



# GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

## **PROGRAMACIÓN DE DISTINTOS PROTOCOLOS PARA LA SIMULACIÓN DE MODELOS 2D/3D DE IMÁGENES DE MICROSCOPIO**

Autor: Antonio Martos Herrero

Director: D. Carlos Óscar Sánchez Sorzano

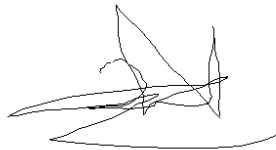
Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

**“Programación de distintos protocolos para la simulación de modelos 2D/3D de imágenes de microscopio”** en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico **2018/19** es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.



Fdo.: Antonio Martos Herrero

Fecha: ...11.../...07.../ ...2019...

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: D. Carlos Óscar Sánchez Fecha: ...11.../...07.../ ...2019...

---

**AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESIS O MEMORIAS DE BACHILLERATO**

***1º. Declaración de la autoría y acreditación de la misma.***

El autor D. **ANTONIO MARTOS HERRERO**

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: **PROGRAMACIÓN DE DISTINTOS PROTOCOLOS PARA LA SIMULACIÓN DE MODELOS 2D/3D DE IMÁGENES DE MICROSCOPIO**, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

***2º. Objeto y fines de la cesión.***

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

***3º. Condiciones de la cesión y acceso***

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

- a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar “marcas de agua” o cualquier otro sistema de seguridad o de protección.
- b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
- c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
- d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
- e) Asignar por defecto a estos trabajos una licencia Creative Commons.
- f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente*).

***4º. Derechos del autor.***

El autor, en tanto que titular de una obra tiene derecho a:

- a) Que la Universidad identifique claramente su nombre como autor de la misma

- b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
- c) Solicitar la retirada de la obra del repositorio por causa justificada.
- d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

#### **5º. Deberes del autor.**

El autor se compromete a:

- a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
- b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
- c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.
- d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.


#### **6º. Fines y funcionamiento del Repositorio Institucional.**

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusiva del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

*Madrid, a .....11..... de .....Julio..... de .....2019....*

**ACEPTA**



Fdo.....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

<p>NO APLICA</p>
------------------



# GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

## **PROGRAMACIÓN DE DISTINTOS PROTOCOLOS PARA LA SIMULACIÓN DE MODELOS 2D/3D DE IMÁGENES DE MICROSCOPIO**

Autor: Antonio Martos Herrero

Director: D. Carlos Óscar Sánchez Sorzano

Madrid

## Agradecimientos

A mi tutor D.Carlos Óscar Sánchez Sorzano por mostrarse siempre tan accesible. Agradezco su inestimable ayuda y compromiso.

Al Centro Nacional de Biotecnología (CNB) por permitirme hacer uso de su programa *Scipion* y confiar en mí para incluir protocolos.

Al grupo Elmlund Lab por permitirme utilizar su programa SIMPLE para el conocimiento y uso de sus protocolos. Les agradezco mucho que siempre hayan estado al otro lado para resolver mis dudas.

**Título: Programación de distintos protocolos para la simulación de modelos 2D/3D de imágenes de microscopio.**

**Autor:** Martos Herrero, Antonio.

**Director:** Sánchez Sorzano, Carlos Óscar.

**Entidad Colaboradora:** CNB – Centro Nacional de Biotecnología, ICAI – Universidad Pontificia Comillas

---

## **RESUMEN DEL PROYECTO**

**Palabras clave:** Microscopía, electrónica, wrapper, *Scipion*, script, streaming, procesar, procesamiento, formato, partícula, volumen, clases.

### **1. Introducción**

La microscopía electrónica, a diferencia de la microscopía óptica, permite mostrar estructuras muy pequeñas debido a que la longitud de onda que utilizan estos microscopios es de 0.5 Angstroms. Existen dos tipos principales: TEM (Microscopía electrónica de transmisión) y SEM (Microscopía electrónica por escaneado).

Sin entrar en detalles técnicos, estos microscopios son herramientas imprescindibles en el ámbito de la investigación puesto que permiten aumentar sensiblemente el tamaño de las partículas, lo que facilita la obtención de imágenes granulares con un elevado detalle.

Sin embargo, la mayoría de dichas imágenes pueden ser muy difíciles de interpretar a simple vista por lo que conviene que sean “tratadas”, mediante algoritmos para conseguir una mejor resolución y para la creación de modelos que permitan identificar las partículas en detalle y con mayor facilidad [1]

*Scipion* es un programa desarrollado por el Centro Nacional de Biotecnología (CNB) cuyo objetivo es la incorporación de paquetes que ejecuten los scripts de varios programas para la mejora de imágenes tomadas por microscopios electrónicos. Cada



paquete ejecuta un script, refiriéndose como *script* a un protocolo (importparticles, cluster2D, refine3D, etc....).

## 2. Definición del proyecto

El trabajo que se va a desarrollar consiste, básicamente, en la creación y configuración de *wrappers* o códigos que permiten la ejecución de los *scripts* de un programa, para su incorporación en *Scipion*.

La función principal de los *wrappers*, mediante la oportuna codificación, consistirá en llamar a los protocolos de diferentes programas con el objetivo de conseguir el adecuado tratamiento de las imágenes de las partículas que previamente habrán sido importadas mediante la utilización de cualquier microscopio electrónico.

Estos protocolos podrán ejecutar distintas funciones tales como la creación de clústeres en dos dimensiones, el desarrollo de modelos iniciales en tres dimensiones, definir procesos de refinamiento en tres dimensiones, etc...

Como punto de partida, se procederá a probar el protocolo que se quiera incluir en *Scipion* con objeto de comprobar que su funcionamiento es correcto y del todo compatible con los objetivos perseguidos en la ejecución del programa referido.

En una segunda fase, se procederá a la definición de las instrucciones o programación del *wrapper*. Es aquí donde debe resaltarse la importancia de que el *wrapper* ejecute únicamente un script, aunque, dependiendo de la configuración del programa de origen, no siempre podrá ser factible por lo que será necesario llamar a más de uno.

Finalmente, y una vez que se ha desarrollado la programación, deberá procederse a la realización de pruebas en *Scipion* con el objetivo de comprobar que el funcionamiento del *wrapper* es correcto y que las salidas creadas son de utilidad y de calidad suficiente para incorporarlos a *Scipion*.

## 3. *Scipion*: Definición y Estructura

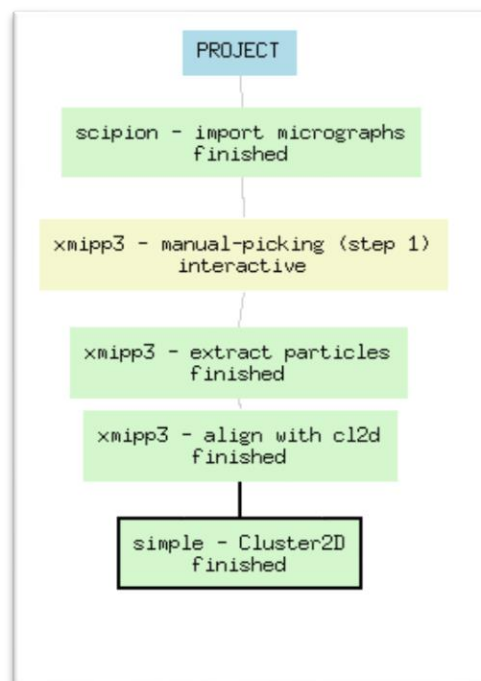
*Scipion* es una plataforma que integra varios paquetes de software para el procesamiento de imágenes digitales.

En la actualidad este programa incorpora más de 20 paquetes y permite al usuario acceder a más de 200 operaciones con el objeto de mejorar la calidad de la imagen en la búsqueda de información no identificable a simple vista.

---

*Scipion* también permite el procesamiento en *streaming*, es decir, facilita procesar las imágenes al mismo tiempo que se están descargando en el microscopio. [2]

La figura que se muestra a continuación, representa las fases de desarrollo y descarga de datos de un proceso muy simple mediante la utilización de *Scipion*. En este caso, la transmisión no es en *streaming*, sino que el proceso comienza una vez que se han obtenido las imágenes con carácter previo.



*Ilustración 1 - Esquema de un proyecto en Scipion*

Como se puede comprobar en la imagen, la primera caja indica el comienzo del proyecto, mientras que el resto de cajas ejecutan un protocolo.

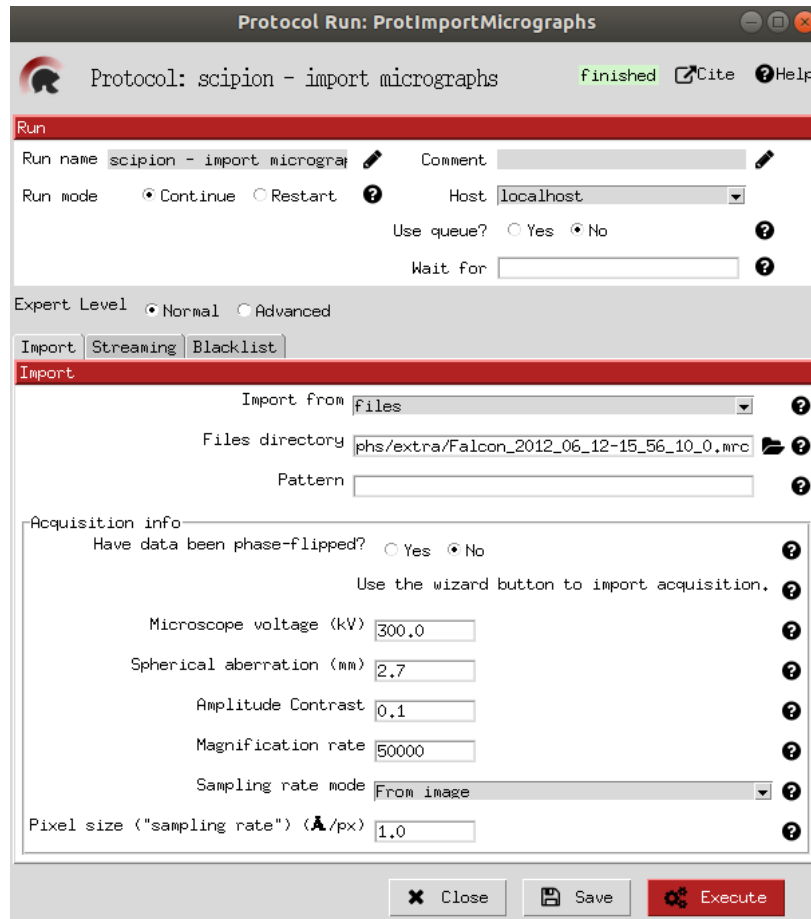
En este proyecto se observan las siguientes acciones a ejecutar:

- ❖ Import micrographs – Importa una imagen del microscopio.
- ❖ Manual picking – El usuario elige ciertas partículas y activa el entrenamiento para que se elijan el resto.

- ❖ Extract particles – Separa las partículas elegidas y crea el fichero correspondiente.
- ❖ Align – Alinea las partículas para obtener resultados más correctos.
- ❖ Cluster2D – Agrupación y alineación 2D.

Para trabajar en *Scipion*, siempre será necesario empezar trabajando sobre un set de imágenes de partículas. Estas partículas se pueden introducir directamente o sacarlas mediante protocolos aplicados a “micrographs” o “movies”. Una vez tenemos las partículas podemos ejecutar los protocolos. *Scipion* sigue una estructura de formatos, es decir, hay ciertos protocolos que trabajan sobre volúmenes, otros sobre las propias partículas y otros sobre sets de clases 2D. Lo normal es trabajar con los siguientes pasos de formatos:

Partículas → Clases 2D → Volúmenes



*Ilustración 2 – Introduzca parámetros de import\_micrographs*

## 4. Wrapper: Estructura

Como ya se ha mencionado anteriormente, un *wrapper* consiste en un código que ejecuta un script. También adecúa las entradas y las salidas para que funcione correctamente ese script.

Cada *wrapper* consta de los siguientes elementos:

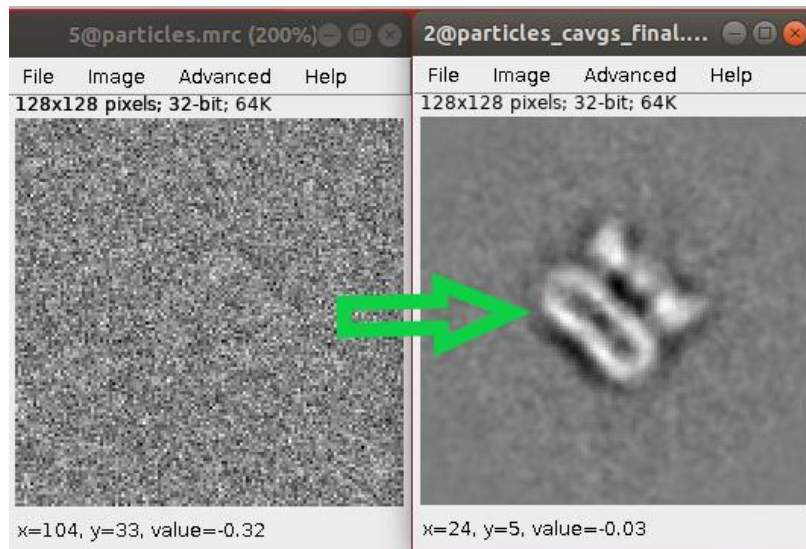
- ❖ **Inicialización:** Se inicia el protocolo con el nombre que se vaya a asignar y en el modo en que aparecerá en *Scipion*.
- ❖ **Definición de parámetros:** Se construye una ventana con los diferentes parámetros que se consideren y que el usuario deberá introducir manualmente. Estos parámetros pueden ser de diferentes tipos, tales como numéricos, archivos, strings, etc.
- ❖ **Pasos o steps:** Deberán definirse las funciones, que siguiendo un orden predeterminado, se ejecutarán. En la interfaz gráfica de *Scipion* aparecerá el número de pasos a seguir que se irá actualizando a medida que finalice su ejecución. Es importante señalar que no se deberán identificar los códigos de las funciones.
- ❖ **Funciones de los steps:** Se trata del código que va a ser ejecutado. Como norma general suele haber 3 funciones: La primera es la denominada *convertInputs* que se encarga de convertir los archivos de entrada, en caso de que los hubiera, a sus correspondientes formatos. La segunda, o función del *protocolo*, que básicamente llama al protocolo que se desea ejecutar como si se estuviese haciendo por consola (cmd). Y, finalmente, la tercera función, *convertOutputs*, que crea los ficheros de salida y obtiene los resultados que el usuario podrá observar en la aplicación.
- ❖ **Otras:** No son obligatorias y son funciones destinadas a la decoración del protocolo que ejecutan la validación de parámetros y añaden información a los usuarios.
- ❖ **Propias:** Existe la posibilidad de crear funciones adicionales a las mencionadas anteriormente con objeto de ejecutar cualesquiera otras acciones que se estimen convenientes. A modo de ejemplo, en el caso de que un protocolo genere varios ficheros que correspondan a numerosas iteraciones, podríamos definir una función que permitiera elegir el fichero que nos interese.

[3]

## 5. Resultados

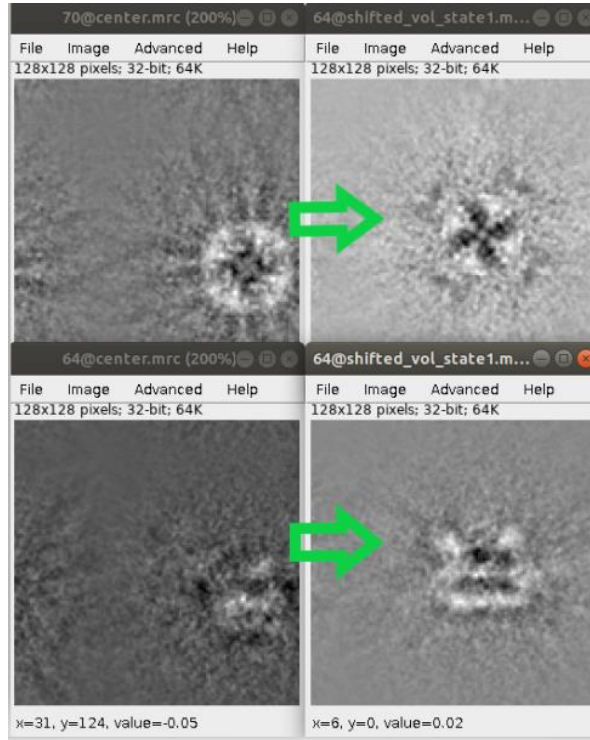
En esta parte se adjuntarán imágenes con los resultados obtenidos tras la ejecución de los distintos protocolos.

- **cleanUp2D:** produce imágenes en dos dimensiones de un set de partículas iniciales.



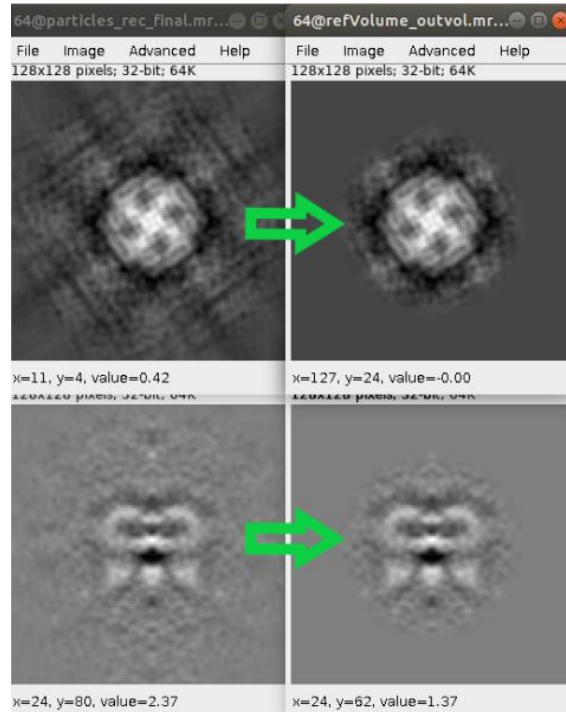
*Ilustración 3 – cleanup2D resultados*

- **Center:** coloca los volúmenes de las partículas en la posición central de la imagen.



*Ilustración 4 – center resultados*

- **Mask:** elimina el ruido de fondo a partir de un radio en píxeles introducido por el usuario:



*Ilustración 5 – mask3D resultados*

## 6. Conclusiones

Para cumplir con el objetivo pretendido en este proyecto, “programar e incorporar en Scipion” se han ejecutado varios programas correspondientes a distintos protocolos del paquete de SIMPLE. En particular, se han programado 5 protocolos denominados cleanUp2D, Center, Mask, Refine3D y Masscen.

Una vez analizados los resultados obtenidos de la aplicación de los mencionados programas, solamente se considera que cleanUp2D y Center como programas válidos para su incorporación a Scipion.



En el caso del programa Mask su incorporación no se considera necesaria puesto que ya existe un programa que realiza funciones similares dentro de Scipion. En cuanto a refine3D, se considera que no es un programa adecuado para su incorporación, debido no solo al modo de ejecución del protocolo sino también porque exige un elevado tiempo de ejecución. Finalmente, el programa Masscen no está incluido en la versión más reciente de SIMPLE por lo que su inclusión carece de sentido práctico.

## 7. Referencias

- [1] Universidad Nacional del Nordeste (1998-2013). “Microscopía electrónica: Definición y tipos”. Sitio Web: <http://www.biologia.edu.ar/microscopia/meb.htm>.
- [2] Sánchez Sorzano, C.O. [Science and Technology Facilities Council]. (23/05/2018). CCP-EM Carlos Óscar Sánchez Sorzano | Spring Symposium 2018. Sitio Web: <https://www.youtube.com/watch?v=wsGlgSoXNI0&t=308s>.
- [3] Anónimo (21/10/2018). “Ejemplo de creación de un wrapper”. Sitio Web: [https://github.com/Scipion-em/Scipion-em-simple/blob/devel/simple/protocols/protocol\\_prime.py](https://github.com/Scipion-em/Scipion-em-simple/blob/devel/simple/protocols/protocol_prime.py).

**Title: Programming of several protocols which simulate 2D/3D models of microscope images.**

**Author: Martos Herrero, Antonio.**

**Supervisor: Sánchez Sorzano, Carlos Óscar.**

**Collaborating Entity: CNB – Centro Nacional de Biotecnología, ICAI – Universidad Pontificia Comillas**

---

## **ABSTRACT**

**Keywords:** Microscopy, electronic, wrapper, *Scipion*, script, streaming, processing, format, particle, volume, classes.

## **1. Introduction**

Unlike optical microscopy, electronics microscopy can show structures with a very small size because this microscope uses a wavelength of 0.5 Angstroms. There are two main types: TEM (Transmission Electronic Microscopy) and SEM (Scanning Electronic Microscopy).

Without getting into technical details, these microscopes are essential tools in the research field since they allow enlarging the size of little particles, which is easier to obtain grainy images.

This does however; algorithms are needed to improve their resolution and to create models which allow human eyes to observe them better and easier. [1]

*Scipion* is a program developed by the Nacional Biotechnology Centre (CNB). The objective of this program is the incorporation of packages which execute scripts from other programs to improve the quality and create models of images taken from electronic microscopes. Each package executes one script, and one script refers to one different protocol (import particles, cluster2D, refine3D, etc.....).

## 2. Project Definition

This project will consist on creating *wrappers*. Wrappers are codes which execute the scripts of a program. Wrappers will be included in *Scipion*.

The main function of these wrappers is to call different protocols for the treatment of images with several particles imported from microscopes. The protocols will be able to simulate several functions such as the creation of two-dimension clusters, three-dimension initial models, three-dimension, etc....

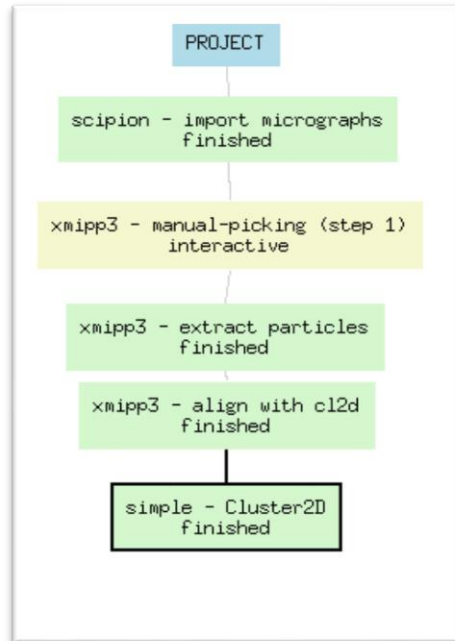
Before programming any code, several tests will be made to check if that protocol is compatible with *Scipion* terms and requirements. The next step will be the programming of the wrapper. In this step will important that the wrapper executes exclusively one script. Sometimes this won't be possible and will depend on the original program, where the script is taken from. After programming, the last step will be testing the wrapper in *Scipion*, checking the performance and the utility of the outputs.

## 3. Scipion: Definition and Structure

*Scipion* is a platform which includes several software packages for image processing. Nowadays, the user can execute more than 20 software packages and over 200 operations to improve the image 'quality in locating information not easily identifiable at a glance.

*Scipion*, also, allows processing in stream, which consists on processing the images while the microscope takes them. [2]

Next, the inside operation of *Scipion* will be explained. Below, an image of a simple project in *Scipion* is shown. It is necessary to say that this project isn't being followed in stream.



*Figure 2 – Scipion project scheme*

Each box executes one protocol, which can be a wrapper or an inside protocol. The project box is exempted as it marks the beginning of a project.

This process comprised the following actions:

- ❖ Import micrographs – Imports an image taken by the microscope.
- ❖ Manual picking – The user chooses some particles within an area and activates the training to choose the rest of the particles.
- ❖ Extract particles – Create a file with the particles picked in step before.
- ❖ Align – Aligns particles to have better results.
- ❖ Cluster2D – 2D group and alignment.

*A set of images which contain particles will be always necessary to start a project in Scipion.* These particles can be directly included or taken from “micrographs” or “movies”. When the particles are ready, we can start playing with the protocols. *Scipion*

follows a format-structure which means that there are some protocols that work on volumes, others on set of classes 2D and others on the particles. Normally, you will have formats ordered in a project as shown:

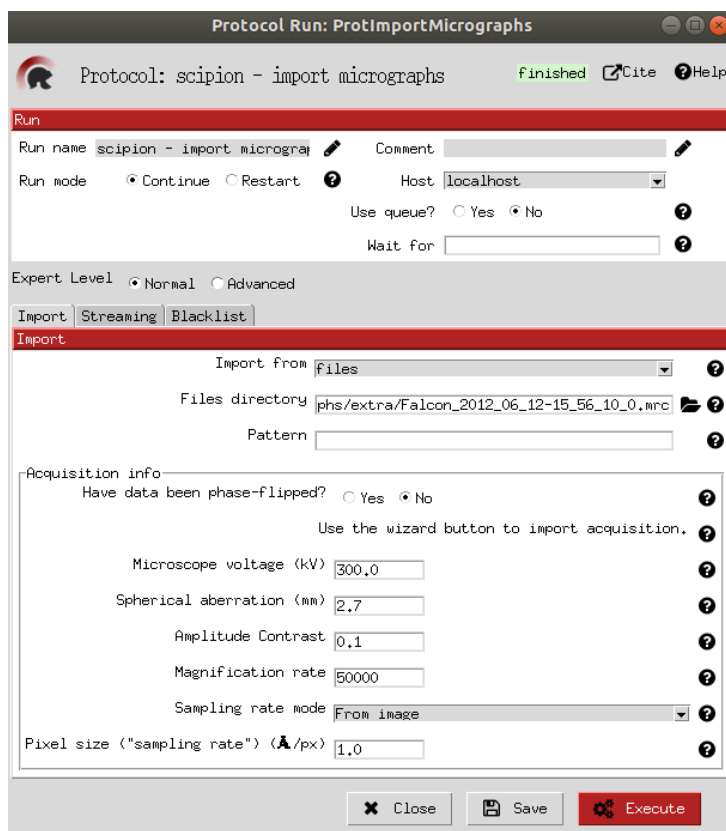


Figure 2 – Introduce parameters for import micrographs protocol

## 4. Wrapper: Structure

As it was mentioned before, a wrapper consists on a code that executes a script. Also, it makes compatible inputs and outputs to make that script go correctly.

Each wrapper has the next structure:

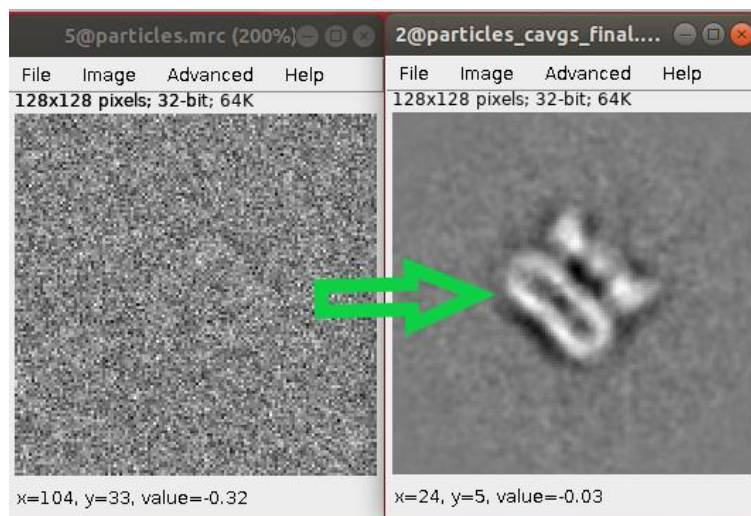
- ❖ **Initialization:** the protocol will begin with the name and a label that the user will see in *Scipion*.
- ❖ **Parameter Definition:** A window is created and where the user will introduce the parameters to use. They can be: numeric, strings, files, etc...
- ❖ **Steps:** Function's definition to be executed in order. Number of steps will appear in the graphic interface of *Scipion* and will be updated when one step is completed. Here only go the name of the functions, not the code.
- ❖ **Functions coded:** Code to be executed. Normally there are three functions. The first one is *convertInputs* and converts the input files to the suitable format in case of having it. Second one calls the protocol as if it was called by console (cmd). Last one is **convertOutputs** and does the same as *convertInputs* but with the outputs and shows the user the results.
- ❖ **Others:** They aren't mandatory and they are functions destined to decorate the protocol, execute a validation process and add information of the programmer.
- ❖ **Last,** there are other functions which generate utility to the programmer. One example of these functions is a function that returns one file when more are generated.

[3]

## 5. Results

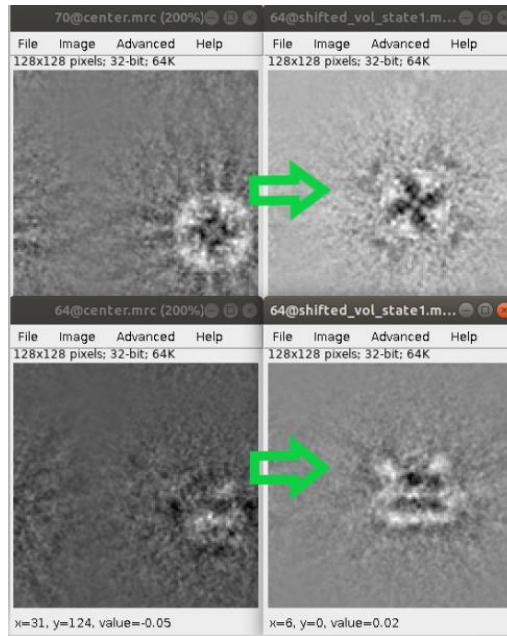
The results obtained in this project are obtained after the execution of the wrappers programmed in Scipion. For each protocol the output images are going to be shown:

- **CleanUp2D:** produces high-quality images from an initial set of particles.



*Figure 3 – cleanUp2D results*

- **Center:** relocates the particles so that they are in the center of the image.



*Figure 4 – center results*

- **Mask:** removes the noise of a volume file within an input ratio in pixels introduced by the user.



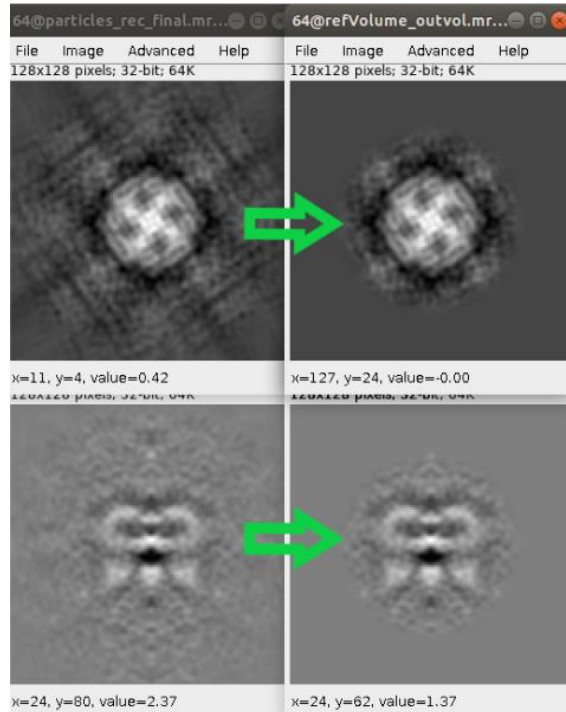


Figure 5 – mask3D results

## 6. Conclusions

To reach the intended objective within this project, "to program and incorporate in Scipion", several programs which correspond to different protocols of the SIMPLE package have been carried out. Specially 5 programmes called cleanUp2D, Center, Mask, Refine3D and Masscen have been run.

According to the main results, only cleanUp2D and Center are considered effective programs to be incorporated in Scipion.

In particular, there is already a program that performs something similar to Mask and refine3D takes too much time to be executed and its execution mode isn't appropriate at all.

Finally, there is no sense to include Masscen as it is a program of a more antique version of SIMPLE.

## 7. References

- [1] Universidad Nacional del Nordeste (1998-2013). “Electronic Microscopy: Definition and types”. Website: <http://www.biologia.edu.ar/microscopia/meb.htm>.
- [2] Sánchez Sorzano, C.O. [Science and Technology Facilities Council]. (23/05/2018). CCP-EM Carlos Óscar Sánchez Sorzano | Spring Symposium 2018. Website: <https://www.youtube.com/watch?v=wsG1qSoXNI0&t=308s>.
- [3] Anonymous (21/10/2018). “Wrapper creation example”. Website: [https://github.com/Scipion-em/Scipion-em-simple/blob/devel/simple/protocols/protocol\\_prime.py](https://github.com/Scipion-em/Scipion-em-simple/blob/devel/simple/protocols/protocol_prime.py).



## *Índice de la memoria*

<b>Capítulo 1. Introducción .....</b>	<b>33</b>
1.1 microscopía electrónica.....	33
1.2 Microscopía cryo-EM.....	36
1.3 Scipion.....	38
1.4 SIMPLE.....	40
1.5 Motivación .....	41
<b>Capítulo 2. Descripción de las Tecnologías.....</b>	<b>42</b>
<b>Capítulo 3. Estado de la Cuestión .....</b>	<b>45</b>
<b>Capítulo 4. Definición del Trabajo .....</b>	<b>48</b>
4.1 Justificación.....	48
4.1.1 “¿Todo en uno?”.....	48
4.1.2 ¿Cómo puede ayudar a los biólogos? .....	48
4.1.3 Factor económico.....	49
4.1.4 Open source.....	50
4.2 Objetivos .....	50
4.3 Metodología.....	51
4.4 Planificación.....	55
<b>Capítulo 5. Sistema: Wrappers.....</b>	<b>57</b>
5.1 Definición.....	57
5.2 Estructura .....	57
5.3 Implementación.....	58
<b>Capítulo 6. Análisis de Resultados.....</b>	<b>59</b>
<b>Capítulo 7. Conclusiones y Trabajos Futuros.....</b>	<b>81</b>
<b>Capítulo 8. Bibliografía.....</b>	<b>83</b>

## *Índice de figuras*

Figura 1. Microscopio T.E.M .....	34
Figura 2. Microscopio S.E.M .....	35
Figura 3. Resolución antes de 2013 vs Resolución cryo-EM.....	38
Figura 4. Ejemplo de proyecto simple en <b>Scipion</b> .....	39
Figura 5. SIMPLE logo .....	41
Figura 6. Ejemplo de búsqueda efectiva en Pycharm.....	42
Figura 7. Creación de carpeta con el programa .....	52
Figura 8. Carpeta con el programa .....	52
Figura 9. Variable de entorno de SIMPLE .....	53
Figura 10. Carpeta link scipion-simple .....	53
Figura 11. Definición protocolo .....	53
Figura 12. Link a protocolos de simple .....	54
Figura 13. Comprobación link correcta.....	54
Figura 14. Representación temporal del trabajo de fin de grado.....	55
Figura 15. Código <code>__init__</code> .....	59
Figura 16. cleanUp2D: definición y paquetes .....	60
Figura 17. cleanUp2D: ventana de parámetros .....	60
Figura 18. cleanUp2D: Pasos .....	61
Figura 19. cleanUp2D: convertInput .....	61
Figura 20. cleanUp2D: cleanUpStep parte 1 .....	62
Figura 21. cleanUp2D: cleanUpStep parte 2 .....	62
Figura 22. cleanUp2D: función getPath .....	63
Figura 23. cleanUp2D: createOutputStep.....	64

---

Figura 24. cleanUp2D-resultados: stack inicial.....	65
Figura 25. cleanUp2D-resultados: parámetros introducidos por usuario .....	66
Figura 26. cleanUp2D-resultados: resultado 5 clusters .....	67
Figura 27. cleanUp2D-resultados: resultado 10 clusters .....	68
Figura 28. Membrana TRPV1 .....	68
Figura 29. Center: Descripción y parámetros usuario .....	69
Figura 30. Center: Funciones y conversión a formato.....	70
Figura 31. Center: Ejecución del script de center.....	70
Figura 32. Center: Volumen de salida .....	70
Figura 33. Center: Volumen de entrada en ejecución-vistas Z e Y.....	71
Figura 34. Center: Parámetros de ejecución .....	72
Figura 35. Center: Resultado final-vistas Z e Y .....	73
Figura 36. Mask: Descripción y parámetros usuario .....	74
Figura 37. Mask: Funciones y conversión.....	75
Figura 38. Mask: Ejecución del script mask.....	75
Figura 39. Mask: Volumen de salida.....	76
Figura 40. Mask: Volumen de entrada en ejecución-vistas Z e Y.....	77
Figura 41. Mask: Parámetros de ejecución del protocolo .....	78
Figura 42. Mask: Resultado final-vistas Z e Y .....	79

## *Índice de tablas*

Tabla 1. Comparación NMR, rayos-X y Cryo-EM .....	37
Tabla 2. Paquetes incorporados a <i>Scipion</i> .....	<b>¡Error! Marcador no definido.</b>



## Capítulo 1. INTRODUCCIÓN

### 1.1 MICROSCOPIA ELECTRÓNICA

A diferencia de un microscopio óptico, la potencia amplificadora de un microscopio electrónico no está limitada por la longitud de onda de la luz visible. El microscopio electrónico utiliza electrones para iluminar un objeto. Como estos electrones tienen una longitud de onda mucho menor que la de la luz, son capaces de mostrar estructuras mucha más pequeñas. Estamos comparando una longitud de onda de 0,5 ángstroms en el microscopio con una longitud de onda aproximada de 4000 ángstroms de la luz visible, siendo esta la longitud de onda más corta.

Los microscopios constan de unos elementos básicos. Disponen de un cañón de electrones que emite los electrones que interfieren con la muestra, generando la imagen. También utilizan lentes magnéticas para crear campos que dirigen y enfocan el haz de electrones aumentando la imagen. Todo ello tiene lugar en el vacío, aspecto imprescindible para conseguir que los electrones no sean desviados por las moléculas de aire. Por último, todos los microscopios electrónicos cuentan con un sistema que registra o muestra la imagen que generan los electrones al chocar contra un sensor bidimensional que posteriormente es presentada digitalmente por el sistema electrónico del microscopio.

Hay dos tipos básicos de microscopios:

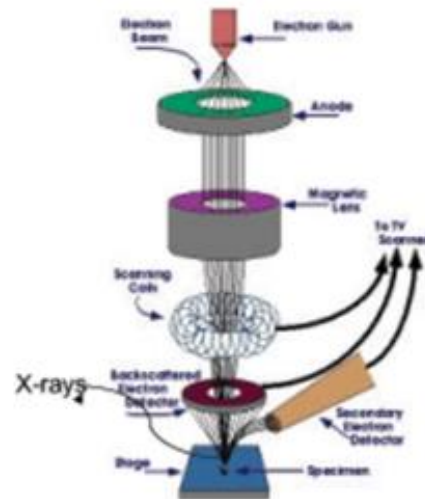
- **Microscopio electrónico de transmisión (MET o TEM):** dirige el haz de electrones hacia el objeto que se desea aumentar. Una parte de estos electrones rebotan o son absorbidos por el objeto y otros lo atraviesan formando la imagen aumentada. Para usar este tipo de microscopio se ha de preparar la muestra y poner una pantalla detrás para registrar la imagen. El aumento de estos microscopios puede llegar a ser de hasta un millón de veces el tamaño de las partículas.



*Figura 1. Microscopio T.E.M*

- **Microscopio electrónico de barrido (MEB o SEM):** crea una imagen ampliada de la superficie de un objeto. A diferencia del TEM, la muestra no requiere de preparativos (o muy pocos). Este tipo de microscopio explora la superficie de las muestras punto por punto utilizando un haz muy concentrado de electrones. Cada punto leído de la muestra corresponde a un píxel de un monitor de televisión. Se va mostrando la imagen en un monitor a la par que se produce el barrido y con una ampliación de hasta 200.000 veces o más. Este tipo de microscopios son útiles para el análisis de la topología de las muestras y para crear imágenes tridimensionales realistas.

### Scanning Electron Microscopy (SEM)



SEM: A focused electron beam (2-10 keV) scans on the surface, several types of signals are produced and detected as a function of position on the surface. The space resolution can be as high as 1 nm. Different type signal gives different information: a. Secondary electrons: surface structure. b. Backscattered electrons: surface structure and average elemental information. b. X-rays and Auger electrons: elemental composition with different thickness-sensitivity.

*Figura 2. Microscopio S.E.M*

Existen otros dos tipos de estos microscopios: el microscopio electrónico de barrido y transmisión (STEM) combina el SEM y el TEM y puede mostrar los átomos individuales de un objeto. También existe el microanalizador de sonda de electrones que sirve para analizar los rayos-X de alta energía que produce el objeto al ser bombardeado con electrones.

[1]

## ***1.2 MICROSCOPIA CRYO-EM***

Algunos materiales, particularmente las biomoléculas, no son compatibles con las altas condiciones de vacío y no soportan los intensos rayos de electrones usados por los microscopios de transmisión electrónica. El agua que rodea las moléculas se evapora y los electrones queman y destrozan las moléculas debido a su alta energía emitida. Cryo-EM soluciona este problema debido a que utiliza muestras congeladas, haces de electrones más suaves y un procesamiento de imágenes más sofisticado.

Comparando con la difracción por rayos-X, estos necesitan estructuras cristalizadas y, muchas veces, el proceso de cristalización altera la estructura, lo que no hace representativa la forma de la molécula. cryo-EM no necesita de estructuras cristalizadas y permite a los científicos ver como se mueven las biomoléculas y como interaccionan a medida que realizan sus funciones. [2]

	Ventajas	Desventajas	Objetos	Resolución
Cristalografía rayos-X	Resolución alta	Difícil cristalización	Muestras cristalinas	
	Alto rango de pesos moleculares	Difícil difracción	Proteínas, membranas, ribosomas, DNA/RNA y complejos proteicos	Alta
	Fácil creación de modelos	Estructura sólida		
		Estructura cristalina		
NMR	Resolución alta	Pureza en muestra	Muestras solubles en agua	
	Estructuras 3D	Preparación muestra	MWs de menos de 40-50 kDa	Alta
	Estudio dinámico	Simulación computacional compleja		
Cryo-EM	Fácil preparación de muestras	Baja resolución	>150 kDa	
	Estructuras semi perfectas	Altas masas moleculares	Virus, membranas proteicas, proteínas largas, ribosomas, compuestos complejos	Baja (<3.5 ángstroms)
	Muestras pequeñas	Dependiente de técnicas EM		
		Coste de equipamiento		

Tabla 1. Comparación NMR, rayos-X y Cryo-EM

En resumen, las técnicas de cryo-EM generan resultados más exactos que el resto de las técnicas cuando se trata del análisis de estructuras biomoleculares. Estas técnicas fueron creadas entre los años 2013 y 2014. Debido a que la resolución no es muy alta, se crearon varios algoritmos de biocomputación que permitieron generar modelos en dos y tres dimensiones para observar las estructuras de una forma más clara.

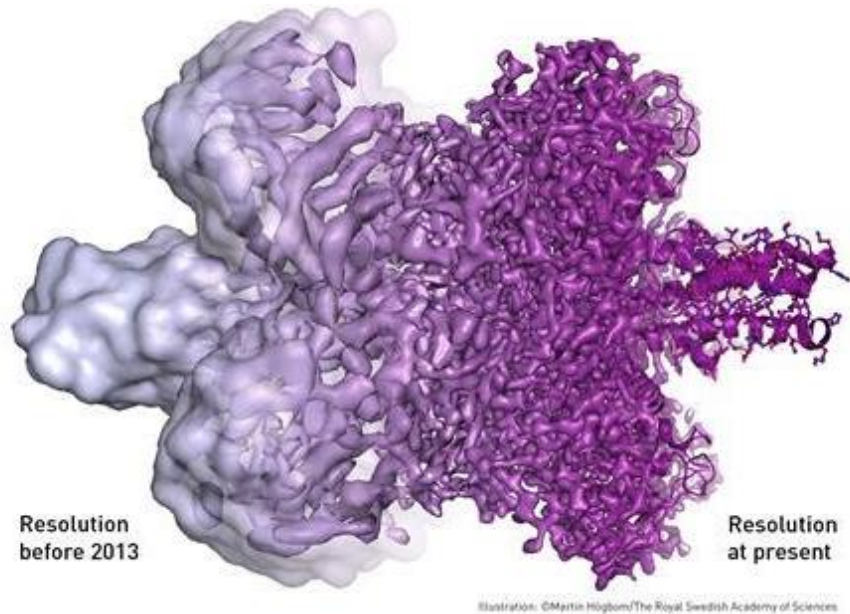


Figura 3. Resolución antes de 2013 vs Resolución cryo-EM

### 1.3 *SCIPION*

- **Descripción:** *Scipion* es un “framework” de procesamiento de imágenes para la obtención de modelos 3D de complejos macromoleculares usando microscopía electrónica (3DEM). Integra varios paquetes de software y presenta una estructura unificada tanto para biólogos como para desarrolladores. *Scipion* permite la ejecución de flujos de trabajo combinando distintas herramientas de software, realizando, al mismo tiempo, el cuidado de formatos y de las conversiones. Todos los pasos seguidos son registrados y el usuario será capaz de reproducirlos cuando desee.

[3]

- **Proyecto:** *Scipion* se basa en la ejecución de programas en un mismo proyecto. A continuación, se muestra un ejemplo de un proyecto en el que se siguen los siguientes pasos:

1. Import micrographs – Importa imagen del microscopio.
2. Manual picking – El usuario elige ciertas partículas y activa el entrenamiento para que se elijan el resto.
3. Extract particles – Saca las partículas obtenidas y crea un fichero.
4. Align – Alinea las partículas para obtener resultados más correctos.
5. Cluster2D – Agrupación y alineación 2D.

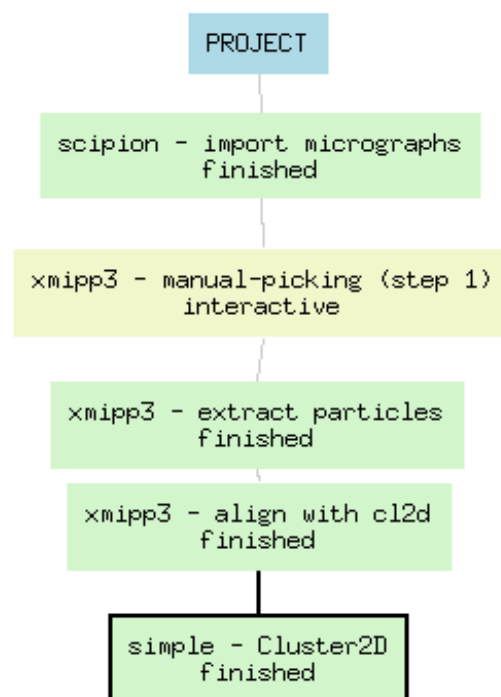


Figura 4. Ejemplo de proyecto simple en Scipion

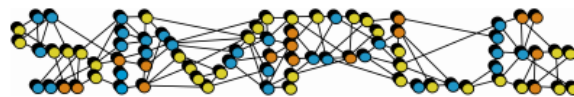
**Paquetes contenidos:** Actualmente, incorpora más de 20 paquetes y más de 200 operaciones a las que el usuario puede acceder. También permite el procesamiento en *streaming*, es decir, puedes procesar las imágenes una vez se empiezan a obtener en el microscopio electrónico. [4]

## 1.4 SIMPLE

SIMPLE es uno de estos paquetes que interesa incorporar en *Scipion*. Sus siglas significan **S**ingle-particle **I**Mage **P**rocessing **L**inux **E**ngine y es un programa destinado al procesamiento de imágenes cryo-EM. Cubre todos los aspectos del análisis individual de partículas y tiene como objetivo principal el crear reconstrucciones en tres dimensiones de partículas con cualquier tipo de simetría (c1-cn, d1-dn, etc.). También, añade herramientas para la reconstrucción de resoluciones atómicas 3D provenientes de series temporales de nanopartículas disueltas en células de grafeno líquido TEM. El back-end consiste en una librería numérica orientada a objetos escrita en Fortran moderno.

La versión actual de SIMPLE es la 3.0 y esta es de donde se van a sacar los protocolos a incorporar en *Scipion*. Esta versión mejora el manejo de metadatos y parámetros de entrada mediante una estructura de proyecto. Este modelo hace el uso más fácil, mejora la organización de los datos de salida y aumenta el rendimiento. [5]

Como ya se ha mencionado, se va a trabajar en la versión 3.0 y su versión actual no está disponible en la página web de SIMPLE y es probable que no se encuentre alguno de los programas en el manual que la propia página suministra.



The SIMPLE 3.0 Command-line Manual

Jan 11, 2019



*Figura 5. SIMPLE logo*

## **1.5 MOTIVACIÓN**

En primer lugar, el tener un programa que simule los mismos pasos mediante distintos algoritmos permite tener varios resultados y elegir el que más se ajuste a nuestra percepción. En segundo lugar, la estructura de *Scipion* permite una ejecución de los pasos de una forma fácil y cómoda para el usuario. En tercer lugar, la mayoría de los microscopios permiten observar las muestras desde una vista y con estos algoritmos se puede ver el completo. En cuarto lugar, es útil tener procesos “secundarios” que realicen funciones como centrar las partículas, eliminar el ruido, calcular ángulos, etc.... Por último, en el ámbito personal, me está viniendo muy bien para practicar la programación en Python y a trabajar en paralelo con distintas personas que trabajan en algoritmos distintos al mío.

## Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este capítulo vamos a describir aquellas herramientas que se han usado a lo largo del proyecto. Se incluirán tanto los protocolos programados como los que son necesarios ejecutar en los wrappers para que funcionen correctamente. En el caso de que el protocolo no haya sido programado en este trabajo se dirá en la descripción. En este capítulo no incluyo los programas *Scipion* y Simple puesto que ya se han explicado en capítulos anteriores. Tampoco describiré la programación en Python ni el sistema operativo LINUX.

- **Jetbrains-pycharm:** Se trata de una herramienta muy útil para programadores en proyectos avanzados. No solo ofrece un entorno para programar en Python, sino que también actúa como herramienta enlazadora de carpetas o proyectos. Ofrece un servicio de búsqueda de clases, ficheros, símbolos y acciones que agiliza el proceso de uso de paquetes, lo que es efectivo cuando se trabaja en proyectos avanzados en los que hay que reutilizar código situado en distintos directorios del equipo.

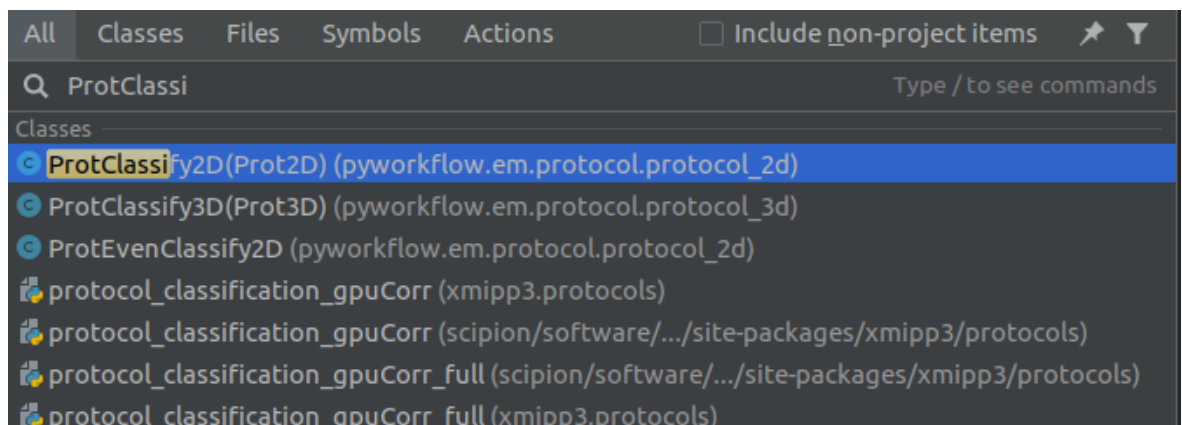


Figura 6. Ejemplo de búsqueda efectiva en Pycharm

La figura 6 muestra un pequeño ejemplo de cómo hacer una “búsqueda inteligente” en Pycharm. Consiste, básicamente, en buscar lo que se quiere en la barra de búsqueda.

➤ **Protocolos SIMPLE:**

- **Refine3D:** Es un protocolo que consiste en un flujo de trabajo distribuido para el refinamiento 3D de partículas basado en la coincidencia de la proyección probabilística. Hay que comentar que este protocolo se ha programado, pero no se tiene pensado incorporar a *Scipion*.
- **Import\_particles:** Se importa un stack de partículas a las que se le asigna una serie de parámetros elegidos por el usuario (o por defecto) como el muestreo, aberración esférica, etc.....
- **Cluster2D:** Clustering y alineación en dos dimensiones de las partículas importadas. Este programa es llamado por refine3D (no se ha programado en este proyecto).
- **Initial\_3D\_model:** Crea un modelo inicial en tres dimensiones a partir de un “set de averages”. Llamado también por refine3D y no programado en este proyecto.
- **cleanUp2D:** Implementa una referencia en dos dimensiones. Como cluster2D, realiza el clustering y la alineación de las partículas importadas. Este programa es una alternativa a cluster2D y ofrece unos resultados más precisos y más realistas en menos tiempo de ejecución.
- **print\_project\_field:** Usado por cleanUp2D y cluster2D para imprimir los ctf o función de transferencia de contrastes. Estos parámetros son necesarios para poder crear la salida de los clusters en dos dimensiones.
- **center:** Ofrece la posibilidad de centrar tanto volúmenes de partículas como las partículas iniciales. Hemos elegido destinarlo únicamente a volúmenes.

- **Mask:** Crea una máscara a partir del radio seleccionado por el usuario, es decir, elimina el ruido y limpia las imágenes. No interesa incorporarlo a *Scipion* porque no aplica la máscara de manera automática.
- **Masscen:** Centra imágenes a partir de su masa en kDa. Este programa sé se ha programado, pero es de una versión antigua por lo que no se incorporará a *Scipion*.
- Además de *Scipion* y Simple se han usado también protocolos de Xmipp3. Una de estas funciones se llama `xmipp_image_convert` y se ha usado en los wrappers para convertir el formato de los archivos de entrada. Las otras son secundarias y se han ejecutado principalmente para realizar pruebas fuera de *Scipion*.

## Capítulo 3. ESTADO DE LA CUESTIÓN

En los últimos años, la microscopía electrónica 3D ha supuesto una revolución por su metodología y su instrumentación. Uno de los factores principales en el cambio que ha supuesto es el desarrollo continuo de software de imágenes. Uno de estos softwares es *Scipion*. Este software incorpora herramientas permitiendo la interoperabilidad entre los diferentes programas pudiendo hacerse uso de distintos protocolos simultáneamente.

Actualmente, incorpora más de 20 paquetes y más de 200 operaciones a las que el usuario puede acceder. La siguiente tabla muestra los paquetes que están disponibles en *Scipion* actualmente con la fecha de sus versiones incorporadas.

	2013	2014	2015	2016	2017	2018
<b>Bsoft</b>		x	x			
<b>cryoEF</b>					x	
<b>CITFIND</b>			x	x	x	x
<b>EMAN</b>			x			
<b>Gautomatch</b>				x		
<b>Frealign</b>		x				
<b>gCIF</b>				x		
<b>gEMpicker</b>			x			
<b>IMAGIC-4D</b>				x		
<b>Localred</b>				x		
<b>Magdistortion</b>				x		
<b>Motioncorr/dosefgpu</b>	x					
<b>Motioncor2</b>					x	x
<b>NYSEC3DFSC</b>					x	
<b>Relion</b>			x		x	
<b>ResMap</b>		x				
<b>SIMPLE</b>	x					
<b>SPIDER</b>	x					
<b>Xmripp</b>				x		
<b>Ublur&amp;summovie</b>			x			

Tabla 2. Paquetes incorporados a Scipion

Aunque *Scipion* es un programa cuyo desarrollo está muy avanzado en lo que respecta a la cantidad de paquetes incorporados, se están desarrollando nuevos programas que salen al mercado y cuyo uso y distribución son gratuitos. Estos programas o bien, ofrecen nuevas funciones o bien las mismas pero mejorando la calidad de los resultados que se obtienen.

Esta es la razón de que resulte interesante para este proyecto el estudio de dichos programas. Por otra parte, *Scipion* no tiene incorporados todos los protocolos de todos los paquetes, por lo que se considera necesario reflexionar si resultaría interesante y efectivo estudiar estos paquetes y sus documentaciones para un mejor desarrollo y ejecución *Scipion*.

## Capítulo 4. DEFINICIÓN DEL TRABAJO

### 4.1 JUSTIFICACIÓN

#### 4.1.1 “¿TODO EN UNO?”

Como se ha visto en apartados anteriores, el objetivo de *Scipion* es que permita disponer de la mayor cantidad de protocolos útiles en un mismo proyecto para su libre uso por los usuarios.

La mayoría de los programas distribuidos hasta la fecha tienen un punto de partida y un punto de final y son bastante lineales y homogéneos en la ejecución de los protocolos. Para su mejor comprensión, expongo el ejemplo de SIMPLE:

SIMPLE es un programa que tiene alguna que otra operación secundaria pero su objetivo principal es el de crear modelos en tres dimensiones que deberán ejecutarse en diferentes pasos. No permite al usuario “jugar” entre estos pasos y el resultado final es el mismo (si se usan las mismas imágenes).

*Scipion*, por el contrario, permite la operación de pasos intermedios y no tiene un objetivo principal, predefinido, sino que es el usuario el que define el objetivo a alcanzar.

En esto consiste mi proyecto: Programar algunos protocolos para incorporarlos a *Scipion* y ayudar y facilitar al usuario a que alcance su objetivo de una forma más rápida y eficiente.

#### 4.1.2 ¿CÓMO PUEDE AYUDAR A LOS BIÓLOGOS?

La biología es un área importante en nuestras vidas. Se dedica a estudiar las estructuras de los seres vivos y de sus procesos vitales. Un área de la biología es la microscopía y esta se dedica a la observación de las células mediante microscopios. Los estudios de esta área



pueden contribuir a la mejora de aspectos importantes de nuestra vida como son la salud o la alimentación.

Todos los microscopios no son suficientemente eficaces, por lo que es muy necesario e importantísimo, disponer de herramientas que mejoren la calidad y resolución de las imágenes que se puedan obtener para su procesamiento.

Además, dichos instrumentos de trabajo deberán permitir la creación de modelos en dos y tres dimensiones para facilitar la tarea de los biólogos en el estudio y deseo de conocer en profundidad las estructuras celulares.

Por tanto, los programas que se intentan desarrollar en este proyecto pretenden ser una respuesta al servicio de ese objetivo vital y que participen en la consecución de resultados que puedan aplicarse a la mejora de la calidad y bienestar de las personas.

#### **4.1.3 FACTOR ECONÓMICO**

El análisis coste- beneficio es importante para el desarrollo de cualquier proyecto. Es importante que exista un equilibrio que permita poner en valor nuestra inversión desde el punto de vista de un inversor.

A este respecto, es interesante mencionar que *Scipion* es una herramienta que no está limitada en uso, es decir, está accesible a todo aquel que quiera utilizarla.

Además, se trata de un programa totalmente gratuito y posiblemente ofrece los mismos servicios, o puede ser que alguno más, que cualquier herramienta por la que hay que pagar un canon o darse de alta como suscriptor.

Finalmente, destacar que se trata de mejorar la “calidad-precio cero” de *Scipion*, lo que podrá conseguirse fácilmente puesto que ya incluye alguna parte de los programas creados en el desarrollo de este proyecto.

#### **4.1.4 OPEN SOURCE**

Hablamos de un proyecto “*open source*”, es decir, cualquier usuario puede desarrollar códigos propios e incorporarlos para su uso en el programa de manera gratuita.

La gran ventaja que se presenta por tanto, es que se ofrece una operabilidad superior y más eficiente a la que puedan presentar otras herramientas similares.

Se trata de una herramienta dinámica puesto que permitirá al usuario desarrollar y utilizar protocolos diseñados “ad hoc”, que no estén disponibles inicialmente, atendiendo a los fines que persigue en su ámbito de actividad.

Se establece así, una relación directa con el desarrollador del programa puesto que el usuario podrá ponerse en contacto con el CNB, mediante correo electrónico, para sugerirle los nuevos códigos que estime útiles y convenientes.

## **4.2 OBJETIVOS**

El objetivo principal de este Trabajo de Fin de Grado es desarrollar e implantar distintos algoritmos utilizados en microscopía electrónica en forma de *wrappers* en *Scipion*, programa desarrollado por el CNB.

Para ello, será necesario realizar previamente una serie de tareas, entre las que se incluyen la realización de pruebas tanto el modo de ejecución como en los resultados obtenidos de los distintos protocolos.

Lo que resulta de interés, es destacar la necesidad de que absolutamente todos los protocolos posibles estén incorporados en *Scipion* puesto que se trata de una herramienta eficaz, rápida y de fácil manejo.

La programación del *wrapper* se desarrollará una vez que se haya elegido y entendido el programa que se haya previamente designado.

Conviene puntualizar, para su consideración, que la programación del *wrapper*, sus resultados o su modo de ejecución, pudieran no ser del todo compatibles con las políticas de *Scipion*, muy estrictas en términos de calidad.

### 4.3 METODOLOGÍA

En primer lugar, para poder trabajar con *Scipion* y los distintos entornos que ofrecerán los algoritmos para la programación de los *wrappers*, se hará uso de una máquina virtual de Oracle instalando el sistema operativo Linux en su versión “Ubuntu 18.04.1 LTS”.

En segundo lugar, se instalarán *Scipion* y los distintos programas de los que habrá que obtener los diferentes algoritmos.

En la elaboración de este proyecto he utilizado, por considerarlos los más adecuados, los programas *SIMPLE*, *goCTF* y *Sphire*.

A continuación, en la lista siguiente, detallo el link correspondiente con la ubicación de las instrucciones que especifican el modo de descarga e instalación de todos los programas mencionados.

Tengo que explicar, no obstante, que no incluyo la referencia de *Sphire* y *goCTF*, puesto que no ha sido posible el desarrollo de ningún *wrapper* en sus protocolos.

- *Scipion*: <https://Scipion-em.github.io/docs/release-2.0.0/docs/Scipion-modes/how-to-install.html>
- *SIMPLE*: <https://simplecryoem.com/> → v.2.5: Se ha utilizado la versión 3.0, que no está disponible en la web.

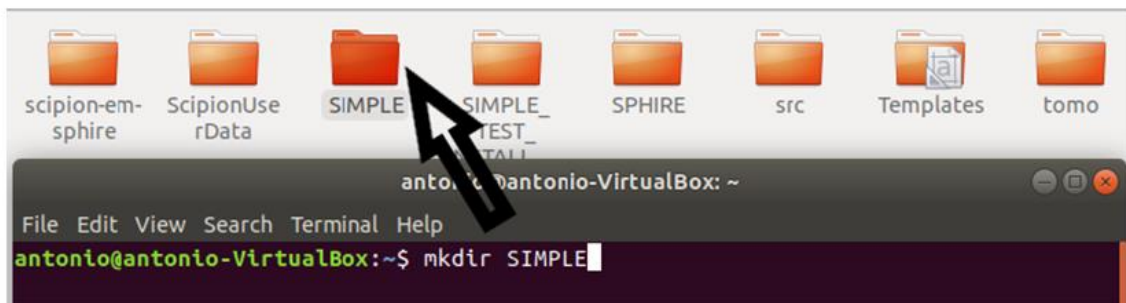
*Scipion* tiene Plugins instalados para los distintos entornos hacia los que se hacen los *wrappers* (*xmipp3*, *simple*, *xmipp*, etc...).

---

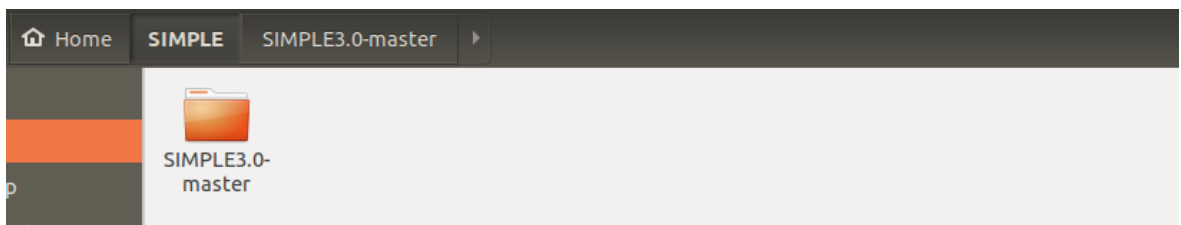
Sin embargo, dichos elementos y configuración no son suficientes para habilitar al usuario a la inclusión de sus programas dentro de la aplicación de *Scipion*, por lo que será necesario desarrollar una serie de etapas o pasos.

Estos pasos, que aplicados al ejemplo SIMPLE, se describen a continuación:

- 1) Crear una carpeta con el programa y sus archivos en el directorio que se desee.



*Figura 7. Creación de carpeta con el programa*



*Figura 8. Carpeta con el programa*

- 2) Añadir el path al sistema.



*Figura 9. Variable de entorno de SIMPLE*

- 3) Crear la carpeta que verá **Scipion**. En este caso se va a llamar **Scipion-em-simple/SIMPLE3.0/simple/protocols**.



*Figura 10. Carpeta link scipion-simple*

En esta carpeta es donde el usuario incluirá sus protocolos como paquetes. Como se está trabajando en Python, se crea un archivo llamado **`__init__.py`** donde se escribirán los nombres de los protocolos que se programan. El código irá en esta misma carpeta.



*Figura 11. Definición protocolo*

- 4) Realizar el link entre Scipion y SIMPLE. Para ello localizamos dónde tenemos descargado el programa de Scipion y buscamos la carpeta Scipion/config/. Abrimos el fichero Scipion.conf y añadimos el PATH a SIMPLE. Este PATH será el mismo que se ha añadido en el paso 2.

`export SIMPLE_PATH=../build`

- 5) Localizar la carpeta `software/lib/python-2.7/site-packages/` y creamos el link ejecutando el siguiente comando:

`ln -s <dirección a link> <directorio actual>`



Figura 12. Link a protocolos de simple

- 6) Comprobar que se observan los protocolos en *Scipion*.

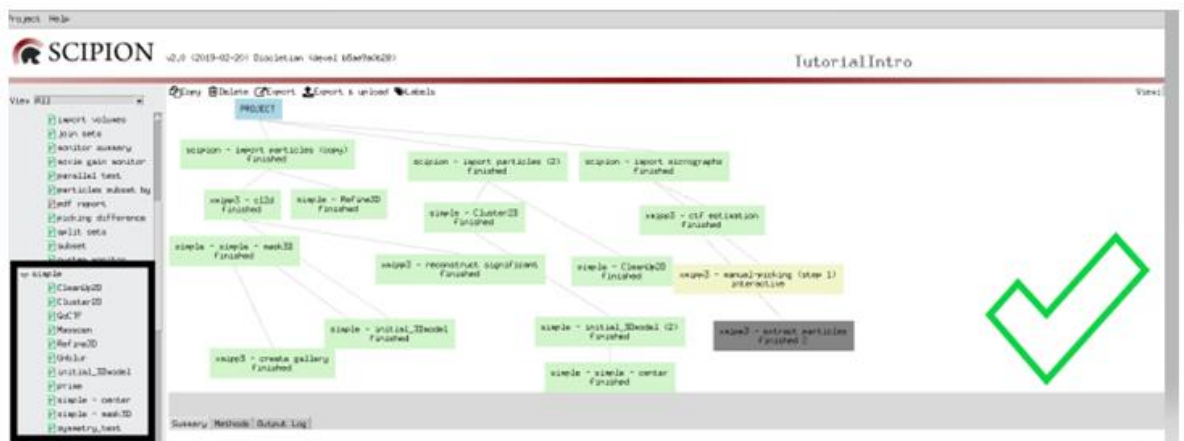


Figura 13. Comprobación link correcta

Ya se puede comprobar la funcionalidad de nuestros programas dentro de Scipion. Ahora será necesario tener programas propios, por lo que se eligen los plugins (simple, sphire, etc...) y los protocolos de interés de estos y se procede a programar los wrappers.

#### 4.4 PLANIFICACIÓN

En este apartado se detallan las diferentes etapas en las que se ha desarrollado el trabajo y su distribución en el tiempo.

ACTIVIDAD	INICIO	FINAL	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo	Abril	Mayo	Junio
Estudio de tecnologías	01/10/2018	15/12/2018	█	█	█						
Python	01/10/2018	01/11/2018	█								
Scipion	02/11/2018	01/12/2018		█							
SIMPLE	02/12/2018	14/12/2018			█						
Pycharm	14/12/2018	15/12/2018			█						
Creación de wrappers	16/12/2018	30/06/2019				█	█	█	█	█	█
Pruebas en origen	16/12/2018	31/05/2019									█
Programación	26/12/2018	20/06/2019				█	█	█	█	█	█
Pruebas en Scipion	01/01/2019	30/06/2019				█	█	█	█	█	█
Duración del proyecto	01/10/2018	30/06/2019	█	█	█	█	█	█	█	█	█

*Figura 14. Representación temporal del trabajo de fin de grado*

La figura 14 representa un diagrama de Gantt en el que se muestra una planificación aproximada de la duración de las tareas ejecutadas a lo largo del año.

El proyecto se ha ejecutado en las diferentes etapas que se relacionan a continuación:

La primera parte del proyecto se ha centrado en el estudio de las diferentes tecnologías. A esta tarea se ha dedicado prácticamente algo más de tres meses puesto que se trata de una materia compleja que requería de un conocimiento exhaustivo. En particular, Python, una de las tecnologías empleadas más importantes puesto que es el lenguaje que se utiliza para programar los wrappers. También se ha profundizado en el estudio de Scipion un programa

que se ha granulado para conocer en detalle el funcionamiento de sus protocolos, y para comprender como se generan las entradas y las salidas y en qué modo trabaja con los formatos.

Asimismo, se han estudiado otras tecnologías tales como Pycharm y SIMPLE. SIMPLE tiene un funcionamiento similar a Scipion aunque con determinadas especificidades que le son propias. A diferencia de este, Simple se ejecuta desde el terminal (cmd), por lo que no ha sido necesario dedicarle tanto tiempo.

En la celda de SIMPLE se ha añadido una etiqueta de “otros” para referirse al otros programas tales como SPHIRE o goCTF que también han sido objeto de estudio aunque algunos de estos no se han llegado a usar durante el proyecto.

La segunda parte del proyecto se ha centrado en la creación de los denominados wrappers y que se ha desarrollado en un plazo aproximado de 6 meses. Se trata de la ejecución de diferentes ciclos cada uno mediante tres pasos: pruebas en origen, programación de los wrappers y pruebas en Scipion.

En la figura 14 se ha dibujado el tiempo total invertido en cada uno de los pasos considerando todos los wrappers conjuntamente, en vez de considerar el tiempo consumido de forma individualizada puesto que muchos de los wrappers se han ido ejecutando de forma simultánea y su duración ha sido diferente para cada uno de ellos.



## Capítulo 5. SISTEMA: WRAPPERS

### 5.1 DEFINICIÓN

El trabajo que se va a desarrollar consiste, básicamente, en la creación y configuración de wrappers o códigos que permiten la ejecución de los scripts de un programa, adecuando las entradas y salidas, para su incorporación en Scipion.

La función principal de los wrappers, mediante la oportuna codificación, consistirá en llamar a los protocolos de diferentes programas con el objetivo de conseguir el adecuado tratamiento de las imágenes de las partículas que previamente habrán sido importadas mediante la utilización de cualquier microscopio electrónico.

Estos protocolos podrán ejecutar distintas funciones tales como la creación de clústeres en dos dimensiones, el desarrollo de modelos iniciales en tres dimensiones, definir procesos de refinamiento en tres dimensiones, etc...

### 5.2 ESTRUCTURA

Cada *wrapper* consta de los siguientes elementos:

- ❖ **Inicialización:** Se inicia el protocolo con el nombre que se vaya a asignar y en el modo en que aparecerá en *Scipion*.
- ❖ **Definición de parámetros:** Se construye una ventana con los diferentes parámetros que se consideren y que el usuario deberá introducir manualmente. Estos parámetros pueden ser de diferentes tipos, tales como numéricos, archivos, strings, etc.
- ❖ **Pasos o steps:** Deberán definirse las funciones, que siguiendo un orden predeterminado, se ejecutarán. En la interfaz gráfica de *Scipion* aparecerá el número de pasos a seguir que se irá actualizando a medida que finalice su

ejecución. Es importante señalar que no se deberán identificar los códigos de las funciones.

- ❖ **Funciones de los steps:** Se trata del código que va a ser ejecutado. Como norma general suele haber 3 funciones: La primera es la denominada *convertInputs* que se encarga de convertir los archivos de entrada, en caso de que los hubiera, a sus correspondientes formatos. La segunda, o función del *protocolo*, que básicamente llama al protocolo que se desea ejecutar como si se estuviese haciendo por consola (cmd). Y, finalmente, la tercera función, *convertOutputs*, que crea los ficheros de salida y obtiene los resultados que el usuario podrá observar en la aplicación.
- ❖ **Otras:** No son obligatorias y son funciones destinadas a la decoración del protocolo que ejecutan la validación de parámetros y añaden información a los usuarios.
- ❖ **Propias:** Existe la posibilidad de crear funciones adicionales a las mencionadas anteriormente con objeto de ejecutar cualesquiera otras acciones que se estimen convenientes. A modo de ejemplo, en el caso de que un protocolo genere varios ficheros que correspondan a numerosas iteraciones, podríamos definir una función que permitiera elegir el fichero que nos interese.

### 5.3 IMPLEMENTACIÓN

Los wrappers se implementarán al programa de Scipion instalado por el propio usuario mediante los pasos mencionados en el apartado 4.3 de “metodología”.

Una vez que los wrappers hayan sido probados y que se haya verificado la calidad de los resultados producidos por los mismos, éstos se subirán al repositorio de github de Scipion donde se encuentran todos los paquetes accesibles al usuario: <https://github.com/I2PC/>.

## Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se explica la realización y desarrollo de los wrappers en la ejecución del proyecto.

La explicación de cada wrapper se dividirá en dos partes: En la primera, se describirá brevemente la funcionalidad del programa y se mostrarán las partes del código. En cada código se incluye un pequeño texto de derechos de autor que no se mostrará en este capítulo. En la segunda se mostrarán los resultados dentro de Scipion y se hará un análisis parcial de los mismos.

Debe resaltarse que el análisis se enfocará en un estudio comparativo de los aspectos relacionados con el procesamiento de imágenes sin entrar a valorar aspectos concretos relacionados con la biología.

### ➤ \_\_init\_\_

```
from protocol_prime import ProtPrime
from protocol_unblur import ProtUnblur
from protocol_Cluster2D import ProtCluster2D
from protocol_init3D import ProtInit3D
from protocol_symmetry import ProtSym
from protocol_refine3D import ProtRefi3D
from protocol_masscen import ProtMasScen
from protocol_mask import ProtMaskSimple
from protocol_goctf import ProtGoCTF
from protocol_cleanup2D import ProtCleanup2D
from protocol_center import ProtCenterSimple
```

Figura 15. Código \_\_init\_\_

La figura 15 representa el código del archivo \_\_init\_\_. En este archivo se escriben los programas que estarán dentro de nuestro módulo de simple.

Este programa únicamente permite que se vean todos los protocolos dentro de la misma carpeta en Scipion y no produce resultados.

## ➤ cleanUp2D

```
import os

import pyworkflow.protocol.params as params
from pyworkflow.em.protocol import ProtClassify2D
from pyworkflow.utils.path import makePath, moveFile
from pyworkflow.em.convert import ImageHandler
import simple

class ProtCleanup2D(ProtClassify2D):
    """
    is a distributed workflow implementing
    a reference-free 2D alignment/clustering
    algorithm suitable for the first pass of
    cleanup after picking

    """
    _label = 'CleanUp2D'

    def __init__(self, **kwargs):
        ProtClassify2D.__init__(self, **kwargs)
```

Figura 16. cleanUp2D: definición y paquetes

La primera parte del código cleanup2D es la mostrada en la figura 16. Aquí, se definen los paquetes, ya sean de Python o de Scipion, que se van a utilizar. Además, se añade una pequeña descripción del programa como comentario.

```
# ----- DEFINE param functions -----
def _defineParams(self, form):
    form.addSection(label='Input')
    form.addParam('inputParticles', params.PointerParam, pointerClass='SetOfParticles', allowsNull=False,
                 label='Input Particles', important=True)
    form.addParam('clusters', params.IntParam, default=5, label='Number of clusters', important=True)
    form.addParam('mask', params.IntParam, default=0, label='Mask radius', help='Mask radius (in Pixels)', expertLevel=params.LEVEL_ADVANCED)
    form.addParam('maxIts', params.IntParam, default=0, label='Iterations', help='Maximum iterations', expertLevel=params.LEVEL_ADVANCED)
    form.addParam('lp', params.IntParam, default=0, label='Low pass filter (Angstroms)', help='Low pass filter (Angstroms)',
                 expertLevel=params.LEVEL_ADVANCED)
    form.addParam('hp', params.IntParam, default=0, label='High pass filter (Angstroms)', help='High pass filter (Angstroms)',
                 expertLevel=params.LEVEL_ADVANCED)
    form.addParam('updatefrac', params.FloatParam, default=0, label='Fraction of updated particles per iteration', help='Value between 0.1 and 1',
                 expertLevel=params.LEVEL_ADVANCED)
    form.addParallelSection(threads=4, mpi=0)
```

Figura 17. cleanUp2D: ventana de parámetros

En esta parte, representada por la figura 17, se pedirá al usuario que introduzca los parámetros necesarios para la ejecución del protocolo. Los parámetros obligatorios son:

- inputParticles: fichero con las partículas (seleccionado de pasos anteriores).
- Clusters: El número de clusters que quiere que el programa cree.

Adicionalmente se añade un menú “LEVEL\_ADVANCED” con los parámetros que no son necesarios (mask, maxits, lp, hp y updatefrac).

- Mask: radio de la máscara de las partículas en píxeles.
- Maxits: si el usuario no consta de un ordenador con suficiente potencia, puede usar esta opción para reducir el tiempo de ejecución del programa disminuyendo el número de iteraciones.
- Lp/hp: Valores en Angstroms del filtrado paso-bajo/alto.
- Update\_frac: porcentaje de partículas actualizadas por iteración.

A continuación, se definen las funciones a ejecutar: convertInput, cleanupStep y createOutputStep.

```
def _insertAllSteps(self):
    self._insertFunctionStep("convertInput")
    self._insertFunctionStep('cleanupStep')
    self._insertFunctionStep("createOutputStep")

# ----- STEPS functions -----
def convertInput(self):
```

Figura 18. cleanUp2D: Pasos

```
def convertInput(self):
    inputPart = self.inputParticles.get()
    inputPart.writeStack(self._getExtraPath("particles.mrc"))

def cleanupStep(self):
```

Figura 19. cleanUp2D: convertInput

La función “convertInputStep” convierte el archivo de las partículas de entrada al formato “.mrc” para evitar errores de formatos.

```
def cleanupStep(self):
    partFile = self.getExtraPath("particles.mrc")
    SamplingRate = self.inputParticles.get().getSamplingRate()
    kV = self.inputParticles.get().getAcquisition().getVoltage()
    partitions = 1
    cs = self.inputParticles.get().getSphericalAberration()
    fracA = self.inputParticles.get().getAmplitudeContrast()
    partName = os.path.basename(partFile)
    partName = os.path.splitext(partName)[0]
    tmpDir = self.getTmpPath(partName)
    makePath(tmpDir)

    paramsOri = 'prg=print project field oritype=ptcl2D > oritab.txt'
    paramsImp = 'prg=import particles cs=%f ctf=no fracA=%f kv=%f smpd=%f stk=%s ' % (cs, fracA, kV, SamplingRate, os.path.abspath(partFile))
    paramsClean = 'prg=cleanup2D ncls=%d nparts=%d nthr=%d ' % (self.clusters.get(),
                                                                partitions, self.numberOfThreads.get())

    if self.maxits.get() > 0:
        paramsClean = paramsClean + (' maxits=%d ' % self.maxits.get())
    if self.mask.get() > 0:
        paramsClean = paramsClean + (' msk=%d ' % self.mask.get())
    if self.lp.get() > 0:
        paramsClean = paramsClean + (' lp=%d ' % self.lp.get())
    if self.hp.get() > 0:
        paramsClean = paramsClean + (' hp=%d ' % self.hp.get())
    if self.updatefrac.get() > 0.1 and self.updatefrac.get() <= 1:
        paramsClean = paramsClean + (' update_frac=%f ' % self.updatefrac.get())

    self.runJob(simple.Plugin.sim_exec(), 'prg=new_project projname=temp', cwd=os.path.abspath(tmpDir),
                env=simple.Plugin.getEnviron())
    self.runJob(simple.Plugin.sim_exec(), paramsImp, cwd=os.path.abspath(tmpDir) + '/temp',
                env=simple.Plugin.getEnviron())
    self.runJob(simple.Plugin.distr_exec(), paramsClean, cwd=os.path.abspath(tmpDir) + '/temp',
                env=simple.Plugin.getEnviron())
    self.runJob(simple.Plugin.sim_exec(), paramsOri, cwd=os.path.abspath(tmpDir) + '/temp',
                env=simple.Plugin.getEnviron())
```

Figura 20. cleanUp2D: cleanUpStep parte 1

```
path = self.getPath(tmpDir)
os.remove(os.path.abspath(self.getExtraPath("particles.mrc")))
moveFile(path, self.getExtraPath(partName + "_cavgs_final.mrc"))
mvRoot2 = os.path.join(tmpDir + '/temp', "oritab.txt")
ih = ImageHandler()
ih.convert(self.getExtraPath(partName + "_cavgs_final.mrc"), self.getExtraPath(partName + "_cavgs_final.mrcs"))
moveFile(mvRoot2, self.getExtraPath(partName + "_oritab.txt"))
createOutputFolder(self)
```

Figura 21. cleanUp2D: cleanUpStep parte 2

Las figuras 20 y 21 corresponden al código de la función cleanUpStep. En la parte 1, se crean parámetros correspondientes a la frecuencia de muestreo, voltaje, aberración esférica y contraste de amplitud de las partículas (guardados en Scipion por otros protocolos). Aquí también se define el script a ejecutar y, en el caso de que el menú LEVEL\_ADVANCED que se ha mencionado antes haya sido rellenado, se añaden sus parámetros a la ejecución del script. Finalmente, se ejecutan los scripts new\_project, import\_particles, cleanup2D y print\_project\_field mediante el Plugin de simple y el comando “runjob”. CleanUp2D genera varios archivos “.mrc” correspondientes a distintas iteraciones del protocolo, por lo que se crea una función

para coger la última iteración denominada getPath(). En la figura 21 se llama a esta función y se mueve el archivo devuelto por esta y el archivo generado por print\_project\_field a otro directorio. De aquí se cogerá lo necesario para generar las salidas.

```
def getPath(self, path):  
    Iter = 1  
    direc = path + '/temp/2_cleanup2D/cavgs_iter00%d.mrc' % Iter  
  
    while (os.path.exists(direc)):  
        Iter += 1  
        if (Iter < 10):  
            direc = path + '/temp/2_cleanup2D/cavgs_iter00%d.mrc' % Iter  
        elif (Iter >= 10 and Iter < 100):  
            direc = path + '/temp/2_cleanup2D/cavgs_iter0%d.mrc' % Iter  
        else:  
            direc = path + '/temp/2_cleanup2D/cavgs_iter%d.mrc' % Iter  
  
    Iter -= 1  
    if (Iter < 10):  
        direc = path + '/temp/2_cleanup2D/cavgs_iter00%d.mrc' % Iter  
    elif (Iter >= 10 and Iter < 100):  
        direc = path + '/temp/2_cleanup2D/cavgs_iter0%d.mrc' % Iter  
    else:  
        direc = path + '/temp/2_cleanup2D/cavgs_iter%d.mrc' % Iter  
  
    return direc
```

Figura 22. cleanUp2D: función getPath

```
def __createOutputStep(self):
    fh = open(self._getExtraPath("particles_oritab.txt"))

    lines = []
    for line in fh.readlines():
        ang = 0.0
        x = 0.0
        y = 0.0
        corr = 0.0
        classId = 0
        for token in line.split():
            if token.startswith("e3="):
                ang = float(token[3:])
            elif token.startswith("x="):
                x = float(token[2:])
            elif token.startswith("y="):
                y = float(token[2:])
            elif token.startswith("corr="):
                corr = float(token[5:])
            elif token.startswith("class="):
                classId = int(float(token[6:]))
                break

        lineItem = dict({"angle": ang, "class": classId, "xTranslation": x, "yTranslation": y, "corr": corr})
        lines.append(lineItem)
    fh.close()

    inputParticles = self.inputParticles.get()
    classes2DSet = self._createSetOfClasses2D(inputParticles)
    classes2DSet.classifyItems(updateItemCallback=self._updateParticle,
                              updateClassCallback=self._updateClass,
                              itemDataIterator=iter(lines))
    result = {'outputClasses': classes2DSet}
    self._defineOutputs(**result)
    self._defineSourceRelation(self.inputParticles, classes2DSet)

# ----- UTILS functions -----
def _updateParticle(self, item, lineItem):
    classNum = lineItem["class"]
    item.setClassId(classNum)

def _updateClass(self, item):
    classId = item.getObjId()
    item.setAlignment2D()
    avgFile = self._getExtraPath('particles_cavgs_final.mrcs')
    rep = item.getRepresentative()
    rep.setSamplingRate(item.getSamplingRate())
    rep.setLocation(classId, avgFile)
```

Figura 23. cleanUp2D: createOutputStep

En la figura 23 se muestra el código para crear las salidas en Scipion. La función principal es createOutputStep y se apoya en las otras dos mostradas. Primero, se abre el archivo “particles\_oritab.txt” creado por print\_project\_field que maneja datos para crear los stacks de clases. A continuación, las coordenadas x e y, el ángulo, la correlación y el id de clase y las guardamos. Por último, se utiliza el archivo creado por cleanUp2D y se le adjunta los parámetros seleccionados para que Scipion cree la salida.



## RESULTADOS

Partimos de un stack inicial de partículas de una proteína denominada trpv1 importado a Scipion.

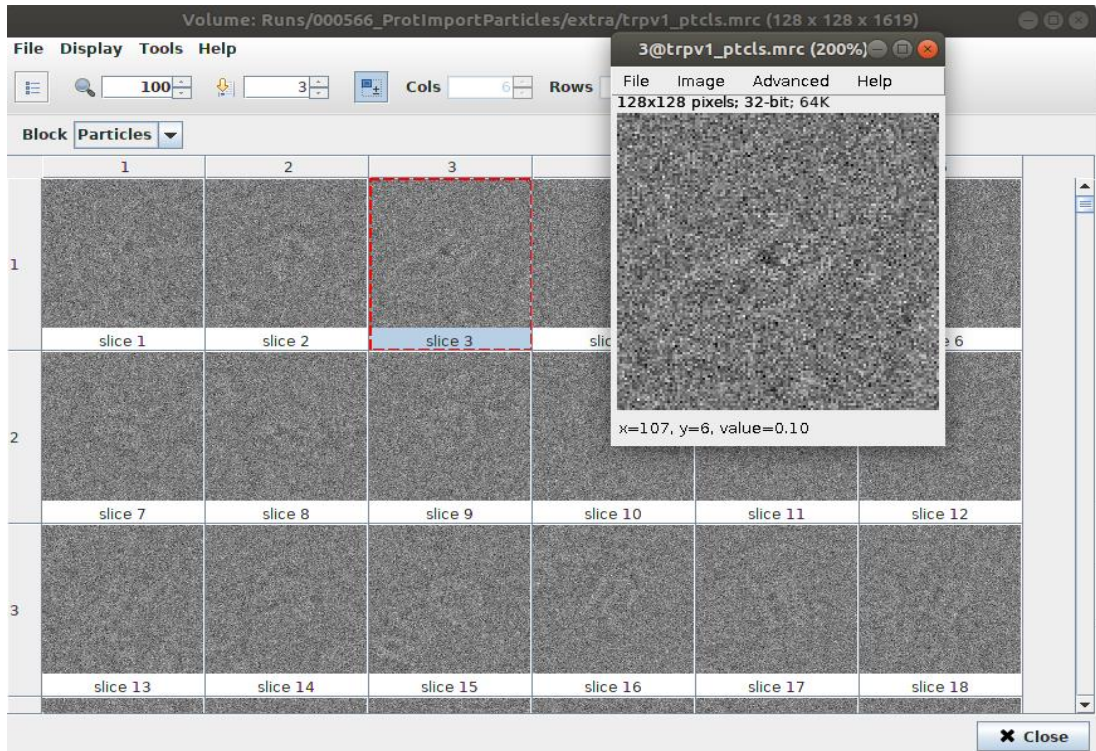


Figura 24. cleanUp2D-resultados: stack inicial

Ahora, ejecutamos el protocolo cleanUp2D con los parámetros que aparecen en la figura 25.

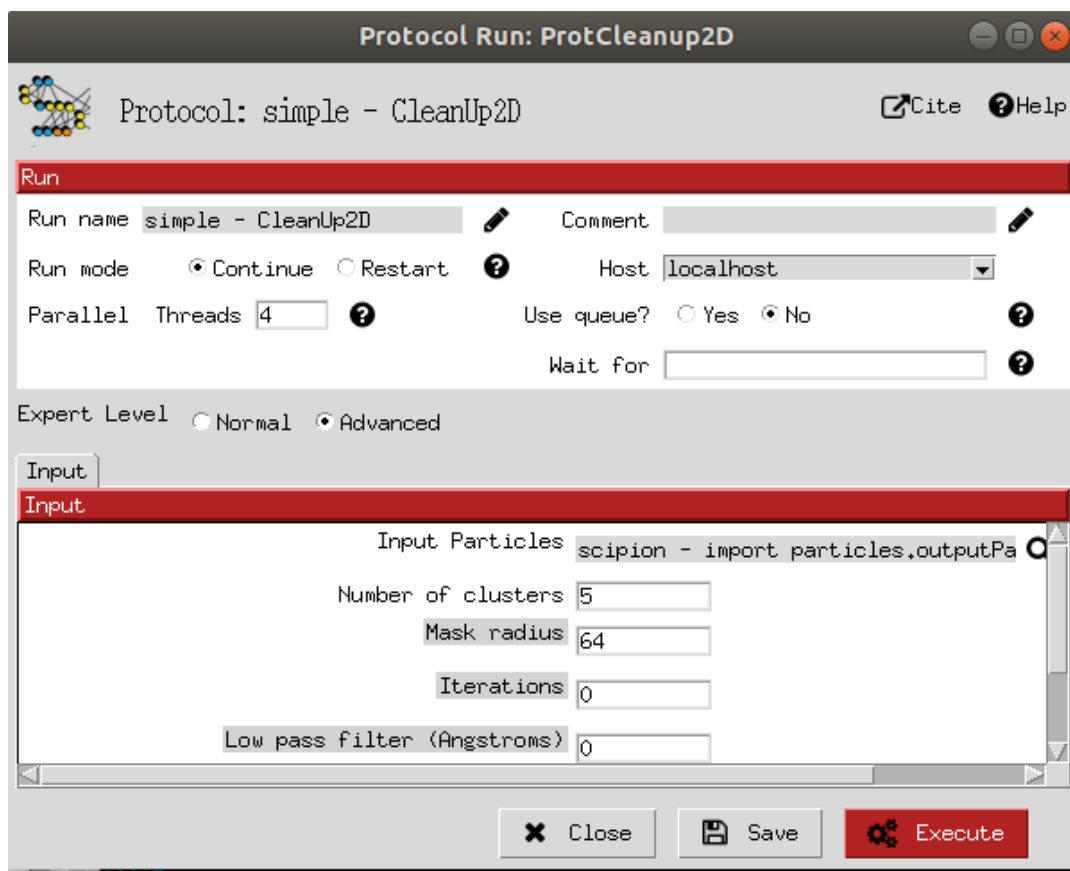


Figura 25. cleanUp2D-resultados: parámetros introducidos por usuario

Como estamos en LEVEL\_ADVANCED, los parámetros que se queden con un valor de 0 se introducirán con sus valores por defecto y el resto se añadirán a la ejecución del script.

Le damos a “Execute” y obtenemos las clases mostradas en la figura 26.

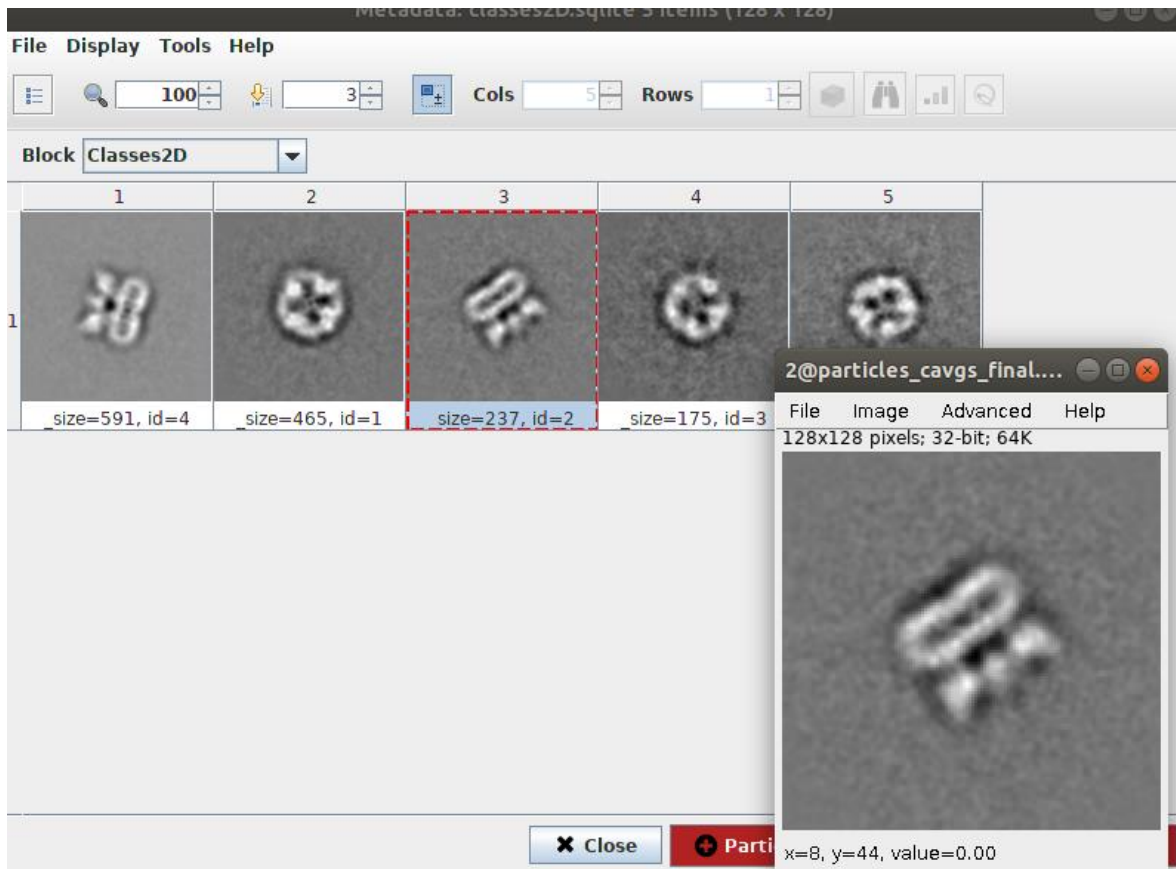


Figura 26. cleanUp2D-resultados: resultado 5 clusters

A continuación probamos a efectuar los mismos pasos con 10 clusters en vez de 5:

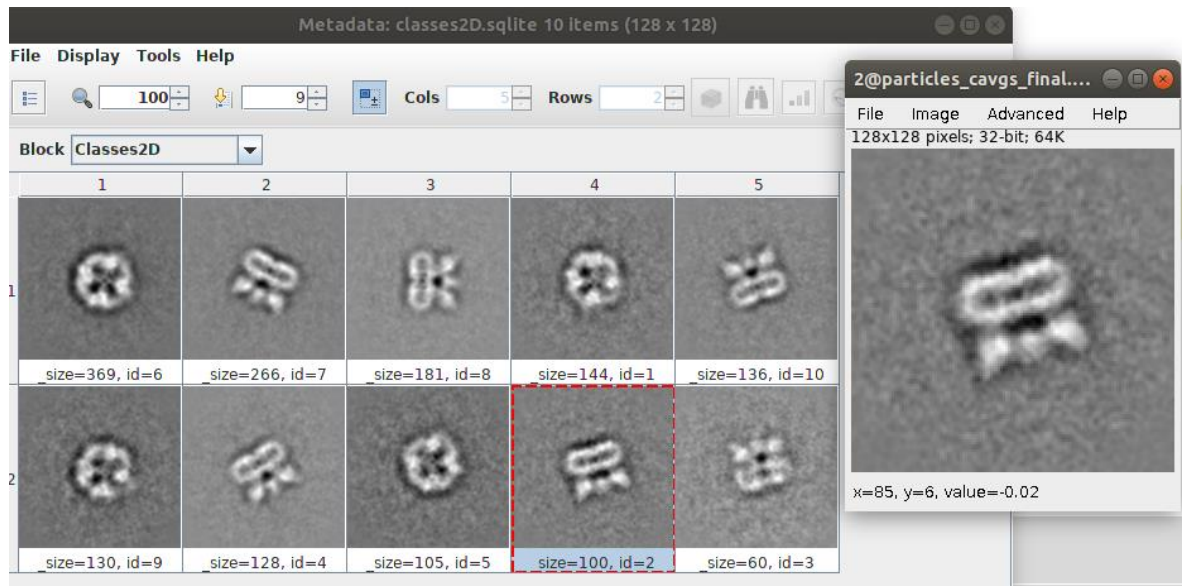


Figura 27. cleanUp2D-resultados: resultado 10 clusters

Se puede observar cómo se crean tantas clases en dos dimensiones (imágenes de las partículas) como el número de clusters que introduce el usuario. También se puede apreciar un cambio significativo en la calidad de las imágenes.

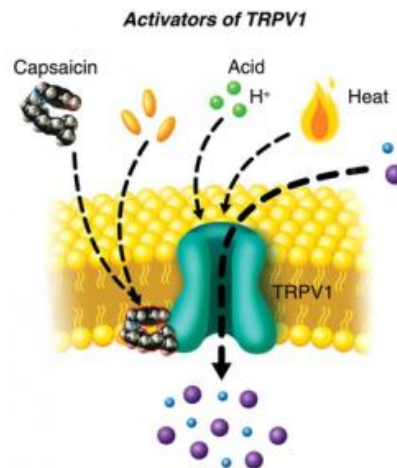


Figura 28. Membrana TRPV1

La figura 28 demuestra que, efectivamente, los resultados obtenidos son correctos y que el protocolo produce imágenes de calidad.

## ➤ Center

Este programa centra los volúmenes obtenidos por otros protocolos. Vamos a comenzar analizando el código:

```
import os
import pyworkflow.em as em
import pyworkflow.protocol.params as params
from pyworkflow.em.protocol import ProtAlignVolume
from pyworkflow.utils.path import makePath, moveFile
from pyworkflow.em.convert import ImageHandler
import simple

class ProtCenterSimple(ProtAlignVolume):
    """
    Center

    Is a program for centering volumes

    """
    _label = 'simple - center '

    def __init__(self, **kwargs):
        ProtAlignVolume.__init__(self, **kwargs)

    # -----Define Params of function-----

    def defineParams(self, form):
        form.addSection(label='Input')
        form.addParam('inputVolume', params.PointerParam, pointerClass='Volume', allowsNull=False,
                      label='Input Volume', important=True, help='Import volume file to apply mask')
        form.addParam('sampling', params.FloatParam, default=2.0, label='Sampling rate in pixels', help='Sampling rate',
                      important='true')

        form.addParallelSection(threads=4, mpi=0)
```

Figura 29. Center: Descripción y parámetros usuario

La parte del código mostrada en la figura 29 corresponde con la descripción del protocolo y con los parámetros que ha de introducir el usuario. En este caso, no hay menú LEVEL\_ADVANCED y ambos parámetros son obligatorios. Estos son:

- inputVolume: Archivo que contiene el volumen sacado por otro protocolo.
- Sampling: Parámetro correspondiente al muestreo de las imágenes cuyo valor viene dado en píxeles.

Las dos siguientes partes son la definición de las funciones y la primera función, dedicada a la conversión del volumen a su formato correspondiente (figura 30).

```
def _insertAllSteps(self):
    self._insertFunctionStep("convertInput")
    self._insertFunctionStep("centerStep")
    self._insertFunctionStep("createOutputStep")
    pass

# -----Define methods of steps-----

def convertInput(self):
    ImageHandler().convert(self.inputVolume.get(), self._getExtraPath("refVolume.mrc"))
```

Figura 30. Center: Funciones y conversión a formato

```
def centerStep(self):
    partFile = self._getExtraPath("refVolume.mrc")
    partName = os.path.basename(partFile)
    partName = os.path.splitext(partName)[0]
    tmpDir = self._getTmpPath(partName)

    makePath(tmpDir)

    center = 'prg=center voll=%s smpd=%f nthr=%d' % (
        os.path.abspath(partFile), self.sampling.get(), self.numberOfThreads.get())

    self.runJob(simple.Plugin.sim_exec(), center, cwd=os.path.abspath(tmpDir),
               env=simple.Plugin.getEnviron())

    path = os.path.join(tmpDir + '/1_center', 'shifted_vol_state1.mrc')
    moveFile(path, self._getExtraPath(partName + "_outvol.mrc"))
```

Figura 31. Center: Ejecución del script de center

La figura 31 muestra el código de la función que ejecuta el script de center, creando el archivo correspondiente a un volumen centrado. Este archivo se moverá al directorio correspondiente para que se vean las salidas en Scipion.

```
def createOutputStep(self):
    vol = em.Volume()
    vol.setLocation(self._getExtraPath('refVolume_outvol.mrc'))
    vol.setSamplingRate(self.inputVolume.get().getSamplingRate())
    self._defineOutputs(outputVol=vol)
    self._defineSourceRelation(self.inputVolume, vol)
```

Figura 32. Center: Volumen de salida

Por último, se adjunta el código de la función que crea la salida en Scipion. Ésta crea un volumen y le asigna el directorio del archivo creado por el protocolo.

## RESULTADOS

Vamos a usar las mismas imágenes de muestra que para el programa de cleanUP2D. A partir de un set de partículas en dos dimensiones crearemos los volúmenes que servirán como archivo de entrada a nuestro programa.

Usamos, mismamente, el archivo salida de cleanUp2D. A este le aplicamos un desplazamiento de las imágenes para poder comprobar el valor de los resultados obtenidos por center. Obtenemos la siguiente muestra:

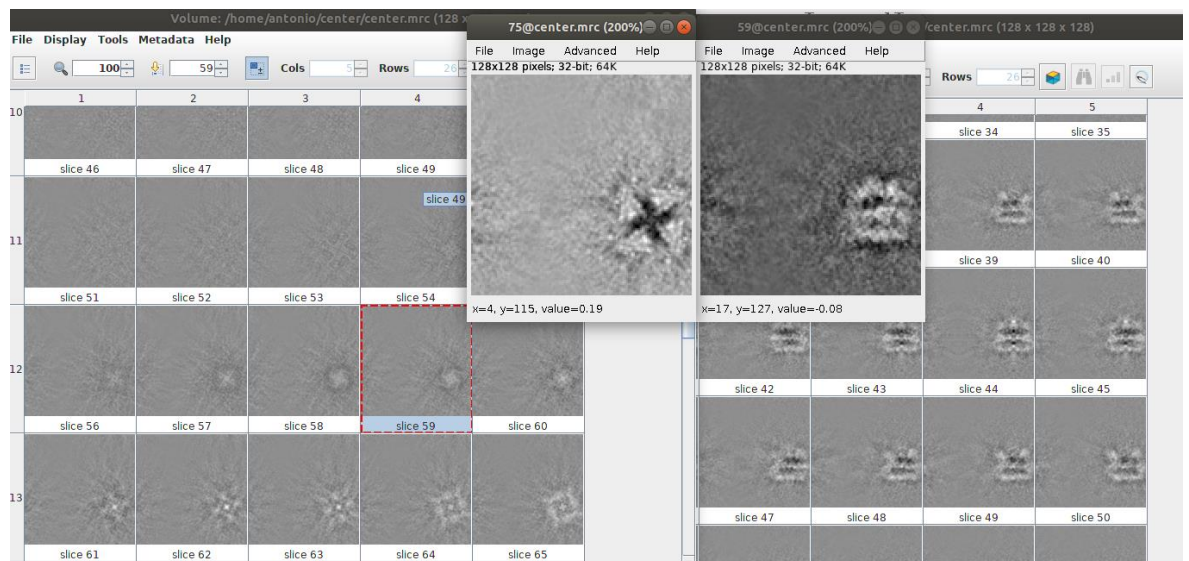


Figura 33. Center: Volumen de entrada en ejecución-vistas Z e Y

Ahora vamos a ejecutar nuestro protocolo con los parámetros de la figura 34:

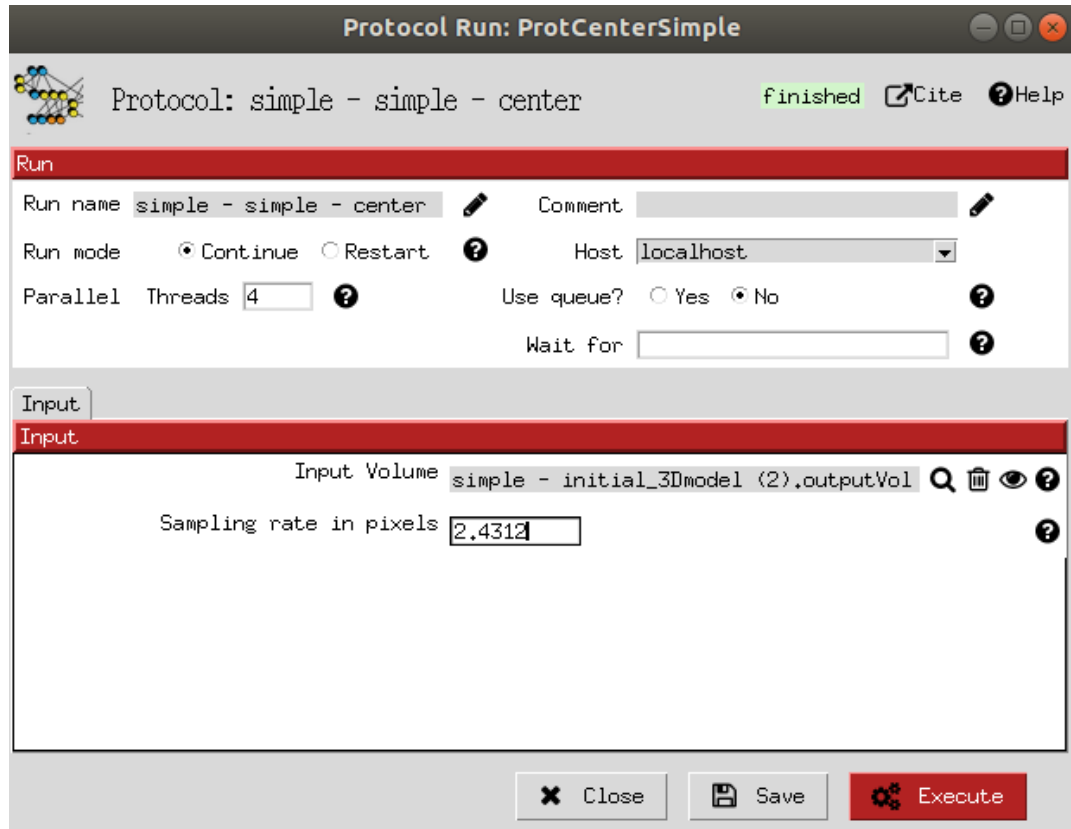


Figura 34. Center: Parámetros de ejecución



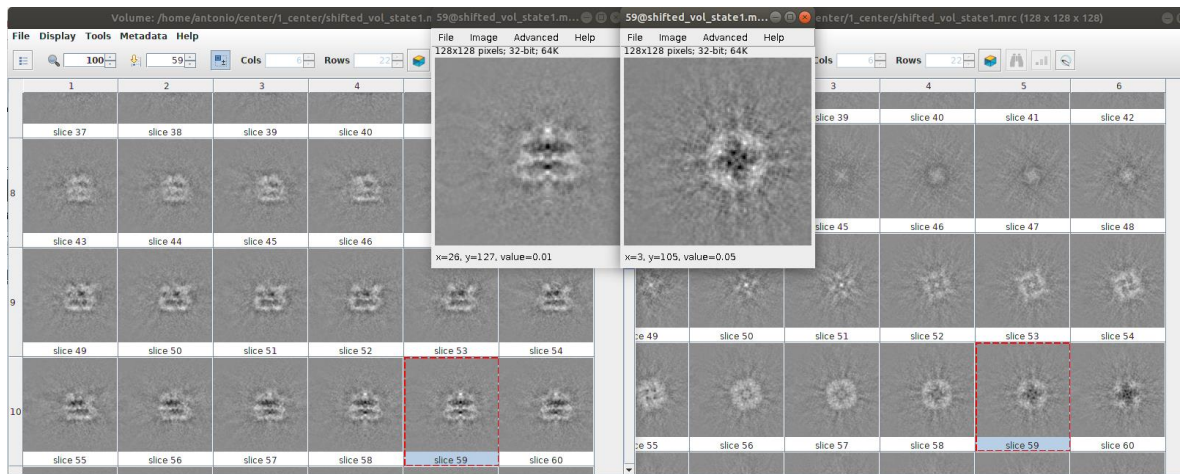


Figura 35. Center: Resultado final-vistas Z e Y

Vemos como el programa a centrado los volúmenes de manera correcta, aunque el resultado podría haber sido mejor. Esto se debe a que, al ejecutar el protocolo que desplaza las imágenes, el desplazamiento ha sido excesivo. De todas formas, los resultados son eficaces y es un protocolo interesante para introducir en Scipion.

## ➤ Mask

Mask es un protocolo que elimina el ruido que se encuentra alrededor de los volúmenes de las partículas.

```
import pyworkflow.em as em
import pyworkflow.protocol.params as params
from pyworkflow.em.protocol import ProtCreateMask3D
from pyworkflow.utils.path import makePath, moveFile
from pyworkflow.em.convert import ImageHandler
import simple

class ProtMaskSimple(ProtCreateMask3D):
    """
    Mask
    Is a program for creating a mask from a pixel size given
    in a volume file
    """
    _label = 'simple - mask3D'

    def _init_(self, **kwargs):
        ProtCreateMask3D._init_(self, **kwargs)

    #-----Define Params of function-----
    def _defineParams(self, form):
        form.addSection(label='Input')
        form.addParam('inputVolume', params.PointerParam, pointerClass='Volume', allowsNull=False,
            label='Input Volume', important=True, help='Import volume file to apply mask')
        form.addParam('sampling', params.FloatParam, default=2.0, label='Sampling rate in pixels', help='Sampling rate',
            important='true')
        form.addParam('mask', params.IntParam, default=0,
            label='Mask',
            help='Mask in pixels. Radio of the particle', important='true')
        form.addParam('lowLimit', params.IntParam, default=0, expertLevel=params.LEVEL_ADVANCED,
            label='Low pass limit',
            help='Low pass limit in Angstroms')
        form.addParam('molW', params.IntParam, default=0, expertLevel=params.LEVEL_ADVANCED,
            label='Molecular Weight',
            help='Molecular weight in kDa')
        form.addParallelSection(threads=4, mpi=0)
```

Figura 36. Mask: Descripción y parámetros usuario

La figura 36 corresponde a la primera parte del código. Aquí se describe el programa y se definen los parámetros que introducirá el usuario. Al igual que cleanUp2D, incluye un menú avanzado (LEVEL\_ADVANCED) con parámetros opcionales en la ejecución del protocolo.

Los parámetros obligatorios son:

- inputVolume: archivo que contiene los volúmenes sacados a partir de otros protocolos.
- Sampling: Parámetro correspondiente al muestreo de las imágenes cuyo valor viene dado en píxeles.

- Mask: radio en píxeles a partir del cual se eliminará el ruido de las imágenes.

Los parámetros opcionales son:

- LowLimit: Parámetro que indica el valor del filtrado paso bajo.
- molW: indica la masa molecular de las partículas en kDa(kilo Daltons).

```
def _insertAllSteps(self):
    self._insertFunctionStep("convertInput")
    self._insertFunctionStep("maskStep")
    self._insertFunctionStep("createOutputStep")
    pass

#-----Define methods of steps-----

def convertInput(self):
    ImageHandler().convert(self.inputVolume.get(), self._getExtraPath("refVolume.mrc"))
```

Figura 37. Mask: Funciones y conversión

```
def maskStep(self):
    partFile = self._getExtraPath("refVolume.mrc")
    partName = os.path.basename(partFile)
    partName = os.path.splitext(partName)[0]
    tmpDir = self._getTmpPath(partName)
    makePath(tmpDir)

    mask = 'ncr=mask_vole%$ msk=%d_smp=%f_nthr=%d' % (os.path.abspath(partFile), self.mask.get(), self.sampling.get(), self.numberOfThreads.get())
    if self.lowLimit.get() > 0:
        mask = mask + 'msklp=%d' % (self.lowLimit.get())
    if self.molW.get() > 0:
        mask = mask + 'mw=%d' % (self.molW.get())

    self.runJob(simple.Plugin.sim_exec(), mask, cwd=os.path.abspath(tmpDir),
               env=simple.Plugin.getEnviron())

    path = os.path.join(tmpDir, './1_mask_lowvol.mrc')
    moveFile(path, self._getExtraPath(partName + "_outvol.mrc"))
```

Figura 38. Mask: Ejecución del script mask

Estas dos figuras corresponden a la conversión del volumen de entrada al formato adecuado y a la función que ejecuta el script de mask. En la ejecución, se verifica si el usuario a agregado valores de los parámetros opcionales y, en caso de que sea así, se añaden al script. Después se ejecuta el script y se mueve lo obtenido al directorio para crear las salidas en Scipion.

```
def createOutputStep(self):  
    vol = em.Volume()  
    vol.setLocation(self._getExtraPath('refVolume_outvol.mrc'))  
    vol.setSamplingRate(self.inputVolume.get().getSamplingRate())  
    self._defineOutputs(outputVol=vol)  
    self._defineSourceRelation(self.inputVolume, vol)
```

Figura 39. Mask: Volumen de salida

Esta última parte, al igual que el programa center, crea un volumen y le asigna el directorio correspondiente para que el usuario pueda ver los resultados de la ejecución del protocolo.

## RESULTADOS

Partimos de un set de volúmenes de creado por otro protocolo de Scipion. Este corresponderá a la misma muestra que los programas cleanUp2D y center.

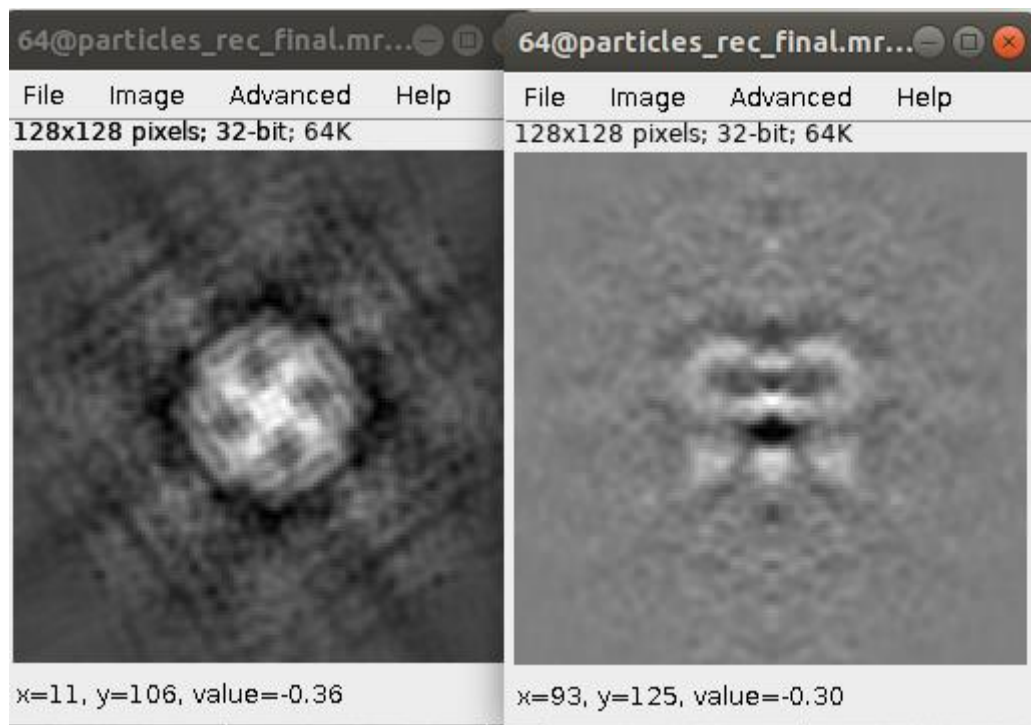


Figura 40. Mask: Volumen de entrada en ejecución-vistas Z e Y

Ahora vamos a ejecutar el protocolo de mask con los valores mostrados en la figura 41.

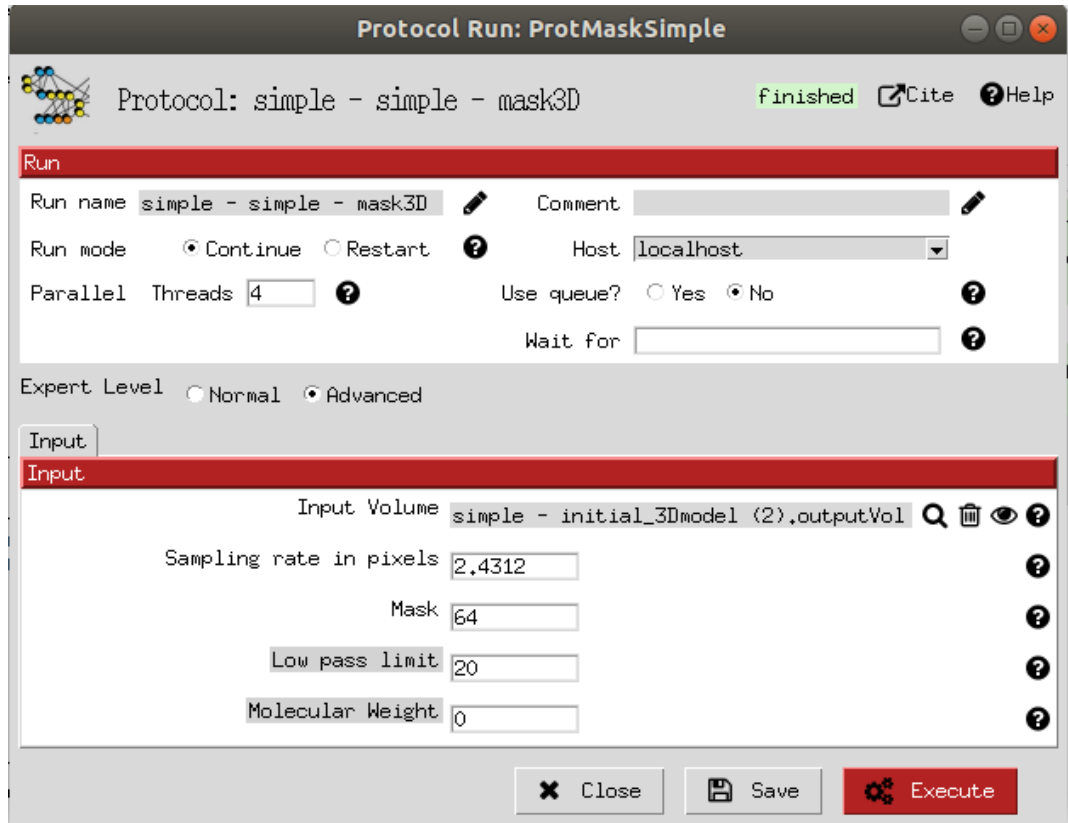


Figura 41. Mask: Parámetros de ejecución del protocolo

Obteniendo así los siguientes resultados:

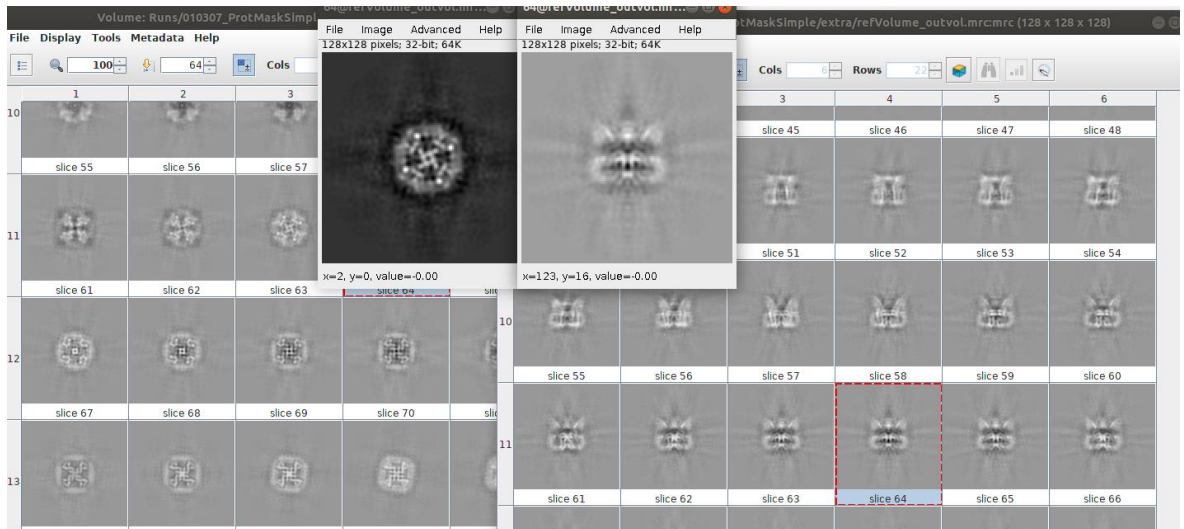


Figura 42. Mask: Resultado final-vistas Z e Y

Si comparamos las 40 y 42 vemos podemos observar como las partículas en la figura 42 son mucho mejor apreciables y la calidad es mejor debido a la máscara a la que ha sido sometida por el protocolo de mask.

➤ Refine3D:

Debido a que este programa no produce resultados buenos, no se va a enseñar el código y solo se enseñará el resultado final.

Partimos de la muestra correspondiente a la figura 24 y ejecutamos el protocolo.

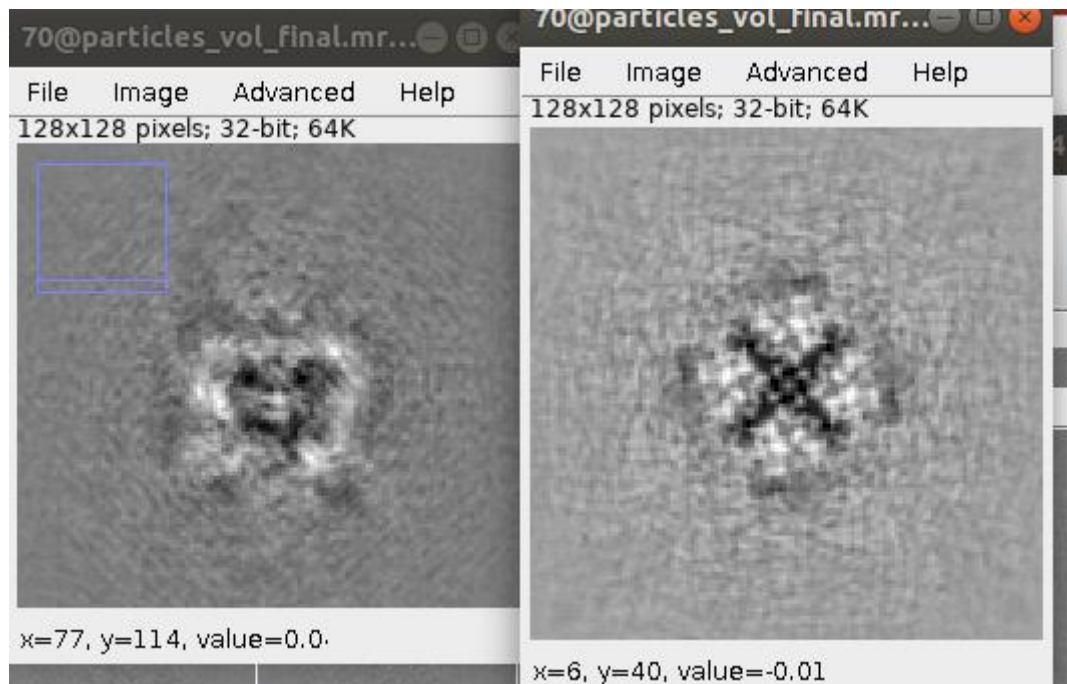


Figura 43. Refine3D: Resultado final-vistas Z e Y

Si observamos ambas vistas del volumen mostrado en la figura 43 y las comparamos con, por ejemplo, las salidas de los programas mask o center, vemos que las imágenes no tienen la misma calidad, pudiendo afirmar que no se aprecia la misma partícula. A esto se le añade el elevado tiempo de ejecución del programa. También, ejecuta varios protocolos a la vez en el wrapper. Esto último no es correcto porque la finalidad de Scipion es ejecutar cada protocolo en pasos distintos.



## Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

Durante este proyecto se ha conseguido desarrollar y ejecutar varios programas correspondientes a distintos protocolos del paquete de SIMPLE con objeto de introducirlos a Scipion y aumentar la cantidad de protocolos destinados al procesamiento de imágenes disponibles para los usuarios. En particular, se han programado cleanUp2D, center, mask, refine3D y masscen.

De todos los programas señalados, solamente cleanUp2D y center se pretenden incorporar a Scipion puesto que resultan ser los más adecuados y válidos al objetivo perseguido: cleanUP2D produce resultados similares a otro protocolo introducido recientemente en Scipion llamado cluster2D reduciendo el tiempo de ejecución y aumentando la calidad de los resultados de las imágenes que se obtienen. En el caso de center, es un protocolo nuevo que proporciona la función de los volúmenes obtenidos con otros protocolos.

Programas como Mask o refine3D no se considera conveniente su incorporación. En el caso de Mask porque ya existe un programa que realice funciones parecidas dentro de Scipion e interesaría otro programa que eliminase el ruido sin pedir al usuario que seleccione un radio (Mask elimina el ruido en función a un radio introducido por el usuario). En lo que respecta a refine3D, no se incorporará a Scipion porque exige un alto tiempo de ejecución.

Tampoco se ha considerado conveniente la inclusión de Masscen puesto que no está incluido en la versión más reciente de SIMPLE que es la que se está usando.

También se procedió a trabajar con los programas motion\_correct\_tomo y simulate\_movies de SIMPLE y con el programa goCTF. Estos programas no se consiguieron ejecutar de manera correcta en el paquete de origen, razón por la cual no se han desarrollado wrappers asociados a ellos.

Con objeto de mejorar y perfilar la calidad y resolución de las imágenes podría ser interesante el desarrollo de trabajos adicionales.

Así y en un futuro no muy lejano, sería apropiado incorporar un programa como Mask con algunas mejoras como por ejemplo la aplicación de una máscara de manera automática de modo que no la tenga que seleccionar el usuario. También, resultaría adecuado mejorar el código de refine3D para que se ejecute de manera que coincida con las políticas de Scipion.

Finalmente, quedaría completar el código de goCTF e incorporarlo a Scipion. Se estableció contacto con los desarrolladores de este programa para que nos enviaran un ejemplo de una ejecución del protocolo, pero no se obtuvo respuesta.

## Capítulo 8. BIBLIOGRAFÍA

- [1] Universidad Nacional del Nordeste (1998-2013). “Microscopía electrónica: Definición y tipos”. Sitio Web: <http://www.biologia.edu.ar/microscopia/meb.htm>.
- [2] The Royal Society of Chemistry (2019). “What is cryo-electron microscopy | News | Chemistry World”. Sitio Web: <https://www.chemistryworld.com/news/explainer-what-is-cryo-electron-microscopy/3008091.article>.
- [3] Anónimo (2019). “Scipión: Descarga, instalación, definición y tutoriales”. Sitio Web: <https://github.com/I2PC/Scipion/wiki>.
- [4] Sánchez Sorzano, C.O. [Science and Technology Facilities Council]. (23/05/2018). CCP-EM Carlos Óscar Sánchez Sorzano | Spring Symposium 2018. Website: <https://www.youtube.com/watch?v=wsG1qSoXNI0&t=308s>.
- [5] Anónimo (2019). “Scipión: Descarga, instalación, definición y tutoriales”. Sitio Web: <https://github.com/I2PC/Scipion/wiki>.
- [6] Elmlund Lab (24/01/2017). “Manual desactualizado de SIMPLE 3.0”. Sitio Web: <https://simplecryoem.com/Oxford/simple3.0manual.pdf>.
- [7] Universidad Nacional del Nordeste (1998-2013). “Microscopía electrónica: Definición y tipos”. Sitio Web: <http://www.biologia.edu.ar/microscopia/meb.htm>.