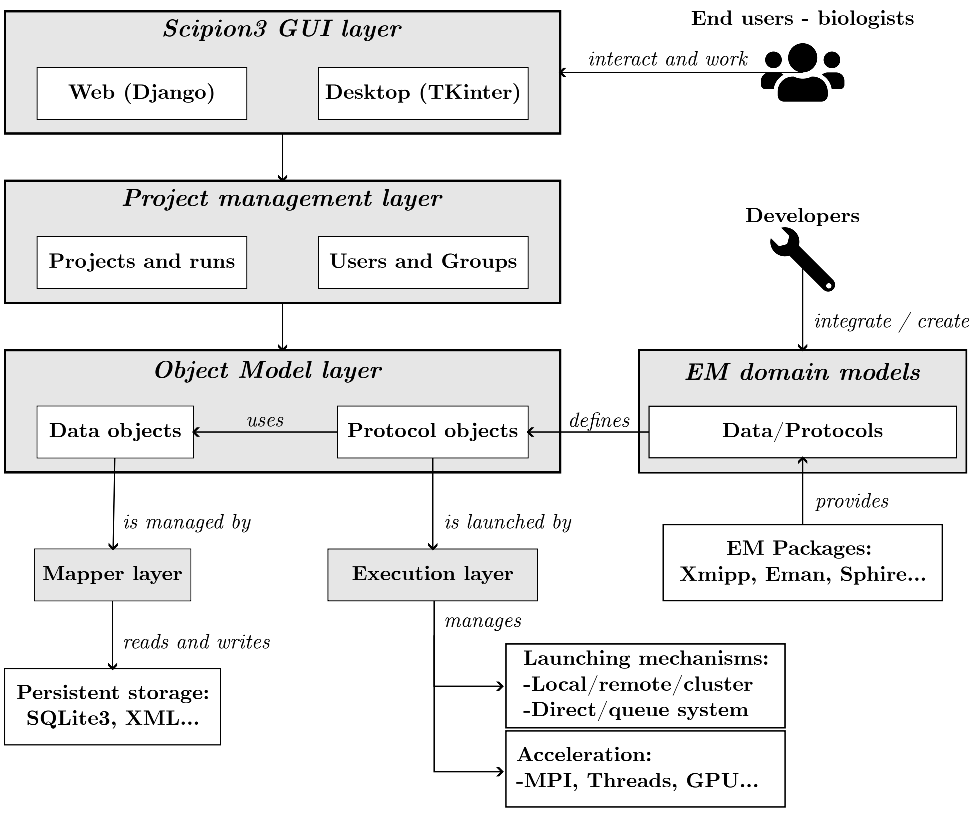


Figure 2.5: Components of the Scipion3 Framework.

2.3.2 The core processing utilities: Xmipp3 and its integration with Scipion

The *X-windows based microscopy image processing package* (Xmipp) was first published back in 2004 [8] as a specialized suite of image processing programs, primarily aimed at obtaining the 3D reconstruction of biological specimens from large sets of projection images acquired by transmission electron microscopy.

Since then, this public-domain and open-source set of programs has evolved into a robust, mature, varied and renowned suite for cryoEM processing. It relies on four repositories: *xmipp*, *xmippCore*, *xmippViz* and *scipion-em-xmipp*. The first two are mainly C/C++ code, with some Python scripting for simpler tasks and CUDA for GPU offload. The two later, however, have little to no parts of C/C++. The visualisation package "Viz" is mainly written in Java, while the integration with Scipion is all Python. Together, these repositories allow for easy cryoEM image handling, visualisation, storage...

The newest addition *scipion-em-xmipptomo* gathers protocols for CryoET image processing. These make use of the programs developed in *xmipp* and extend the xmipp to the tomography domain. This includes protocols for each stage of the pipeline described in 2.2 movie alignment, tilt-series alignment, CTF calculation, reconstruction...

As the names indicate, Xmipp exposes its general cryoEM protocols in Scipion through the *scipion-em-xmipp* plugin and the specific cryoET ones through the *scipion-em-xmipptomo* (mostly beta versions as of redaction date) plugin.

2.4 Identifying the problem - proposed solution

So, the current panorama is: There is a demand for validation in the CryoET field, to make better use of the wide variety of developed programs for each step of the process. Scipion has some of the workflow parts or stages covered by validation protocols in its scipion-em-xmipptomo plugin, but picking still does not have one. This results in structural biologists having a really hard time - or not being able to do it at all - when looking for automatized, better, consensuated results from different algorithms and programs.

Low signal-to-noise ratio (SNR) found in cryoET images makes the picking step a very complicated task. Often pickers select high contrast zones (gold beads used for align tasks, ice contaminants, the support of the containing grid) instead of the regions of interest. As no picker is perfect, a consensus program is mandatory to improve the election of the interesting particles.

Thus, this project is conceived. The proposal is as follows: A new protocol will be introduced into the existing scipion-em-xmipptomo plugin, written in Python. It will be accessible from the Scipion GUI, allowing both basic and advanced parameter tuning. The inputs, coming from two or more previously existing picking protocols will be converted into a common convention, respecting the Scipion internal object structures for coordinates, tomograms and subtomograms. All coordinates referring to a same structure will have a representative assigned. A neural network model will be processing the data, and the user will be able to load a previously trained one or train it in-situ with the input data. In addition to the sets to be evaluated, the user will be able to import previously detected negative or positive examples to aid the training in difficult situations. The output will include the pickings that pass the quality threshold. In the end, the user will have a **consensus** of 3D particle picking with a set of coordinates ready for further processing and refining.

The proposal includes collaborations with several people from the BCU and CNB-CSIC. **Federico P. de Isidro** provided with the formation on CryoET workflows and validation concepts [10]. **Jorge Jiménez** and **Oier Lauzirika** gave an introduction to coding inside Scipion and Xmipp. **Patricia Losana**, **Ana Cuervo** and **Jose Luis Vilas** reviewed the scientific results of the project. Carlos Óscar Sorzano and Roberto Marabini (directors) supervised the advances of the work and its integration with Scipion, as well as the conceptual mechanisms of the protocol.

3
Design and development

This chapter describes the program design and coding effort carried out for this project to be completed. The following sections will expose the followed methodology, as well as the particularities of the integration of each part of the software inside of Scipion and Xmipp and the functioning of the protocol.

3.1 Followed methodology

There is no strongly defined methodology in the Scipion and Xmipp development teams. Thus, the decision on how to carry the development was open and based on following the cues found on previous work of other developers. The following cycle was established and followed:

- The development is divided into three main tasks T: Scipion code (GUI, form...), integration code (conventions, objects, classes...) and the work code (TensorFlow, data conversion...). Each of the tasks is independent and can be worked on in parallel.
- Each task is then divided into smaller subtasks t (as long as possible and convenient).
- The possible dependencies between subtasks are calculated and the following mechanism is started:
- For each main task T...
  - For each subtask t that composes T...
    1. Study the existing code and determine needed libraries.
    2. Define the basic structure in a "fast" way to define the code layout, mainly through comments in the code.
    3. Materialize the concepts into actual code.
    4. Create tests and code until it's working.
    5. Refine and comment the code.

6. Re-run the tests, code and refine stages until it's fully working.
7. Integrate the subtask code with the surrounding pieces of code.
8. Integrate task work with the previous one.

- Refine the code as a whole and implement general tests.
- Integrate into Scipion through the scipion-em-xmipptomo plugin.

3.2 Extra code

Besides the main code, additional pieces of software have been independently developed for the deep consensus protocol to work. This section presents the approach and some technical details of this additional code.

3.2.1 Coordinate consensus 3D

At the beginning of the execution pipeline, the protocol needs to determine the duplicities in the input sets, as different pickers may pick the same region with a small offset. Originally, it was thought that other existing utilities could be used for the task, but finally a dedicated program was developed for this, called *xmipp_ coordinates_ consensus_ tomo*.

This Python Xmipp script does not have a GUI, and needs to be launched directly from the terminal. It takes the route to an Xmipp MetaData (XMD) file containing all of the coordinates picked in a certain tomogram as an input. The metadata file must also contain the reference to the picker of origin. Given certain threshold, box size and sampling rate values it will determine which coordinates from different pickers refer to the same region/structure of interest. The output is written in four separate files. The first one contains the consensus coordinates that surpassed the credibility threshold, and are thus deemed to be *correct* pickings. The second one contains the coordinates below the credibility threshold, but still above an inferior limit, which makes them *doubtful* pickings. The third one contains all the coordinates that do not pass any threshold and are marked as *negative* pickings. The fourth one contains the full set of consensus coordinates regardless of their classification. The output format is also XMD, so it's fully compatible with the rest of the Xmipp ecosystem.

Additionally, this program allows the input of previously labelled data.

3.2.2 Noise picking 3D

Data augmentation is necessary for the network to generalise in a better way. A way of augmenting the set of negative examples is picking noise from the input tomograms themselves. This script is called *xmipp_ pick_ noise_ tomo*.

This Python Xmipp script is also called from the terminal, and works on a per-tomogram basis. Similarly to what happens with the coordinate consensus 3D program, this one needs the route to an XMD file containing all of the picked coordinates inside a certain tomogram. It also receives the tomogram's size and the picking parameters. It then follows a Monte-Carlo approach, generating random coordinate points inside the tomogram and then evaluating if they fulfil the requirements. Even though it can be configured to use other metrics, the one actually implemented is the surpassing of a threshold in the (euclidean) distance between the generated point and all the picked coordinates. As an output, it will write an XMD file containing the almost-certain negative coordinates for the input tomogram.

3.2.3 Subtomogram extraction

Once all the work with coordinates is done, a program is needed to extract the actual image data form the original tomograms. This program already exists in the Xmipp codebase and it is called *xmipp_tomo_extract_subtomograms*. This program receives an original tomogram, the coordinates file and the box size. It then extracts all of the specified points to the selected output folder in MRC format, and also generates a file for each tomogram referencing all its extracted subtomograms.

3.2.4 Networks Manager - NetMan

The network manager (or NetMan) is a class specifically developed for this project. It receives a plethora of parameters regarding picking and the neural network model. It is able to create and load networks, manipulate the input 3D images and feed them to the network to either train or score. It includes several pieces of integration with Keras and TensorFlow, such as the model itself (keras.models.Sequential) or the wrapper for the dataset (tf.data.Dataset). By default, NetMan will create a neural network of the following characteristics:

- Type of network: Deep 3D Convolutional Neural Network.
- Input layer topology: 1x Resize layer.
- Convolving layers topology: 3x Convolution, max pooling and normalisation.
- Finishing touches layers topology: 1x Global average pooling, dense, dropout, dense, dropout, dense.
- Output form: A vector containing the probability of being a good picking for each input subtomogram.

The structure can be better seen in Table 3.1, which shows the example network created for an input of shape (50x50x50x1), as well as Figure 3.1, which visualises the shape of the images as they traverse the network.

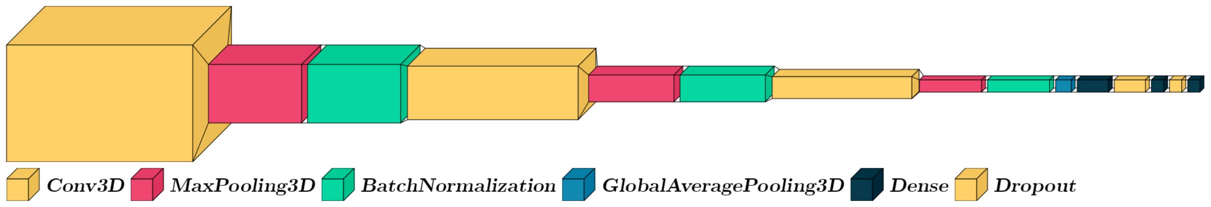


Figure 3.1: Shapes of the inputs as they traverse the network, as printed by VisualKeras.

This code makes use of TensorFlow *Strategies*, which enables the use of multiple GPUs as well as CPU threads to accelerate the training and scoring of a single model. A Python

Layer type	**Output shape**	**Param #**
Input	(None, 50, 50, 50, 1)	0
Conv3D	(None, 48, 48, 48, 64)	1792
MaxPool3D	(None, 24, 24, 24, 64)	0
Batch Norm	(None, 24, 24, 24, 64)	256
Conv3D	(None, 22, 22, 22, 128)	221312
MaxPool3D	(None, 11, 11, 11, 128)	0
Batch Norm	(None, 11, 11, 11, 128)	512
Conv3D	(None, 9, 9, 9, 256)	884992
MaxPool3D	(None, 4, 4, 4, 256)	0
Batch Norm	(None, 4, 4, 4, 256)	1024
GlobAvgPool 3D	(None, 256)	0
Dense	(None, 512)	131584
Dropout	(None, 512)	0
Dense	(None, 128)	65664
Dropout	(None, 128)	0
Dense	(None, 2)	256

Table 3.1: Structure of the model. In total, it has 1.307.392 parameters.

Generator integrated with Xmipp image handling is employed to feed the TensorFlow Dataset object without leading the machine to an *Out Of Memory (OOM)* situation. This data generating object receives a batch size and will yield a pack of labelled images (unlabelled in the case of the scoring data Generator) every time it is called.

3.3 Code design and organisation

3.3.1 Choice of repository

As stated at the end of Chapter 2.3.1, the integration of new features into Scipion is done through its addition to the code of a previously existing plugin or through the creation of a whole new one. This decision has to be done after analysing the possible groupings: by software suite (such as scipion-em-sphire that gathers all programs developed by the Sphire team), field (such as scipion-em-tomo that gathers basic functions and object definitions for CryoET) or mixed groupings (such as scipion-em-xmipptomo that gathers all of the extensions done to Xmipp but are too specific to be included in the main xmipp or scipion-em-tomo).

An analogous version of this project exists for Single Particle Analysis (SPA) in 2 dimensions. It is included in the scipion-em-xmipp project, while the executed libraries and programs are stored in the xmipp project. This one differs because the protocol is added to the scipion-em-xmipptomo plugin instead, but the heavy-duty code and libraries are still found in xmipp.

The tag to find this project in the repositories is *mit_ deepconsensus_ 3d*[1][2].

[1] `https://github.com/I2PC/xmipp`

[2] `https://github.com/I2PC/scipion-em-xmipptomo`

3.3.2 Code found in each repository

The protocol is thus distributed amongst several repositories, and once the protocol is launched from inside Scipion they start interacting. The code that runs from inside of the scipion-em-xmipptomo package is called *the protocol*. It defines the elements shown in the GUI, as well as the I/O management and the creation of the needed data structures. Minor calculations are done inside of this code, such as generating lists and writing them into Xmipp-formatted metadata files. The code found in xmipp is divided in two. Part of it, called *the program* is found in the *applications/scripts* folder and consists on the code that receives the orders from the protocol, doing the biggest part of the processing. The other part, called *the library* resides in the *libraries/py_xmipp/deepPickingConsensusTomo* folder and contains the logic of the neural network as well as the definition of the internal data management mechanisms. This structure is more clearly seen in Figure 3.2.

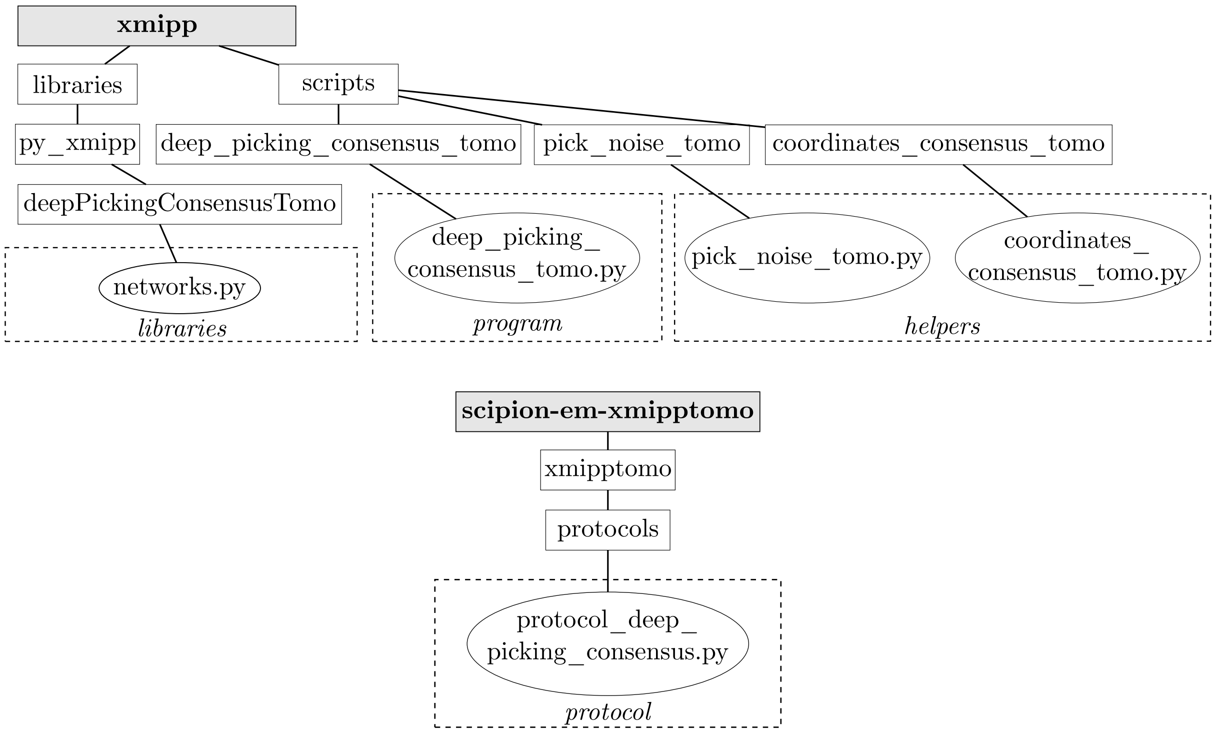


Figure 3.2: Actual code files supporting the "libraries", "program" and "protocol".

3.3.3 Code interactions

From the moment the Scipion GUI starts the processing pipeline, there are several handover moments in which a piece of code delegates the work on another or returns the results to other programs that are within the project scope (but on another code realm). This is better understood when speaking in terms of ***steps***. There are several steps defined on the protocol code, many of them resulting in calls to other parts of the project (and even the libraries from xmipp developed outside of it). Each of the steps, as well as the concrete interactions, are later explained in detail.

3.4 Functioning

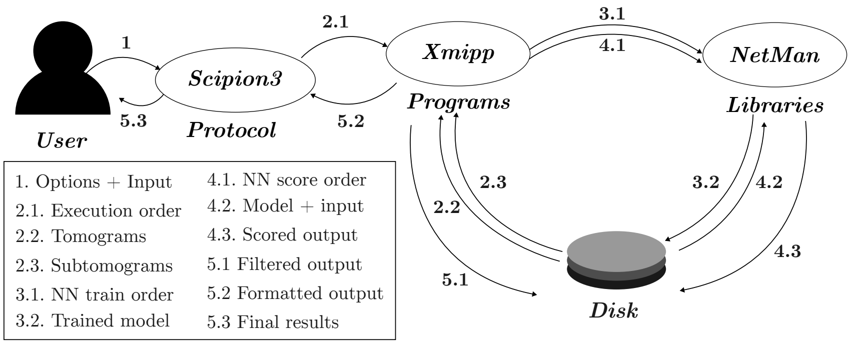
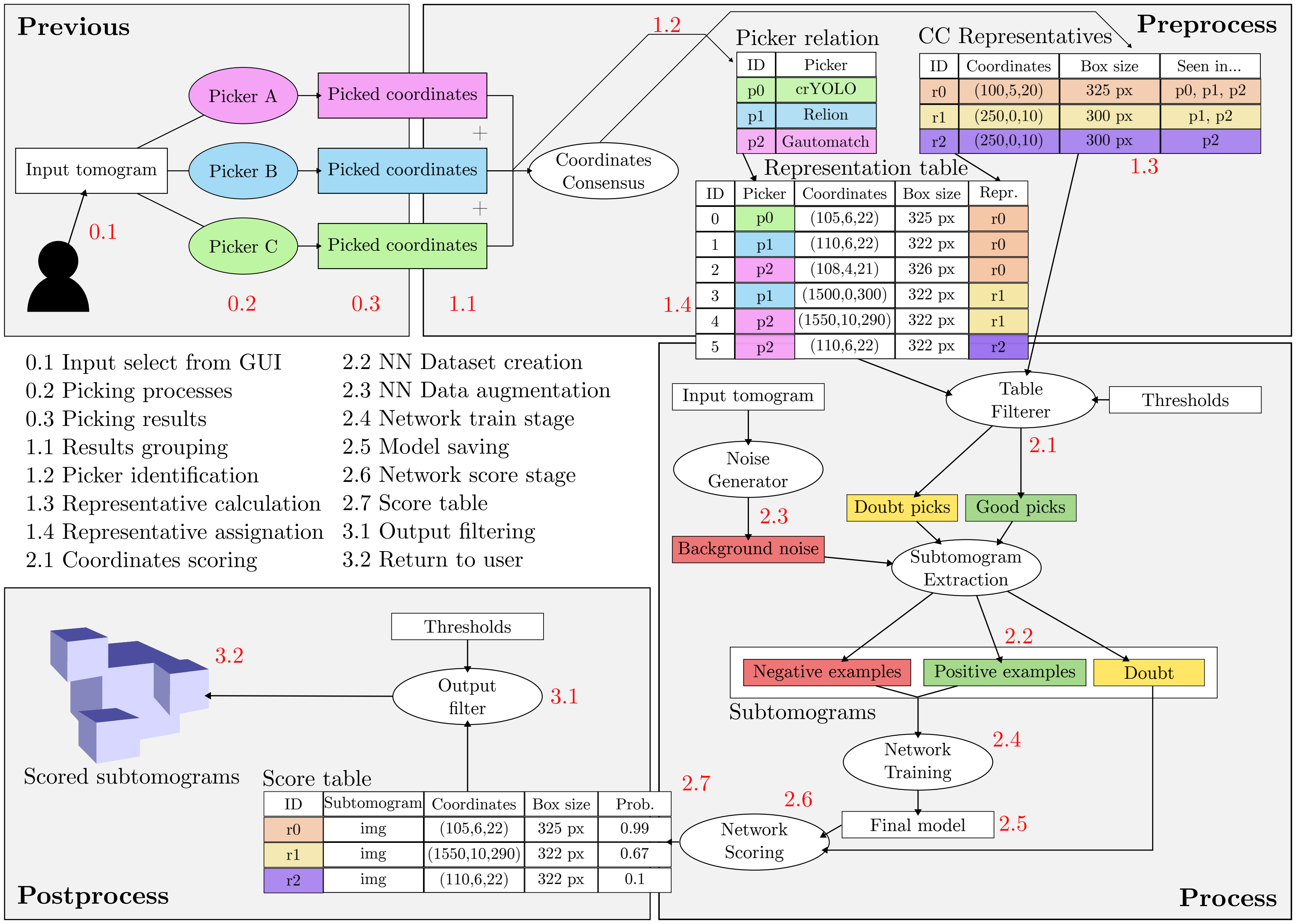


Figure 3.3: Information flow between the different parts of the project during the processing.

The project can be seen from two perspectives. The first one (applied in the previous section) looks at the code files themselves and the come-and-go from one to another during their interaction. The second one summarizes the implemented solution from a functional point of view, describing the "story" of the protocol from start to finish without paying much attention to which code is actually executing it. Figure 3.3 shows the first point of view, while in the next horizontal page, Figure 3.4 illustrates the grouping of the main functional blocks and their smaller tasks.



Picker relation

ID	Picker
p0	crYOLO
p1	Relion
p2	Gautomatch

CC Representatives

ID	Coordinates	Box size	Seen in...
r0	(100,5,20)	325 px	p0, p1, p2
r1	(250,0,10)	300 px	p1, p2
r2	(250,0,10)	300 px	p2

Representation table

ID	Picker	Coordinates	Box size	Repr.
0	p0	(105,6,22)	325 px	r0
1	p1	(110,6,22)	322 px	r0
2	p2	(108,4,21)	326 px	r0
3	p1	(1500,0,300)	322 px	r1
4	p2	(1550,10,290)	322 px	r1
5	p2	(110,6,22)	322 px	r2

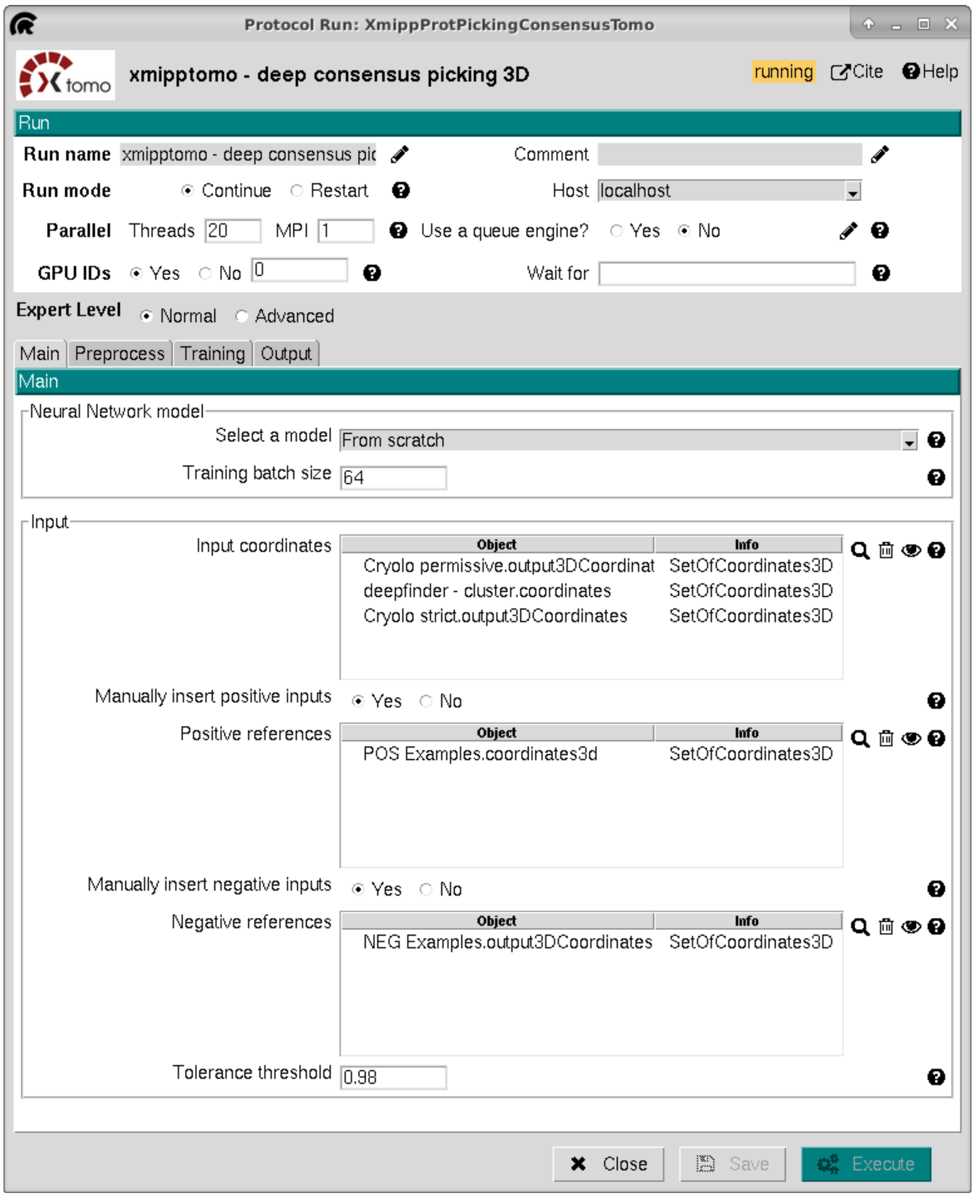
Score table

ID	Subtomogram	Coordinates	Box size	Prob.
r0	img	(105,6,22)	325 px	0.99
r1	img	(1550,10,290)	322 px	0.67
r2	img	(110,6,22)	322 px	0.1

0.1 Input select from GUI
0.2 Picking processes
0.3 Picking results
1.1 Results grouping
1.2 Picker identification
1.3 Representative calculation
1.4 Representative assignation
2.1 Coordinates scoring
2.2 NN Dataset creation
2.3 NN Data augmentation
2.4 Network train stage
2.5 Model saving
2.6 Network score stage
2.7 Score table
3.1 Output filtering
3.2 Return to user

Figure 3.4: Diagram definition of the protocol and its phases.

3.4.1 Parameter gathering and launch



xmipptomo - deep consensus picking 3D

Run

Run name	xmipptomo - deep consensus pic	Comment	
Run mode	[✓ Continue] Restart	Host	localhost
Parallel	Threads 20 MPI 1	Use a queue engine?	Yes [✓ No]
GPU IDs	[✓ Yes] No 0	Wait for	

Expert Level [✓ Normal] Advanced

Main | Preprocess | Training | Output

Main

Neural Network model

Select a model	From scratch
Training batch size	64

Input

Input coordinates

Object	Info
Cryolo permissive.output3DCoordinat	SetOfCoordinates3D
deepfinder - cluster.coordinates	SetOfCoordinates3D
Cryolo strict.output3DCoordinates	SetOfCoordinates3D

Manually insert positive inputs [✓ Yes] No

Positive references

Object	Info
POS Examples.coordinates3d	SetOfCoordinates3D

Manually insert negative inputs [✓ Yes] No

Negative references

Object	Info
NEG Examples.output3DCoordinates	SetOfCoordinates3D

Tolerance threshold 0.98

Figure 3.5: The tool is integrated into Scipion through an user-friendly GUI.

Upon introducing an instance of the protocol in an active Scipion project, the GUI will present the user with a form that allows the tuning of many parameters in its various tabs, as illustrated in Figure 3.5. The full explanation of each of the parameters is found in Appendix B. Parameters are described inside of the protocol in the following way:

```
def _defineParams(self, form : params.Form):
  #...
  group_input.addParam('inputSets', params.MultiPointerParam,
            pointerClass = SetOfCoordinates3D, allowsNull=False,
            label = 'Input coordinates',
                help = 'Select the set of 3D coordinates that represent
                                                        ↪ the
                                                        ↪ subtomograms
```

```
                                                                  ↪ to be used as
                                                                  ↪ input data.')
  #...
```

After all of the parameters are correctly set and the *Execute* button is hit, the execution of the protocol begins. Scipion will internally call the protocol's class function **_insertAllSteps()**, which contains a call to the main step functions mentioned later in this chapter.

3.4.2 Pre-process step

The first block of execution begins with the call to **preProcessStep()** from within the protocol. At this moment, Scipion is able to give the protocol a handle to every variable stored in the input form by calling its *get()* method. As an example, the following code snippet provides the input from all of the chosen picking protocols (Scipion maps the objects at runtime by the name given in the form description):

```
# Function that reads the GUI-defined parameters
def preProcessStep(self):
  # The form has a parameter called inputSets that carries the 3D coordinates
                                              ↪ from the input pickers
  self.inputSetsOf3DCoordinates = [item.get() for item in self.inputSets]
```

In a similar fashion, every input parameter can be read and stored in the protocol's memory for modifications. Once all the needed parameters are read, two tables are created: *self.untreated* and *self.pickerMD*. The first one (self.untreated) holds a set containing the coordinates found by all input pickers together. It contains the ID of the picker, the separate 3D coordinates, an identifier for the tomogram of origin and the box size and sampling rate of the regions. The second table (self.pickerMD) contains metadata information from each picker: the ID, size in pixels of the box that contains the picked structures (called box size) and the measurement of each voxel (in Å). These two tables are preallocated with the total amount of input coordinates to avoid excessive overhead in case the input is too big, by using the length of the input set of sets for the pickerMD table and the sum of the lengths of each of the contained sets for the untreated table.

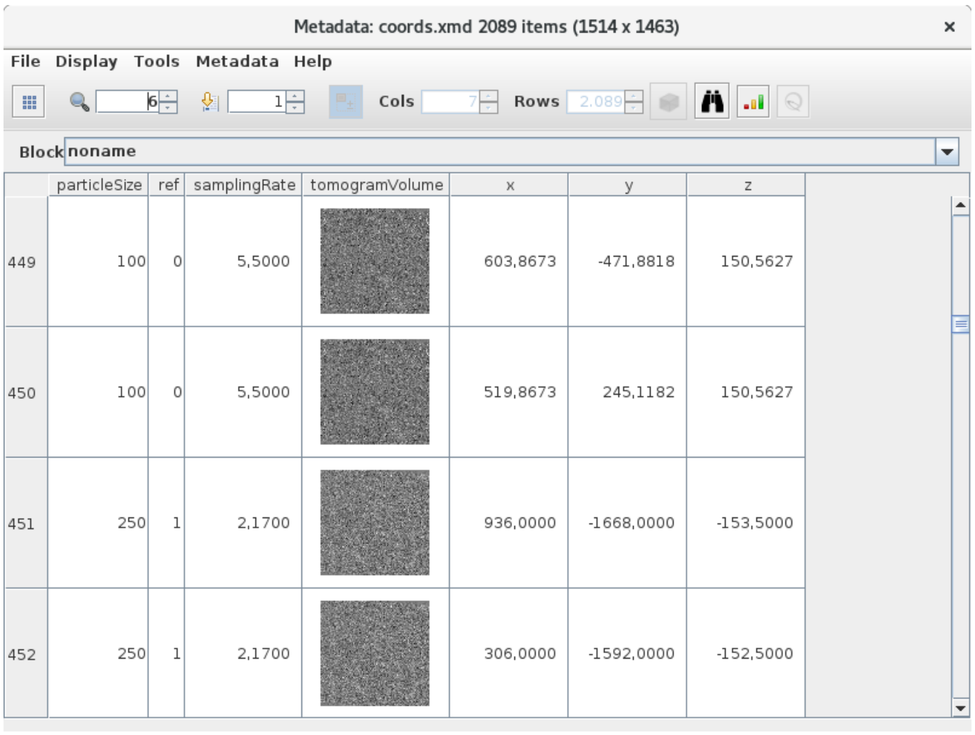
```
# GENERATE THE NEEDED TABLES TO START ---------------------------------
# Combined table of untreated data
colnames = ['pick_id','x', 'y', 'z', 'tomo_id', 'boxsize', 'samplingrate']
self.untreated = pd.DataFrame(index=range(self.totalROIs),columns=colnames)

# Pickers data table
colnames_md = ['boxsize', 'samplingrate']
self.pickerMD = pd.DataFrame(index=range(self.nr_pickers), columns=colnames_md)
```

The population of both tables is done in the same 2-level nested *for* loop. Each of the inputs will be of class *SetOfCoordinates3D*, and the contained objects are all of class *Coordinate3D*. They both provide useful getter functions to obtain the needed information such as the associated tomogram or the coordinates. In addition, when previously labelled data is introduced using the GUI, the corresponding structures are created at this point.

The auxiliary function *BSSRConsensusStep()* is then called for setting a common sampling rate and box size for the neural network used later. This function also accepts an optional parameter to modify which is the criteria for the consensus: the biggest, the smallest, a mean value or the first encountered value, for example.

At the end of this step, the auxiliary function *writeCoords()* is called to persist the generated tables. A Pandas DataFrame is created for every tomogram, containing all of its picked coordinates and the associated data. This requires the use of the Xmipp *MetaData Library (MDL)*, as in the following example:

```
from pwem import emlib
# This is done for every input tomogram
# setColumnValues(XMIPP MD LIBRARY LABEL, VALUE)
# Create a Xmipp MD Object
outMD = emlib.MetaData()
outMD.setColumnValues(emlib.MDL_REF, df['pick_id'].tolist())
outMD.setColumnValues(emlib.MDL_XCOOR, list(map(int, df['x'])))
outMD.setColumnValues(emlib.MDL_YCOOR, list(map(int, df['y'])))
outMD.setColumnValues(emlib.MDL_ZCOOR, list(map(int, df['z'])))
outMD.setColumnValues(emlib.MDL_PICKING_PARTICLE_SIZE, df['boxsize'].tolist())
outMD.setColumnValues(emlib.MDL_SAMPLINGRATE, df['samplingrate'].tolist())
outMD.setColumnValues(emlib.MDL_TOMOGRAM_VOLUME, df['tomo_id'].tolist())

outMD.write(outpath)
```



	particleSize	ref	samplingRate	tomogramVolume	x	y	z
449	100	0	5,5000		603,8673	-471,8818	150,5627
450	100	0	5,5000		519,8673	245,1182	150,5627
451	250	1	2,1700		936,0000	-1668,0000	-153,5000
452	250	1	2,1700		306,0000	-1592,0000	-152,5000

Figure 3.6: Storing the intermediate data tables in Xmipp MD format allows for a friendlier user experience.

At the end of this step, the protocol's *extra* folder will contain a *pickedpertomo* subfolder with all of the per-tomogram picked coordinates **_allpickedcoords* in XMD format. Figure 3.6 illustrates how convenient is for the user to navigate through the generated XMD files just by clicking around in the Scipion GUI.

3.4.3 Coordinates consensus step

This step acts as a wrapper for the aforementioned coordinates consensus program developed inside of the Xmipp repository. The command to be run is constructed with the appropriate file paths and other parameters. It is then launched on the Xmipp side by means of the *xmipp_coordinates_consensus_tomo* Python script, which is called once per tomogram.

When launched, it will assess every tuple of coordinates and store them in XMD format, using the next logic:

- Create an empty sack S of coordinates
- For each candidate coordinate c...
  - If sack is empty: add c to S
  - If sack is not empty: for each consolidated coordinate cc...
    1. If dist(c,cc) <= threshold: assimilate c as part of the items represented by cc
    2. (if "centroid" option is active) Recalculate centroid for cc
    3. If dist(c,cc) > threshold: carry on with next consolidated coordinate
  - If c was not assimilated by any consolidated coordinate: c becomes cc
- For each consolidated coordinate cc...
  - If representedBy(cc) => threshold: add to "positive" input list
  - If representedBy(cc) < threshold: add to "doubt" input list
- Write all input lists to disk

The distance function to determine if two regions refer to the same one is modular, so that it could be changed in the future for a more complex one, and is originally defined as a the norm of the subtraction of both position arrays:

```
def distance(a: np.ndarray, b: np.ndarray) -> float:
    return np.linalg.norm(a-b)
```

At the end of this step, the *extra* folder will contain another folder called *coordconsensus*, with three subfolders *pos, neg* and *doubt* that will contain the consensus coordinates per tomogram, as well as a file with all of the consensus coordinates.

3.4.4 Prepare NN step

This step is meant to leave everything prepared for the execution of the neural network training and scoring. The metadata referring to the whole dataset is combined. Auxiliary functions *tomogramExtract()* and *noisePick()* are called right after the combination has been done.

The noise picking algorithm will generate a set of negative examples, as stated previously in this work, through the *xmipp_pick_noise_tomo* script. The extraction step function will then act as a wrapper of the *xmipp_tomo_extract_subtomograms* script and generate a .mrc file for each subtomogram, separated in *positive, doubt* and *negative* examples.

At the end of this step, the neural network has everything it needs to work.

3.4.5 Process train step

This wrapper gathers all of the needed parameters and launches the learning step of the process through the *xmipp_deep_picking_consensus_tomo* program. From within the Xmipp realm, the launched program will also make use of the created library: Network Manager or *NetMan*. As its name implies, it contains a class with functions capable of managing both the input data and the neural network. Models created with this library are capable of running on several GPUs at a time, ensuring the maximum usage of the resources for a faster execution. This is done through mirrored TensorFlow strategies.

After detecting the input flag that asks for a training session, the program will either load or create the required network, and feed all of the subtomograms (positive, negative and doubtful). This feeding must be done by using a *tf.data.Dataset* object, so it can be run in parallel. The data is prepared (shuffled and augmented as needed) using a Python Generator, and then wrapped by the *from_generator* method of the Dataset class. This ensures a correct parallelisation of the task.

The training step ends once the epochs are exhausted or convergence is reached, saving the best model into a file under the *extra/nn* folder.

3.4.6 Process score step

In a similar fashion as in the train step, *xmipp_deep_picking_consensus_tomo* is called, this time specifying that a scoring is needed to be done through the flags. All of the input subtomograms will be fed to the loaded network, but no noise will be picked or inserted. Finally, the scored consolidated coordinates results as well as the raw results will be written using the XMD format.

3.4.7 Post-process step

The files generated in the previous step contain the probability of being a good pick for the consensus coordinates as well as all raw input coordinates. During this step, the results that will be presented to the user are generated by means of filtering the data. The user is presented with the consolidated coordinates and not all the raw coordinates.

3.4.8 Create output step

After the results are prepared, this last function will save them into a final file and define the outputs, relating them to the original tomograms for Scipion to show them to the user via GUI. The user will then be able to create subsets and further process the data.

4
Testing and results

Several tests have been run during the development of the project. This section aims to explain how were these tests conducted as well their context and the obtained results. A discussion regarding the results and based on the opinion of CryoET experts is conducted at the end.

4.1 Real life case study

EMPIAR, the Electron Microscopy Public Image Archive[1], is a public resource for raw images underpinning 3D cryo-EM maps and tomograms. EMPIAR also accommodates 3D datasets obtained with volume EM techniques and soft and hard X-ray tomography. All data archived in EMPIAR can be re-used freely without any conditions or restrictions ("CC0" license model). Source: EMPIAR *About* page[2].

Proteins are essential components of all living organisms, performing a wide range of cellular functions. The synthesis of proteins is a complex process that begins with the genetic information encoded in DNA. The DNA sequence serves as a template for the production of messenger RNA (mRNA) molecules through a process called transcription, which occurs in the nucleus of eukaryotic cells. mRNA molecules carry the genetic instructions from the DNA to the site of protein synthesis.

Ribosomes play a crucial role in protein synthesis by facilitating the translation of mRNA into a linear sequence of amino acids, ultimately leading to the formation of a mature protein. Ribosomes bind to the mRNA decoding the information carried by nucleotide sequence. This decoding involves matching the sequence of three nucleotides, known as codons, with tRNAs (transfer RNAs) carrying the corresponding amino acids.

Through their intricate machinery, ribosomes ensure accurate and efficient protein synthesis, allowing for the diverse array of proteins necessary for the proper functioning of living organisms.

[1] `https://empiar.org/`
[2] `https://www.ebi.ac.uk/empiar/about/`

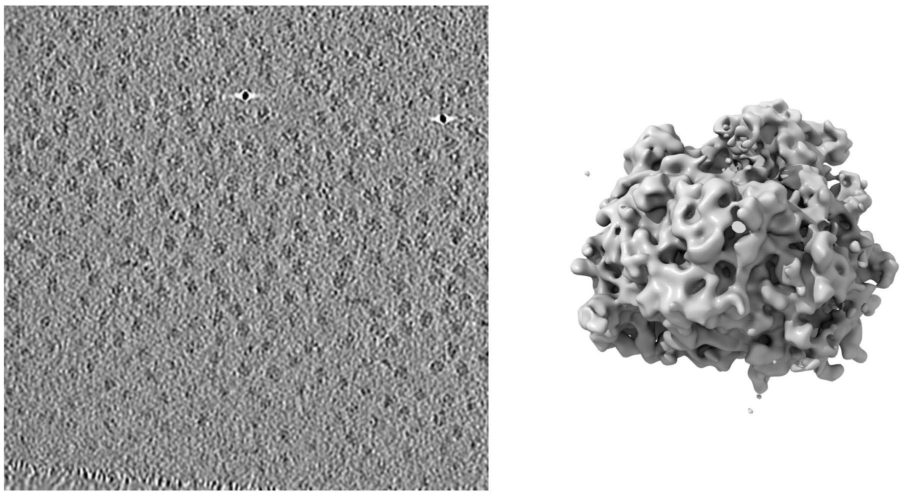


Figure 4.1: Tomogram slice (left) and reconstructed ribosome (right) from EMPIAR-10045.

Due to its importance and its relatively simple structure, ribosomes make a great testing structure for various processing pipelines.

The EMPIAR 10045 dataset[3] first appeared in late 2015 [21]. It comes from an experiment called *Resolving macromolecular structures from electron cryo-tomography data using STA in RELION*. The information contains specimens of 80S-type ribosomes from Saccharomyces cerevisiae cells (a common yeast also known as *brewer's yeast* or *baker's yeast*). As the title of the experiment suggests, the experiment was carried by means of cryoET techniques. Apart from the resulting structure[4] of the ribosome found in the Electron Microscopy Data Bank (EMDB), it also provides public access to raw tomogram data, shown in Figure 4.1. The interesting information in the context of this Thesis is:

- Microscope of origin: FEI Titan Krios 300kV
- Acquisition device: GATAN K2 Summit
- Resolution: 4K x 4K x 300-600 frames
- Size of reconstructed tomograms: 30-60GB per tomogram

4.2 Experimental setup

The Biocomputing Unit has several machines available for both developing software and running image processing algorithms. The early development has been done in the following laptop called **Pitts** kindly lent by the Biocomputing Unit for the whole duration of the project:

- Brand: Dell Alienware
- Model: 15 R4
- CPU: Intel Core i7-8750H 6-core 12-threads @ 2.20GHz

[3]`https://www.ebi.ac.uk/empiar/EMPIAR-10045`
[4]`https://www.ebi.ac.uk/emdb/EMD-3228`

- RAM: 32GB DDR4
- GPU: Nvidia GTX 1060M 6GB
- Disk: 1TB NVMe 3.0
- OS: Ubuntu 22.04 LTS desktop, kernel 5.19.0-41
- Compilation environment: CUDA 11.7 and GCC/G++ 11.3.0
- Python: python 3.9.13, conda 3.22.0

As the chosen dataset (and in general cryoET datasets) is very heavy at 30-60GB per tomogram, the experiments could not correctly run due to both system and GPU RAM issues even before the consensus was run. To overcome that, the final tests and execution were carried out in a bigger workstation called **Milgrom**[5] that consists of:

- Brand: AzkenMuga
- Model: T-Series Xeon SP
- CPU: 2 x Intel Xeon Gold 6230 20-core 40-threads @ 2.10GHz
- RAM: 384GB DDR4
- GPU: 4 x Nvidia RTX 2080Ti 11GB
- Disk: 2TB SATA SSD, 80TB SATA HDD
- OS: Ubuntu 18.04.5 LTS
- Compilation environment: CUDA 11.2 and GCC/G++ 8.4.0
- Python base: python 3.11.4, conda 23.5.2

This machine is powerful enough for any picking program to be run, and even several of them in parallel, allowing a faster cycle of development-testing to be established.

For the development, Microsoft Visual Studio Code[6] was chosen for its weightlessness together with its SSH feature to remotely work within the workstation and laptop from the convenience of the personal computer. Work and tests from the exterior were done using a VPN connection together with a VNC server running on both machines. This allowed easy movements between the developing and testing machines.

Even though the debugging (neural network building, xmipp script testing...) has been done outside of Scipion, all of the final experiments have been done inside of it. As the aim of the project is to provide an integrated solution, it only made sense to do it like so. The testbench consists of a Scipion project, as it can be seen in Figure 4.2.

Colours are used to distinguish the nature of each element of the project. **Light blue** indicates the imported data from the EMPIAR deposition. **Deep purple** is used for picking programs that are going to feed the protocol. **Grey** is for hand-picked coordinates. **Green** is for the picking consensus itself, while light green is used for subtomogram extraction protocols. **Lilac** is used for the posterior refinement activities used to provide the evaluation metrics and **light brown** is for the previous work needed to get the picking program *deepfinder* working.

[5]Thanks to A. Cuervo and P. Losana for lending this machine.

[6]`https://visualstudio.microsoft.com/`

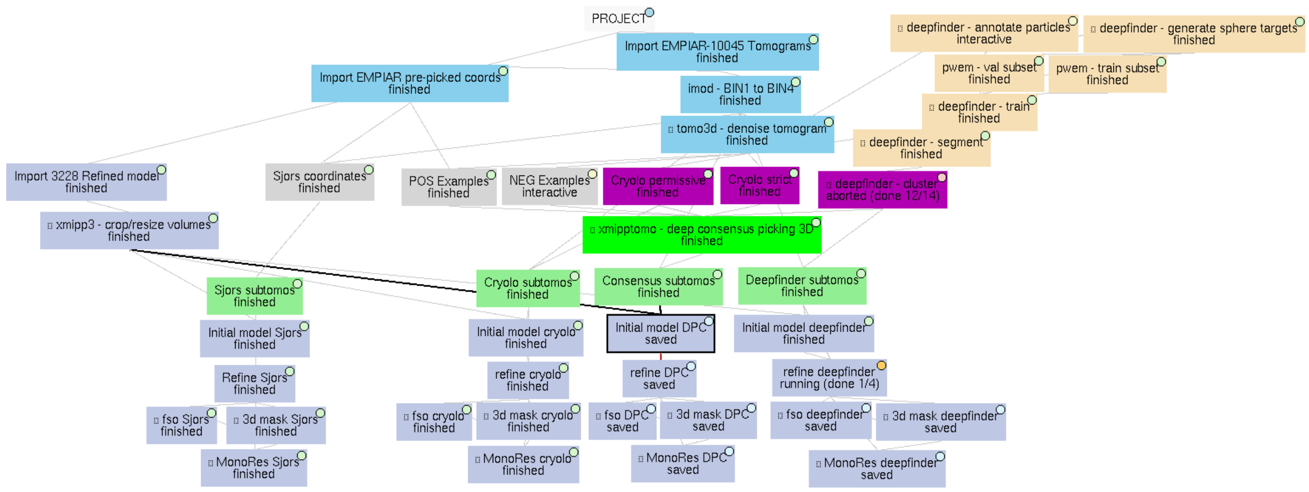

PROJECT
Import EMPIAR-10045 Tomograms finished
Import EMPIAR pre-picked coords finished
imod - BIN1 to BIN4 finished
tomo3d - denoise tomogram finished
deepfinder - annotate particles interactive
deepfinder - generate sphere targets finished
pwem - val subset finished
pwem - train subset finished
deepfinder - train finished
deepfinder - segment finished
Import 3228 Refined model finished
Sjors coordinates finished
POS Examples finished
NEG Examples interactive
Cryolo permissive finished
Cryolo strict finished
deepfinder - cluster aborted (done 12/14)
xmipp3 - crop/resize volumes finished
xmipptomo - deep consensus picking 3D finished
Sjors subtomos finished
Cryolo subtomos finished
Consensus subtomos finished
Deepfinder subtomos finished
Initial model Sjors finished
Initial model cryolo finished
Initial model DPC saved
Initial model deepfinder finished
Refine Sjors finished
refine cryolo finished
refine DPC saved
refine deepfinder running (done 1/4)
fso Sjors finished
3d mask Sjors finished
fso cryolo finished
3d mask cryolo finished
fso DPC saved
3d mask DPC saved
fso deepfinder saved
3d mask deepfinder saved
MonoRes Sjors finished
MonoRes cryolo finished
MonoRes DPC saved
MonoRes deepfinder saved

Figure 4.2: Scipion project deployed both in Pitts and Milgrom.

First, the 7 tomograms that make up the EMPIAR-10045 dataset (as well as the "ground truth" coordinates picked by Sjors Scheres) are imported into the workspace using *scipiontomo*. As the original 3D images are too big, they are reduced using *imod tomo preprocess* together with the ground truth coordinates. By applying a 4x binning factor, the dataset is reduced to 4.61GB. The original tomograms have a very low contrast, so in order to facilitate the automated picking a denoise algorithm is applied using *tomo3d denoise tomogram*. The comparison can be

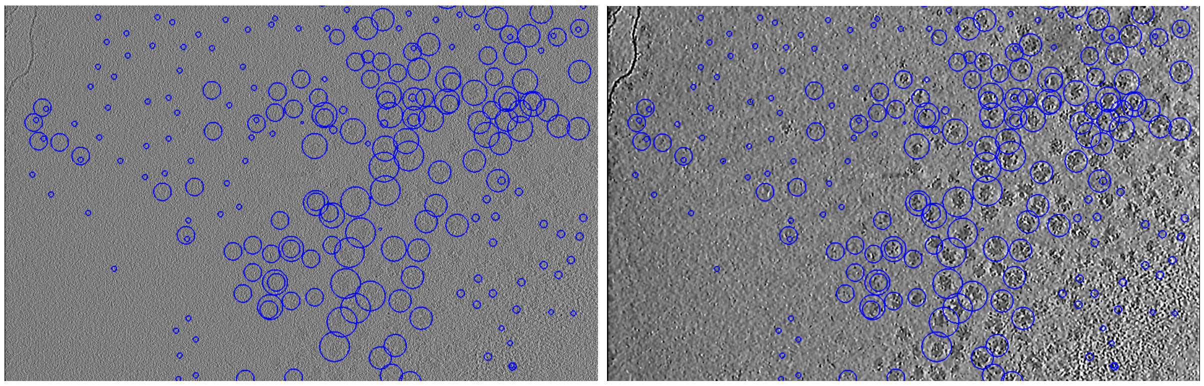


Figure 4.3: A higher contrast facilitates figure detection.

seen in Figure 4.3. This denoised data is fed to the different pickers. Once finished picking, the **deep consensus picking 3D** (the subject of this thesis) protocol is launched, with all of the coordinates as inputs. Finally, a set of protocols is launched in order to provide evaluation metrics. The subtomograms are extracted from the deep consensus using *xmipptomo* in order to be used for a reconstruction. As the picking in this project is still not directional, the 3-D data from EMDB-3228 (related to EMPIAR-10045) is also imported, to provide a guide on the reconstruction of the initial model. The final model obtained from only the hand-picked coordinates is compared to the one given by the consensus picking and by single automated pickers by means of the the visual analysis of the resolution maps obtained from the MonoRes protocol *(more on that in Section 4.3)* and a human inspection.

The experiment consists on the reconstruction of the target structure parting from three scenarios, all using the 7 tomograms from EMPIAR-10045. One of them with the coordinates coming from cryolo, another with data coming only from deepfinder, and finally with the coordinates consensus output coordinates. The amount of coordinates in each set is (respectively) 3.600, 14.432 and 24.238. The final result metrics come from the consensus protocol being fed the set of 24.000 coordinates, but tests have been done with 5.000 and 10.000 elements to see if the algorithm scales properly.

4.3 Metrics

It is a hard task to determine which are the metrics that can be used as success markers for the experiments, as their election is subjective. The following guidelines were proposed in conjunction with experts in the field of cryoEM imaging:

- **Ground truth assumption** - It is assumed that the manual pickings made by professionals are completely correct. This includes the original pickings obtained from the EMPIAR-10045 dataset, as well as pickings done by cryoET experts at the BCU-CNB.
- **Results review** - Once the picking consensus is done, a team of experts reviews the output. The resulting pickings as well as the reconstructions from the different branches of the protocol will give an insight on how good the picking was. The opinion of experts on results is a widespread metric in the field.
- **Fourier Shell Correlation** - The FSC[7] measures the quality of the reconstructed 3D model. This is done by dividing the set of subtomograms in two equal-sized subsets, doing independent reconstructions and performing a cross-correlation calculation between them two in the Fourier space. This allows to discern between the invariant higher resolution features of the model and noise, and gives a global number for the resolution.
- **Fourier Shell Occupancy** - The FSO [22] is a novel measure of resolution anisotropy for CryoEM 3D maps. The Scipion protocol for FSO calculation outputs a curve for a given 3D model, describing the distribution of the resolution in the Fourier space. The curves obtained from different models can be compared to determine which reconstruction is better.
- **MonoRes local resolution map** - MonoRes [23] is an algorithm used to estimate the local resolution in electron density maps. It can give an image of the resolution along all of the reconstructed model in a graphical way.

[7] `https://en.wikipedia.org/wiki/Fourier_shell_correlation`

4.4 Results

Picked coordinates

The amount of picked coordinates for each of the tested pickers are: 3.120 (hand-picked), 3.666 (cryolo), 14.436 (deepfinder) and 3.616 (consensus with a minimum prediction probability of 0.75). During the execution of the coordinates consensus, more than 6.000 duplicate coordinates were detected and combined.

Time

	N	**GPU**	**T(min)**	**SpdUp**	**GPU-Eff**	**CPU**	**T(min)**	**SpdUp**	**CPU-Eff**
M	10.000	1	12	1,00	1,00	20	12	1,00	1,00
M	10.000	2	10	1,2	0,6	40	10	1,2	0,6
M	10.000	4	10	1,2	0,3	80	7	1,71	0,42
L	24.000	1	31	1,00	1,00	20	31	1,00	1,00
L	24.000	2	27	1,14	0,57	40	20	1,55	0,78
L	24.000	4	23	1,34	0,34	80	16	1,93	0,48

Table 4.1: Execution times, speed-up and efficiency for different GPU (fixed at 20 CPUs) and CPU (fixed at 1 GPU) configurations for a 50x50x50 input.

Time measurements have been taken in several scenarios. Table 4.1 shows the observed execution times when using different amount of GPUs and a fixed amount of CPU threads on the left side (20), while on the right it shows the speed-up and efficiency with different amount of CPU threads but a fixed number of GPUs (1).

Raw FSC/FSO numbers

Figure 4.4 shows the FSO curves obtained from the three different reconstruction branches. In terms of FSC (with cutoff at cross-correlation value 0.5), the obtanied numbers are: 21.0Å (hand-picked), 21.0Å (cryolo), 19.20Å (deepfinder) and 19.10Å (consensus).

Local resolution (MonoRes)

Histograms in Figure 4.5 show how much of each of the reconstructed maps is at which resolution. This can be interpreted visually with the plotting of the coloured resolution map, either by slices (like in Figure 4.6) or in 3D space (like in Figure 4.7).

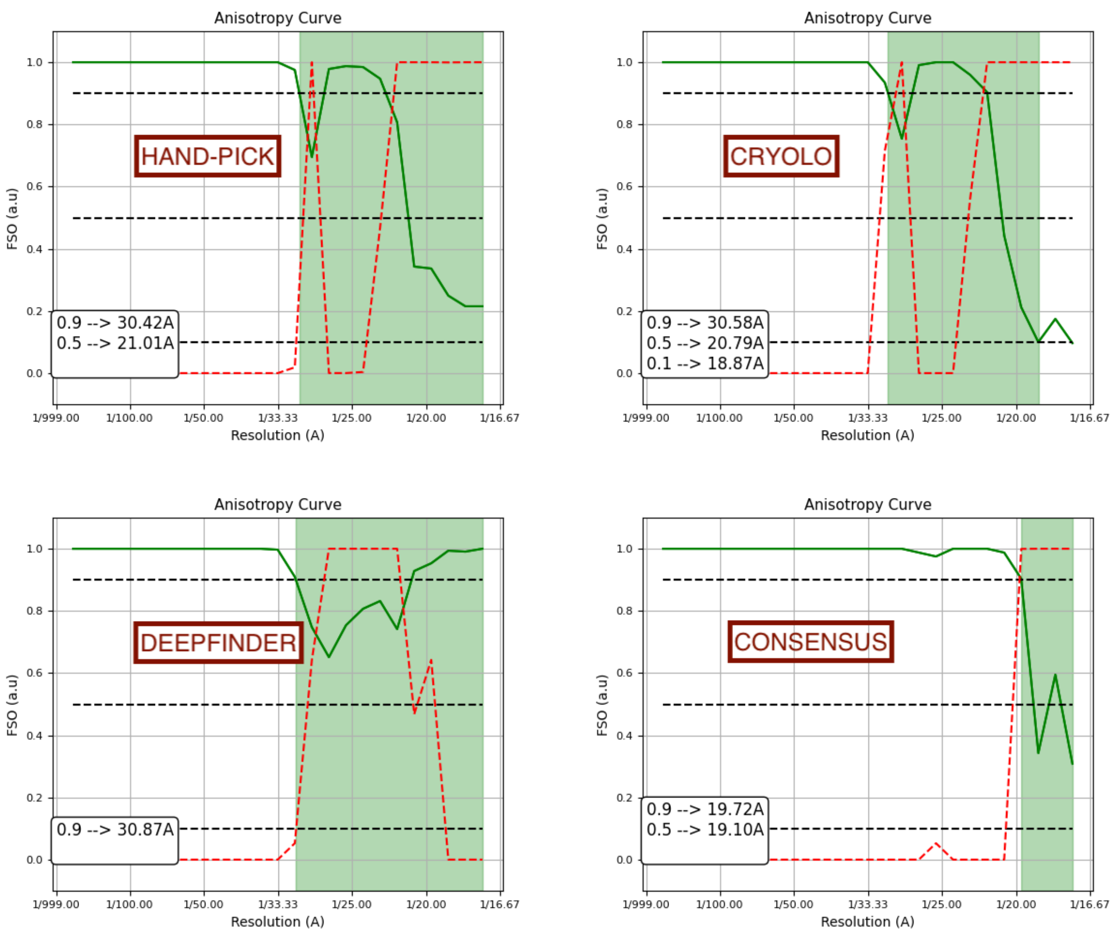


Figure 4.4: FSO curves of the different reconstruction branches.

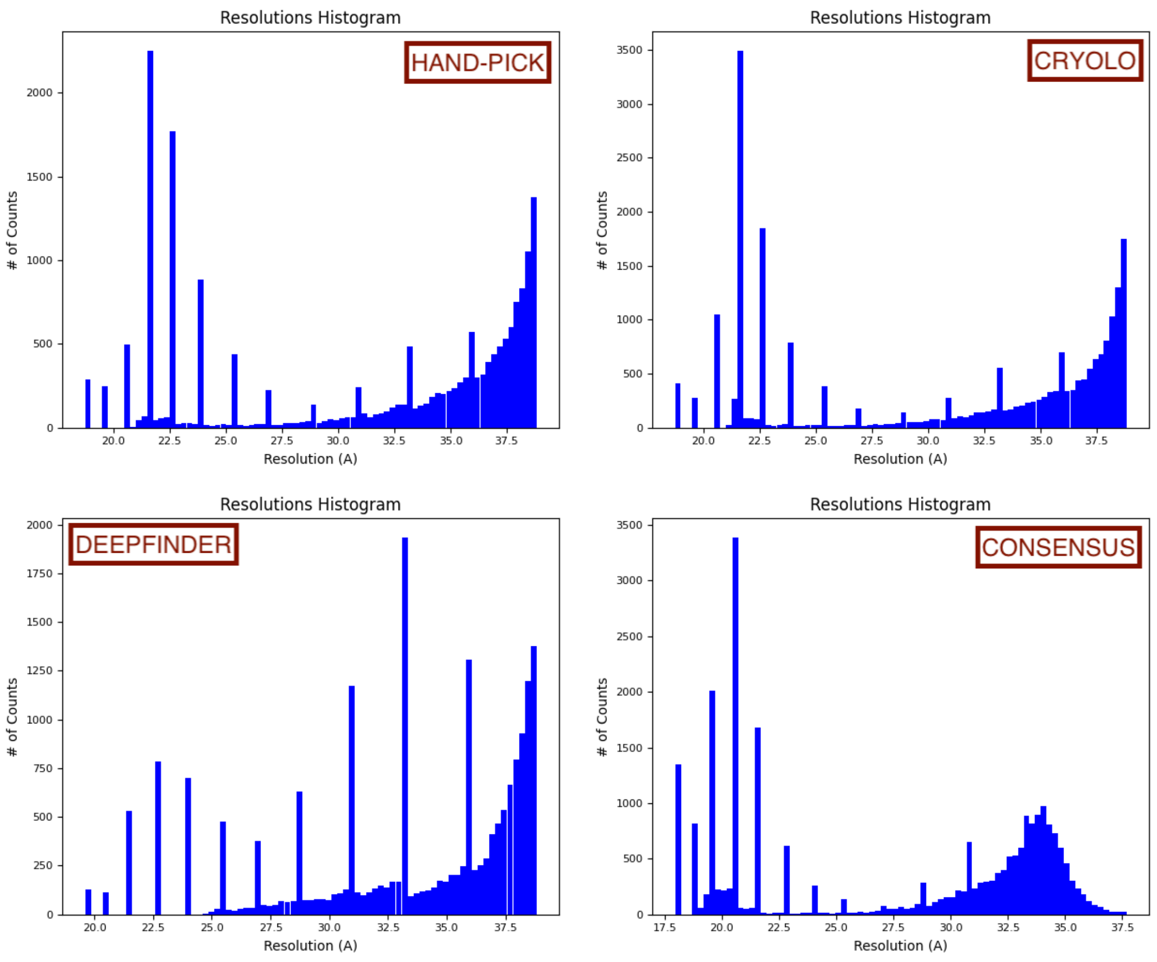


Figure 4.5: Resolution histograms of the different reconstruction branches.

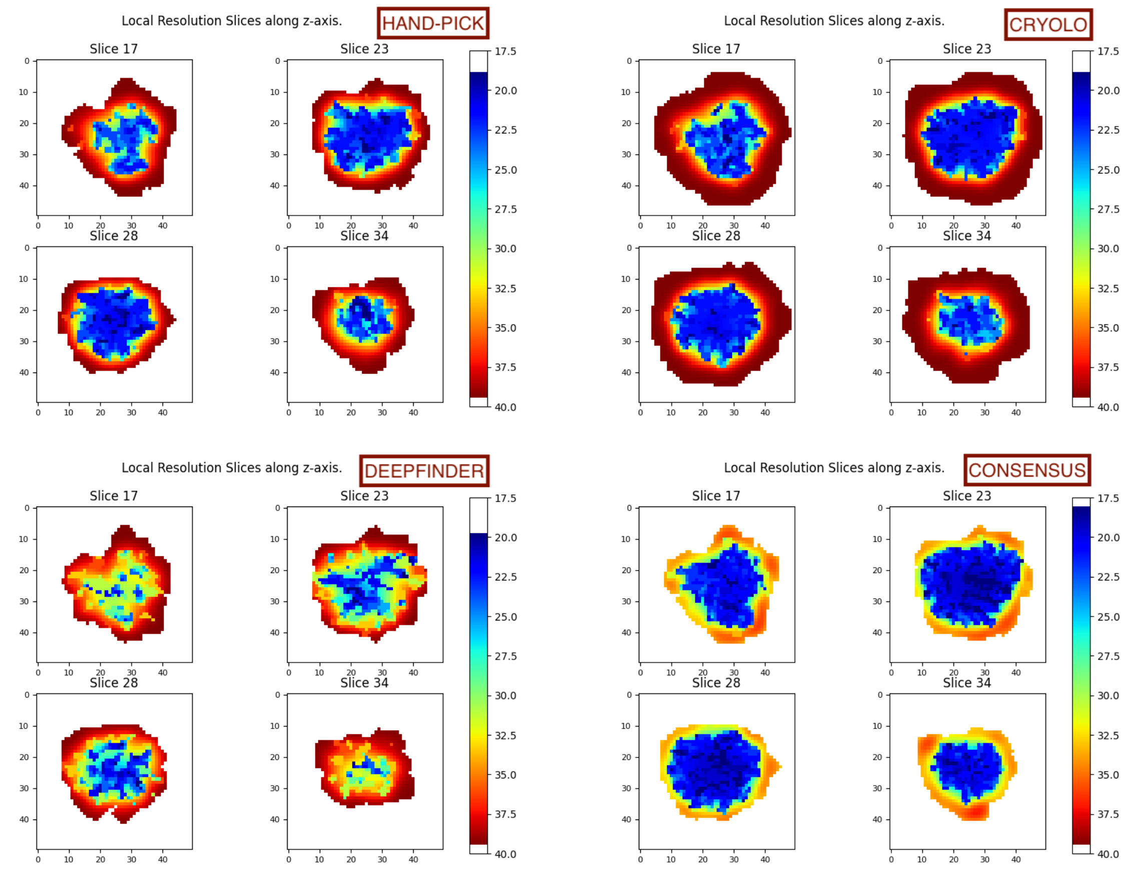


Figure 4.6: Resolution by slices of the different reconstruction branches, in Angstroms.

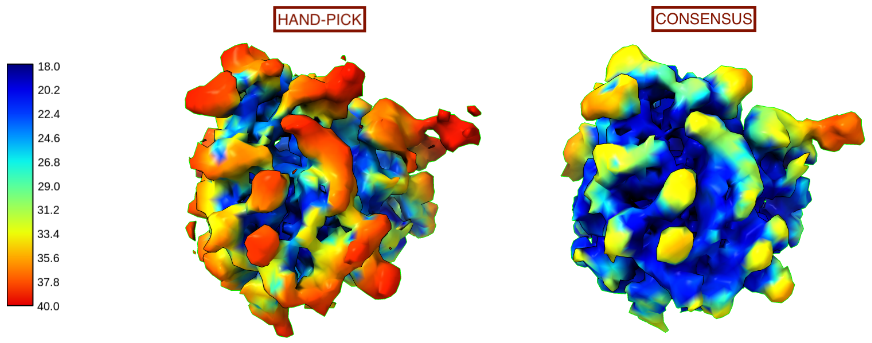


Figure 4.7: Visualisation of the local resolutions of the reconstructed volumes from manual and consensus pickings, as shown in ChimeraX.

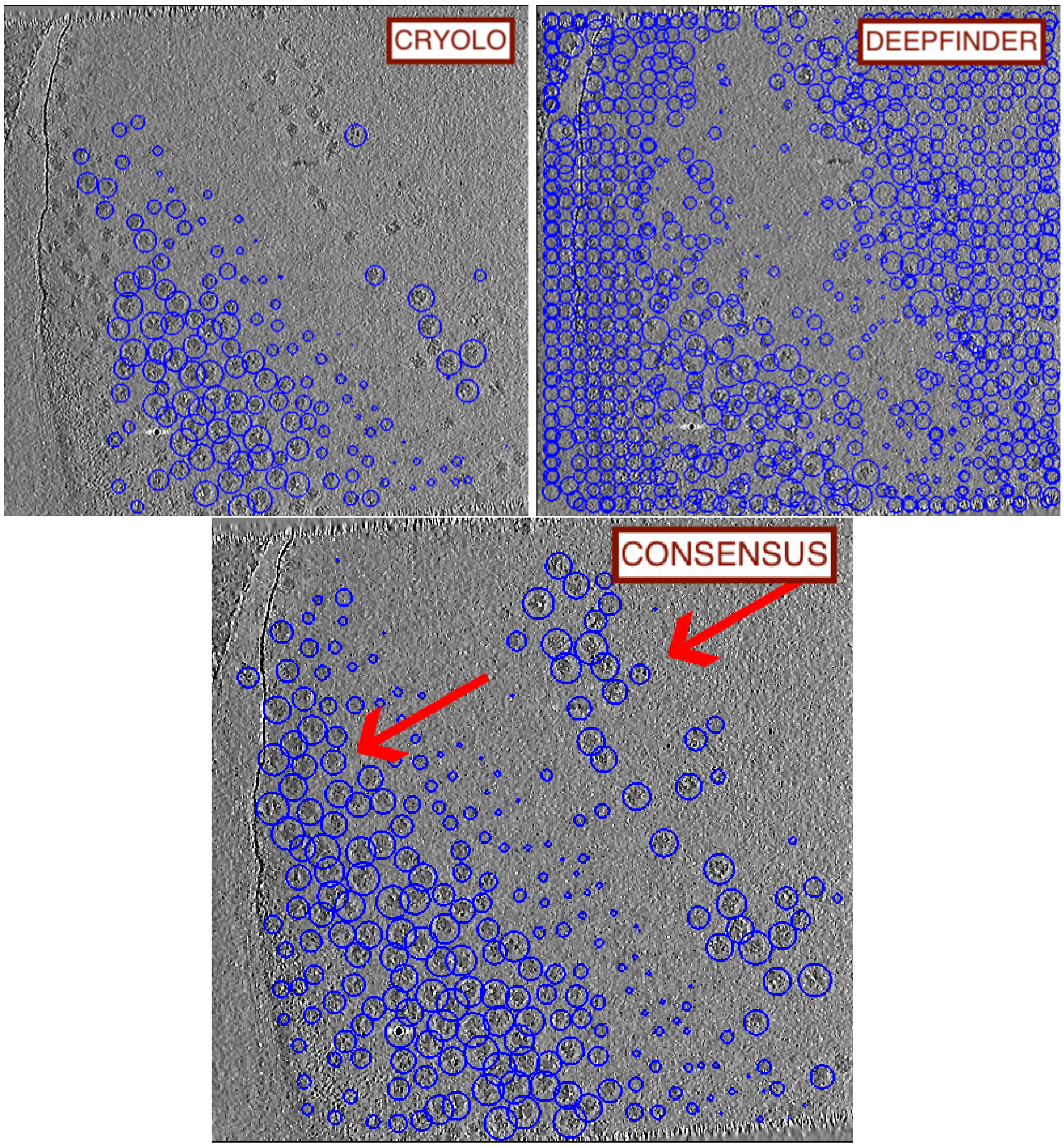


Figure 4.8: Picked coordinates in the same tomogram and same slices from the different reconstruction branches. Arrows in red indicate the pickings missed by cryolo but not the consensus.

4.5 Discussion and results conclusions

The following discussion has been conducted with experts in CryoET and CryoEM. Any of the following subsections should not be used alone to emit a verdict on the consensus. Only the combination and balance analysis between the different measured parameters are the key to understanding how good (or bad) is the developed product.

4.5.1 On the execution time

One of the most interesting factors to look at is the elapsed time needed for the protocol to conclude. Picking is itself a very time-consuming process, so it is important that the training and scoring process does not take too much time or it will lose its sense. The observed mean execution times can be found on Table 4.1 along with two different input sizes. The experiment labelled as *size L* corresponds to the amount of subtomograms used regularly for real-life projects, so it is the one under discussion. Compared to the execution times of the pickers (cryolo: 3h, deepfinder: 15h), the 30 minutes spent by the consensus picking do not pose a risk to the attractiveness of the product.

On the compute scalability side, it is clear that the protocol gets much more of a boost from more CPU cores than from adding more GPUs. In fact, the speed-up of using additional GPUs and its related efficiency are very poor. The preprocessing code should get more attention that the neural network system in future performance analyses, such as the auxiliary programs for 3D coordinates consensus and noise picking which still work on a single thread. The noise picking algorithm does also play a crucial role in the execution time. The algorithm struggles to find noise particles in densely populated tomograms, so the problem resolution time is not deterministic.

As a conclusion on this matter, the time spent is reasonable and will affect positively on the amount of end-users that will be attracted to it, but it can be improved.

4.5.2 On the raw resolution numbers (FSC, FSO)

The FSO curves and the FSC should not be interpreted alone by themselves while judging the correctness or quality of a model. The quality of a model can be poor while having a good FSC resolution number.

On the side of the **FSO**, the desired curve is the one that "stays up" as much as possible along the resolution axis and declines very quickly when it does, as this means that the highest percentage of the model is represented at higher resolutions. Figure 4.4 shows how the reconstruction from the consensus picking is superior to any other method alone, as the curve stays up until the range of 20 Å and then declines very quickly. This elongated upper line together with the very narrow decline band (the one with green background) seen in Figure 4.4 speaks about the advantage of the consensus over the other methods. The **FSC**, on the other hand, corresponds to the FSO curve at cut point 0.5, and is widely used as a metric of the overall resolution of the model.

The histograms from Figure 4.5 also support the statement of the consensus being superior. While the other three methods accumulate low-resolution fractions of the model at the right tail, the consensus manages to keep it down. Also, the amount of better resolution fractions of the model is much higher in the consensus, as it can be seen in the counts for resolutions lesser than 21Å, which are not as populated in the other methods by far.

Moreover, as a result of the application of the Nyquist-Shannon theorem[8], the maximum achievable resolution would be at much 2x the sampling rate (8,68Å/px). That is to say, 17,36Å. The FSC from the consensus stops at 19,10Å, which is 1,75Å away from this limit.

Good FSO curves and high FSC resolutions are a good sign, and together with the visual analysis done in the next subsections confirm that the quality of the obtained reconstruction is superior to the one from other methods.

4.5.3 On the picked regions

One of the main reasons for this project to exist is the combination of the good pickings from several inputs, while leaving their bad elections out. Figure 4.8 shows the output of two pickers on the top part - cryolo and deepfinder - while the bottom presents the combined output from the consensus picking. The scene can be seen in any of the remaining tomograms: the consensus keeps the good pickings from cryolo, which was more conservative, and discards the noise picked by deepfinder, which was excessive in its work. As a downside, the consensus still inserts gold beads and some high-contrast artefacts into the final coordinates, as it is not given many negative examples to avoid them.

The consensus succeeds at effectively combining the outputs of different pickers, keeping only the best parts of each and every one of them.

4.5.4 On the reconstructions

The obtained structures can be compared between each other, as the processing done in all branches of the tests was the same. Figure 4.6 shows the shape and resolution distribution of the images. The consensus reconstruction has a higher quality, with much better overall resolution, and specifically in the outer shell of the ribosome. Figure 4.7 illustrates the difference in 3D, where colours for higher resolutions are much more present in the consensus reconstruction than in the hand picked one. This means that the developed product is able to generate better results than any other method alone by itself.

Still, the resolutions are not the best, and the experiment could be refined more from start to finish to see which is the limit of the consensus picking.

[8]`https://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem`

5 Project conclusions and future work

5.1 Conclusions

This project culminates with the creation of a picking consensus program for electron cryotomography inside of the Scipion plugin *scipion-em-xmipptomo*. Together with experts in the matter of CryoEM processing, some insights can be provided as to the utility of the project:

- **Time consumption** - This protocol is not a substitute of any other program but an addition to the CryoET pipeline inside Scipion. Thus, the sense of using it or not will depend on the complexity of the studied structures. Simpler (from the perspective of picking) images might not need an ensemble of pickers to find structures, while bigger projects with complex backgrounds can benefit from it. ***The picker takes a short time to execute in the context of common tomography workflow times.***
- **Usefulness** - Being a novel approach to validation in cryoET, the protocol has proven to be a useful piece of software, although the accuracy and correctness can still be improved (removal of gold bead picking is pending).***The protocol effectively aids in the elimination of unwanted pickings in an automated manner.***
- **Stronger together** - the enhancements obtained while using this protocol show that there is no perfect program for picking. The flaws of each and every picker can result in worse classifications but they can be solved with the combination of results by means of neural networks. ***The protocol enhances the achievable results when compared to single picking programs.***
- **Neural networks help** - The use of neural networks over simple voting mechanisms introduce an enhancement: a bad region that is picked by more than one (or multiple times by one) picker is later discarded by the neural network due to its score, while in voting mechanisms it would still be in the output set. ***The neural networks suppose an improvement over simple voting mechanisms.***

Thus, it can be concluded that the project *Improvement of the validation capability for*

CryoET structure of interest detection workflows inside of Scipion was successful at reaching its aims and establishing a line of action for the future.

5.2 Future work

Due to the nature of the work, its future is quite limited once the original set of requirements is fulfilled. Nevertheless, there are three clear paths of future work for this particular project:

Picky Blinder

- **Picky Blinder** (DeepConsensus 3D V2) - this is already planned as it is the natural way of maintaining the created piece of software. It includes the incorporation of new compatible pickers to the protocol, as well as further applying enhancements in a similar manner to what was introduced in the Task 5 of this project. This includes fine tuning of the GUI for a better user experience, neural network hyperparameters auto-tuning, CPU optimisation for the auxiliary programs (noise picking and 3D coordinates consensus still run in a single thread), further automation, offering of pre-trained models and noise examples, image filters, further incorporation of the Scipion team coding philosophy... As well as features left out in this Master's Thesis due to time constraints or new ones such as incorporating membrane/directional picking support.

PICKEL

- **Pickel** - The neural network developed for the DeepConsensus 3D project and its knowledge could be used as the base of a fully fledged tomogram picker. This picker would be fully autonomous.

TomoMocho

- **Deep TomoMocho** - This naturally comes by following the effort of "porting" all existing capacities found in Xmipp for Single Particle Analysis (SPA) to tomography. Segmenting the picking space into good and bad (carbon, bad-contrast, contamination) regions would help ruling out incorrectly picked coordinates.

6
Project Management

The amount of information and tasks generated during the lifespan of a project can be overwhelming if not carried correctly. Thus, project management plays a vital role in maintaining a working order and assuring the correct advancement of the project itself.

6.1 WBS Diagram

The ***Work Breakdown Structure*** diagram is a tree in which the project itself is the root. The second level of the tree contains the main phases of the project, and the following levels are comprised of the sub-tasks of each phase. This projects WBS diagram can be seen on 6.1 and contains the basic and simplified structure of the work.

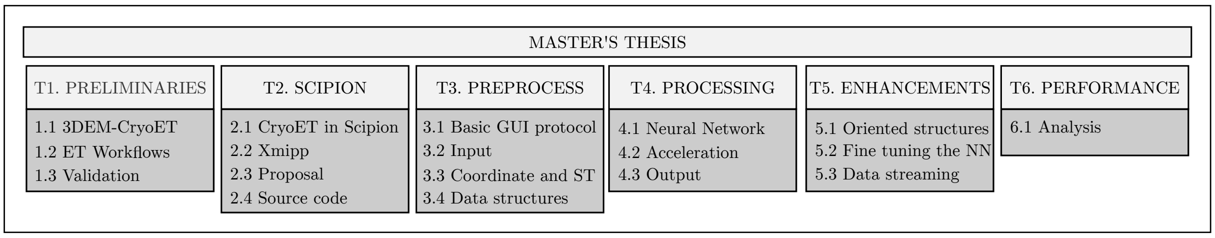


Figure 6.1: Work Breakdown Structure of the Master's Thesis.

Each of the tasks shown in the WBS diagram is further described in the next section, giving a more detailed insight of the defined requirements.

6.2 Work packages and their tasks

6.2.1 Preliminary Study

Coming from the field of Informatics Engineering, there is a big amount of concepts and lessons to be learned before starting the actual work. In cases like this, where two disciplines are merged in a project, it is very important to acquire the needed knowledge in both of them. This work package is devoted to that knowledge gain on the field of structural biology, microscopy, electron microscopy, cryogenic microscopy, and finally electron cryotomography. During this phase, the different parts of the typical CryoEM and CryoET workflows will be analysed. This learning includes the material from the Master's ***Procesamiento de Imágenes Biomédicas y sus Aplicaciones*** subject as well as training received from BCU personnel. This part is considered done once there is a basic knowledge on the biological basis of CryoET, the workflows and the programs used in every step, and concludes with the presentation of the problem to be solved.

6.2.2 The Scipion Framework

Scipion is quite a popular processing framework among structural biologists - more precisely among people involved in electron microscopy. Although it is a friendly tool for end users, its innards are more complex from the point of view of a developer, as it is formed by several sub-projects that generate a confluence of C/C++, CUDA, Java and Python code. Learning the internal structure is important as the code written in this project must be integrated in the correct way with the rest of the system. This part is considered done once there is a knowledge on the internal structures of the Scipion and Xmipp code, as well as the interaction between the different parts that partake on its functioning. Also, this part needs a functioning GUI interface for the protocol to be created and accessible to be deemed as finished, and concludes with the proposal of the implementation of the meta-picking protocol. Enhancements to the GUI's user adaptation would be made afterwards with the knowledge gained in the ***Sistemas Adaptativos y Modelado de Usuario*** subject.

6.2.3 Preprocessing code development

The inputs from the different picker programs can be very varied. Thus, the protocol developed in this project must include three preprocessing layers: one that converts every format into common ones, another that detects duplicity in the inputs and assigns a centroid for each structure of interest that is represented by different coordinates, and a final one to extract the sub-volumes from the tomogram and gives it to the processing part of the pipeline. This part is considered **almost** done once the aforementioned code is developed and tested to be working. For this part to be **fully** done, the preprocessing must have the ability to be run either sequentially or in parallel (or if it's not possible, it must be correctly justified why).

6.2.4 Processing code development

Once the coordinate table is processed and the subtomograms are extracted and stored as 3D images, they can be fed onto a neural network. The processing code must set up this neural network, train it by splitting the input dataset (train and validation) and adding some positive and negative examples, save and load the weights and other parameters and be worked against later with other datasets. This part is deemed **almost** finished once an initial acceptable scoring is obtained with the Ribosome dataset, and a readable output is generated for Scipion to show it to the user and be transferred to other protocols for further processing. For this part to be **fully** finished, the processing step must be able to be run in Nvidia GPUs. Techniques to process data in parallel were learnt from the ***Cálculo Intensivo y Manejo de Datos a Gran Escala*** subject from the Master.

6.2.5 Enhancements

This section includes some desired enhancements that would make the protocol more useful for the real-life user:

- Some structures are floating in the cells, whilst some others might be attached to a membrane or other kind of cellular body. In the later case, some pickers can provide some information on the orientation of each picked structure inside of the tomogram, relative to the body they are attached to. This is very useful information as it allows some alignments to be done before feeding the images to the neural network, providing (potentially) much better scoring results. This goal would be achieved by implementing directionality management at the preprocessing step.
- Neural networks and their complex parameters (called *hyperparameters*) are still a mysterious black box for the majority of the scientific community. Nevertheless, some of the parameters can be tuned to better adjust it to the type of data they will be working with. This goal would be achieved by finding a way of optimising the hyperparameters. Knowledge from the ***Aprendizaje Automático: Teoría y Aplicaciones*** subject was used in this task.
- One of the raising use cases for Scipion is the processing of data as it is being captured in a microscope. This goal would be achieved by adding the capacity of working against a dynamic dataset instead of a fixed one.

6.2.6 Performance assessment

The balance between accuracy and processing speed must be bore in mind as one of the main goals of Scipion is to provide useful results - in a considerate amount of time. Taking into account that a previous unknown amount of time has been spent in each of the pickers that act as inputs, this protocol must be able to work efficiently to justify its use. If the program is too slow, it is most probable it will fall into the *"works well but takes too long"* category of software and not be used at all. This part is considered done when the performance assessment is done and reviewed by BCU personnel expert in the field.

6.2.7 Meta-management

This is a transversal part in the project. It includes all of the tasks needed to define and control the advance of the project. Its goal is to make sure the work packages are completed. Due to its cross-package nature, it is not contained in the WBS diagram, even though it is important. The most notable parts of this package are:

- **Documentation and meta-documentation:** This task includes all of the written material - internal Scipion code documentation/comments, guides and this document itself.
- **Code/information repositories:** This task regards the creation of the two main GitHub structures needed for the project - the LaTeX document personal repository and the needed branches in the xmipp and scipion-em-xmipptomo I2PC repositories.

Concepts from ***Dirección y Gestión de Proyectos Científicos y Tecnológicos*** and ***Iniciación a la Investigación y la Innovación*** were used in this task.

6.3 Time estimation and deviations

An initial estimation of the time needed for each defined task is performed in order to evaluate and adjust the scope of the project in an easier way. This estimation *(a priori time)* is found in Table 6.1 and presents it next to the real measurements *(a posteriori time)*.

The project was planned from winter 2022 to summer 2023. Figure 6.2 shows the time milestones for each part of the project in a Gantt *fashion*.

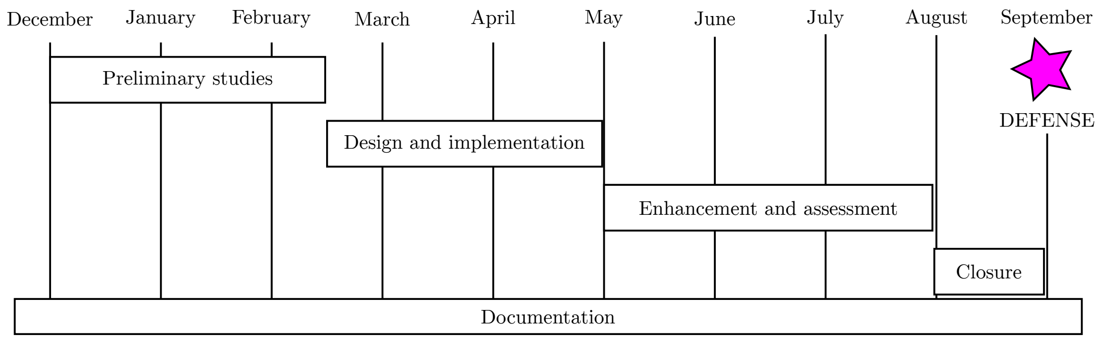


Figure 6.2: Simplified Gantt diagram of the Master's Thesis.

6.4 Deviation analysis

Table 6.1 shows a 26 hour time deviation from the originally expected 300 hours. Even though deviations were in mind since the beginning, 10 out of 22 tasks have been affected. As multiple factors - both internal and external - have affected the completion time of several tasks, and this section is devoted to analysing them one by one.

6.4.1 Minor deviations

This subsection includes those work packages with a deviation of 5 hours or less. The affected packages and their comments are the following:

- T1.1 Introduction to 3DEM and CryoET (+5h) - Even with a introduction to 3DEM during the Master's course, structural biology, 3D electron microscopy and specifically electron cryotomography are very complex, with a lot of biological and technical concepts that seem to be easy to understand under a shallow perspective, but take some more (much more) time to learn when trying to have a deeper grasp of the field for an interdisciplinary project.

- T1.2 Introduction to CryoET workflows (+3h) - Similarly to what happened with T1.1, the difficulty of learning what is really done in each of the steps of the CryoET workflow was underestimated. Studying how errors can happen in each step and their effect on later steps of the process has proven to be a bit harder than expected, as different programs work in a similar but not equal way.
- T3.1 Create a protocol accessible from Scipion GUI (+5h) - In order to make a program able to show a GUI in Scipion, several existing projects have been checked to get a better idea on how the developers usually do it. There are general guidelines, but not strict standards on how to write a Scipion protocol. Thus, more time was spent on comparing different approaches found in the checked existing protocols.
- T3.2 Input data conversion and conventions (+5h) - There were some basic errors on the original study of the code and the way it handles the data, leading to inconsistencies between what was thought and what was observed. This meant further investigation was needed to clarify the differences and come to a consensus.
- T3.4 Internal data structures (+5h) - *In the same fashion as in T3.2*, errors in the initial understanding of the internals of Scipion and Xmipp brought some inconsistencies.
- T4.3 Output data conversion and conventions (+5h) - Initially, this package was meant to generate a correct output containing all of the needed information. But a mistake was made: the output had to be read by next steps of the process, so the output information did not only have the requirement of being in a single format and with conventions, they also had to follow some rules for the next programs in the pipeline to be able to read them.
- T5.2 Fine-tuning of the neural network hyperparameters (+5h) - Techniques such as *grid search* can be computationally heavy. Running the tests and waiting between executions introduced a notable overhead responsible for this deviation.

6.4.2 Major deviations

Other work packages have suffered a difference in the completion time of 10 hours or more. This is considered a major deviation end the affected packages and their comments are the following:

- T5.1 Support for oriented structures (-15h) - What originally was thought to be a moderately hard and needed implementation task ended up being not very relevant. It ended up "only" being studied and left for a future update of the protocol out of the scope of this Master's Thesis.
- T5.3 Data streaming (-15h) - After inspecting some protocols with streaming capabilities it seemed like a hard task. This was not the case as "streaming" in the end just meant to introduce the code logic for the protocol to keep waiting for new inputs based on data timestamps and stopping when a threshold or signal were raised.
- Code performance analysis (+10h) - Time by itself was not regarded as sufficient for a performance analysis. As experts processing were introduced into the task to get their

input on the measured times an overhead was introduced due to intermediate result reports preparation, meetings...

6.4.3 External factors and deviation conclusions

Even though each task time deviation has its own reason, another culprit is found when looking at the bigger picture. This work was originally designed while working a 20h/week, but it was executed while working a 40h/week job. This introduced a higher workload that required more time, slowing down the advancements in the Master's Thesis development.

In general, it can be said that all of the minor deviations were caused due to the lack of previous knowledge on the field of biocomputing as well as the Scipion source this project interacts with and integrates into. Some cuts were done in the field of enhancements due to the already suffered slippages in other previous work packages and because of their optional nature. A lot of time was also spent doing installation tasks. The status of the whole cryoET ecosystem is **beta**, so every week there was a need of merging the project with the newest changes in the surrounding programs. Personally, I do not consider this a disastrous deviation, as I already expected a time excess in doing an interdisciplinary work in which I was lacking a big part of the knowledge.

Even though there is a high number of tasks that have suffered from deviation, it is not worrying as it is concentrated in "minor" deviations, as it can be seen in 6.3. Approximately two thirds of the deviations are classified into the category of minor.

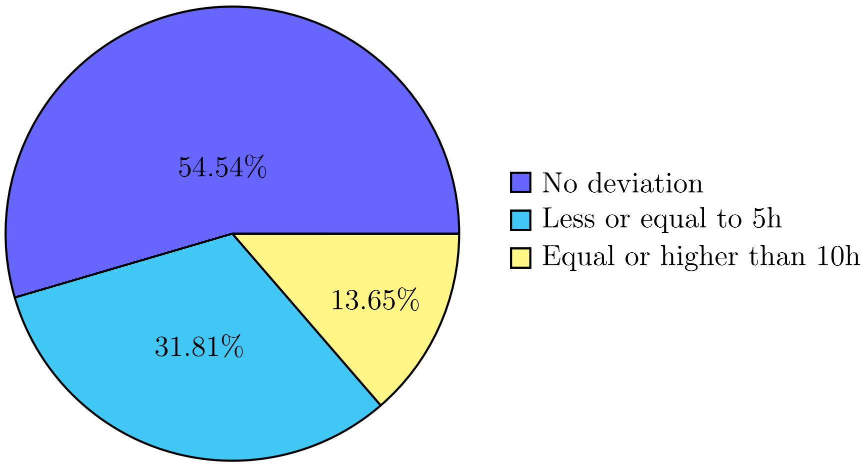


Figure 6.3: Pie chart showing the distribution of the deviations.

	Estimation (h)	Real (h)	δ(h)
T1. PRELIMINARY STUDY	30	38	+8
T1.1 Introduction to 3DEM and CryoET	20	25	*+5*
T1.2 Introduction to CryoET workflows	5	8	*+3*
T1.3 The problem of multi-picking and validation	5	5	-
T2. THE SCIPION FRAMEWORK	35	35	-
T2.1 Study of the CryoET tools in Scipion	10	10	-
T2.2 The Xmipp image processing package	2	2	-
T2.3 Proposal of the meta-picking protocol	3	3	-
T2.4 The Scipion source code	20	20	-
T3. PREPROCESSING CODE	75	90	+15
T3.1 Create a protocol accessible from Scipion GUI	15	20	*+5*
T3.2 Input data conversion and conventions	15	20	*+5*
T3.3 Coordinates consensus and STA	40	40	-
T3.4 Internal data structures	5	10	*+5*
T4. PROCESSING CODE	50	55	+5
T4.1 Initial neural network model implementation	40	40	-
T4.2 GPU acceleration	5	5	-
T4.3 Output data conversion and conventions	5	10	*+5*
T5. ENHANCEMENTS	60	35	-25
T5.1 Support for oriented structures	20	5	*-15*
T5.2 Fine-tuning of the neural network hyperparameters	10	15	*+5*
T5.3 Data streaming	30	15	*-15*
T6. PERFORMANCE ASSESSMENT	10	20	+10
T6.1 Code performance analysis	10	20	*+10*
PROJECT MANAGEMENT	40	43	-
Code documentation	2	5	-
Code/information repository maintenance	3	3	-
Dissertation document redaction	30	30	-
Slides/defence preparation	5	5	-
Total	**300**	**326**	

Table 6.1: "Estimated vs real time" comparison for each work package group. 12 ECTS correspond to 300 hours.

Bibliography

[1] H.-W. Wang, "Current status and future perspective of cryo-electron microscopy in structural biology," *Science China. Life sciences*, vol. 44, 04 2015.

[2] P. R. Baldwin, Y. Z. Tan, E. T. Eng, W. J. Rice, A. J. Noble, C. J. Negro, M. A. Cianfrocco, C. S. Potter, and B. Carragher, "Big data in cryoem: automated collection, processing and accessibility of em data," *Current Opinion in Microbiology*, vol. 43, pp. 1–8, 2018, environmental Microbiology * The New Microscopy. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1369527417301315

[3] W. Kühlbrandt, "The resolution revolution," *Science (New York, N.Y.)*, vol. 343, pp. 1443–4, 03 2014.

[4] T. Nakane, A. Kotecha, A. Sente, G. McMullan, S. Masiulis, P. M. Brown, I. T. Grigoras, L. Malinauskaite, T. Malinauskas, J. Miehling, L. Yu, D. Karia, E. V. Pechnikova, E. de Jong, J. Keizer, M. Bischoff, J. McCormack, P. Tiemeijer, S. W. Hardwick, D. Y. Chirgadze, G. Murshudov, A. R. Aricescu, and S. H. Scheres, "Single-particle cryo-em at atomic resolution," *bioRxiv*, 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/05/22/2020.05.22.110189

[5] P. Mahdavi Sharif, M. Nematizadeh, M. Saghazadeh, A. Saghazadeh, and N. Rezaei, "Computed tomography scan in covid-19: a systematic review and meta-analysis," *Polish Journal of Radiology*, vol. 87, pp. 1–23, 01 2022.

[6] C. Sorzano and J. Carazo, "Cryo-electron microscopy: The field of 1,000+ methods," *Journal of Structural Biology*, vol. 214, no. 3, p. 107861, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047847722000314

[7] J. de la Rosa-Trevín, A. Quintana, L. del Cano, A. Zaldívar, I. Foche, J. Gutiérrez, J. Gómez-Blanco, J. Burguet-Castell, J. Cuenca-Alba, V. Abrishami, J. Vargas, J. Otón, G. Sharov, J. Vilas, J. Navas, P. Conesa, M. Kazemi, R. Marabini, C. Sorzano, and J. Carazo, "Scipion: A software framework toward integration, reproducibility and validation in 3d electron microscopy," *Journal of Structural Biology*, vol. 195, no. 1, pp. 93–99, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S104784771630079X

[8] J. de la Rosa-Trevín, J. Otón, R. Marabini, A. Zaldívar, J. Vargas, J. Carazo, and C. Sorzano, "Xmipp 3.0: An improved software suite for image processing in electron microscopy," *Journal of Structural Biology*, vol. 184, no. 2, pp. 321–328, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047847713002566

[9] D. Strelak, A. Jiménez-Moreno, J. L. Vilas, E. Ramírez-Aportela, R. Sánchez-García, D. Maluenda, J. Vargas, D. Herreros, E. Fernández-Giménez, F. P. de Isidro-Gómez, J. Horacek, D. Myska, M. Horacek, P. Conesa, Y. C. Fonseca-Reyna, J. Jiménez, M. Martínez, M. Harastani, S. Jonić, J. Filipovic, R. Marabini, J. M.

Carazo, and C. O. S. Sorzano, “Advances in xmipp for cryo–electron microscopy: From xmipp to scipion,” *Molecules*, vol. 26, no. 20, 2021. [Online]. Available: https://www.mdpi.com/1420-3049/26/20/6224

[10] F. P. de Isidro, D. Hanein, and N. Volkmann, *Cryo-electron Tomography: A Journey from Sample Preparation to Data Mining - Validation chapter.* Elsevier, 2023.

[11] J. Jiménez de la Morena, P. Conesa, Y. Fonseca, F. de Isidro-Gómez, D. Herreros, E. Fernández-Giménez, D. Strelak, E. Moebel, T. Buchholz, F. Jug, A. Martinez-Sanchez, M. Harastani, S. Jonic, J. Conesa, A. Cuervo, P. Losana, I. Sánchez, M. Iceta, L. del Cano, M. Gragera, R. Melero, G. Sharov, D. Castaño-Díez, A. Koster, J. Piccirillo, J. Vilas, J. Otón, R. Marabini, C. Sorzano, and J. Carazo, “Scipiontomo: Towards cryo-electron tomography software integration, reproducibility, and validation,” *Journal of Structural Biology*, vol. 214, no. 3, p. 107872, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047847722000429

[12] T. Wagner, F. Merino, M. Stabrin, T. Moriya, C. Antoni, A. Apelbaum, P. Hagel, O. Sitsel, T. Raisch, D. Prumbaum, D. Quentin, D. Roderer, S. Tacke, B. Siebolds, E. Schubert, T. R. Shaikh, P. Lill, C. Gatsogiannis, and S. Raunser, “Sphire-cryolo is a fast and accurate fully automated particle picker for cryo-em,” *Communications Biology*, vol. 2, no. 1, p. 218, 2019. [Online]. Available: https://doi.org/10.1038/s42003-019-0437-z

[13] G. Tang, L. Peng, P. R. Baldwin, D. S. Mann, W. Jiang, I. Rees, and S. J. Ludtke, “Eman2: An extensible image processing suite for electron microscopy,” *Journal of Structural Biology*, vol. 157, no. 1, pp. 38–46, 2007, software tools for macromolecular microscopy. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047847706001894

[14] Y. Hao, X. Wan, R. Yan, Z. Liu, J. Li, S. Zhang, X. Cui, and F. Zhang, “Vp-detector: A 3d multi-scale dense convolutional neural network for macromolecule localization and classification in cryo-electron tomograms,” *Computer Methods and Programs in Biomedicine*, vol. 221, p. 106871, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016926072200253X

[15] L. R. F.R.S., “Lvi. investigations in optics, with special reference to the spectroscope,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 8, no. 51, pp. 477–486, 1879. [Online]. Available: https://doi.org/10.1080/14786447908639715

[16] C. Sorzano, J. Vargas, J. Otón, J. Rosa-Trevín, J. Vilas, M. Kazemi, R. Melero, L. del Caño, J. Cuenca, P. Conesa, J. Gomez-Blanco, R. Marabini, and J. Carazo, “A survey of the use of iterative reconstruction algorithms in electron microscopy,” *BioMed Research International*, vol. 2017, pp. 1–17, 09 2017.

[17] K. Sandberg, D. N. Mastronarde, and G. Beylkin, “A fast reconstruction algorithm for electron microscope tomography,” *Journal of Structural Biology*, vol. 144, no. 1, pp. 61–72, 2003, analytical Methods and Software Tools for Macromolecular Microscopy. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047847703001710

[18] M. C. Scott, C.-C. Chen, M. Mecklenburg, C. Zhu, R. Xu, P. Ercius, U. Dahmen, B. C. Regan, and J. Miao, “Electron tomography at 2.4-ångström resolution,” *Nature*, vol. 483, no. 7390, pp. 444–447, 2012. [Online]. Available: https://doi.org/10.1038/nature10934

[19] “Scipion-em repository,” https://github.com/scipion-em, accessed: 2023-05-19.

[20] "Scipion-chem repository," https://github.com/scipion-chem, accessed: 2023-05-19.

[21] T. A. M. Bharat and S. H. W. Scheres, "Resolving macromolecular structures from electron cryo-tomography data using subtomogram averaging in relion," *Nature protocols*, vol. 11, no. 11, pp. 2054—2065, November 2016. [Online]. Available: https://europepmc.org/articles/PMC5215819

[22] J.-L. Vilas and H. D. Tagare, "New measures of anisotropy of cryo-em maps," *Nature Methods*, vol. 20, no. 7, pp. 1021–1024, 2023. [Online]. Available: https://doi.org/10.1038/s41592-023-01874-3

[23] J. L. Vilas, J. Gómez-Blanco, P. Conesa, R. Melero, J. Miguel de la Rosa-Trevín, J. Otón, J. Cuenca, R. Marabini, J. M. Carazo, J. Vargas, and C. O. S. Sorzano, "Monores: Automatic and accurate estimation of local resolution for electron microscopy maps," *Structure*, vol. 26, no. 2, pp. 337–344.e4, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0969212617304434

[24] "Scipion 3 official install guide," https://scipion-em.github.io/docs/release-3.0.0/docs/scipion-modes/how-to-install.html, accessed: 2023-05-10.

[25] "Xmipp 3 official install guide," https://github.com/I2PC/xmipp#xmipp, accessed: 2023-05-10.

Appendices

A

Project dependencies

Figure A.1: Logo of the Scipion framework and Xmipp.

Although several guides exist for the installation of Scipion and its related plug-ins, setting everything up can be quite challenging for beginners in the world of Linux. The present appendix shows the installation process followed during this project, mainly for the sake of traceability and reproducibility but also to provide a guide for whoever finds it useful. The laptop used for the development is the one described in Section 4.

The machine has been provided by the Bio Computing Unit with the sole purpose of doing this work, so the only installed software packages are the ones needed by Scipion and Xmipp to ensure the best working order.

A.1 Scipion3 and Xmipp3

A virtual environment providing system is required for the installation. This includes *venv, anaconda, miniconda, mambaforge, miniforge...* For the duration of the project, miniconda3 was employed as the python virtual environment manager. The rest of the requirements and installation process are explained in their respective webpages [24] [25].

First, Scipion3 must be installed as per its guide. The installation of Xmipp in developer mode is straightforward. After installing the requirements stated in the documentation, it can be cloned from GitHub.

```
# Retrieve the repository
git clone https://github.com/I2PC/xmipp xmipp-bundle
# Trigger the compilation, using Scipion to pass the needed
    ↪ environment variables
scipion3 run ./xmipp-bundle/xmipp
```

A.2 Needed plugins to reproduce the experiments

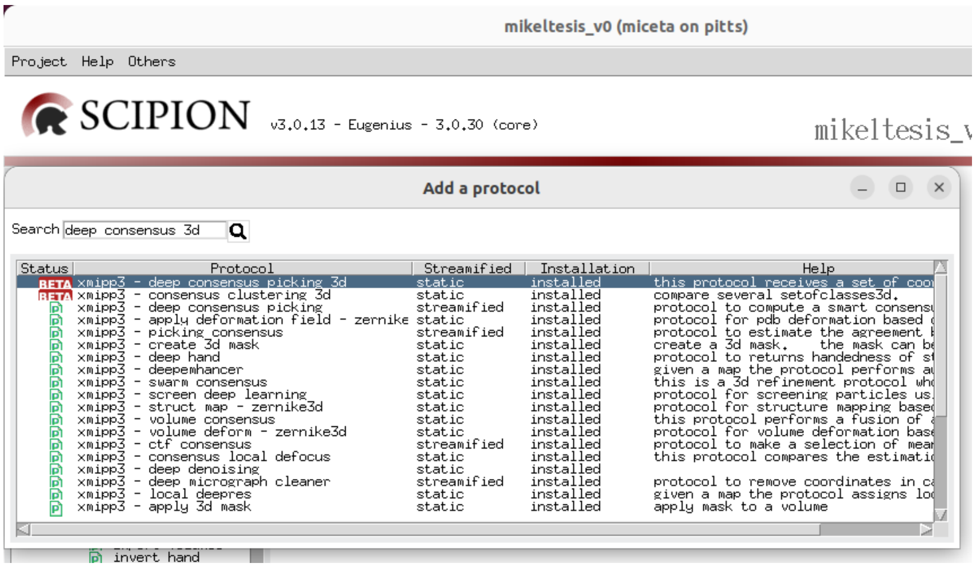
This projects makes use of crYOLO, DeepFinder, Dynamo, Eman, EmanTomo, IMOD, ScipionTomo, Xmipp and XmippTomo.

```
# Install the base Xmipp plugin in devel mode
scipion3 installp -p xmipp-bundle/src/scipion-em-xmipp --devel

# Retrieve the repositories for the developer-mode plugins
git clone https://github.com/I2PC/scipion-em-xmipptomo -b
    ↪ mit_deepconsensus_3d
git clone https://github.com/scipion-em/scipion-em-deepfinder


# Install the needed plugins
scipion3 installp -p scipion-em-tomo
scipion3 installp -p scipion-em-sphire
scipion3 installp -p ./scipion-em-deepfinder --devel
scipion3 installp -p scipion-em-dynamo
scipion3 installp -p scipion-em-eman2
scipion3 installp -p scipion-em-emantomo
scipion3 installp -p scipion-em-imod
scipion3 installp -p ./scipion-em-xmipptomo --devel
```

From this point, any IDE such as PyCharm or VSCode can be used to open the xmipp source folder. As this project relies only on the modification of Python code, there is no need of recompiling and/or linking the C and Java parts of Xmipp. The branch used for this project is called *mit_ deepconsensus_ 3d* in both the *scipion-em-xmipptomo* and in the *xmipp* repositories.

A.3 Using the protocol

After installing the Scipion plugin *scipion-em-xmipptomo*, the protocol developed in this project becomes available in the GUI for its use. It can be found in the protocols list (Ctrl+F) by looking for the keywords "deep consensus 3d", as it is shown in figure A.2


mikeltesis_v0 (miceta on pitts)
Project Help Others
SCIPION
v3.0.13 - Eugenius - 3.0.30 (core)
mikeltesis_v
Add a protocol

Search deep consensus 3d

Status	Protocol	Streamified	Installation	Help
BETA	xmipp3 - deep consensus picking 3d	static	installed	this protocol receives a set of coo
BETA	xmipp3 - consensus clustering 3d	static	installed	compare several setofclasses3d.
p	xmipp3 - deep consensus picking	streamified	installed	protocol to compute a smart consensu
p	xmipp3 - apply deformation field - zernike	static	installed	protocol for pdb deformation based
p	xmipp3 - picking consensus	streamified	installed	protocol to estimate the agreement
p	xmipp3 - create 3d mask	static	installed	create a 3d mask. the mask can b
p	xmipp3 - deep hand	static	installed	protocol to returns handedness of s
p	xmipp3 - deepemhancer	static	installed	given a map the protocol performs a
p	xmipp3 - swarm consensus	static	installed	this is a 3d refinement protocol wh
p	xmipp3 - screen deep learning	static	installed	protocol for screening particles us
p	xmipp3 - struct map - zernike3d	static	installed	protocol for structure mapping base
p	xmipp3 - volume consensus	static	installed	this protocol performs a fusion of
p	xmipp3 - volume deform - zernike3d	static	installed	protocol for volume deformation bas
p	xmipp3 - ctf consensus	streamified	installed	protocol to make a selection of mea
p	xmipp3 - consensus local defocus	static	installed	this protocol compares the estimati
p	xmipp3 - deep denoising	static	installed	
p	xmipp3 - deep micrograph cleaner	streamified	installed	protocol to remove coordinates in c
p	xmipp3 - local deepres	static	installed	given a map the protocol assigns lo
p	xmipp3 - apply 3d mask	static	installed	apply mask to a volume

invert hand

Figure A.2: The protocol developed in this project is listed as *xmipptomo - deep consensus picking 3d.*

B
Tunable parameters at GUI-level

The Graphical User Interface (GUI) exposed through Scipion allows the user to modify several parameters that affect the execution of the protocol developed in this project. In this appendix you will find these parameters together with their explanation.

B.1 Main parameters

This section contains the parameters that are indispensable to the execution of the program. This includes:

- **Voting instead of DNN** (Hidden by default and set to "No") - This parameter will disable the machine learning algorithm and fall back to a voting mechanism. This is ideal for testing purposes or getting a consensus picking in a faster way.
- **Model selection** - The model can be either a new one, trained *in situ* for the input dataset or an existing model that was saved in *.h5* format.
- **Training batch size** - This parameter sets the amount of input volumes that will be fed each time to the neural network during training.
- **Input coordinates** - Multiple doubtful inputs can be added here for the neural network to be fed.
- **Positive references** - Similar to the input coordinates field, but only for adding optional coordinates that are verified to be positive (e.g. hand-picked volumes).
- **Negative references** - same as the previous parameter, but for negative labelled data.
- **Tolerance threshold** - Sets the score threshold above which the subtomograms are accepted as good pickings at the output of the neural network during the scoring stage.

B.2 Preprocess options

Some intermediate programs executed during the course of the main protocol need some parameters to work. They are:

- **Same-element relative radius** - Represents how near two different picks need to be in order to be considered the same real-life region of interest. It is represented in fractions of the total size of the box size.
- **Representative choosing method** - Representatives are used to generate clusters of coordinates. The position of the region represented can be chosen at the end of the consensus either by assigning it the one of the first element that entered that cluster or a centroid of all elements of the cluster.
- **Boxsize choosing method** - Input pickers might have different box sizes and sampling rates. In order to homogenize the items, all are resized to have the same parameters. Which size and sampling rate to choose is up to the user: first one found, biggest, smallest or a mean.
- **Amount of noise picked for negative input** - Represents, in fractions of the total size of the dataset, how many regions of noise need to be found. Value *1.0* will preserve the balance between positive and negative inputs.
- **Noise picking evasion radius** - Similarly to what happens in the coordinates consensus, a radius is also required during the random noise picking step. This ensures that the picked noise is far away from any picked coordinate. It is also expressed in fractions of the consensus box size.

B.3 Training

These parameters are very specific to the training stage of the neural network, and rule the learning process of the model:

- **Cycles (epochs)** - How many epochs to run. An epoch ends when the model has trained over the whole dataset once.
- **Validation fraction** - Fraction of the dataset to be reserved for validation purposes. Validation is done once an epoch ends, to calculate how accurate the network is with data it has not seen.
- **L2 regularisation strength** - A higher value keeps the model simpler and smaller, by penalizing the storage of bigger weights inside of the neural network.
- **Learning rate** - Determines at which rate the weights are allowed to change. A higher learning rate makes the network train faster, but it might converge to a local minima (or not converge at all). Smaller values will help the model to converge easier, but at the cost of needing many more epochs to do so.
- **Stop on convergence** - Introduces a callback function that executes at the end of every epoch. If convergence is reached, the training stops early.

- **Force data augmentation** - This will force data augmentation to be performed, even if the dataset is large.

C Auxiliary programs and their parameters

C.1 Coordinate Consensus 3D

This script, created for this project to have a way of determining if two coordinates refer to the same position in real life, specifies the following parameters:

```
def defineParams(self):
    self.addUsageLine('Coordinates Consensus Tomo - Unify coordinates for a
                                              ↪ single tomogram\. This program
                                              ↪ accepts an XMD file containing
                                              ↪ the following information:\n'
    '- 3D coordinates in separate columns (X, Y, Z)\n'
    '- Integer identifier of the picker who saw it'
    )

    self.addParamsLine('--input <path> : input')
    self.addParamsLine('--outputPos <path> : output path for positive subtomos')
    self.addParamsLine('--outputDoubt <path> : output path for doubtful subtomos
                                              ↪ ')
    self.addParamsLine('--outputAll <path> : output path for all subtomos')
    self.addParamsLine('--boxsize <int> : boxsize')
    self.addParamsLine('--samplingrate <double> : sampling rate')
    self.addParamsLine('--radius <double> : radius')
    self.addParamsLine('--number <int> : number')
    self.addParamsLine('--constype <int> : type of consensus (0 for first, 1 for
                                              ↪  centroid)')
    self.addParamsLine('[ --inputTruth <path> ] : Optional path to XMD file
                                              ↪ containing truthful coordinates.
                                              ↪ Added to POS automatically.')
    self.addParamsLine('[ --inputLie <path> ] : Optional path to XMD file
                                              ↪ containing lie(negative)
                                              ↪ coordinates. Added to NEG
                                              ↪ automatically.')
    self.addParamsLine('[ --outputNeg <path> ] : Output path for negative
                                              ↪ subtomos')
    self.addParamsLine('--startingId <int> : Initial number to use for
                                                ↪ MDL_PARTICLE_ID field in XMD.')
```

C.2 Noise Picking 3D

```
def defineParams(self):
    self.addUsageLine('PickNoiseTomo - picks random coordinates for
                                                ↪ DeepConsensusTomo. This program
                                                ↪ takes an XMD formatted file
                                                ↪ containing coordinates for picked
                                                ↪  subtomos and uses brute force to
                                                ↪  obtain coordinates representing
                                                ↪ "noise" for use as negative input
                                                ↪  in the DeepConsensusTomo NN
                                                ↪ training.')

    self.addParamsLine('--input <inputFile> : Path to the XMD file containing
                                                ↪ MDL_X MDL_Y and MDL_Z')
    self.addParamsLine('--output <outputFile> : Path for the resulting XMD file
                                                ↪ to be written.')
    self.addParamsLine('--radius <radius> : Radius (in fraction of boxsize) for
                                                ↪ thresholding the distances')
    self.addParamsLine('--boxsize <boxsize> : Size in pixels of the square box
                                                ↪ for subtomograms')
    self.addParamsLine('--samplingrate <srate> : Sampling rate in Angstroms/
                                                ↪ pixel')
    self.addParamsLine('--size <x> <y> <z> : Triplet representing the tomo size'
                                                ↪ )
    self.addParamsLine('[ --limit <limit=0.7> ] : Amount (in fraction of total
                                                ↪ input coords) of pickings to do.
                                                ↪ Default=0.7')
    self.addParamsLine('[ --threads <int=4> ] : Number of threads to parallelize
                                                ↪  the search. Default=4')
    self.addParamsLine('[ --static ] : flag to stop the algorithm from reducing
                                                ↪ the radius')

    self.addExampleLine('xmipp_pick_noise_tomo --input tomo05_coords.xmd --
                                                ↪ boxsize 200 --samplingrate 2.17')
```

C.3 Subtomogram Extraction

This program was written by F.P. de Isidro and J.L. Vilas from the BCU-CNB. It accepts the file name for the tomogram, as well as the file name for the coordinates file, the box size and the path in which the subtomograms will be stored once extracted.