UNIVERSIDAD SAN PABLO - CEU

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN E INGENIERÍA INFORMÁTICA

PROYECTO FINAL DE CARRERA

Integración de diferentes paquetes de Tomografía Electrónica

Autor: Javier Sánchez Jiménez

Director: Carlos Óscar Sánchez Sorzano

Julio 2012

UNIVERSIDAD SAN PABLO-CEU

ESCUELA POLITÉCNICA SUPERIOR

División de Ingeniería Informática y de Telecomunicación

Calificación del Proyecto Fin de Carrera

Datos del alumno

NOMBRE:

Datos del Proyecto

TÍTULO DEL PROYECTO:

Tribunal calificador

PRESIDENTE:	FDO.:

SECRETARIO:	FDO.:

VOCAL:	FDO.:

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

RESUMEN

La biología estructural, cuyo objetivo es la determinación de la estructura tridimensional de los complejos macromoleculares y orgánulos celulares sometidos a estudio, tiene un gran interés debido a que la función que realiza una proteína en el interior de una célula está muy ligada a la estructura tridimensional que adquiere en el espacio, por lo que gracias a la reconstrucción de estas estructuras es posible el estudio y la investigación de nuevos fármacos basándose en las funciones que realizan estas proteínas en el organismo. El análisis de los orgánulos celulares proporciona el contexto adecuado en el que se desenvuelven estas proteínas.

En este contexto, la tomografía electrónica de transmisión (TEM) y la tomografía de Rayos-X (TomoX) son técnicas que permiten, a partir de un conjunto de pasos, obtener una tomografía a partir de una serie de proyecciones tomadas a partir de un microscopio electrónico en el caso de TEM ó a la exposición de la muestra a Rayos-X en el caso de TomoX.

Existes diversos programas y paquetes software cuyo objetivo es el de obtener las reconstrucciones tridimensionales de orgánulos celulares a partir de una serie de proyecciones bidimensionales. Durante este proyecto, el alumno se ha integrado en el equipo de desarrollo de Xmipp (CNB-CSIC) para colaborar activamente en la implementación del software TomoJ, desarrollado por los miembros del Instituto Curie (París) y que permite, desde una única interfaz realizar todos los pasos que se realizan en tomografía para llevar a cabo la reconstrucción tridimensional de la muestra, con el objetivo de integrar funcionalidad del paquete Xmipp en este software para mejorar su rendimiento y prestaciones.

Concretamente, el alumno ha integrado en el programa TomoJ la capacidad de trabajar con series de proyecciones procedentes de Tomo-X, algoritmos de preprocesado de imágenes como filtros gaussiano y mediano o corrección Gamma entre otros, así como el algoritmo de reconstrucción Tomo3D, utilizado por Xmipp, y que obtiene buenos resultados en un tiempo computacional considerablemente inferior a los obtenidos a través de algoritmos de reconstrucción convencionales.

ABSTRACT

The structural biology, aimed at determining the three dimensional structure of macromolecular complexes and cellular organelles under study, is of great interest because the function that performs a protein inside a cell is closely related to the acquired three-dimensional structure space, so thanks to the reconstruction of these structures is possible to study and research of new drugs based on the functions performed by these proteins in cells. The analysis of cellular organelles provides the proper context in which they operate these proteins.

In this context, transmission electron tomography (TEM) and X-ray tomography (TomoX) are techniques that enable, from a set of steps, to obtain a tomography from a tilt-series taken from a electron microscope in the case of TEM or exposure of the sample to X-rays in the case of TomoX.

Some software programs and packages aimed to obtain three-dimensional reconstructions of cellular organelles from a series of two-dimensional projections. During this project, the student has been integrated into the development team of Xmipp (CNB-CSIC) to collaborate actively in the implementation of TomoJ, software developed by the members of the Institut Curie (Paris) and allows, from a single interface to perform all steps that take place in tomography to perform the three-dimensional reconstruction of the sample, with the aim of integrating the package Xmipp functionality in this software to improve its performance and benefits.

Specifically, the student has integrated into the program TomoJ the ability to work with tilt-series from Tomo-X, image pre-processing algorithms such as Gaussian and Medium filters and Gamma correction among others, as well as the reconstruction algorithm Tomo3D used by Xmipp, with which we obtained good results in a computational time significantly lower than those obtained through conventional reconstruction algorithms.

Agradecimientos

Este trabajo no podría haber sido realizado sin la colaboración de muchas personas que, desde el día en el que comencé mis estudios, no han dejado de apoyarme en todo momento. Quiero agradecerles todo lo que han hecho por mi durante todo este tiempo sin esperar recibir nada a cambio más que estas palabras de agradecimiento.

Gracias a mi director de proyecto, el Dr. Carlos Óscar Sánchez Sorzano, profesor en la Universidad San Pablo CEU, que me acogió en su centro de trabajo donde, durante un año, me ha formado y preparado en cierto modo para el mundo laboral, enseñándome los valores del trabajo en equipo. A su equipo, investigadores del CNB-CSIC quedo especialmente agradecido por solucionarme cualquier problema cuando era necesario, en especial quería mencionar a los investigadores Jesús Cuenca, Joaquín Otón y José Miguel De La Rosa por su dedicación, esfuerzo y apoyo durante todo este tiempo.

Gracias a mis profesores de la Universidad San Pablo CEU, que, durante toda mi estancia en la universidad siempre mantuvieron su puerta del despacho abierta para cualquier necesidad, gracias por enseñarme a ingenieros, y, sobre todo, gracias por enseñarnos a ser personas.

Gracias a mis compañeros, con los cuales he compartido largas horas de estudio durante toda mi estancia en la universidad, sin vuestro apoyo este momento nunca hubiera llegado.

Finalmente me gustaría agradecer profundamente a toda mi familia todo el apoyo y esfuerzo que me han brindado desde el día que nací. Ellos son los verdaderos artífices de todo este trabajo. Gracias de todo corazón.

Soy muy afortunado.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

1.1 Introducción a la biología estructural

En la actualidad, la biología molecular permite el estudio y el conocimiento de los componentes elementales en el desarrollo de la vida celular. Su objetivo es la caracterización de los complejos moleculares (proteínas, DNA, RNA, así como sus complejos macromoleculares) y su interrelación.

En el amplio campo de la biología molecular se sitúa la biología estructural, cuyo objetivo es la determinación de la estructura tridimensional de los complejos macromoleculares sometidos a estudio. Este campo tiene un gran interés debido a que la función que realiza una proteína en el interior de una célula está muy ligada a la estructura tridimensional que adquiere en el espacio, sin desdeñar por ello otros factores claramente importantes como pueda ser su composición bioquímica. Por lo tanto el conocimiento de la estructura de las proteínas y células es necesario, por ejemplo, para el diseño de medicamentos cuyos objetivos son proteínas específicas

La unidad de medida que determina la calidad de la reconstrucción tridimensional es la resolución, que mide el nivel de detalle que contiene la estructura, más concretamente la menor distancia que separa dos puntos permitiendo distinguirlos como separado. Una alta resolución permite la posibilidad de interpretar la estructura con un alto nivel de detalle, y esto es imprescindible para obtener su función.

Para obtener la conformación estructural que presentan las macromoléculas en el espacio se siguen diferentes aproximaciones, la primera de ellas consiste en predecir su estructura en base a una información a priori de los datos bioquímicos que poseen tales como su secuencia de aminoácidos, función o familia por medio de la resolución de ecuaciones complejas que definen la interacción de los diferentes átomos que componen la macromolécula. La segunda de ellas, permite la reconstrucción de la molécula utilizando para ello distintas técnicas biofísicas como son la cristalografía de rayos X, la espectroscopia por resonancia magnética nuclear y la microscopía electrónica de transmisión. Cada una de estas técnicas tienes sus limitaciones, y, por

tanto, su rango de aplicación: los rayos X permiten la obtención de información estructural a muy alta resolución con la condición de que la molécula de estudio pueda ser cristalizada, por lo que su rango de aplicación está limitado a macromoléculas con un peso molecular bajo (únicamente con moléculas con peso molecular inferior a 30kDa. La resonancia magnética nuclear también permite la obtención de datos a alta resolución, pero con unas limitaciones de tamaño mucho mayores que en rayos X. A diferencia de estos, la microscopía electrónica de transmisión no presenta limitaciones en cuanto al tipo de espécimen a tratar, se necesita menor cantidad de muestra y no requiere un estado cristalino de la muestra. Sin embargo, tiene el inconveniente de que la resolución alcanzada con esta técnica es menor que en los otros casos. Este impedimento se puede paliar, no obstante, a través de potentes algoritmos de procesamiento de imagen y de teoría de la señal. Es en este campo donde los ingenieros informáticos y de telecomunicaciones juegan un papel crucial en la biología estructural y, por tanto, éste será el tema de estudio a lo largo de este proyecto.

1.2 Introducción a la microscopía electrónica de transmisión

Como ya se ha introducido en la sección anterior, la microscopía electrónica de trasmisión es una técnica estructural de baja-media resolución que cada vez se perfila más como una herramienta poderosa para realizar estudios estructurales a media-alta resolución. Su funcionamiento es similar al de la microscopía óptica salvo por el hecho de que el haz de luz incidente no se compone por fotones de una determinada longitud de onda sino que son electrones acelerados por una considerable diferencia de potencial, la energía cinética adquirida es la que les confiere su poder resolutivo al llevar asociadas longitudes de onda muy inferiores a las de la luz visible ultravioleta.

Otro factor limitante de la resolución en la reconstrucción tridimensional de partículas aisladas es la variabilidad biológica. Para realizar tal reconstrucción se combinan imágenes de proyección procedentes de multitud de distintas macromoléculas, supuestas todas iguales en forma. No obstante, esto no es sino una aproximación puesto que diferencias en el agente de tinción o de su entorno pueden

modificar ligeramente la estructura flexible de la proteína en estudio. Además debemos mencionar la acción destructiva que la radiación electrónica ejerce sobre la muestra biológica. Dicha destrucción impide la visualización de detalles estructurales de alta resolución, por lo que para minimizar su impacto y obtener imágenes de alta calidad se disminuye la dosis de radiación empleada, sin embargo estas micrografías con menor dosis también presentan un menor contraste.

Las micrografías obtenidas por microscopía electrónica se caracterizan por un bajo contraste de la señal frente al ruido, presentando una relación señal a ruido, SNR, por debajo de 1/2 y por una resolución muy por debajo de la teórica impuesta por las limitaciones del microscopio.

La tomografía electrónica permite reconstrucciones tridimensionales de un objeto simple una vez realizada la adquisición de la serie de inclinación (*tilt series*), correspondiente al conjunto de proyecciones 2D de este objeto simple, realizando rotaciones sobre la muestra a lo largo del eje de inclinación. A diferencia de éstas, cuando la muestra está compuesta de múltiples copias de un mismo objeto, como es el caso de proteínas y complejos macromoleculares, la estrategia de la colección de datos y la reconstrucción tridimensional es diferente y necesita del cálculo de un mapa de promedios 3D. De nuevo, en este contexto pueden aparecer diferentes aproximaciones, dependiendo de la simetría interna de las partículas y la organización regular de éstas.

Como ya se ha ido mencionando a lo largo de este apartado, la microscopía electrónica presenta una serie de limitaciones que dificultan el hecho de poder obtener una alta resolución de reconstrucciones tridimensionales, impedimentos, sin embargo, que en su mayoría pueden solucionarse aplicando poderosos algoritmos de procesado de imágenes. Ejemplos de estos problemas son los daños producidos por radiación sobre la muestra, heterogeneidad estructural de la muestra, ruido aditivo debido al microscopio o aberraciones producidas por el microscopio.

1.3 Introducción a la reconstrucción tridimensional

Durante el desarrollo de este capítulo veremos de forma general los pasos necesarios para realizar una reconstrucción tridimensional a partir de las muestras de laboratorio. Desde el momento en el que se digitaliza la muestra para poder ser interpretada por un ordenador hasta que se aplican los algoritmos de reconstrucción. Son muchos y diferentes los algoritmos, métodos y herramientas los que se utilizan para llevar a cabo cada uno de los pasos que se van a citar a continuación, sin embargo, para llevar a cabo una reconstrucción tomográfica con EMT es necesario realizar todos estos pasos.

Como veremos, algunos de estos pasos están fuera del alcance del proyecto, por lo que a lo largo del proyecto nos centraremos únicamente en aquellos implementados en el software TomoJ. Sin embargo, conviene comentarlos para no perder una visión general de cómo se realiza el proceso de reconstrucción.

La técnica matemática conocida que resuelve la estructura tridimensional de la macromolécula que generó proyecciones con las que trabajamos en TEM es conocida como tomografía, y existen muchas aproximaciones a la solución de este problema así como diferentes algoritmos de reconstrucción que estudiaremos a lo largo de este proyecto.

Sin embargo, son muchas las etapas por las que pasa un conjunto de micrografías antes de producir una estructura tridimensional, que son comunes a todas las técnicas y algoritmos que permiten realizar la reconstrucción tridimensional. Un breve repaso del estado actual de las mismas contextualizará convenientemente este proyecto.

1.3.1 Adquisición de datos

La colección de datos a partir de la cual se realizará la reconstrucción tomográfica, consistirá en una serie de proyecciones bidimensionales de la misma muestra que han sido tomadas rotadas gradualmente sobre un eje perpendicular al haz de electrones. Es importante recalcar además que antes de realizar dichas proyecciones

la muestra debe ser protegida frente a la radiación generada por el microscopio, en otro caso, la interacción entre la muestra y el haz de electrones dañará su estructura, para ello se utilizan métodos como *Cryoelectron microscopy (cryo-EM)*, técnica con la que no nos extenderemos por no formar parte de los objetivos del proyecto.

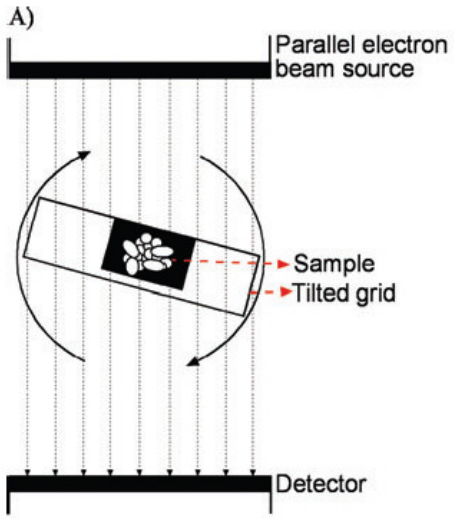


Fig. 1.1: Principios de la adquisición de datos en microscopía electrónica de transmisión.

El haz de electrones (electron beam) y el detector son estacionarios, mientras que la muestra es rotada con un ángulo de inclinación constante que rota sobre un eje perpendicular al haz de electrones, tal y como podemos observar en la figura [1.1].

Para hallar la serie de proyecciones bidimensionales hemos de partir de la suposición de que el espécimen estudiado es relativamente fino con respecto a la profundidad de foco del microscopio, *Weak Phase Object*, y, de esta forma se puede demostrar que la imagen adquirida es una imagen de proyección. Un ejemplo cotidiano de imágenes de proyección son las obtenidas por radiografía convencional.

Suponiendo que la micrografía se encuentra en soporte fotográfico, el primer paso hacia una reconstrucción tridimensional consiste en la digitalización correcta de la misma de forma que pueda ser tratada por un ordenador.

Posteriormente se procede a la selección y extracción de aquellas regiones dentro de la micrografía que efectivamente corresponden a proyecciones de la proteína

estudiada, obteniendo de esta forma las proyecciones bidimensionales a partir de las cuales trabajaremos en el resto del ciclo de reconstrucción.

1.3.2 Normalización

A continuación se realiza una normalización de las imágenes de proyección con el objetivo de homogeneizar los niveles de densidad de gris de imágenes procedentes de distintas micrografías.

El conjunto de proyecciones utilizado para la reconstrucción de un espécimen puede llegar a tener varios miles de partículas provenientes de diferentes micrografías, las cuales han sufrido distintas condiciones de exposición en el microscopio, revelado y digitalización. El efecto de estas diferencias es observable en ligeras diferencias entre el contraste de las micrografías. Incluso dentro de una misma micrografía se pueden observar variaciones importantes de contraste.

Existen diferentes tipos de normalización que se realizan en tomografía electrónica tridimensional, este tipo de normalización se escoge dependiendo de la variabilidad del conjunto, donde las más comunes son estas dos:

- *Zero-One*, que aplica una media de cero y desviación típica de uno sobre todas las proyecciones de la serie.
- *Electron tomography*, que aplica una media de cero y una desviación típica que varía en función del coseno del ángulo de inclinación.

1.3.3 Pre-procesado

La primera etapa de procesado se realiza en 2D tratando de mejorar la calidad de las imágenes obtenidas a partir de la mejora del factor SNR (relación señal a ruido) de la serie de proyecciones. En primer lugar, basándose en el hecho de que la transformada de Fourier de una señal periódica es nula salvo en un conjunto de ∂'s (llamados *spots* o reflexiones) podemos realizar un fuerte filtrado del ruido.

Este ruido es causado principalmente por un muestreo insuficiente de de la dispersión de electrones desde la muestra y por las imperfecciones en el detector.

Además, en muchos casos es conveniente enventanar las imágenes para eliminar aquellas zonas en las que, con seguridad, no hay proyección de la proteína. De este modo se elimina parte del ruido.

Para eliminar este ruido, los diferentes programas y paquetes software que permiten realizar reconstrucciones tomográficas permiten realizar diferentes tipos de filtrado con el fin de mejorar la relación señal a ruido. Estos filtros los estudiaremos detenidamente en los capítulos [2.3.2] y [2.5].

1.3.4 Alineamiento

Durante el desarrollo de este paso, el objeto es centrado y alineado rotacionalmente con respecto al eje principal y la dirección de proyección de cada imagen es determinada para toda la serie.

Existen dos técnicas diferenciables a la hora de determinar la posición y orientación de las proyecciones bidimensionales en el espacio tridimensional: El alineamiento con referencia (*reference-based)* y el alineamiento sin referencia (*reference-free).* Los algoritmos *reference-free* son usados para determinar los parámetros de alineamiento para obtener un primer modelo tridimensional de baja resolución. Ejemplos de estos algoritmos son *el método de los momentos* (Gelfand y Goncharov 1990) y *random conical tilt series* (Radermacher 1987).

Por su parte, los algoritmos *reference-based* requieren un modelo tridimensional de referencia para perfeccionar de manera iterativa los parámetros de alineamiento y para obtener un modelo tridimensional de alta resolución.

En tomografía, todas las proyecciones de una serie comparten una línea común correspondiente al eje de inclinación, por lo tanto, la determinación de este eje es esencial para llevar a cabo la reconstrucción. Idealmente, la muestra no se ha movido durante la adquisición de datos y, por lo tanto, la orientación de los ángulos de

inclinación pueden ser obtenida como una línea común presente en todos los espectros de potencia de la serie. Sin embargo, en la práctica este caso ideal no se da, por lo que se tiene que realizar un alineamiento traslacional y rotacional de las proyecciones para determinar la orientación del eje de inclinación debido a las desviaciones que se producen durante el proceso de adquisición de datos.

El proceso de alineamiento suele realizarse a través del alineamiento de pequeñas partículas de oro que se encuentran en la muestra, que sirven como puntos de referencia bien definidos (fiducial markers). Sin embargo, en algunas ocasiones estas marcas no son visibles o, simplemente no se encuentran. De forma alternativa, se pueden usar como puntos de referencia las esquinas de las imágenes o, incluso, el contenido de la información de la imagen se puede utilizar a través de la correlación cruzada.

Ha sido demostrado que la información de la imagen es la información más valiosa en el registro de la imagen, por lo tanto, los algoritmos que utilizan la correlación cruzada con un volumen de referencia o técnicas similares son los que proporcionan los mejores resultados y, por ello constituyen el alineamiento estándar en microscopía electrónica de partículas individuales. Sin embargo, cuando se realiza el alineamiento de una serie de proyecciones no existe un volumen previo que pueda ser utilizado como referencia y, por esta razón, el algoritmo empleado debe de realizar el alineamiento con información exclusiva de la propia serie. Existen tres posibilidades diferentes para utilizar la correlación cruzada en este contexto.

La primera de las alternativas es la de alinear secuencialmente la primera imagen con la segunda, la segunda con la tercera y así sucesivamente. Esta aproximación compara imágenes similares, y tiene el gran inconveniente de que puede producir grandes derivas (causadas por la propagación de errores) en los parámetros de alineamiento dando lugar a estimaciones inexactas del eje de inclinación.

La segunda alternativa consiste en utilizar la correlación cruzada para realizar un seguimiento de las características de las diferentes proyecciones. Estas características se definen como puntos de referencia en las imágenes (como puedan ser cambios de

contraste, bordes o esquinas), que son buscados realizando la correlación cruzada en proyecciones vecinas. Esta función de seguimiento permite definir los marcadores que son visibles en una sóla subsecuencia de la serie.

Por último, la tercera alternativa construye una reconstrucción aproximada de la tomografía y realinea la serie de proyecciones con respecto a la reconstrucción aproximada.

El proceso de alineamiento no siempre proporciona una serie totalmente alineada, por lo que hay que realizar comprobaciones antes de pasar al proceso de reconstrucción, ya que pueden existir variaciones de alrededor de cinco grados entre proyecciones.

Para llevar a cabo el alineamiento, cada programa o paquete software de reconstrucción implementa sus propios algoritmos, permitiendo al usuario realizar el alineamiento de la serie de forma automática o manual. En el capítulo [2.3] veremos cómo se realiza este alineamiento en el programa TomoJ y qué alternativas propone.

1.3.5 Asignación angular

La asignación angular es una pieza clave en el proceso de reconstrucción, ya que permite fijar el ángulo de inclinación desde el que se ha tomado cada proyección. En ciertas ocasiones, estos valores son conocidos por el usuario, por lo que únicamente tendrá que fijarlos en el programa de reconstrucción que esté utilizando, ó puede que, en algunos casos, los propios ficheros de datos ya contengan esta información, tan y como ocurre en los ficheros SEL. Este proceso se realiza habitualmente indicando el ángulo de inclinación de la primera proyección y el índice de salto angular que se va a tomar entre dos proyecciones consecutivas.

Si los ángulos de inclinación son desconocidos, pueden ser obtenido por medio de dos procedimientos diferentes. El más sencillo, basado en la transformada de Fourier de la serie de proyecciones requieren unos puntos de referencia bien definidos (*fiducial markers)* como pueden ser partículas de oro. El ángulo de inclinación es definido por la

línea vertical que cruza la densidad espectral de potencia que se muestra en la pantalla y la línea de inclinación que aparece sobre ella.

El segundo procedimiento utiliza las coordenadas de un conjunto de puntos seleccionados de los puntos de referencia. Estas coordenadas se utilizan para obtener por medio del método de regresión lineal la ecuación de la recta perpendicular al eje de inclinación. Esta recta representa la dirección media de los desplazamientos. De esta forma, el eje de inclinación se obtiene como la recta perpendicular a la recta de desplazamiento promedio.

1.3.6 Reconstrucción

El último paso de este proceso es el de realizar la reconstrucción tridimensional de la serie. Para ello, la serie debe de estar bien alineada y los ángulos de inclinación bien fijados, sólo así obtendremos una reconstrucción tridimensional que se corresponda con la serie de proyección.

Existen multitud de algoritmos de reconstrucción tomográfica aunque la mayoría de ellos se pueden clasificar en dos grandes categorías: métodos transformados y métodos basados en una expansión en serie del volumen a reconstruir. Los primeros resuelven el problema en base a invertir o pseudo-invertir una estimación de la expresión de dicho volumen en algún dominio transformado, típicamente el espacio de Fourier. Para ello, se encargan de estimar convenientemente la expresión transformada del volumen solución y de realizar la transformada inversa correspondiente, teniendo en cuenta que tan sólo se pueden proporcionar soluciones aproximadas para evitar las inestabilidades numéricas de la solución exacta asociadas al hecho del posible desconocimiento de alguna región del mencionado espacio transformado.

Los algoritmos del segundo grupo aproximan el volumen solución por una serie de funciones base desplazadas en el espacio y correctamente ponderadas, por lo que su cometido es el de tratar de encontrar los pesos adecuados para cada una de las funciones base atendiendo a la minimización de algún funcional.

Entre los algoritmos del primer grupo destacan la inversión directa de la transformada de Fourier y el método de retroproyección ponderada (*WBP, Weighted BackProjection).* Estos algoritmos se basan en el teorema central del límite, según el cual, cada proyección obtenida en el microscopio constituye una medida de la transformada de Fourier del volumen a reconstruir, por ello, si colocamos apropiadamente dicha estimación en el espacio de Fourier tridimensional y hacemos la transformada inversa de Fourier obtendremos una estimación del volumen que constituye la estructura tridimensional de las proyecciones obtenidas.

Existen multitud de variantes a estos dos métodos que buscan cuál es la mejor forma de interpolar aquellos valores del espacio de Fourier tridimensional que no han sido medidos por ninguna proyección, otros trabajan con transformadas de Fourier en coordenadas cilíndricas debido a que en dicha geometría pueden expresarse mejor ciertas características de los datos.

WBP (Weighted BackProjection) (Rademarcher 1992) es el algoritmo de reconstrucción más utilizado entre los programas y paquetes software que permiten realizar reconstrucciones tomográficas, ya que es el algoritmo que menos tiempo computacional requiere. Este algoritmo compensa las bajas frecuencias en el dominio de Fourier multiplicando la Transformada de Fourier de cada proyección bidimensional por una función ponderada a partir de los pesos obtenidos y sumándoles la Transformada de Fourier inversa de las proyecciones ponderadas.

En el segundo grupo de algoritmos podemos encontrar la mayoría de los algoritmos de reconstrucción iterativos. Su principal característica es que trabajan en el espacio real y que realizan una aproximación gradual de la solución, por contra, este acercamiento gradual supone un mayor coste en tiempo aunque la existencia de algoritmos que dan buenos resultados con una única iteración como ART, (*Algebraic Reconstruction Technique*sis) supone una importante reducción del tiempo de cómputo acercando los algoritmos iterativos al nivel de los métodos transformados en cuanto a tiempo de ejecución se refiere. Los algoritmos basados en expansión del volumen solución permiten la inclusión de restricciones espaciales de manera más sencilla que los métodos transformados.

Habitualmente estos algoritmos de expansión minimizan una función en el que participan la discrepancia entre las proyecciones experimentales obtenidas y las proyecciones obtenidas a partir del volumen solución en esa misma dirección. Sin embargo, se pueden añadir términos adicionales a dicho funcional que minimicen algún aspecto que se considere importante como puede ser la varianza de la solución ó términos que realzan los bordes basándose en gradientes. En esta línea encontramos métodos como ART y SIRT (*Simultaneous Iterative Reconstruction Technique)*, los cuales, al igual que WBP, son los más utilizados entre los programas de tomografía.

1.3.7 Post-procesamiento

Es posible realizar mejoras sobre una reconstrucción ya realizada, basadas básicamente en el refinamiento del alineamiento de la estructura. Existen diversos algoritmos que permiten refinar la orientación y el posicionamiento de imágenes de partículas con respecto a un volumen de referencia.

Existen dos tipos de estos algoritmos, los cuales estiman los parámetros de alineamiento de una imagen experimental minimizando la diferencia entre ésta y las proyecciones bidimensionales del volumen de referencia.

El primer tipo de algoritmos muestrea el espacio de los ángulos de prueba y el espacio de traslaciones de prueba uniformemente comparando la imagen experimental únicamente con las proyecciones de referencia que fueron calculadas utilizando un conjunto de referencia. En cada iteración, se obtiene una reconstrucción tridimensional a partir de imágenes experimentales usando los parámetros de alineamiento actuales y este volumen es tomado como referencia en la siguiente iteración. El muestreo realizado para obtener el cálculo de las proyecciones de referencia se reduce gradualmente durante el proceso iterativo para mejorar la precisión de alineamiento.

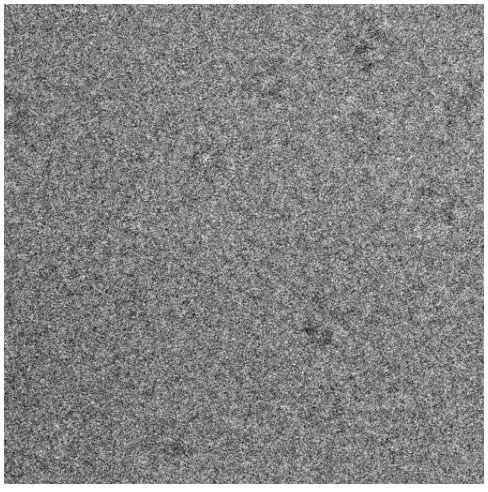
El segundo tipo de algoritmos de refinamiento no requieren ningún tipo de parámetro para realizar el proceso de muestreo. Las proyecciones de referencia para realizar la comparación con imágenes experimentales son calculadas a partir de parámetros de orientación y posición determinados por algoritmos de optimización

basados en gradientes. Estos métodos obtienen los parámetros de alineamiento en el espacio de Fourier utilizando el teorema central del límite. Esto hace que estos algoritmos sean más rápidos que los anteriores pero requieren unos esquemas de interpolación más complicados que los primeros, por lo que consumen una gran cantidad de memoria.

Habitualmente, el refinamiento se realiza utilizando el primer tipo de algoritmos hasta que este se acerca a la solución. Después, para realizar un refinamiento más rápido, una vez que el proceso se acerca a la solución, se continua el refinamiento con el segundo tipo de algoritmos.

Como se puede comprobar, las técnicas de teoría de señal, procesamiento de imágenes y las técnicas matemáticas relacionadas juegan un papel fundamental en el proceso ya que posibilitan una máxima extracción de información en un entorno nada favorable como son las imágenes de proyección obtenidas en el microscopio electrónico de transmisión con todas sus restricciones de muy baja relación señal a ruido, SNR, y ancho de banda frecuencial limitado.

En resumen, podemos definir la etapa de reconstrucción tridimensional como aquella que pasa de las micrografías como la mostrada en la figura [1.2] a volúmenes como el mostrado en isosuperficie en la figura [1.3]



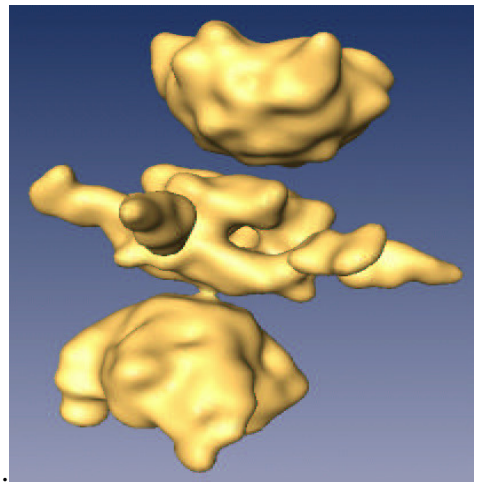


Figura 1.2: Criomicrografía de ejemplo Figura 1.3:Reconstrucción tridimensional de ejemplo.

1.4 Introducción la microscopía de Rayos-X

La técnica de microscopía de Rayos-X permite visualizar células en condiciones criogénicas con una alta resolución de entre 50 y 15nm. Este rango de resolución la hace situarse entre las microscopías confocal y electrónica permitiendo apreciar con mayor facilidad texturas y objetos tridimensionales que hayan sido pulverizados metálicamente antes de su observación. Para ello, idealmente, esta técnica reconstruye con coeficientes de absorción de las proyecciones de la muestra, sin embargo, las micrografías de rayos X son sólo una aproximación de las proyecciones del espécimen, lo que lleva a cometer imprecisiones en el proceso de reconstrucción no se tienen en cuenta la incorporación explícita de estas aproximaciones.

A diferencia de la microscopía electrónica de transmisión, las imágenes tomadas a partir de TomoX presentan un mayor contraste y un menor ruido, además, la geometría de la colección de datos (generalmente de un sólo eje de inclinación) ayuda a reducir el espacio de soluciones posibles. Por contra, las imágenes obtenidas a partir de TomoX son una aproximación más pobre a las proyecciones ideales de la muestra que las ofrecidas por la microscopía electrónica de transmisión, por lo que en este campo, el reto no es el de mejorar la relación señal-ruido de la serie de proyecciones, si no la caracterización del microscopio PSF y su correspondiente incorporación a los métodos de reconstrucción.

1.5 Objetivos

Una vez realizada una introducción acerca de la biología estructural y más concretamente de cómo realizar reconstrucciones tridimensionales a través de la microscopía electrónica de transmisión vamos a fijar los objetivos que persigue este proyecto dentro de este contexto.

La idea inicial del proyecto era la de colaborar en el desarrollo de un nuevo software elaborado íntegramente a partir de las herramientas de Xmipp que permitiese realizar todos los pasos necesarios para realizar reconstrucciones tomográficas desde una única interfaz de usuario sencilla. Este último factor, el de la sencillez era la pieza clave en la que se basaba el proyecto, ya que desde hace años han sido implementados

algunos programas similares que no han triunfado debido a la complejidad de sus interfaces.

La implementación de este software ya había sido empezada por los miembros de Xmipp, tomando por nombre Xmipp_TomoJ, programa que se estaba desarrollando en lenguaje de programación Java, utilizando el framework de programación JNI (Java Native Interface) para realizar llamadas al código de Xmipp implementado en su mayoría con el lenguaje C++ y que utilizaba un patrón de diseño VMC (vista-modelo-controlador) para separar la interfaz de usuario con los datos del programa.

La idea de este software era la de competir con el programa *TomoJ*, implementado por los miembros del Instituto Curie y que se encontraba prácticamente finalizado. Sin embargo, ambas partes acordaron la elaboración de un software común que reuniese toda la funcionalidad implementada ya en TomoJ, todas las herramientas, métodos y algoritmos de Xmipp que pudieran complementarlo, así como todas las ideas acerca de esta interfaz que se estaban poniendo en práctica en el proyecto *Xmipp_TomoJ*.

De esta forma, a través de este proyecto, el alumno colaborará en la integración de código entre TomoJ y Xmipp en los puntos en los que se ha considerado que la calidad que ofrece Xmipp en el punto en cuestión es superior a la que ofrece el programa TomoJ. Durante el desarrollo de este proyecto trataremos cada uno de estos puntos, por qué se ha considerado que es interesante integrarlos en TomoJ, qué ofrecen al usuario y qué resultados ofrecen.

A continuación se citan las funcionalidades que se van a añadir a este software.

- Capacidad de preprocesar la serie de proyecciones desde la misma interfaz de usuario. Este preprocesado se puede realizar también a través de la herramienta ImageJ, pero consideramos que será más cómodo para el usuario realizar estos pasos desde una única interfaz. Las técnicas de preprocesado que serán implementadas en TomoJ por parte del alumno son las siguientes: herramienta de recortado de imágenes *crop*, filtros mediano, gaussiano y paso banda, corrección

Gamma y substract background. Cada una de ellas explicada en el capítulo [2.5.1]

- Integración del programa Tomo3D, con el cual se puedrán realizar reconstrucciones tridimensionales a partir de una serie bien alineada, permitiendo realizar dicha reconstrucción con diferentes algoritmos. El funcionamiento de este programa se verá más detalladamente en los capítulos [2.3] y [2.5.2].

- Integración de la capacidad de cargar, preprocesar, alinear y reconstruir series de proyecciones en formato SEL, formato con el que trabaja Xmipp.

- Integración de funcionalidad capaz de cargar imágenes de rayos X así como de realizar sus reconstrucciones tridimensionales. [2.5.3]

2 Integración de software

2.1 Xmipp

2.1.1 Introducción a Xmipp

X-Window based microscopy processing package (Xmipp) es un conjunto de programas especializados en el procesamiento de imágenes que permite obtener reconstrucciones tridimensionales de espécimenes biológicos a partir de un conjunto de proyecciones adquiridas a través de microscopía electrónica por transmisión (TEM) [1.1]. Al estar soportado por un equipo muy activo de trabajo, éste está continuamente sometido a cambios y se incorporan por tanto con mucha frecuencia nuevas metodologías para el análisis de proyecciones de partículas simples, asignación angular, reconstrucciones, etc.

Su implementación en C++ con un diseño jerárquico de estructuras y funciones bien documentados, ofrece un gran ambiente de trabajo para el desarrollo de nuevos algoritmos.

Xmipp está orientado al completo procesamiento de partículas aisladas por microscopía electrónica, desde su adquisición a su reconstrucción tridimensional. Proporciona un conjunto de programas para realizar todos los pasos necesarios en el workflow del procesamiento de imágenes.

Además de Xmipp existen otros paquetes de procesado de imágenes en el ámbito de EM, como por ejemplo Spider (Frank et al., 1996), Imagic (van Heel et al., 1996), o Eman (Ludtke et al., 1999). Con el objetivo de que Xmipp sea compatible con otros paquetes software como los nombrados, Xmipp puede leer de forma nativa cualquiera de los formatos de fichero utilizados por cualquiera de estos paquetes. A continuación se explican los puntos a favor que permite comparar Xmipp con el resto de paquetes software dedicados al procesamiento de imágenes EM:

- Clasificación de imágenes 2D y 3D: La clasificación de imágenes corre un papel fundamental en el paso de preprocesamiento en EM. Su

objetivo es el de ordenar la población original en diferente subpoblaciones homogéneas, produciendo dos ventajas: ayuda a distinguir las proyecciones que provienen de diferentes especímenes o por diferentes proyecciones del mismo espécimen y permite agrupar proyecciones con direcciones de proyección parecidos.

- *Contrast transfer function (CTF):* Xmipp permite la estimación y la incorporación de esta función en el algoritmo de reconstrucción. La estimación de *CTF* se realiza usando un modelo paramétrico 2D para obtener la densidad espectral de potencia seguida de un ajuste 2D de un modelo teórico *CTF*. Una vez estimada esta función Xmipp proporciona métodos para realizar su correción.

- Asignamiento angular: Xmipp ha desarrollado un potente algoritmo de asignamiento angular basado en la correlación de las imágenes experimentales.

- Métodos de reconstrucción 3D: Se han desarrollado algoritmos iterativos de reconstrucción en los cuales las funciones base que se usan para describir los volúmenes no son voxels sino funciones suaves. En estos métodos, el volumen que va a ser reconstruido es representado como la suma de los pesos de funciones base, reduciendo el problema de la reconstrucción a la estimación de pesos. Esto se realiza en un proceso iterativo en el que las proyecciones del volumen reconstruido calculado a través de un modelo de formación de la imagen se asemejan a las proyecciones experimentales ofrecidas por el microscopio. Xmipp ha implementado diferentes algoritmos iterativos de reconstrucción como: algebraic reconstruction technique (ART, Herman, 1980), simultaneous iterative reconstruction technique (SIRT, Gilbert, 1972), simultaneous algebraic reconstruction technique (SART, Andersen and Kak, 1984), component averaging (CAV, Censor et al., 2001a), block-iterative component averaging (BICAV, Censor et al., 2001b), y Averaging strings (Censor et al., 2000).

- Comparación objetiva de métodos de reconstrucción 3D: El análisis del rendimiento de dos algoritmos diferentes en un fundamento clave en el procesamiento de imágenes. Xmipp ofrece diferentes aproximaciones para la comparación del rendimiento de estos métodos. Estos programas permiten el diseño de volúmenes simulados (ya sea a través de elementos geométricos como cilindros o conos o a través de estructuras atómicas PBD), generar proyecciones con cualquier tipo de colección geométrica, añadir diferentes tipos de ruido, simular el efecto de *CTF,* etc. Con el objetivo de evaluar la calidad de los diferentes algoritmos, su salida puede ser comparada con el volumen original (la reconstrucción ideal) a través de figuras de mérito.

2.1.2 Estructura de Xmipp

Xmipp es un paquete portable a cualquier máquina que tenga instalado el compilador C++ de GNU y las librerías gráficas Qt. En la práctica esto incluye cualquier máquina UNIX y Windows vía Cygwin. Este paquete contiene una jerarquía de clases organizadas en librerías que implementan toda la funcionalidad del paquete.

En la actualidad, el core de Xmipp está compuesto por diferentes librerías separadas por funcionalidad: Librería de funciones, librería de programas externos, librería de programas Java, librería de reconstrucción, librería de clasificación y librería de estructura datos entre otras

La librería de funciones, proporciona más de 150 programas distintos. Estos programas no son más que los front-ends de las bibliotecas del núcleo y forman la interfaz de usuario a través de la línea de comandos UNIX, proporcionando una amplia gama de herramientas para la comunicación entre procesos y para la automatización de tareas. Durante la implementación de este proyecto se han utilizado varios de estos programas, realizando desde Java llamadas a la línea de comandos de UNIX, algunos de estos ejemplos son los programas image_convert o xray_import, cuya función explicaremos detalladamente en capítulos [2.5.2] y [2.5.3]

La librería de programas externos contiene los ejecutables o las carpetas donde se ubican los programas con los que interactúa Xmipp, como es el caso de ImageJ, muy interesante para este proyecto ya que la implementación a realizar se hará como plugin de este programa. La librería de programas Java, ubicada en el directorio *java* de Xmipp contiene todas las clases Java implementadas en Xmipp, con las clases que permiten además el uso de JNI para realizar llamadas a código C++ de otras librerías de Xmipp desde Java.

Por otra parte, la librería de estructuras de datos proporciona estructuras de datos y funcionalidades como el uso de vectores, matrices, volúmenes, transformadas de Fourier, ficheros Xmipp I/O (selection files), funciones de error, conversiones de tipos, etc. Mientras que la librería de clasificación maneja estructuras de datos y funcionalidades para dar soporte a los algoritmos de clasificación implementados.

Por último, la librería de reconstrucción implementa todas las funcionalidades relacionadas con la obtención de la estructura tridimensional, incluyendo funciones de asignamiento angular, detección y corrección de *CTF,* reconstrucción 3D y evaluación de la calidad de las reconstrucciones.

Esta jerarquía de estructuras de datos y funciones proporciona a los programadores una base sólida para la creación de nuevos algoritmos relacionados con el procesamiento de imágenes. A continuación se muestra la jerarquía de paquetes de Xmipp. Figura . En la que únicamente han sido incluidas aquellas con las que se ha trabajado a lo largo del proyecto y sobre la que hablaremos más detalladamente en el capítulo [5] en el que veremos qué contiene cada carpeta y para qué que usa en este proyecto.

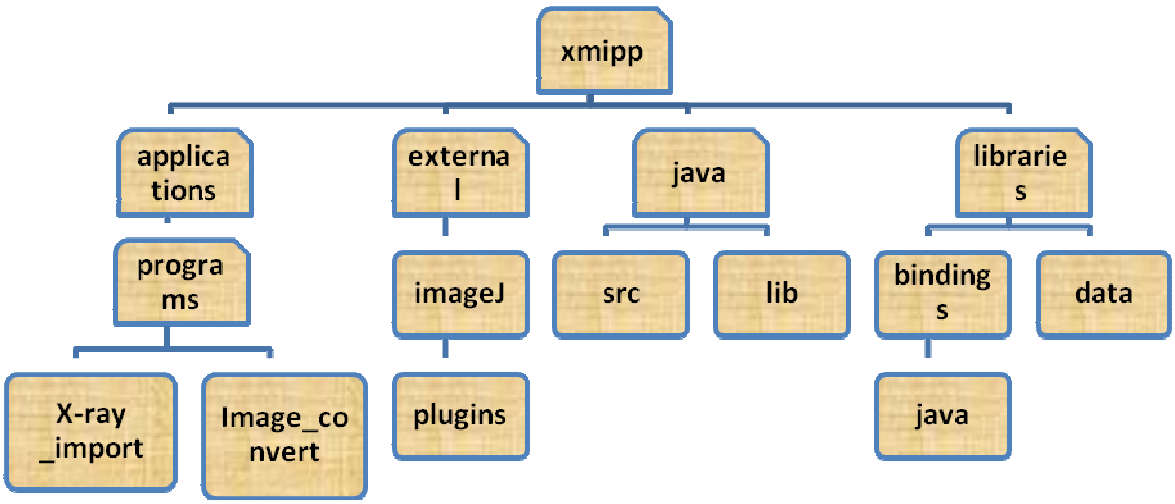


Figura 2.1.2.1: Jerarquía de clases de Xmipp. Únicamente se muestran los directorios usados durante el proyecto

2.2 ImageJ

ImageJ es un conocido programa de procesamiento digital de dominio público programado en Java desarrollado por el Instituto Nacional de la Salud, que ofrece un gran número de posibilidades para el análisis de imágenes, incluyendo capacidad para mostrar, editar, analizar, procesar, guardar, e imprimir imágenes de 8, 16 y 32 bits. Puede leer varios formatos de imagen incluyendo TIFF, PNG, GIF, JPEG, DICOM, FITS, así como formatos RAW.

ImageJ puede calcular el área y las estadísticas de valor de píxel de selecciones definidas por el usuario y la intensidad de objetos umbral (*thresholded objects*). Puede medir distancias y ángulos. Se pueden crear histogramas de densidad y gráficos de línea de perfil. Es compatible con las funciones estándar de procesamiento de imágenes tales como operaciones lógicas y aritméticas entre imágenes, manipulación de contraste, convolución, análisis de Fourier, nitidez, suavizado, detección de bordes y

filtrado de mediana, tal y como podemos apreciar en la figura [2.1] Permite además realizar transformaciones geométricas como ampliar y rotar. Es compatible con cualquier número de imágenes al mismo tiempo, limitado solamente por la memoria disponible.

Lo interesante de esta herramienta en cuanto a este proyecto, radica en que se pueden desarrollar plugins usando el editor incluido en ImageJ y un compilador Java. Los plugins desarrollados por usuarios permiten amoldarse a las necesidades de cada uno ofreciendo todas las funciones y algoritmos implementados en ImageJ.

La limitación principal de este programa, como en cualquier otra aplicación software Java, es el tratamiento de la memoria RAM. La tomografía electrónica requiere un alto uso de memoria, que está limitada debido a los procesadores de 32-bit de Java. En el caso de trabajar con gran cantidad de datos la máquina requiere procesadores de 64-bit y, al menos, 4 Gigabytes de memoria RAM.

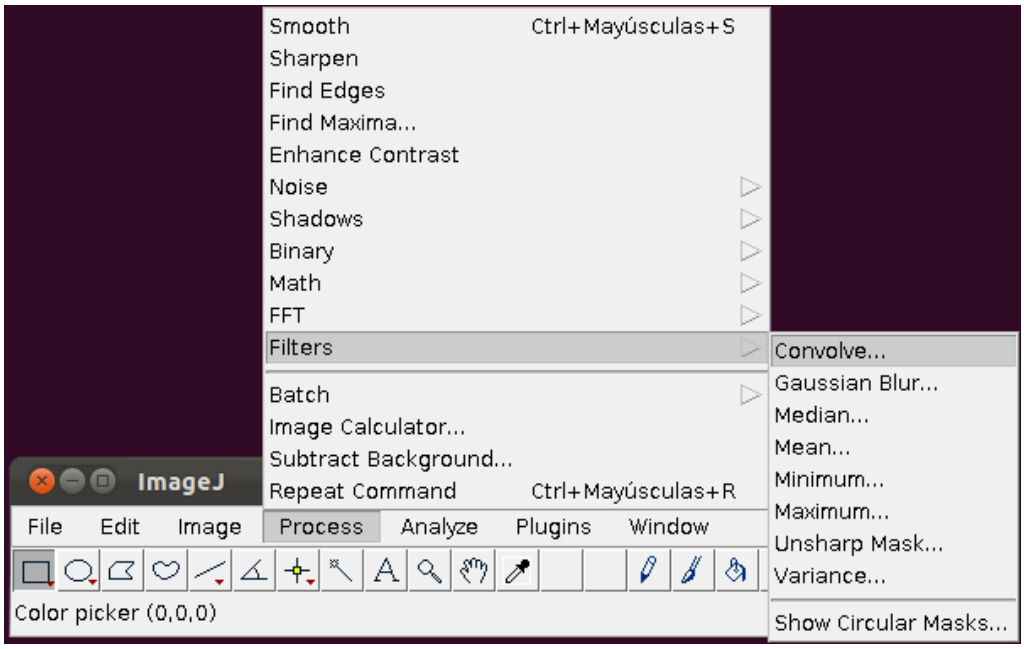


Figura 2.1: Interfaz de usuario de ImageJ.

2.3 TomoJ

2.3.1 Introducción a TomoJ

Tomography software for three-dimensional reconstruction in transmission electron microscopy (TomoJ) es un plugin para el programa de análisis de imágenes ImageJ que proporciona una interfaz para el preprocesamiento, alineamiento, reconstrucción y combinación de múltiples volúmenes tomográficos, incluyendo los más recientes algoritmos de reconstrucción, tanto algebraicos como iterativos, así como la técnica *WBP* (weighted back-projection)

Como plugin de ImageJ, TomoJ mantiene todas las ventajas de ImageJ en cuanto al análisis de imágenes, ofreciendo además una fácil instalación, portabilidad (al estar implementado en Java puede correr en distintos sistemas operativos) y la simplicidad que ofrece su interfaz de usuario.

En cuanto a la lectura y escritura de ficheros, TomoJ mantiene todos los formatos que proporciona ImageJ, añadiendo además el formato estándar de imágenes de microscopía electrónica MRC y SPIDER.

Para realizar la instalación de este software previamente deberemos instalar el programa ImageJ y acto seguido copiar el fichero de distribución TomoJ_.jar en el directorio plugins de ImageJ.

Como ya hemos mencionado en capítulos anteriores, TomoJ será el programa base de este proyecto, ya que a partir de él se implementarán funciones extras implementadas en Xmipp para aumentar su funcionalidad.

2.3.2 Interfaz de usuario de TomoJ

Una vez cargada una imagen a partir de ImageJ, podemos abrir el plugin TomoJ a partir de la pestaña *plugins* de la interfaz de ImageJ, lo cual abrirá la interfaz de usuario del plugin TomoJ. (ver figura 2.3.2.1)

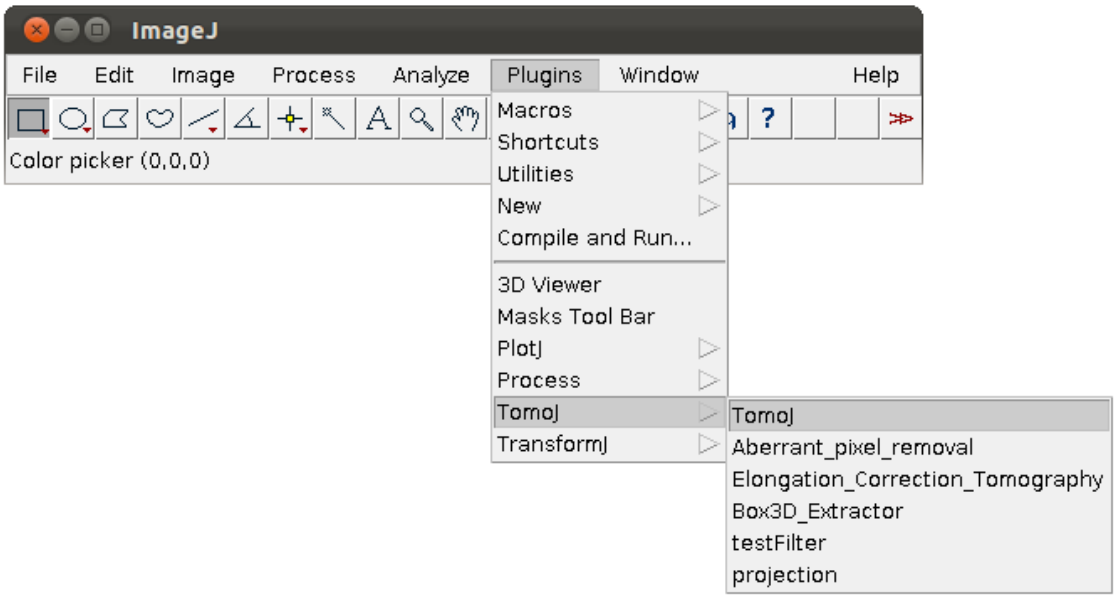


Fig. 2.3.2.1. Abriendo el plugin TomoJ a través de ImageJ

La interfaz de usuario de TomoJ, Figura [2.3.2.2] está dividida en dos partes: La parte superior, que permite al usuario realizar funciones relacionadas con la carga y el almacenamiento de imágenes reconstruidas, y la parte inferior, que contiene diferentes pestañas para realizar los diferentes pasos a seguir en la reconstrucción de una imagen. En este punto es importante resaltar que, en este contexto, cuando hablamos de una imagen no hablamos de una proyección particular, sino que llamamos imagen al conjunto de proyecciones realizadas desde diferentes ángulos que conforman una pila de proyecciones.

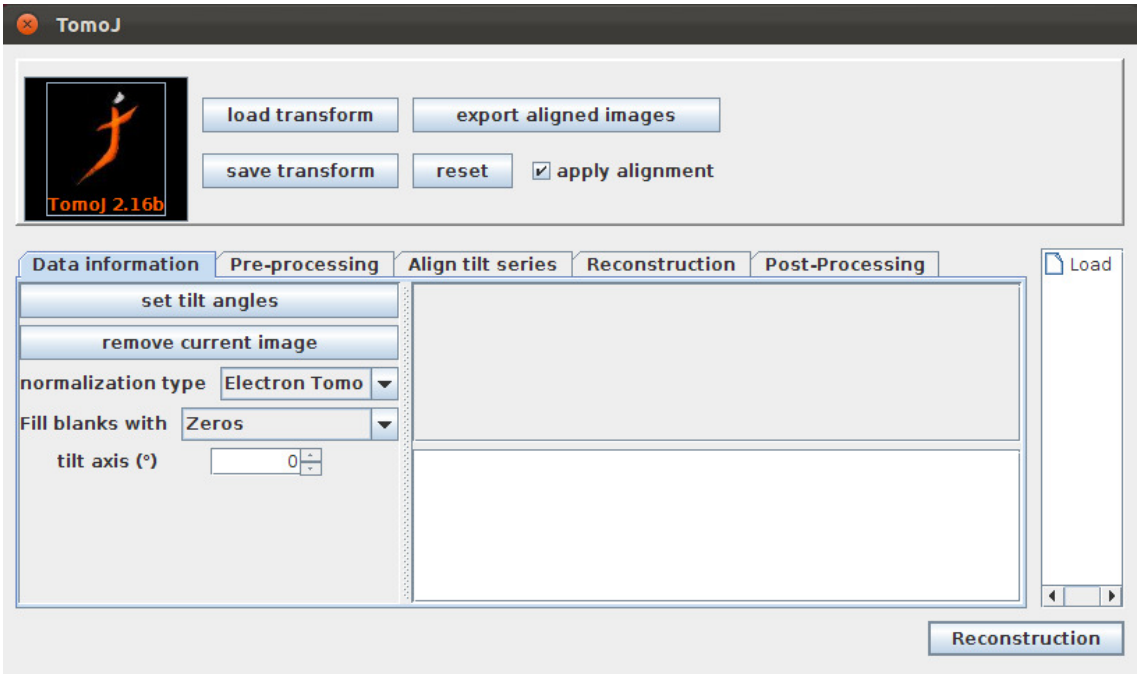


Figura 2.3.2.2 Interfaz de usuario de TomoJ

La parte superior de la interfaz contiene opciones generales que pueden ser utilizadas por el usuario en cualquier momento:

- *load transform*: Permite la carga de transformaciones que ya han sido realizadas. Llamamos transformación en este contexto a la imagen alineada. Los algoritmos de alineamiento crean un fichero TXT con la matriz de alineamiento para la serie de proyecciones. A partir de esta opción podemos aplicar la matriz obtenida a la serie de proyecciones para alinearla.

- *save transform*: Almacena la transformación obtenida en un fichero. Este fichero consistirá en un fichero de texto que almacenará en cada fila la imagen relativa a cada proyección así como su ángulo y eje de inclinación.

- *export aligned images*: Este botón permite exportar una serie de inclinación alineada. Si el nombre termina con .mrc, .spi, .xmp ó .sel el

archivo se guadará en el formato correspondiente. En el caso de que tenga otra extensión diferente se guardará en formato .tiff.

- *reset:* resetea la transformación.

- *apply alignment:* permite mostrar las imágenes alineadas si está activo o las no alineadas si no está activado.

La parte inferior de la interfaz permite al usuario realizar la reconstrucción tridimensional de la imagen cargada ofreciendo la posibilidad de realizarla en función de las necesidades del usuario, permitiendo diferentes tipos de pre-procesado, alineamiento y algoritmos de reconstrucción. A continuación detallaremos cada una de estas pestañas relativas a los pasos del proceso de reconstrucción, herramientas que utilizaremos posteriormente para realizar una reconstrucción tridimensional a partir de una serie de proyecciones.

La pestaña *data information* contiene información general sobre la visualización de la imagen y permite realizar modificaciones sobre la serie de inclinación.

- *set tilt angles:* En el momento en el que se abre el plugin de TomoJ, antes de mostrarse la interfaz, el usuario debe introducir el ángulo de inclinación con el que comienza la serie así como la separación entre ángulos, de este modo y en función del número de proyecciones que contiene la serie, se fijan los ángulos de inclinación para cada proyección. Este botón permite realizar cambios sobre el ángulo de inclinación en el que comienza la serie una vez abierto el programa.

- *remove current image:* Elimina la proyección que se está mostrando y toda su información asociada. A diferencia de esta filosofía, Xmipp propone descartar en lugar de eliminar proyecciones, permitiendo al usuario habilitarla de nuevo si es necesario.

- *normalization type:* La serie de proyecciones debe de situarse en un mismo marco numérico con el fin de eliminar las diferencias entre los distintos rangos de las diferentes proyecciones. Estas diferencias se deben principalmente a que el haz de electrones del microscopio atraviesa diferente grosor de la muestra en función al ángulo de inclinación con el que se está realizando la proyección. TomoJ permite realizar dos tipos de normalizaciones: *Zero-One*, que deja todas las proyecciones de la serie con una media de cero y desviación típica de uno y *Electron tomography*, que deja a las proyecciones con una media de cero y una desviación típica que varía en función del coseno del ángulo de inclinación. Además, permite la posibilidad de no aplicar ningún tipo de normalización, utilizada en el caso en que el que la normalización ya se haya realizado antes de abrir el programa a través de ImageJ u otra aplicación.

- *fill blanks with:* Este botón permite rellenar los espacios en blanco producidos por el alineamiento de diferentes formas: *Zeros*, que pone ceros en lugar de los espacios en blanco, *Mean,* que rellena los espacios en blanco con la media de la imagen y *NaN* , que rellena los espacios en blanco con un NaN, de este modo, cuando se realiza la reconstrucción, algoritmos como SIRT no tienen en cuenta estas posiciones y otros como WBP los reemplaza por la media de la imagen automáticamente.

- *tilt axis:* Permite realizar cambios sobre el eje de inclinación de la serie.

La pestaña de preprocesamiento permite realizar funciones de preprocesado de imágenes con el objetivo de mejorar la relación señal a ruido SNR de la serie de proyecciones. Para ello, TomoJ propone tres técnicas diferentes de preprocesado. En este punto es imprescindible recordar que uno de los objetivos de este proyecto era el de ampliar los algoritmos de preprocesado de TomoJ, como se verá en el capítulo [2.5.1]

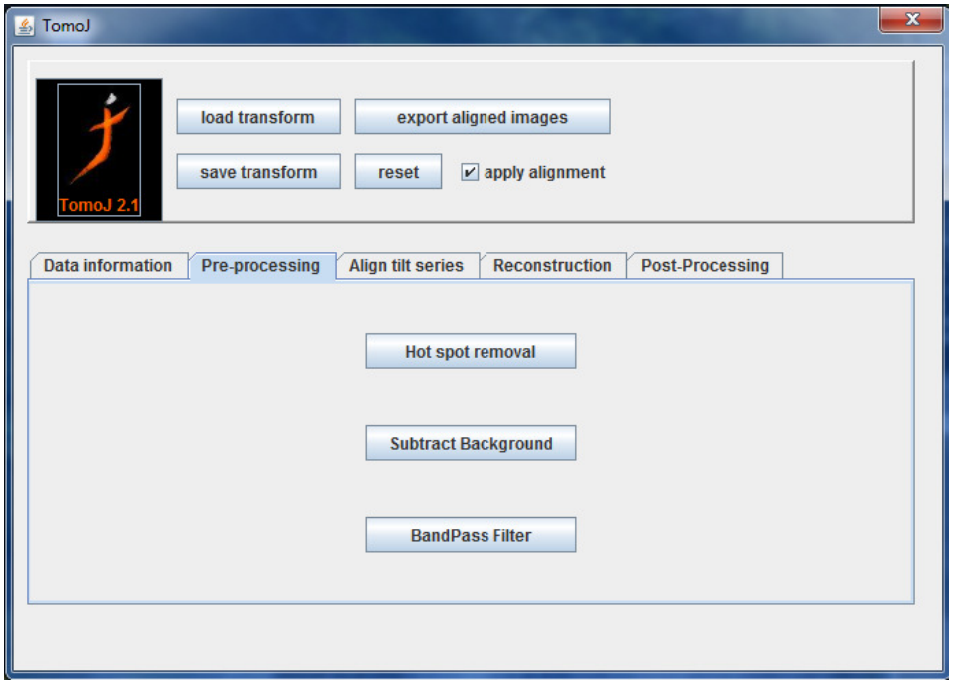


Fig. 2.3.2.3 Pestaña de pre-procesamiento de la interfaz de TomoJ

Tal y como observamos en la figura [2.3.2.3], los algoritmos de preprocesado implementados por *TomoJ* son: *hot spot removal*, *subtract background* y *band pass filter*

- *Hot Spot Removal:* Se trata de una herramienta que permite eliminar píxeles aberrantes (blancos ó negros) debidos entre otros por los rayos X. Para ello calcula la diferencia con sus vecinos y detecta los píxeles aberrantes como los que tengan los valores mayores que un cierto umbral definido en función de la desviación estándar de la imagen. Después reemplaza estos píxeles con el valor resultante del promedio de sus vecinos. Este algoritmo recibe como parámetro el radio de vecindad, es decir, el número de pixeles que se desplaza para obtener los vecinos.

- *Subtract Background:* A través de este algoritmo se suaviza el fondo de la serie de proyecciones.

- *BandPass Filter:* Permite aplicar filtros sobre la imagen en el espacio de Fourier. Recibe como parámetros: *low cut (pixels)*, valor mínimo del radio a partir de cual, por debajo, todos los valores frecuenciales son puestos a cero y los superiores son multiplicados por una función coseno para ir desde cero hasta el valor de la señal, *low pass y high pass,* radio mínimo y máximo a partir del cual todos los valores por encima se mantienen intactos, y *high cut*, radio máximo en el cual, por encima, todas las frecuencias son puestas a cero y por debajo la señal es multiplicada por una función coseno para ir desde cero hasta el nivel de la señal. Estos parámetros son recogidos por la interfaz que podemos observar en la figura [2.3.2.4]

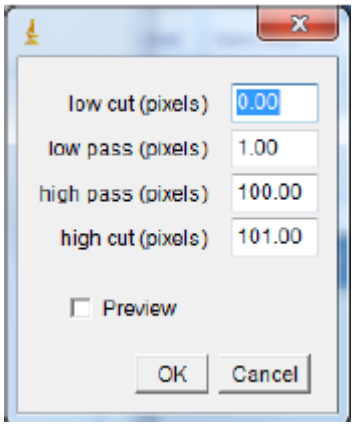


Fig. 2.3.2.4 Dialogo que ofrece TomoJ al aplicar un filtro paso banda

El panel de alineamiento es el que observamos en la figura [2.3.2.5], está dividido en dos partes. La primera de ellas, denominada *without landmarks* es relativa a los procesos de alineamiento que realizan sin la necesidad de usar cadenas Landmarks, mientras que la segunda hace referencia a los procesos que se utilizan para realizar el alineamiento de una serie utilizando estas cadenas.

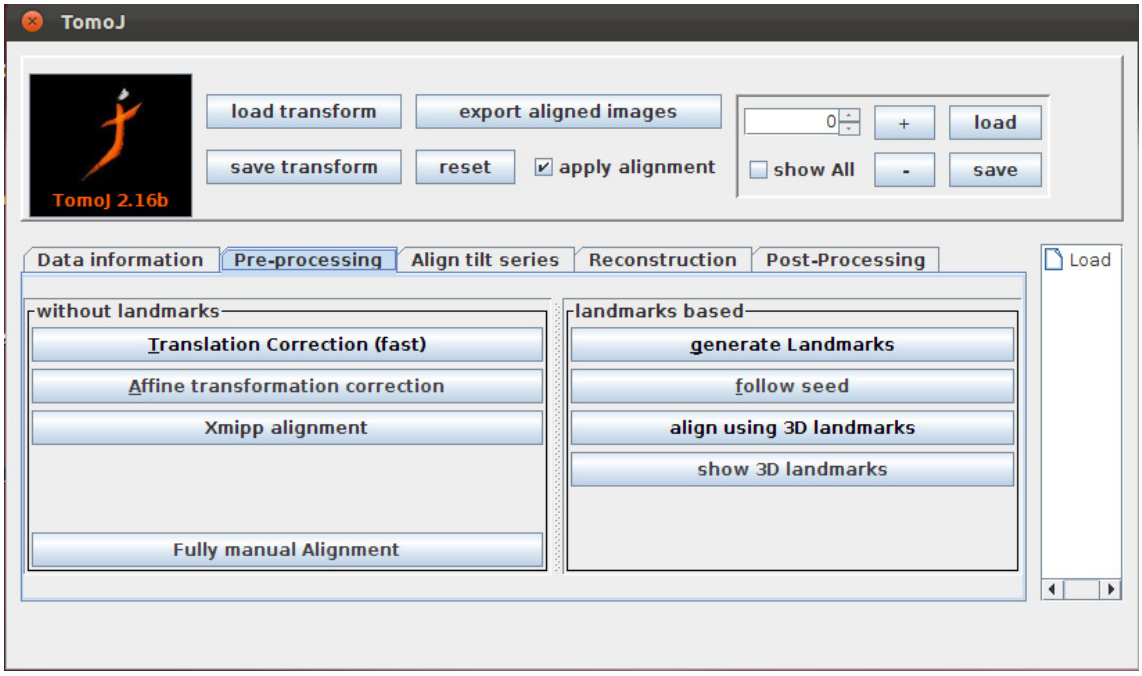


Figura 2.3.2.5 Pestaña de alineamiento proporcionada por TomoJ

El panel *without landmarks* permite varias formas de realizar el alineamiento de la serie sin emplear el uso de landmarks:

- *Translation correction*: Corrige la translación de la serie utilizando la correlación cruzada en el espacio de Fourier. Para ello obtiene los coeficientes de correlación cruzada (CCC) entre cada dos imágenes consecutivas y, a continuación se mueve una de las imágenes el número de píxeles necesarios para maximizar la CCC. Este paso se lleva a cabo en el dominio de la frecuencia utilizando la propiedad de la correlación de la transformada de Hartley (HT).

- *Affine transformation correction*: Corrige la translación así como la rotación de la serie de proyecciones. Esta imagen corregida se usa sólo para la detección de landmarks, puesto que las imágenes utilizadas para reconstruir son alineadas a partir de la información estándar de las

propias landmarks. Los parámetros utilizados para utilizar en este algoritmo son los que podemos apreciar en la figura [2.3.2.6].

- Para el uso de los procesos anteriores, TomoJ muestra una interfaz al usuario para obtener una serie de parámetros para realizar el alineamiento. Estos parámetros permiten seleccionar si se desea utilizar la parte central de la imagen o bien una zona seleccionada (*roi centered)*, la posiblidad de utilizar un filtro paso banda en espacio de Fourier como el que se puede utilizar en la pestaña de procesamiento, permite realizar *downsampling* sobre la imagen, seleccionar si se desea realizar una translación entera o de punto flotante (*integer translation)*, indicar si se desea expandir la imagen en la dirección perpendicular al eje de inclinación por medio de un factor dependiente del ángulo de inclinación (*expand images)*, y, por último, a través del checkbox *cumulative reference,* permite que en lugar de alinear las proyecciones de dos en dos, tome la proyección de inclinación cero como la primera referencia para alinear la imagen al siguiente ángulo más negativo. A continuación se añade esta proyección a la referencia para alinear la siguiente imagen. Para cada proyección, la referencia será la suma de las proyecciones que ya han sido alineadas. Cuando se llega a la proyección con el ángulo más negativo, el procedimiento es repetido desde la proyección cero hasta aquella que contenga la inclinación mayor.

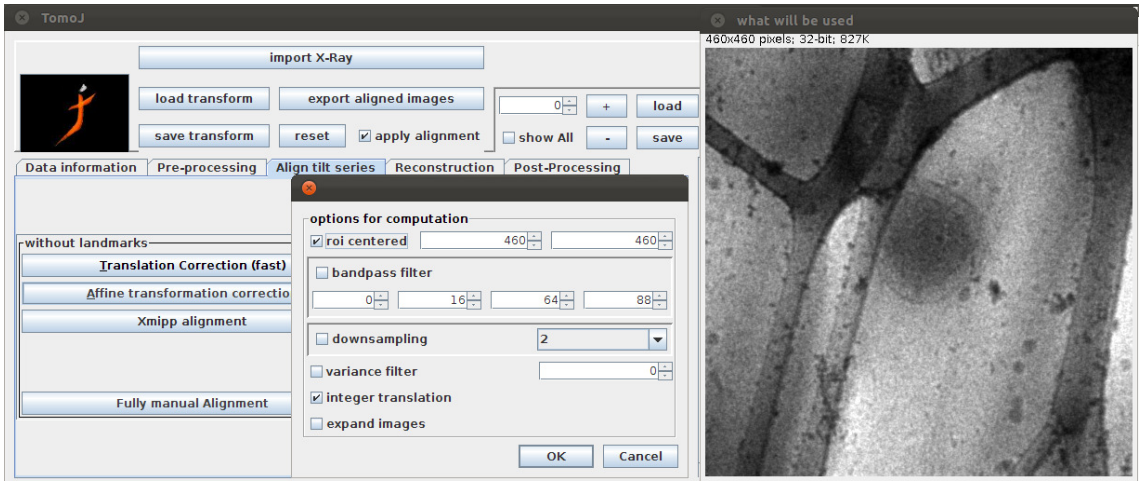


Fig. 2.3.2.6: Interfaz proporcionada por TomoJ para realizar el alineamiento de la serie de proyecciones a través del algoritmo *Affine Transformation Correction*

- *fully manual alignment:* Este método permite realizar el alineamiento de la serie de forma totalmente manual. Se suele utilizar como reemplazamiento del método *translation correction* cuando este no funciona y no necesita tener en cuenta ningún alineamiento previo. Cuando el usuario utiliza este método, TomoJ proporciona una ventana en la cual se realiza el alineamiento. Esta nueva ventana muestra distintas imágenes: La imagen que está siendo alineada (parte superior izquierda) y la imagen anterior que le sirve a esta como referencia para realizar el alineamiento (parte superior derecha y parte inferior izquierda). Además, en la parte inferior derecha podemos encontrar una superposición de las dos imágenes anteriores. En el caso en que las dos imágenes estén bien alineadas el usuario verá únicamente tonos grises, mientras que en otro caso contrario se verán tonos verdes y magentas. Esta nueva interfaz proporciona herramientas para realizar la traslación y la rotación de la imagen, aunque también se puede realizar directamente con el ratón. Por último hay que destacar que TomoJ, para facilitar el alineamiento al usuario ofrece además la posibilidad de realizar *downsampling* y de expandir la imagen en dirección

perpendicular al eje de inclinación para compensar la diferencia de ángulos de inclinación entre imágenes.

- *xmipp alignment:* A través de este botón el usuario puede realizar el alineamiento de la serie con un algoritmo de alineamiento desarrollado por los miembros del equipo de desarrollo de Xmipp. Este algoritmo no ha sido integrado en el software TomoJ por parte del alumno, aunque sí ha colaborado en esta tarea, por lo que no nos centraremos en su integración, aunque sí en su funcionalidad.

De entre las diferentes posibilidades de alineamientos posibles que introdujimos en el capítulo [1.3.4] podemos ubicar el algoritmo de alineamiento de Xmipp en aquellos que utilizan la correlación cruzada para realizar un seguimiento de las características de las diferentes proyecciones, definiendo estas características como los puntos de referencia en las imágenes (como puedan ser cambios de contraste, bordes o esquinas) que son buscados realizando la correlación cruzada en proyecciones vecinas. Para ello, este algoritmo lleva a cabo un registro general de la imagen para predecir la posición de cualquier región de la proyección en cualquiera de sus proyecciones adyacentes. Si todas las proyecciones están numeradas de acuerdo a su ángulo de inclinación una región de la imagen *i* es buscada en la proyección *i+1,* prediciendo antes su posición a través de la transformación afín y, después a través del refinamiento de su correlación cruzada. Una vez que se identifica la región de la proyección *i* en la proyección *i+1*, se solicita su búsqueda en la región *i+2* y así sucesivamente. De esta forma, el algoritmo propaga una coordenada de la proyección *i* en las proyecciones *i+1, i+2...* hasta que la región buscada no se pueda encontrar debido a su oscurecimiento por otros objetos, su cambio de forma o el ruido. De la misma forma se realiza el proceso hacia atrás. Todas las coordenadas correspondientes a regiones equivalentes en distintas proyecciones forman lo que se denominan cadenas de

Landmarks, y, lo más probable es que no cubra toda la serie de inclinación. Estas cadenas de Landmarks se refinan de nuevo para garantizar que corresponden a regiones equivalentes. Finalmente todas las cadenas de Landmarks se utilizan para realizar una regresión robusta de los parámetros de alineamiento.

Una gran ventaja de este enfoque reside en que el proceso es totalmente automático y puede producir miles de cadenas Landmarks para alinear la serie.

Además de los algoritmos que hemos comentado, TomoJ proporciona sus propios algoritmos de alineamiento basados en cadenas Landmarks. Como podemos observar en la figura [2.3.2.7] al situarse en la pestaña de alineamiento, el programa muestra un menú al usuario en la parte superior derecha de la interfaz.

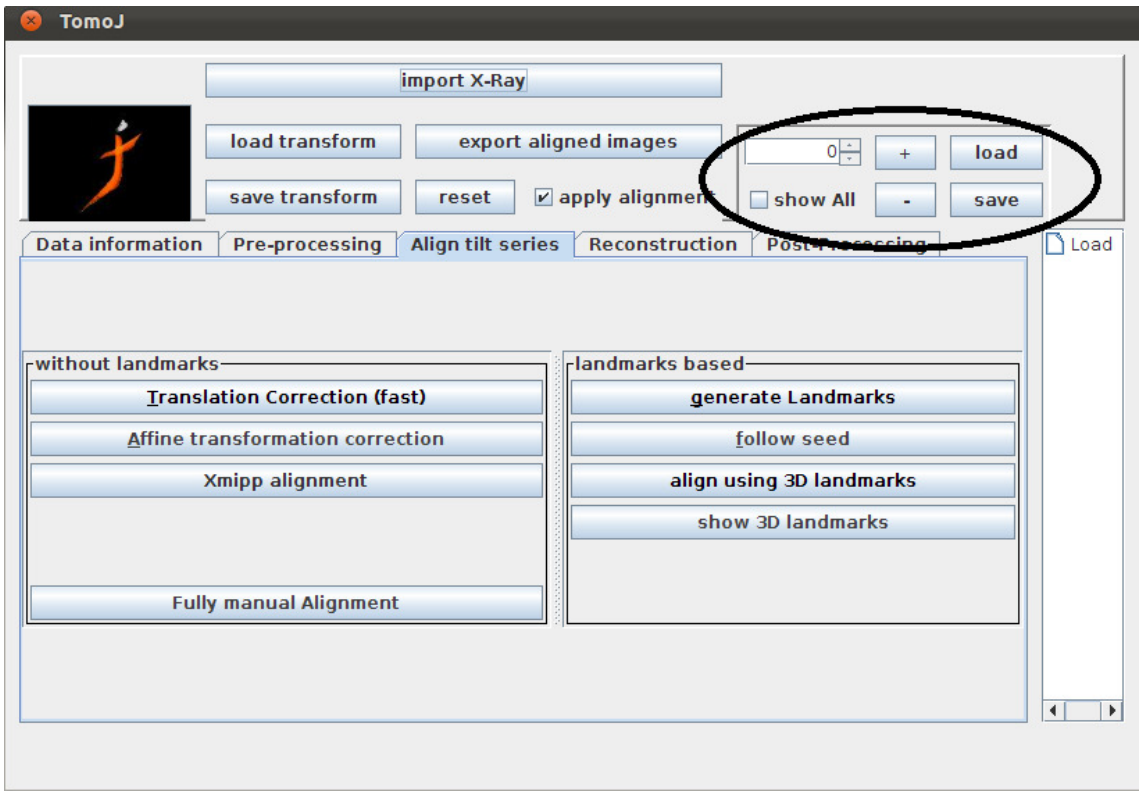


Figura 2.3.2.7: Menú de alineamiento de TomoJ para realizar el alineaiento de la serie a trvés de landmarks

Este menú se utiliza para la creación, modificación y eliminación de cadenas Landmarks. Estas cadenas vienen dadas como puntos ROI de ImageJ (figura 2.3.2.8) y, para que un punto pueda ser considerado como Landmark, únicamente puede haber uno en la imagen. Las opciones disponibles en este menú son:

- +: Permite agregar una cadena Landmark vacía, poniendo un nuevo punto ROI sobre la imagen que permitirá definir el punto de referencia
- -: Permite eliminar la cadena que se muestra actualmente. Si además se pulsa CTRL se eliminarán todas las cadenas de referencia.
- *load* y *save:* Permiten cargar y guardar cadenas Landmark desde ó a un archivo
- *show all:* Muestra todos los puntos de referencia seleccionados para una imagen.

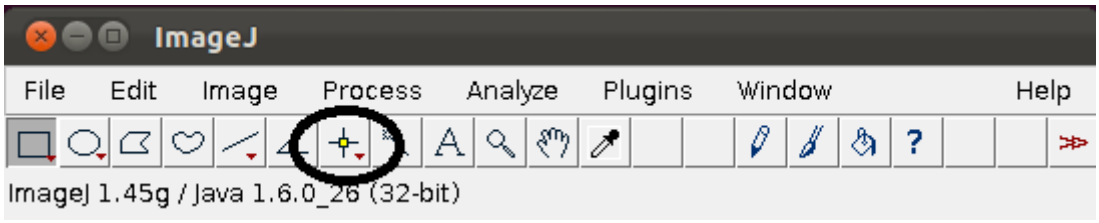


figura 2.3.2.8: Herramienta de utilización de puntos ROI en la interfaz de usuario ImageJ

Además de estas herramientas que permiten crear cadenas Landmarks a través de los puntos de referencia de forma manual, TomoJ proporciona algoritmos para realizarlo de forma automática además de otra serie de herramientas.

A través del botón *generate Landmarks* podemos crear automáticamente los puntos de referencia. Dos ventanas de vista previa aparecerán, la primera permite comprobar el tamaño de la ventana usado y la segunda permite el ajuste de los puntos de detección.

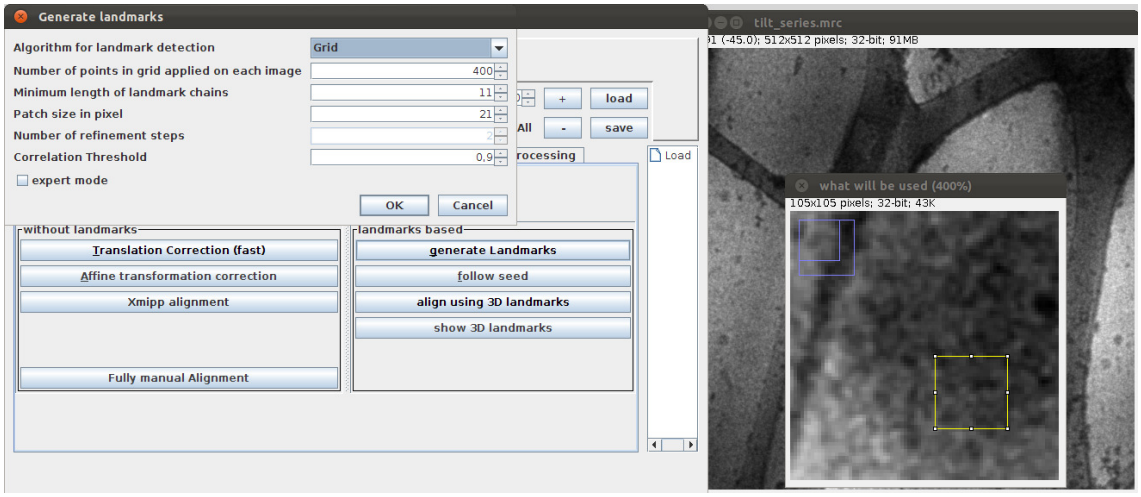


Figura 2.3.2.8: Interfaz proporcionada por TomoJ para la detección automática de Landmarks

A través de la interfaz de usuario para la generación automática de landmarks que podemos observar en la figura [2.3.2.8] podemos ejecutar diferentes tipos de algoritmos para realizar este cometido: Algoritmo grid y algoritmo de puntos críticos (usando máximos o mínimos). La interfaz de recogida de parámetros es la que podemos observar en la figura [2.3.2.8], mientras que la interfaz utilizada para ejecutar el algoritmo de puntos críticos es la que podemos observar en la figura [2.3.2.9]

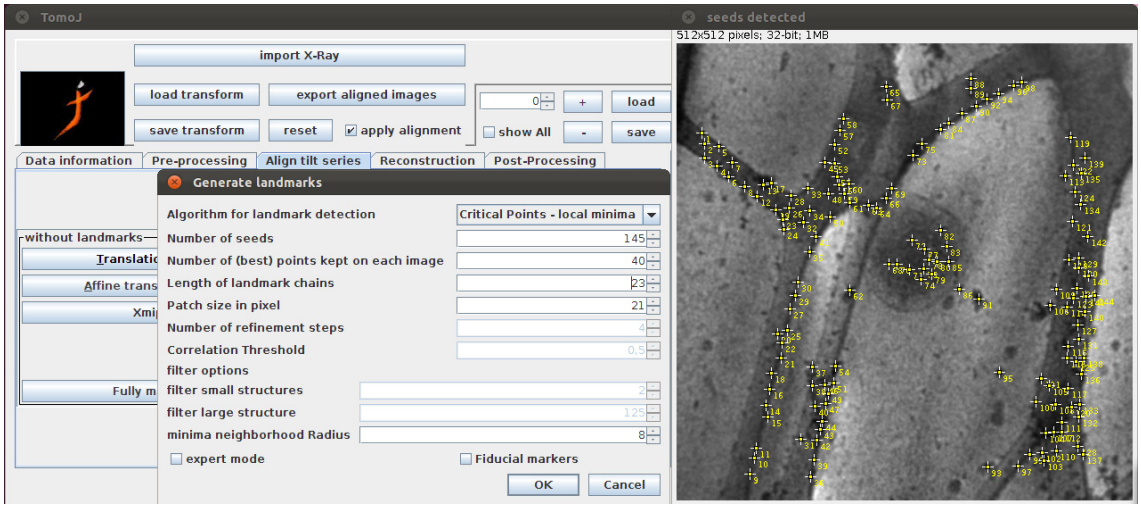


Figura 2.3.2.9: Interfaz de usuario para el uso del algoritmo de puntos críticos, en este caso se utiliza el algoritmo de mínimos locales y, como podemos observar, el propio TomoJ muestra al usuario el número de puntos críticos que el usuario haya introducido en el campo *Number of seeds*, ofreciéndole estos puntos críticos que van a ser seguido a lo largo de la serie de proyección.

A través del botón *Follow Seed* de la interfaz *Landmarks based* podemos realizar el seguimiento de cadenas de Landmarks definidas en proyecciones anteriores y sucesivas a la proyección en la que hemos establecido estos puntos de referencia. Para ello necesitamos una serie de proyecciones que haya sido alineada anteriormente.

Por su parte, el botón *Center landmarks* permite poner el baricentro de todas las cadenas landmarks en cada proyección para centrar la serie. *Align using 3D landmarks* realiza un refinamiento de la serie alineada a través de la determinación de la posición tridimensional de cada landmarks, y, por último, el botón *Show 3D landmarks* permite visualizar la posición de las Landmarks 3D obtenidas después de realizar el filtrado con *Align using 3D landamrks.*

La pestaña de reconstrucción permite al usuario realizar la reconstrucción tridimensional de la serie una vez alineada permitiendo utilizar diferentes algoritmos de reconstrucción como BP, WBP, ART y SIRT . El panel de reconstrucción, que podemos observar en la figura [2.3.2.10] está dividido en dos partes, la primera de ellas ofrece información general para todos los algoritmos de reconstrucción, mientras que la segunda permite seleccionar el tipo de algoritmo que se va a utilizar.

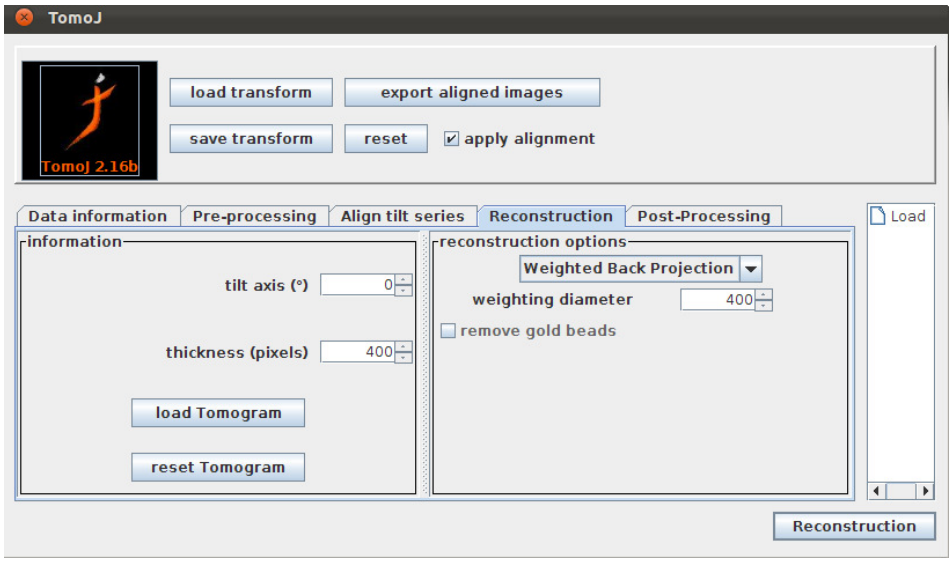


Fig.: 2.3.2.10: Panel de reconstrucción ofrecido por TomoJ

El panel de información permite al usuario seleccionar el eje inclinación con el que se va a realizar la reconstrucción a través del botón *tilt axis (º)* , ofreciendo una línea que muestra el eje de inclinación sobre la serie. Además, sobre la introducción de este valor TomoJ ofrece la posiblidad de realizar rotaciones sobre las imágenes clickeando con el botón izquierdo del ratón sobre el texto *tilt axis (º)*, recomendado para series en las que se ha realizado un alineamiento automático, así como obtener la suma de todas las imágenes alineadas de la serie clickeando esta vez sobre el texto con el botón derecho ó el mínimo de todas las imágenes alineadas si además el usuario mantiene pulsada la tecla *Ctrl*.

El botón *thickness* de este mismo panel permite indicar el volumen de espesor en píxeles que corresponde al número de voxels esperado que la reconstrucción ocupará en la dirección del eje Z. Este valor puede ser obtenido de forma aproximada dividiendo el espesor de la muestra en nm por el tamaño de la muestra en nm/pixel.

Por último, el panel de información permite cargar una reconstrucción calculada anteriormente para mejorarla a través de ART/SIRT a través del botón *load tomogram* y la posibilidad de volver a realizar una reconstrucción a través del botón *reset tomogram.*

Por su parte, la ventana de reconstrucción permite al usuario utilizar el algoritmo de reconstrucción que considere necesario. WBP es el algoritmo más utilizado en tomografía electrónica debido a su velocidad de procesamiento. Este algoritmo compensa las bajas frecuencias en el dominio de Fourier utilizando un sistema de ponderación de pesos antes de realizar la reconstrucción, acto seguido las proyecciones alineadas y ponderadas son proyectadas hacia atrás en un volumen tridimensional utilizando interpolación bilineal. Por su parte, los algoritmos ART y SIRT son realizados de manera iterativa ya que las proyecciones del volumen reconstruido calculado a través de un modelo de imagen parecido a las proyecciones obtenidas a partir de microscopio, asumiendo un modelo de proyección lineal con ruido gaussiano. Este modelo de proyección lineal se trata de una aproximación de primer orden de un proceso de formación de imágenes no lineal producido en el microscopio.

En el caso de elegir el algoritmo WBP, la interfaz permite al usuario introducir el parámetro *weighting diameter*, utilizado para reducir el diámetro del filtro usado para realizar la ponderación de los pesos.

Por su parte ART y SIRT son dos algoritmos iterativos en los cuales es necesario proporcionar el número de iteraciones con el que se desea realizar la reconstrucción, tal y como se muestra en la figura [2.3.2.8]. Se recomienda que este número de iteraciones nunca sea menor que cuatro para obtener un buen tomograma, aunque el valor más utilizado por los usuarios de TomoJ suele ser de diez iteraciones. La elección del número de iteraciones depende de la media de los errores cuadráticos

obtenidas por las diferencias entre las proyecciones del volumen reconstruido y los datos experimentales. Una vez realizado el proceso de reconstrucción TomoJ ofrece una curva de error mostrando el error obtenido en función del número de iteraciones.

El segundo parámetro que utilizan estos algoritmos es el coeficiente de relajación, se trata de un factor de ponderación utilizado para mejorar la calidad de la reconstrucción, habitualmente a expensas de la convergencia. TomoJ propone para el algoritmo ART un coeficiente de relajación equivalente a uno dividido entre el número de iteraciones que hayamos elegido y de uno para SIRT, que es menos sensible al ruido. Experimentalmente estos son buenos valores para la mayoría de los casos. Sin embargo, estos valores pueden ser modificados manualmente a través de la ventana de reconstrucción.

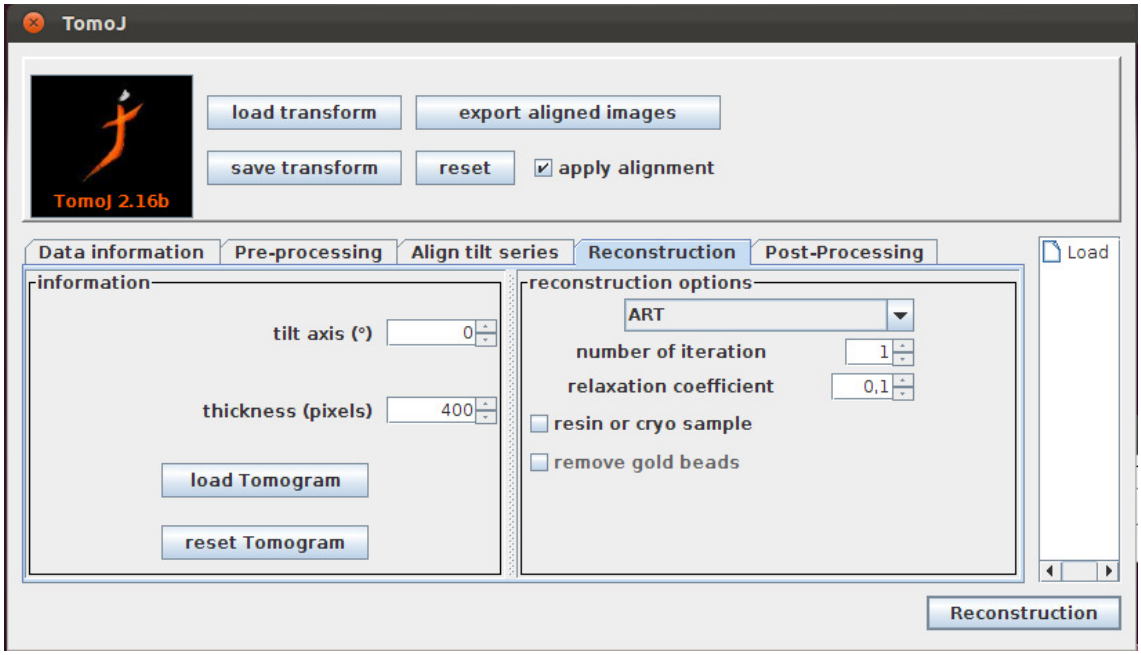


Fig. 2.3.2.11: Interfaz de reconstrucción a partir de ART proporcionado por TomoJ

En general, se utilizan valores de coeficientes de relajación bajos en el caso en que la serie tenga una baja relación señal a ruido y valores mayores en otro caso. La

curva de error también se puede mostrar en función del coeficiente de relajación, una rápida convergencia significa que se ha elegido un coeficiente de relajación demasiado alto, mientras que una convergencia lenta implica que el coeficiente de relajación es demasiado bajo.

Por último, el botón *resin or cryo sample* comprueba si la muestra tiene un espesor constante mientras que el botón *remove gold beads* permite la posibilidad de eliminar las partículas de oro a través de la reconstrucción.

Por último, la pestaña de post-procesamiento (figura 2.3.2.12) permite corregir fallos que se hayan podido producir durante el proceso de reconstrucción, permitiendo al usuario:

- *elongation correction:* En el caso de realizar reconstrucciones sobre series en las que no disponemos de todas las proyecciones se puede producir un alargamiento de la reconstrucción en el eje Z (perpendicular al plano 0º), que puede ser corregido escalando la reconstrucción por un factor a lo largo del eje Z. Esto es usado sobre todo cuando se utiliza el algoritmo WBP ya que es un método muy sensible al fenómeno *missing wedge*.

- *box extractor:* Permite la extracción de fragmentos de la reconstrucción tridimensional una vez computada.

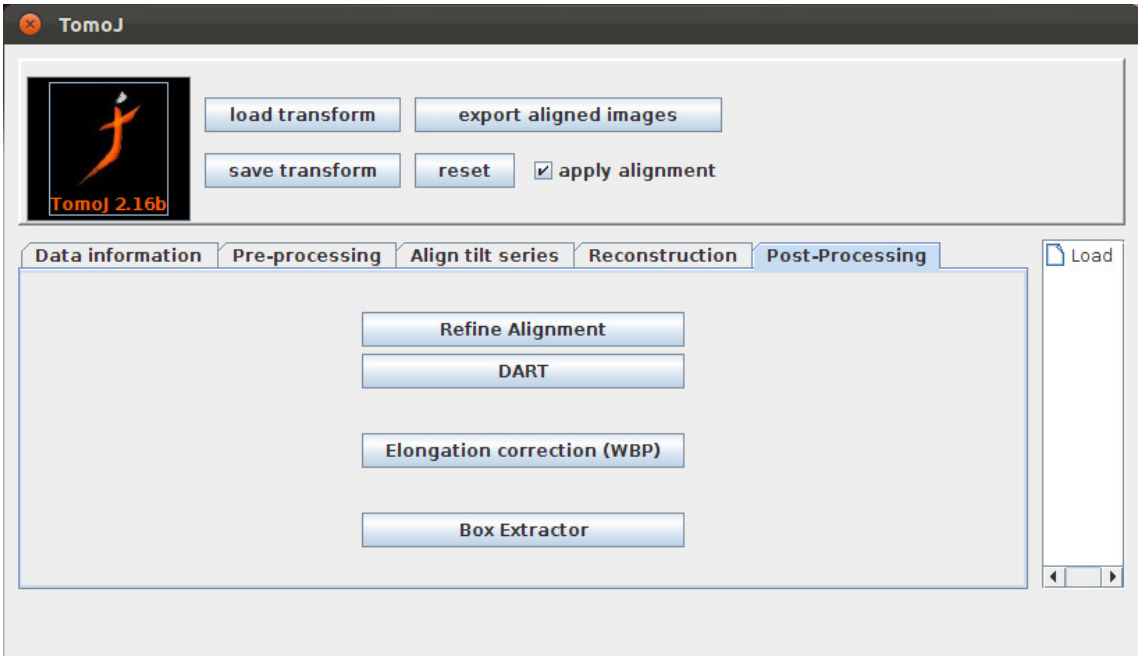


Fig. 2.3.2.11: Pestaña de postprocesamiento proporcionada por TomoJ

- *DART:* A través del botón DART podemos realizar una reconstrucción tridimensional de la serie utilizando para ello el algoritmo iterativo de reconstrucción *DART* (*Discrete ART)*, el cual es capaz de realizar reconstrucciones más precisas en series que cuentan con pequeño número de proyecciones, o que tienen un rango angular pequeño, que los algoritmos convencionales. Además, trata con eficacia el posible ruido que pueda tener la serie de proyecciones y se trata de un algoritmo muy robusto con respecto a errores de estimación de gris.

- *Refine Alignment:* A partir de este botón el usuario de TomoJ es capaz de, una vez obtenida la reconstrucción tridimensional de la serie, realizar mejoras sobre una reconstrucción ya realizada, basadas básicamente en el refinamiento del alineamiento de la estructura. Este tipo de algoritmos estiman los parámetros de alineamiento de una imagen experimental minimizando la diferencia entre ésta y las proyecciones bidimensionales del volumen de referencia.

Para finalizar este capítulo del proyecto considero necesario recalcar que cada una de las funcionalidades que hemos ido mencionando estaban ya implementadas antes de la llegada del alumno al proyecto y que, por lo tanto no nos hemos centrado en gran profundidad sobre cada una de ellas, cosa que sí haremos con la funcionalidad que el el alumno ha integrado en este software.

En capítulos posteriores veremos además cómo se realiza el *workflow* de trabajo con TomoJ para obtener una tomografía a partir de una serie de proyecciones (capítulo 4) .

2.4 Tomo3D

Fast tomographic reconstruction on multicore computers, Tomo3D, se trata de un software que permite realizar reconstrucciones tomográficas aprovechando el poder computacional de la tecnología multicore para reducir el tiempo de procesamiento.

Tomo3D implementa los métodos de reconstrucción WBP y SIRT, obteniendo reconstrucciones en el orden de segundos en el caso de WBP y del orden de pocos minutos en el caso de SIRT. Para ello recibe como entrada la serie de proyecciones y genera un fichero en formato MRC. Además este programa es compatible con paquetes estándares permitiendo su integración fácilmente en otros programas relacionados con la tomografía electrónica. Éste, es uno de los puntos claves de este proyecto ya que, uno de los objetivos de este es precisamente el de integrar este programa dentro de TomoJ.

Este programa consta básicamente de tres ficheros binarios que permiten ejecutar el programa en función del procesador de la máquina en la que se vaya a ejecutar:

- *tomo3D:* Utilizado para procesadores de 64-bits. Está optimizado particularmente para el procesador Intel Pentium IV aunque también funciona con procesadores AMD.

- *tomo3D.core2*: Utilizado para procesadores de 64-bits y optimizado para procesadores Intel Core 2 en adelante

- *tomo3D.ia32:* Utilizado para procesadores de 32-bits (Intel Pentium III en adelante)

El problema de la reconstrucción tridimensional en ET se puede descomponer en una serie de sub-reconstrucciones 2D correspondientes a las distintas proyecciones perpendiculares al eje de inclinación. Cada una de estas proyecciones 2D del volumen pueden ser obtenidas a partir de cualquier método de reconstrucción, ya sea WBP ó SIRT, pero trabajando ahora en el espacio bidimensional. La ventaja de este software reside precisamente en que la reconstrucción de las proyecciones independientes puede realizarse en paralelo por medio de los diferentes niveles de paralelismo que proporcionan los ordenadores multicore.

Tomo3D utiliza un enfoque multiproceso vectorizado para realizar reconstrucciones tomográficas creando para ello un número de hilos de ejecución (habitualmente el número de cores de los que disponga la computadora) que corren en paralelo. El programa mantiene en cada momento un pool con las proyecciones a reconstruir, agrupadas en bloques de cuatro. Estos bloques son enviados a los hilos de ejecución en cuanto estos estén inactivos. Cuando un bloque es adjudicado a un hilo de ejecución sus cuatro proyecciones son reconstruidas simultáneamente gracias a las instrucciones SSE. Cuando un hilo de ejecución finaliza con la reconstrucción de un bloque solicita otro proporcionando un mecanismo de balanceo de carga y haciendo que Tomo3D sea flexible y se adapte a situaciones en las que la computadora esté siendo utilizada por otros programas o usuarios.

Tomo3D ha sido desarrollado en C sobre Linux, utilizando Pthreads así como técnicas de optimización de código Singlecore para llevar a cabo un mayor aprovechamiento de la memoria y de las unidades de procesamiento internas dentro de los diferentes cores. El consumo de memoria ha sido reducido al mínimo, almacenando únicamente un pequeño conjunto de bloques y las tablas precalculadas (por ejemplo cosenos) cuyos cálculos se repiten a lo largo del proceso de reconstrucción. Además, las operaciones de entrada/salida se han optimizado para reducir latencias de disco.

Este programa se ejecuta desde la línea de comandos, recibiendo la serie de proyecciones en formato .MRC así como un fichero con extensión .TLT que contiene los ángulos de las distintas proyecciones (una por línea) y genera como salida la reconstrucción en formato .MRC. Además permite seleccionar el algoritmo de reconstrucción que se va a utilizar así como introducir parámetros propios de cada algoritmo.

Uno de los objetivos de este proyecto es el de realizar la integración de este software dentro de TomoJ, por lo que se ha desarrollado una interfaz de usuario para introducir los parámetros que se han considerado necesarios para realizar reconstrucciones tomográficas de calidad. La integración de este programa con TomoJ se explicará detalladamente en el capítulo [2.5.2].

2.5 Integración realizada

Como ya se ha introducido a lo largo del proyecto, el objetivo de éste es el de aumentar la funcionalidad y el rendimiento del programa TomoJ a través de algoritmos y funciones implementadas en el paquete de Xmipp.

Durante el desarrollo de este capítulo trataremos cada uno de estos puntos, por qué se ha considerado que es interesante integrarlos en TomoJ, qué ofrecen al usuario y qué resultados ofrecen.

La forma en la que han sido implementadas estas integraciones serán explicadas detalladamente en el capítulo [3]

2.5.1 Algoritmos de preprocesado

Tal y como vimos en el capítulo [2.3.2] el programa TomoJ implementa los filtrados *hot spot removal, substract background* y *bandpass filter*, pero estos no son los únicos filtros que se pueden aplicar a través de TomoJ, ya que como hemos mencionado anteriormente este se trata de un plugin de ImageJ y, como tal, para poder ejecutarse tiene que estar arrancado el programa ImageJ previamente.

Este programa permite realizar otro tipo de filtrados tales como el filtro gaussiano (*gaussian blur)*, filtro mediano *(median)* entre otros. Hemos creído conveniente integrar estos filtros en la interfaz de usuario de TomoJ con el fin de proporcionar al usuario mayor comodidad a la hora de realizar el pre-procesado de la serie proporcionándole todos los tipos de filtrado que este pueda utilizar en una única interfaz.

Además de estos filtrados también se ha creído conveniente integrar herramientas como el recortado de imágenes o algoritmos más la mejora del contraste de la imagen como es el caso de la corrección gamma.

Una vez realizada la integración de estas funciones, la pestaña *preprocesing* de la interfaz de usuario de TomoJ será la siguiente (Figura 2.5.1.1).

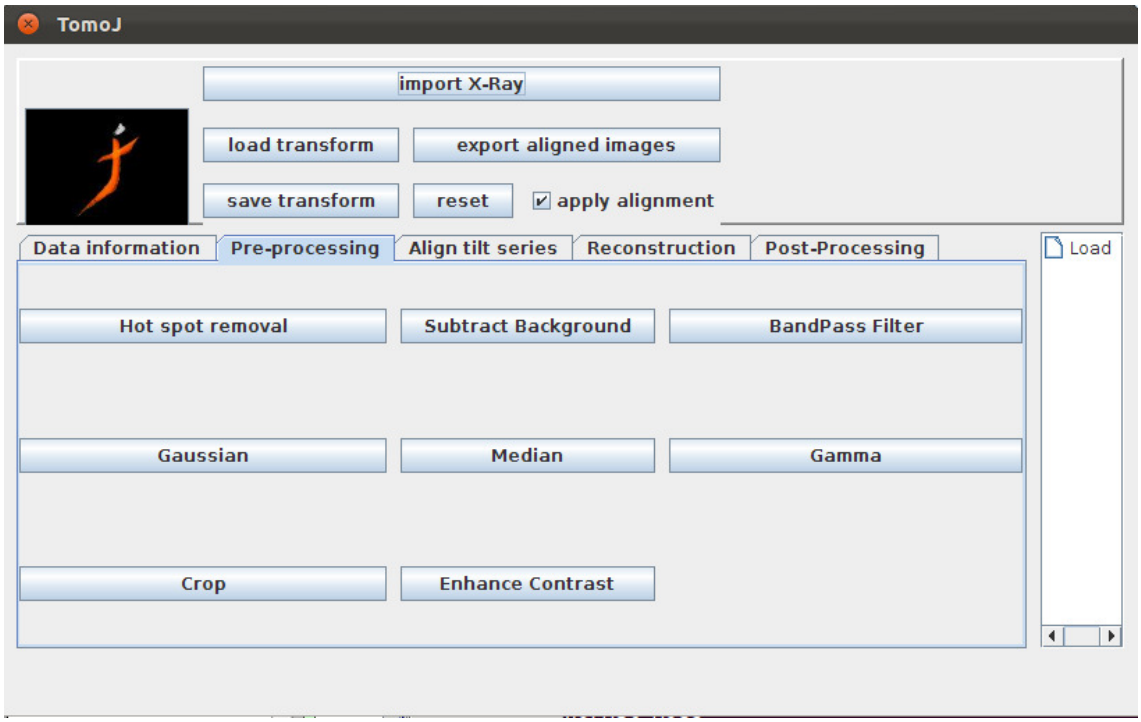


Fig. 2.5.1.1: Interfaz preprocesamiento TomoJ una vez integrados los cambios

A partir del nuevo botón *gaussian* el usuario puede aplicar un filtro gaussiano a la serie de proyecciones o a una proyección concreta. Este tipo de filtro realiza la

convolución entre la serie de proyecciones y una gaussiana para suavizar la serie. Para ello recibe el parámetro Sigma (Radius) que se trata de la desviación típica de la función gaussiana y que el usuario deberá introducir.

Como mencionamos en el capítulo [2.2] el código de ImageJ es abierto, y por tanto totalmente reutilizable a la hora de desarrollar nuevos plugins. Por esta razón, para realizar la implementación de estos filtros se ha utilizado la librería *IJ de ImageJ,* que proporciona un conjunto de funciones al desarrollador entre las que se encuentran métodos necesarios para realizar este tipo de filtrados.

Al ejecutarse directamente el comando con los métodos de la librería IJ, la ejecución es equivalente a la que obtendríamos en el caso de pinchar sobre la interfaz ImageJ en la pestaña filters->Gaussian Blur... por lo que el control del proceso pasa a ser de nuevo de ImageJ, que proporcionará al usuario las ventanas necesarias para introducir el valor del parámetro Sigma (Radius).

Además, para este tipo de operaciones, ImageJ ofrece al usuario la posibilidad de realizar el filtrado sobre todas las proyecciones o no (Figura 2.5.1.3). En la figura 2.5.1.2 podemos observar un ejemplo del uso de esta aplicación.

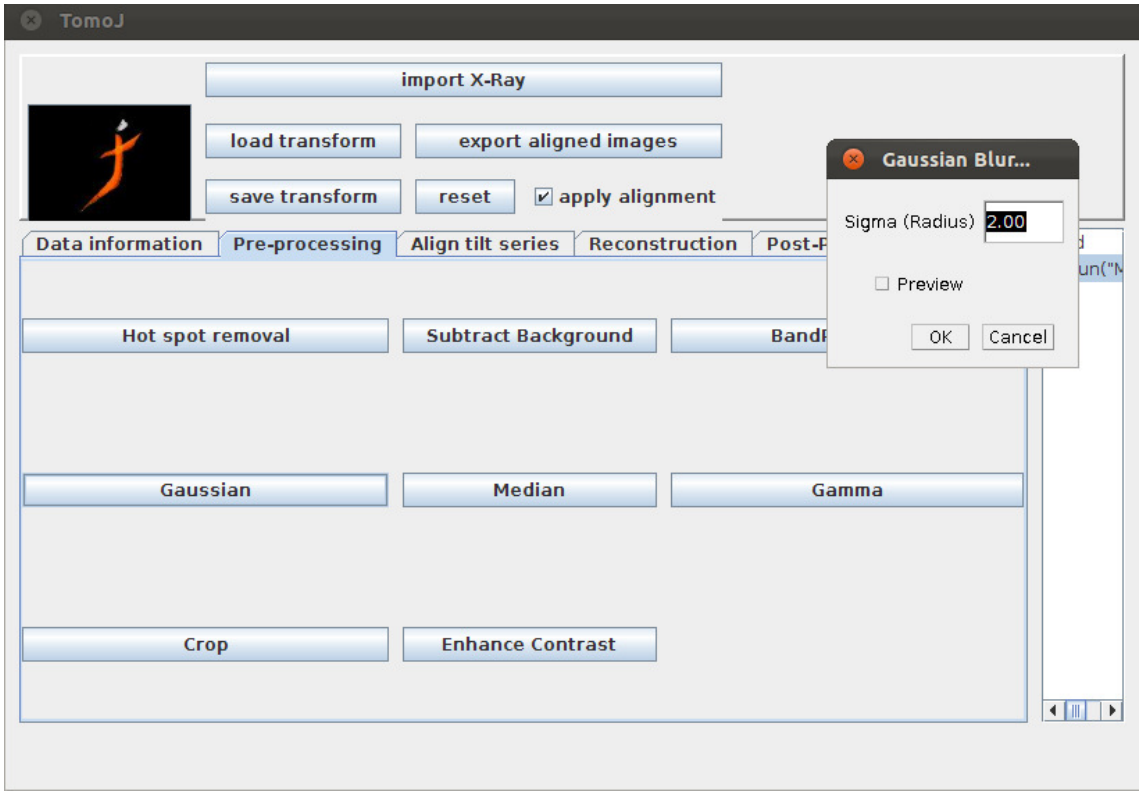


Fig. 2.5.1.2: Interfaz proporcionada por ImageJ para introducir el valor del parámetro Sigma(Radius). En el que introduciremos el valor 4.00 para visualizar de una manera clara los cambios producidos por el filtro.

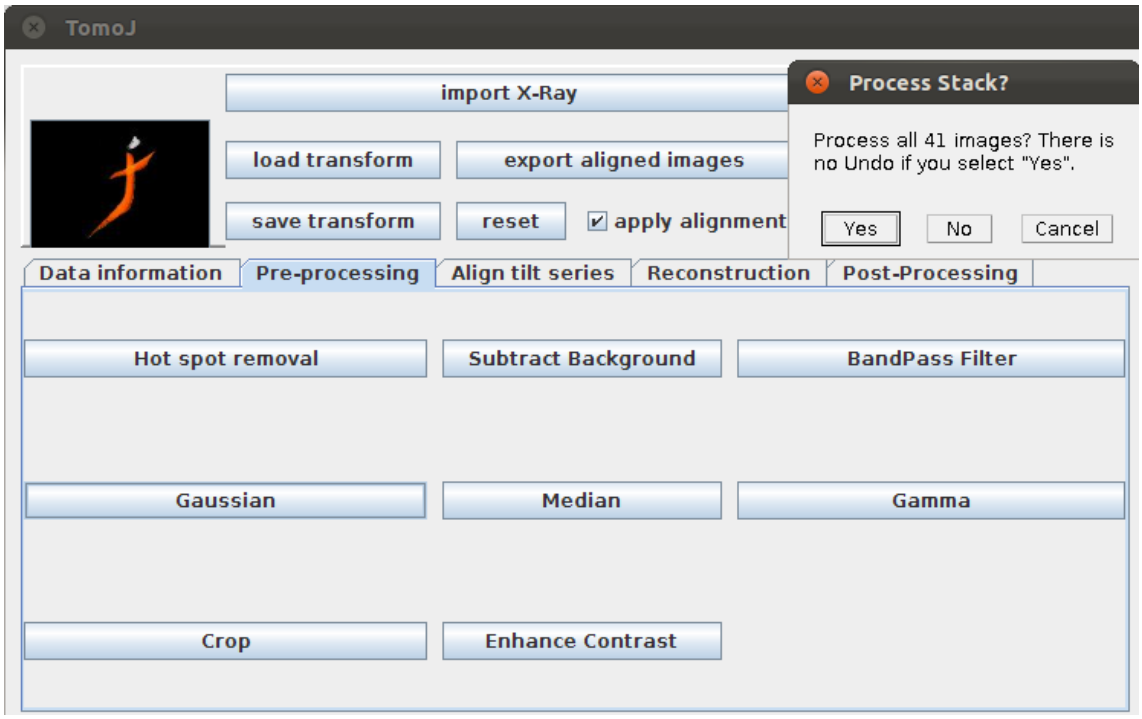


Fig. 2.5.1.3: Interfaz proporcionada por ImageJ verificar si se desea realizar el filtrado sobre todas las proyecciones

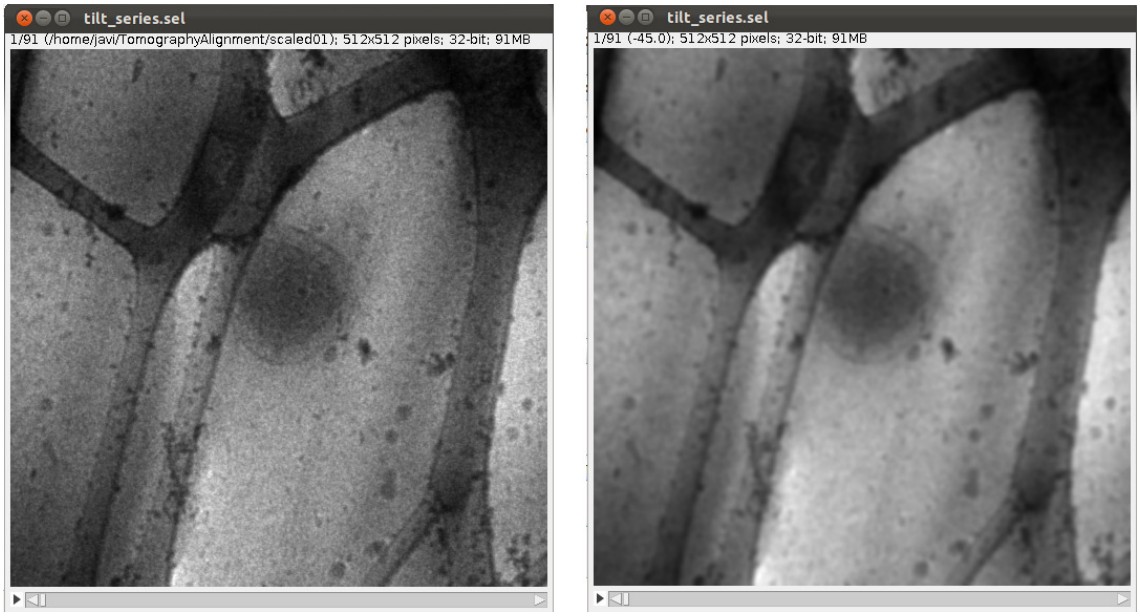


Fig. 2.5.1.4: A la izquierda imagen original, a la derecha imagen tras aplicar un filtro gaussiano con Sigma=2.0

Como podemos observar en la figura [2.5.1.4], el efecto de aplicar un filtro gaussiano es el de añadir detalles de baja frecuencia, por lo que puede producir un efecto nebuloso en la imagen. El filtro elimina especialmente el ruido gaussiano producido por la digitalización de las imágenes actuando sobre cada píxel de la capa activa o selección, estableciendo su valor como el promedio entre los valores de todos los píxeles incluidos el valor de Sigma (Radius) establecido. Un valor alto producirá un mayor efecto de desenfoque.

El segundo filtro que se ha añadido al programa TomoJ es el filtro de la mediana, el cual se puede aplicar a través del botón *median*, este tipo de filtros elimina especialmente el ruido aleatorio y el ruido sal y pimienta calculando los píxeles de la nueva imagen a partir de a mediana del conjunto de píxeles del entorno de vecindad del píxel correspondiente a la imagen origen. De esta forma se homogeneízan los píxeles de intensidad muy diferente con respecto a la de los vecinos. Este entorno de vecindad se

obtiene a partir del parámetro Radius, valor que deberá de introducir el usuario, este valor deberá de ser bajo para eliminar el ruido aleatorio.

Al igual que en el caso del filtro gaussiano, para realizar la implementación de este filtro se ha utilizado la librería de clases de ImageJ *IJ,* por lo que de nuevo el control del proceso vuelve a ImageJ y este es el que proporciona las ventanas de recogida de datos tal y como ocurría con el filtro gaussiano (figuras 2.5.1.5 y 2.1.5.6)

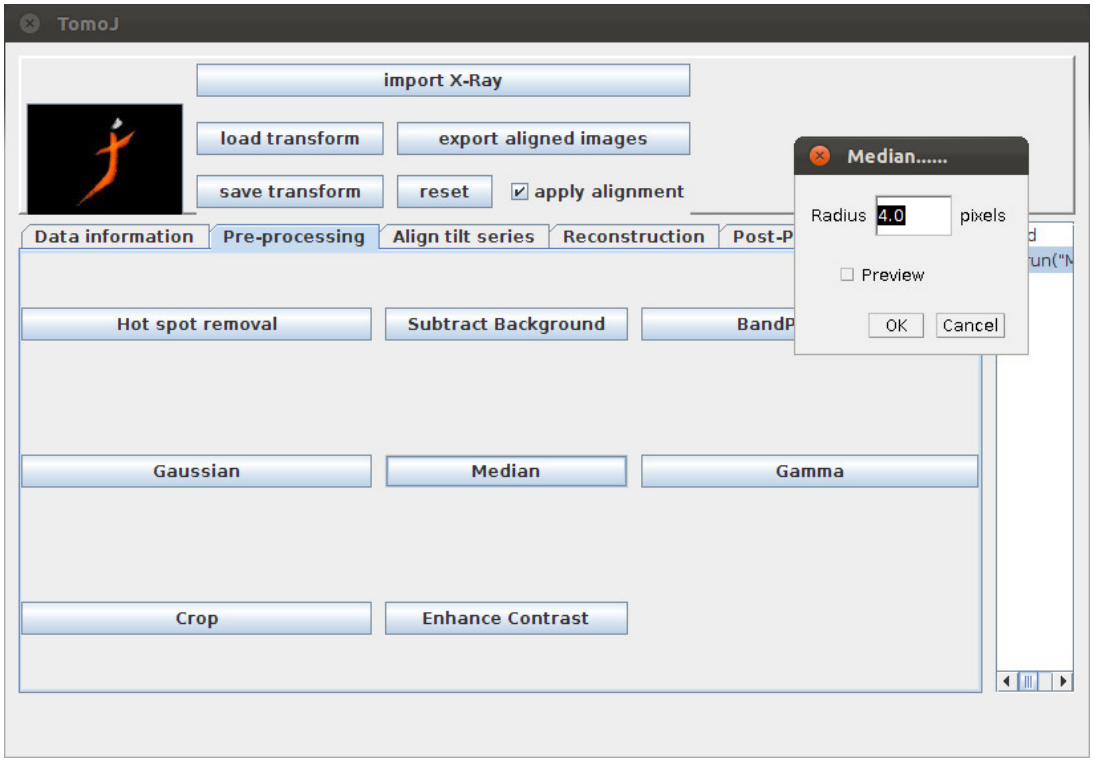


Figura 2.5.1.5: Interfaz de ImageJ sobre TomoJ para introducir el valor del parámetro Radius (radio de vecindad)

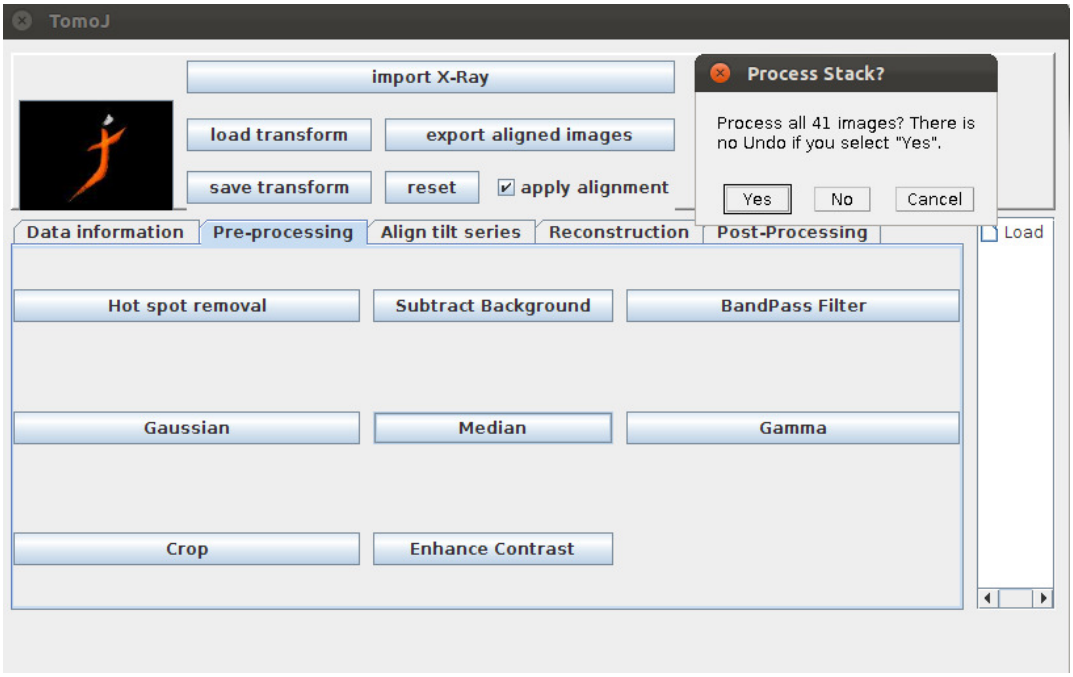


Figura 2.5.1.6: Al igual que en el filtro gaussiano se nos plantea la posibilidad de realizar el filtrado a toda la serie o no

Si realizamos el filtrado con los parámetros introducidos anteriormente obtenemos el resultado ofrecido por la figura [2.5.1.7]

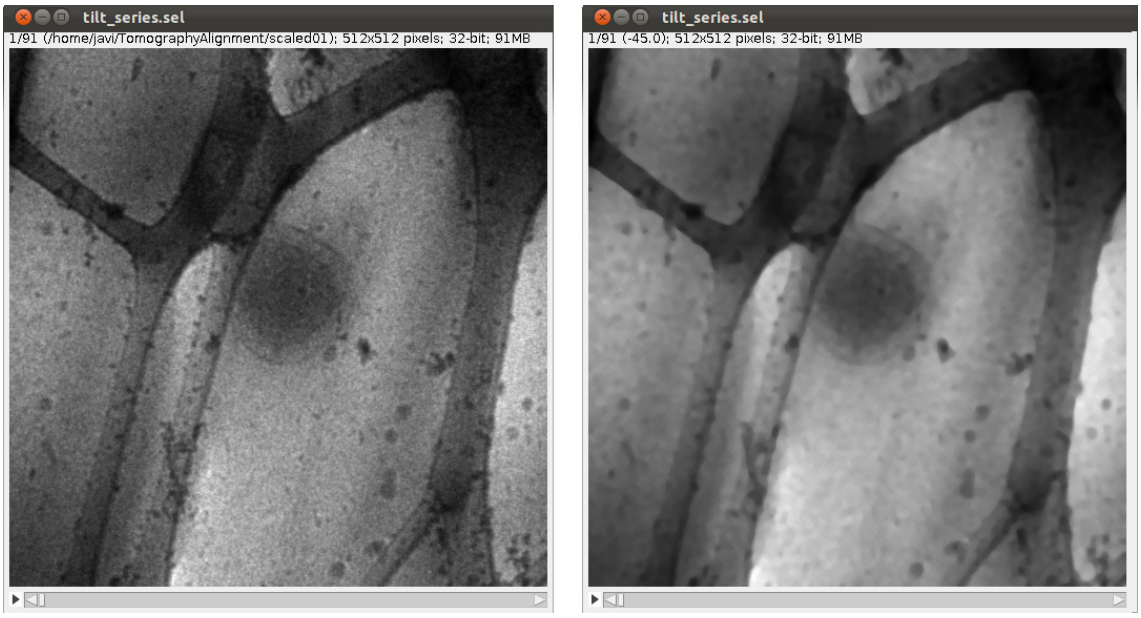


Figura 2.5.1.7: A la izquierda la imagen original, a la izquierda la imagen después de ser filtrada con un radio de 4 píxeles.

El tercer algoritmo de pre-procesamiento implementado para TomoJ ha sido la corrección gamma, disponible a través del botón de la interfaz *gamma*. Este algoritmo de procesamiento de imágenes es una forma especial de aumento de contraste diseñada para mejorar el contraste en áreas muy claras o muy oscuras. Esto se logra modificando los valores medios, particularmente los medios-bajos, sin afectar el blanco (255) ni el negro (0). Puede utilizarse para mejorar el aspecto de una imagen, o para compensar el rendimiento de diferentes dispositivos frente a una imagen. En este contexto, es necesario definir el concepto de contraste como el grado de diferencia entre los componentes más oscuros y los más claros en una imagen. Una imagen tiene contraste pobre si contiene solamente transiciones bruscas entre el blanco y el negro, o contiene valores dentro de una gama estrecha (una imagen cuyos valores se extienden entre 100 a 140 tendría contraste pobre). Una imagen tiene un buen contraste si se compone de una amplia gama de valores desde el negro al blanco. La amplitud de la escala de intensidad usada por una imagen se llama rango dinámico. Una imagen con un buen contraste tendrá un mayor rango dinámico.

La corrección gamma es el resultado de aplicar a la serie de proyecciones la función de transformación gamma, la cual viene dada por la siguiente expresión:

$$d = k \cdot r^{g}$$

Donde *k* es una constante y *g* es el valor *gamma*, que introducirá el usuario a partir de la interfaz.

Para valores de gamma superiores a la unidad, hay una gran corrección en el contraste para valores pequeños del color de entrada mientras que se realiza una pequeña corrección en el contraste para valores grandes. El brillo aumenta más para valores intermedios del color de entrada. Por lo tanto se producirá un aclarado de la imagen y aumentará el contraste en las áreas más oscura.

Para valores de gamma inferiores a la unidad se producirá una pequeña corrección en el contraste para valores pequeños del color de entrada mientras que se producirá una gran corrección en el contraste para valores grandes. El brillo disminuirá más para valores intermedios del color de entrada. En resumen, este rango de valores oscurece y enfatiza el contraste en las áreas más claras.

En el caso en el que gamma sea igual a la unidad se aplicará la función identidad (d=r) y por lo tanto será equivalente a la señal original.

Al igual que los filtros anteriores, esta corrección también está implementada en ImageJ, por lo que utilizaremos la librería *IJ* para llevar a cabo su implementación (figuras 2.5.1.8 y 2.5.1.9)

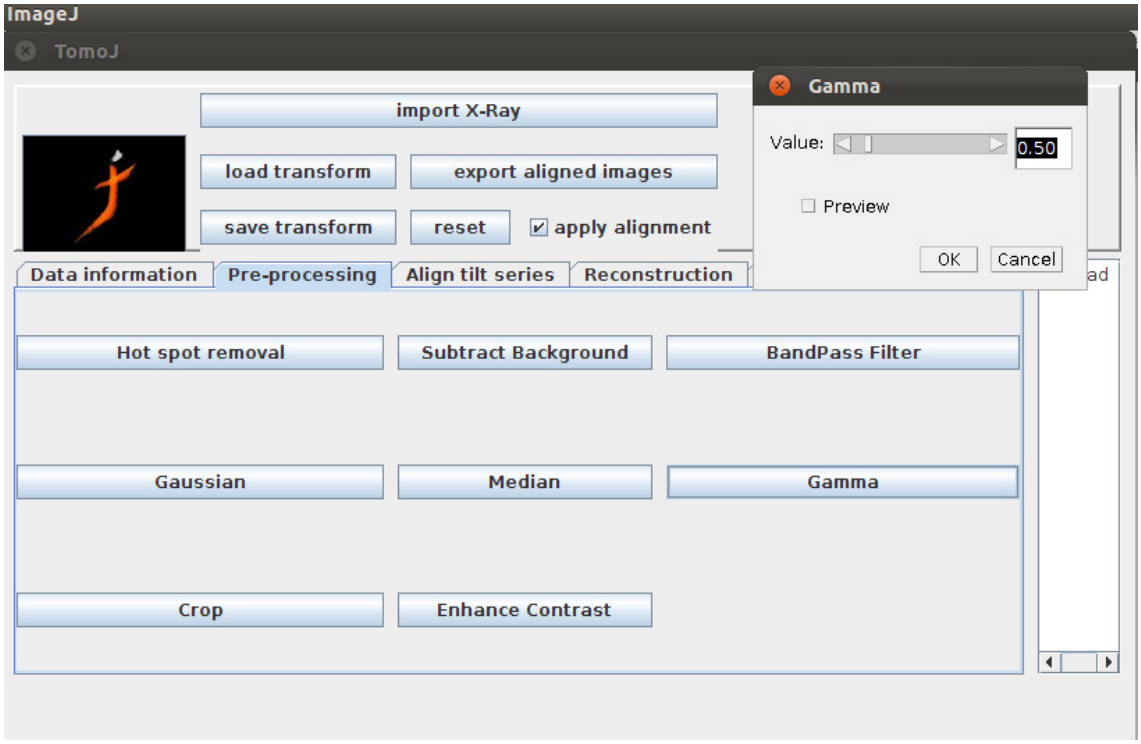


Figura 2.5.1.8: Interfaz gráfica proporcionada por ImageJ para introducir el valor del parámetro *gamma*.

En las figuras 2.5.1.9, 2.5.1.10 y 2.5.1.11 podemos observar el resultado de la imagen tras aplicar la corrección gamma sobre la serie de proyecciones con valores de gamma de 0.5, 1 y 1.5 respectivamente.

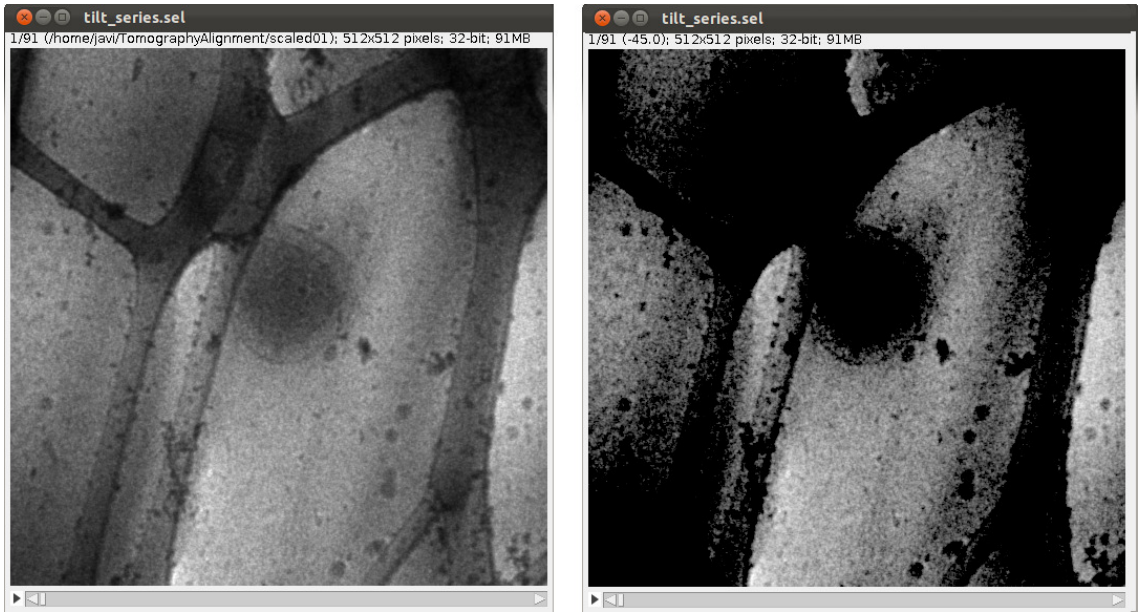


Figura 2.5.1.9: A la izquierda la imagen original, a la derecha la imagen tras realizar la correción gamma con gamma=0,5

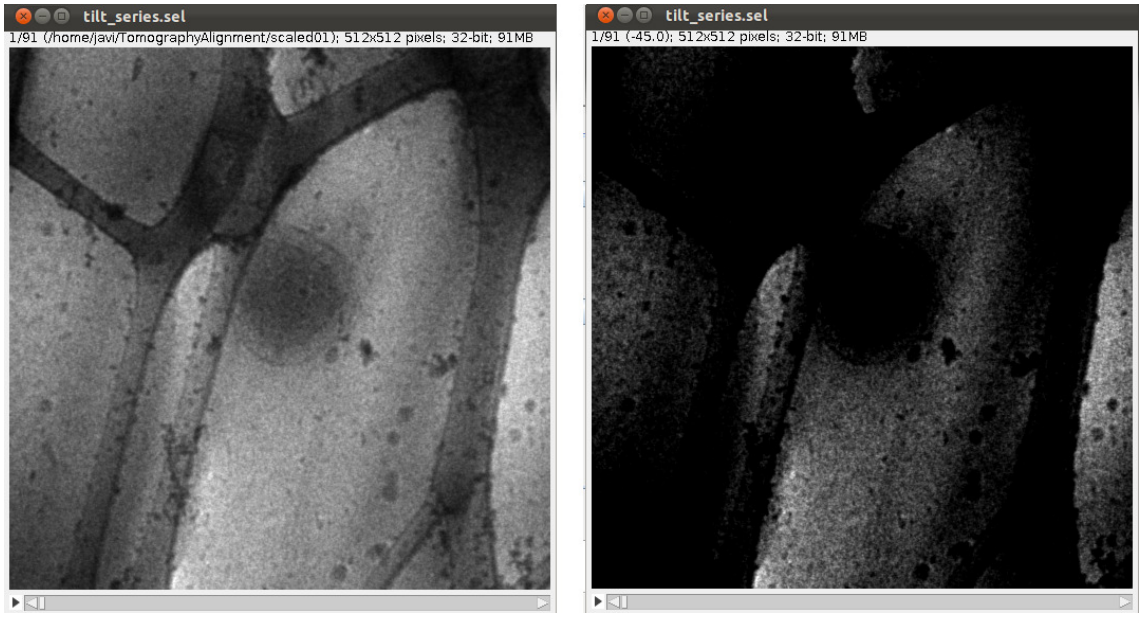


Figura 2.5.1.10: A la izquierda la imagen original, a la derecha la imagen tras realizar la correción gamma con gamma=1,5

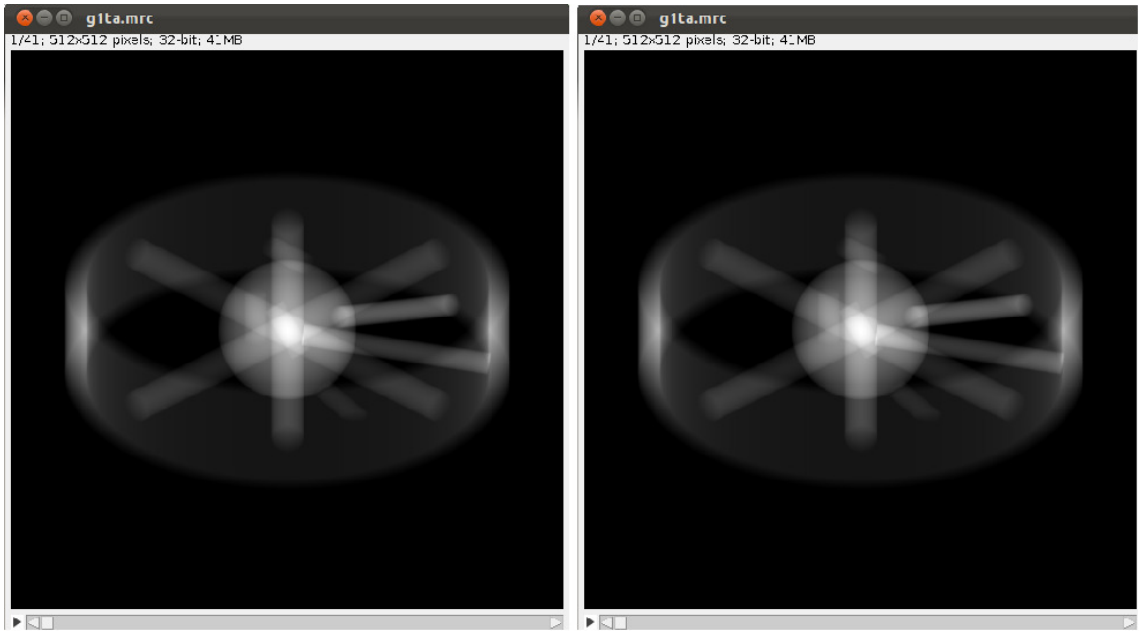


Figura 2.5.1.11: A la izquierda la imagen original, a la derecha la imagen tras realizar la correción gamma con gamma=1.

El último algoritmo de pre-procesado que se ha implementado ha sido *Contrast Enhance*, que permite realizar mejoras en el contraste de la serie de proyecciones utilizando técnicas de igualación de histograma (*histogram equalization)* ó estiramiento de histograma *(histogram stretching)*. Estas técnicas de modelado de histograma proporcionan un método sofisticado para modificar el rango dinámico y el contraste de una imagen mediante la modificación de dicha imagen de tal forma que el histograma de su intensidad tenga la forma deseada.

Las técnicas estiramiento de histograma (también llamadas técnicas de normalización de histogramas) mejoran el contraste de una imagen por el estiramiento de los valores de intensidad para abarcar un intervalo deseado. Se diferencia de las técnicas de igualación de histograma en que sólo es posible aplicar una función de

escalado lineal a los valores de píxel de la imagen, haciendo que estas técnicas permitan realizar mejoras de contraste más pobres que las obtenidas con técnicas de igualación de histogramas.

Por su parte, las técnicas de igualación de histogramas permiten realizar mejoras en el contraste de la imagen, utilizando para ello funciones de transferencia no lineales y no monotónicas para mapear los valores de intensidad de cada píxel en la imagen original y la que se va a generar.

Al igual que el resto de algoritmos de pre-procesado implementados, *Enhance Contrast* es una técnica implementada en ImageJ, por lo que podemos utilizar la librería IJ para realizar la mejora de contraste a través de este programa.

Para ejecutar este algoritmo, ImageJ proporciona el diálogo de recogida de datos que muestra la figura [2.5.1.12]

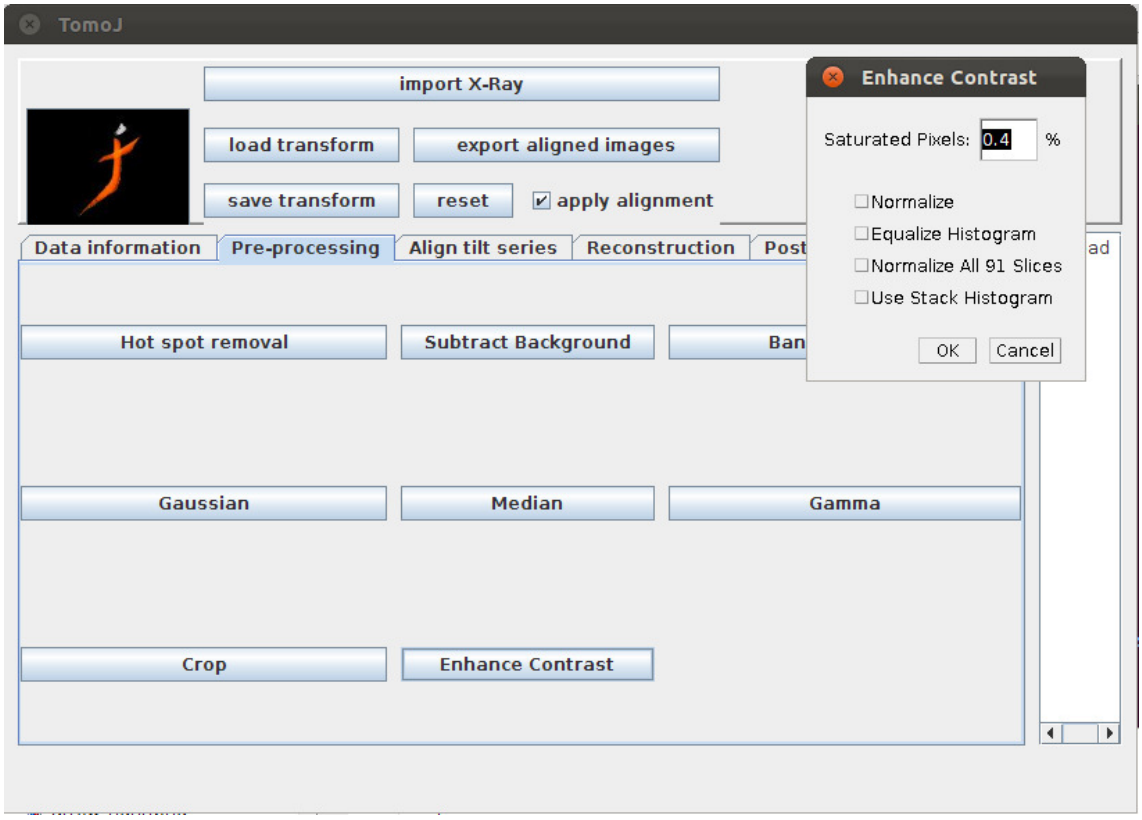


Fig. 2.5.1.12: Diálogo proporcionado por ImageJ para recoger las opciones con las que se ejecuta el algoritmo de *Enhance Contrast*

Donde el parámetro *Satured pixeles* determina el número de píxeles de la imagen que se puede saturar, por lo que cuanto más alto sea este valor más aumentará el contraste realizado. Este valor debe ser siempre mayor que cero para obtener un histograma satisfactorio. El checkbox *normalize* permite que ImageJ recalcule los valores de los píxeles de la imagen hasta que su rango sea el máximo posible para cada tipo de datos (0-255 para imágenes de 8-bits y 0-65535 para imágenes de 16-bits). Habilitando la opción *Equalize histogram* ImageJ realizará la mejora de contraste utilizando métodos de *histogram equalization,* método comentado anteriormente, y que pasará por alto el valor de *satured pixeles* para utilizar un algoritmo de *histogram equalization* estándar. En el caso de no estar habilitada esta opción se utilizarán técnicas de igualación de histogramas.

A continuación se mostrará un ejemplo de aplicación de esta mejora de contraste utilizando cada uno de los métodos expuestos:

Fig. 2.5.1.13

Como se ha ido mencionando a lo largo del capítulo, para realizar la integración de estos algoritmos se ha utilizado la librería de clases *IJ* de ImageJ, por lo que la implementación ha resultado relativamente sencilla ya que al ejecutar el comando correspondiente el control del proceso es tomado por ImageJ, que se encargará de proporcionar al usuario las ventanas emergentes necesarias para realizar la entrada de parámetros y ejecutar el comando indicado. Toda la parte relacionada con la implementación de la integración de estas funciones se verá detalladamente en el capítulo [3.1].

Es muy importante recalcar en este punto una de las ideas principales en las que se basaba en software *Xmipp_TomoJ* y que se ha decidido introducir en el software TomoJ que, aunque el alumno durante este proyecto no haya realizado esta integración sí que ha participado activamente en ella en este contexto. Esta idea es la permitir al usuario llevar un seguimiento de todas las funciones que haya utilizado a lo largo de sus interacciones con el programa, desde la carga de la serie de proyecciones hasta realiizar su reconstrucción, permitiéndole volver sobre cualquiera de los pasos que haya realizado en cualquier momento. Este concepto de *workflow* proporcionará al usuario flexibilidad y facilidad para deshacer los cambios que han producido efectos no deseados sobre las proyecciones. Para ello, TomoJ almacenará en memoria una instancia del objeto que representa la serie de proyecciones aumentando el uso de esta de forma considerable, creando una estructura jerárquica con los diferentes estados por los que pasará la serie de proyecciones.

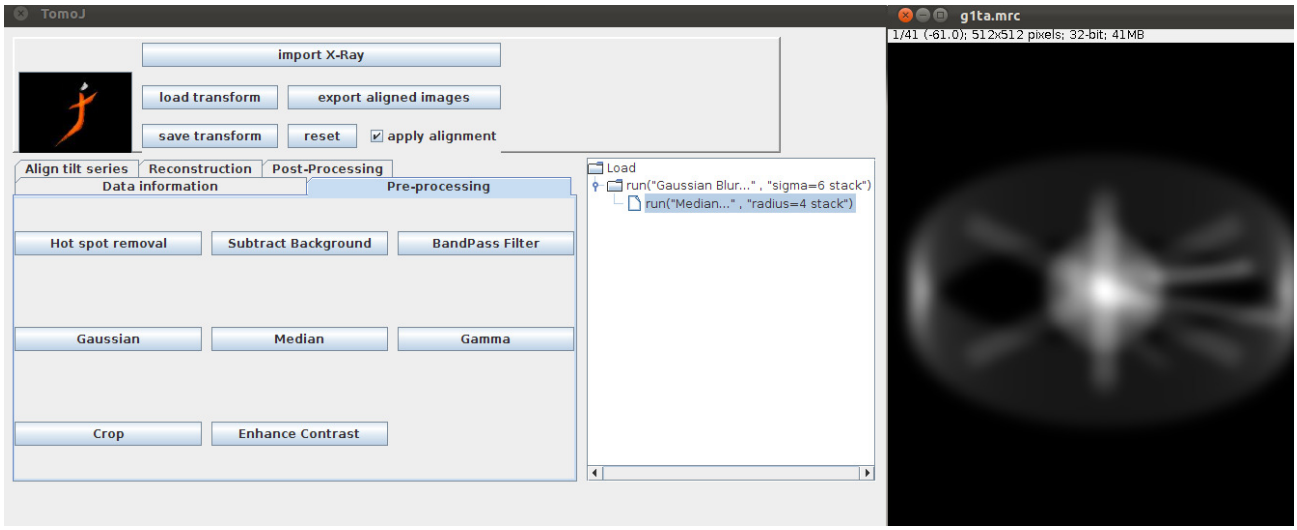


Fig. 2.5.1.14 En el panel derecho sobre fondo blanco podemos apreciar cómo a medida que filtramos la serie con los diferentes algoritmos se genera la estructura jerárquica que en el fututo (cuando esté implementado) permitirá al usuario volver al paso que convenga oportuno

La importancia que acarrea la integración de esta idea en el software TomoJ en el contexto de los algoritmos de pre-procesamiento reside en que es necesario conocer los parámetros de entrada que haya introducido el usuario (la desviación típica en el caso de el filtrado gaussiano, el radio de vecindad en el caso del filtrado de la mediana y el valor de gamma en el caso de la corrección gamma), ya que este valor será el que distinga los diferentes pasos que formarán parte de la estructura arbórea que generará el workflow. Sin embargo, al ejecutar el comando de ImageJ éste es el que proporciona la recogida de estos parámetros y, por lo tanto, como desarrolladores de TomoJ los desconocemos. Para solucionar este problema se ha implementado un método que permite capturar los diálogos que corren dentro de un terminal para obtener el valor de dichos parámetros. Toda esta implementación se verá detalladamente en la sección [3.1].

Por último, además de estos algoritmos de pre-procesamiento se ha implementado a su vez la herramienta *crop*, disponible también en ImageJ y que permite al usuario recortar la imagen para obtener una pequeña región de interés. Esta herramienta es utilizada en casos en los que el tamaño de la imagen supera la memoria disponible del ordenador en el que se está ejecutando el programa.

Para utilizar esta herramienta es necesario seleccionar un área rectangular a través de ImageJ que será la que forme la nueva imagen y pulsar el botón *Crop.* En las figuras [2.5.1.15] y [2.5.1.16] podemos observar cómo se realiza el proceso de recortar una proyección.

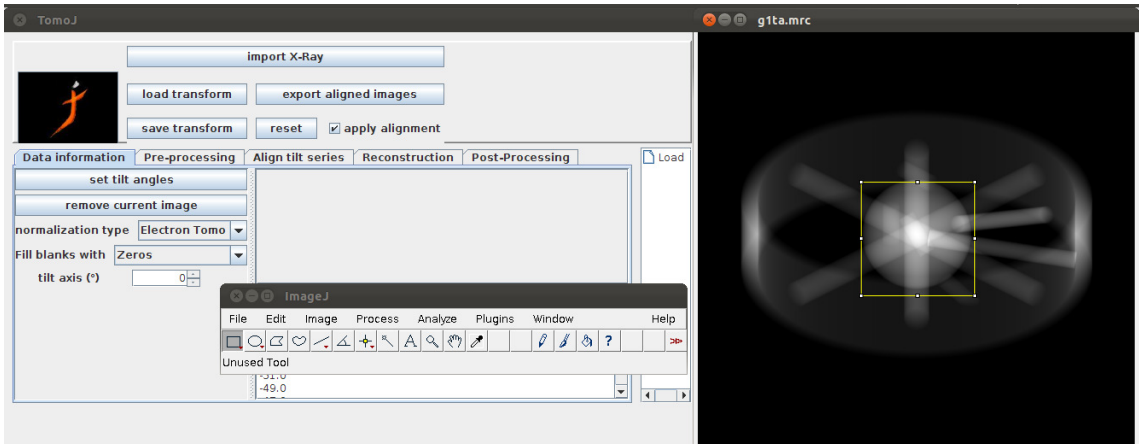


Fig. 2.5.1.15 Seleccionamos el área que deseamos recortara trvés de ImageJ

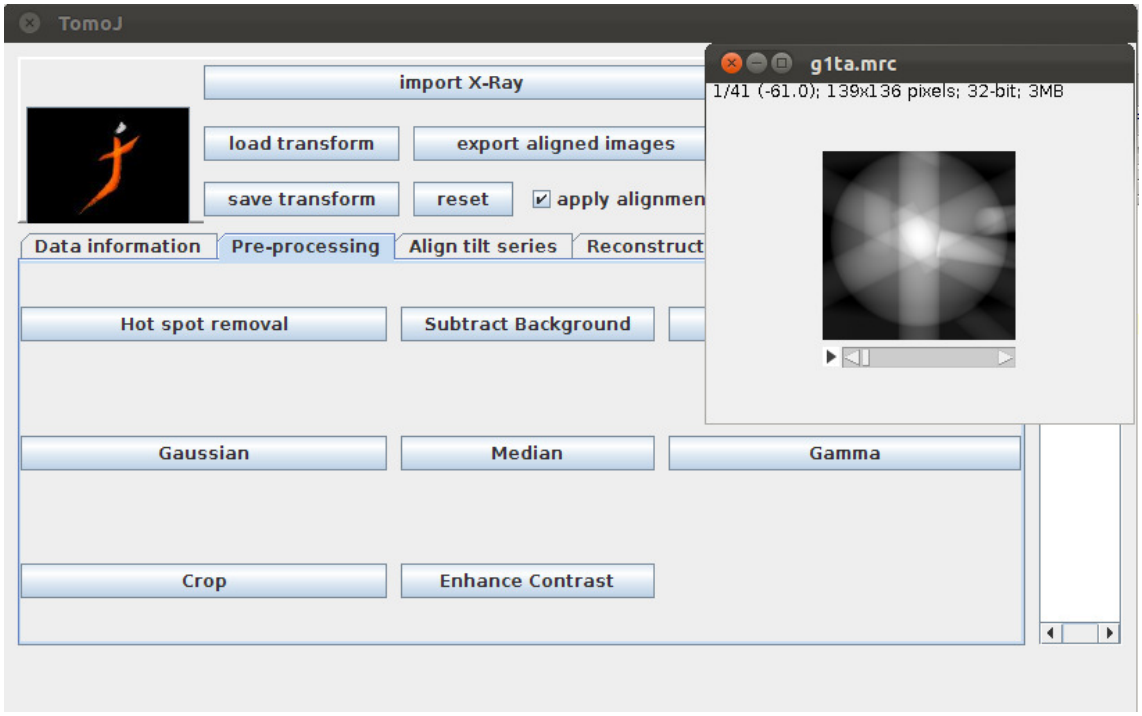


Fig. 2.5.1.16 Imagen recortada a partir de la herramienta crop.

2.5.2 Integración Tomo3D

Tal y como introdujimos en el capítulo [2.4], Tomo3D es un programa que permite realizar reconstrucciones tridimensionales a partir de una serie de proyecciones bien alineada aprovechando el poder computacional de la tecnología multicore, permitiendo obtener tomografías de calidad con un tiempo computacional mucho menor que los algoritmos convencionales.

Tomo3D permite realizar reconstrucciones a partir de los algoritmos *WBP* y *SIRT*, empleando un tiempo del orden de segundos para el primer algoritmo y del orden de pocos minutos para el segundo. A lo largo de este capítulo veremos la interfaz que se ha creado para ejecutar este programa desde TomoJ, así como los resultados obtenidos a partir de su ejecución, dejando su implementación para el capítulo [3.3].

Como algoritmo de reconstrucción, el programa Tomo3D tiene cavidad en la pestaña de TomoJ *reconstruction*, que, como vimos en el capítulo [2.3.2] ofrece al usuario la capacidad de ejecutar los diferentes algoritmos de reconstrucción. Tal y como vemos en la figura [2.5.2.1] han sido añadidas a la lista de algoritmos de reconstrucción las opciones *WBP (tomo3D)* y *SIRT (tomo3D)*.

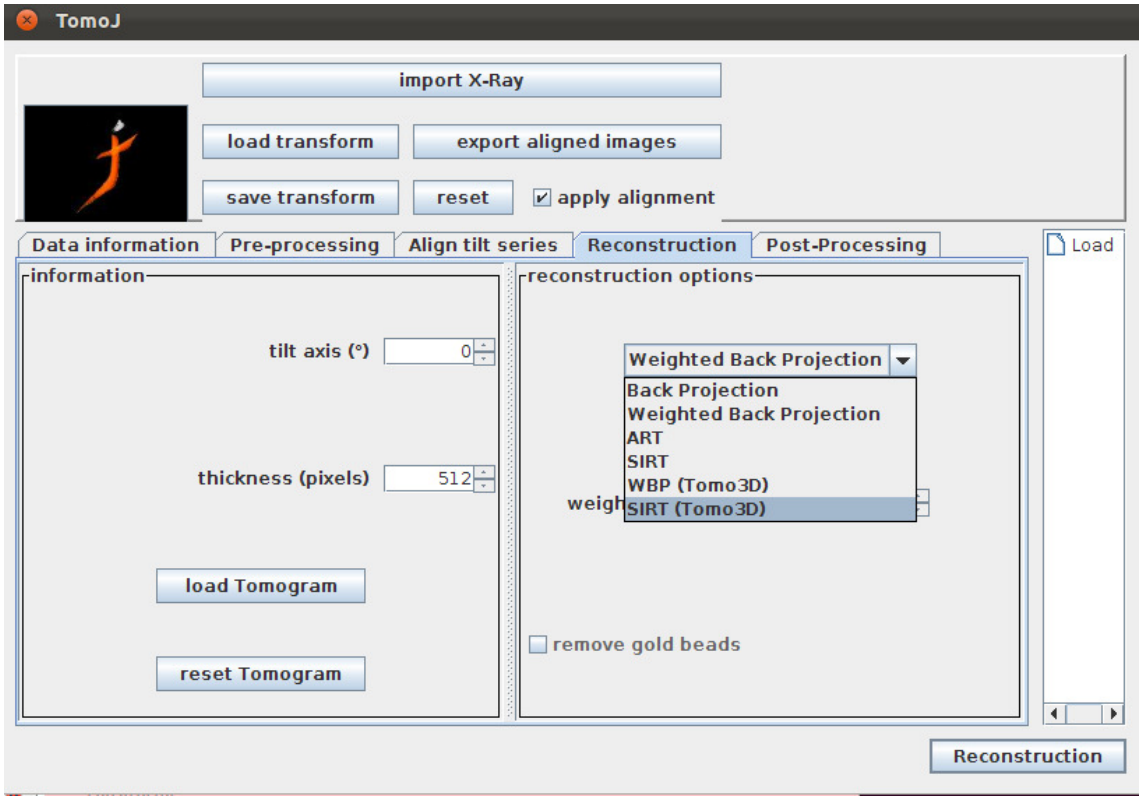


Figura 2.5.2.1: Pestaña de reconstrucción de la interfaz de usuario de TomoJ una vez integrado el programa Tomo3D

Tomo3D consta de tres ficheros binarios que se lanzarán desde la línea de comandos para ejecutar el proceso. Dependiendo del procesador de la máquina que se esté utilizando se utilizará el fichero binario adecuado, tal y como se adelantó en el capítulo [2.4]. El funcionamiento básico de este programa es el recibir la serie de proyecciones en formato MRC así como un fichero de texto IMOD (extensión TLT) que contenga los ángulos de inclinación de cada una de las imágenes que contiene la serie en el mismo orden (una línea por proyección), generando como salida el tomograma de la serie en formato MRC.

El uso general de este programa es por tanto el siguiente:

tomo3D.ia32 -a tilt_angle_file.tlt -i tilt_series.mrc

Donde *tomo3D.ia32* es el fichero binario que deben utilizar las máquinas de 32-bits, *tilt_angle_file.tlt* es el fichero que contiene los ángulos de inclinación de la serie y *tilt_series.mrc* es la serie de proyecciones.

Además, Tomo3D permite introducir otra serie de opciones como parámetros, por ejemplo, este programa utiliza por defecto el algoritmo *WBP* para reconstruir, ofreciendo la posibilidad de introducir a través del flag *-m* la frecuencia del filtro de Hamming que se aplica en este algoritmo. Por defecto, en *WBP,* el programa aplica un filtro Ramp junto con un filtro de Hamming. Este último permite la atenuación de las frecuencias altas, que, habitualmente contienen mucho ruido. Este parámetro permite la creación de una frecuencia inicial a partir de la cual el filtro Hamming no se aplica. Este valor debe estar en el rango [0,0.5], donde 0,5 representa el valor de la frecuencia de Nyquist y, con el cuál no se aplicaría el filtro de Hamming. Por lo que para el caso del siguiente ejemplo tomo3D utilizaría el algoritmo *WBP* con frecuencia inicial de 0,23 para el filtro de Hamming.

tomo3D.ia32 -a tilt_angle_file.tlt -i tilt_series.mrc -m 0.23

Si el usuario desea utilizar el algoritmo *SIRT* se utilizará el flag *-S.* Recordaremos en este punto que este algoritmo, al igual que *ART,* es un proceso iterativo, por lo que es posible a su vez indicar el número de iteraciones que el usuario desea realizar a través del flag *-l*. En el caso de no habilitar este flag se tomará un total de 40 iteraciones como valor por defecto. A continuación mostramos un ejemplo del uso del programa Tomo3D utilizando el algoritmo de reconstrucción *SIRT* y un total de 15 iteraciones

tomo3D.ia32 -a tilt_angle_file.tlt -i tilt_series.mrc -S -l 15

De entre todos los flags que permiten aumentar la funcionalidad de tomo3D a la hora de usarse, además de los ya mencionados, se ha decidido utilizar el flag *-c* que en el caso de que esté activado, restringe el volumen que se va a calcular. Este flag es utilizado habitualmente en el caso del uso de algoritmos *SIRT*, donde en cada iteración se asegura de que las densidades de los voxels del volumen que se calcula sean positivas, dejando a cero las que sean negativas.

Para permitir al usuario aplicar cada uno de estos flags y, en función del tipo de algoritmo de reconstrucción utilizado se han habilitado las siguientes interfaces de usuario que permitan habilitar estas funcionalidades. En la figura [2.5.2.2] podemos observar la interfaz ofrecida para utilizar el algoritmo *WBP* mientras que en la figura [2.5.2.3] observamos la interfaz ofrecida para utilizar el algoritmo *SIRT*. En el capítulo [3.3] explicaremos detalladamente cómo, a partir de los datos recogidos de la interfaz de usuario, el programa construye y ejecuta el comando que permite la utilización de este programa.

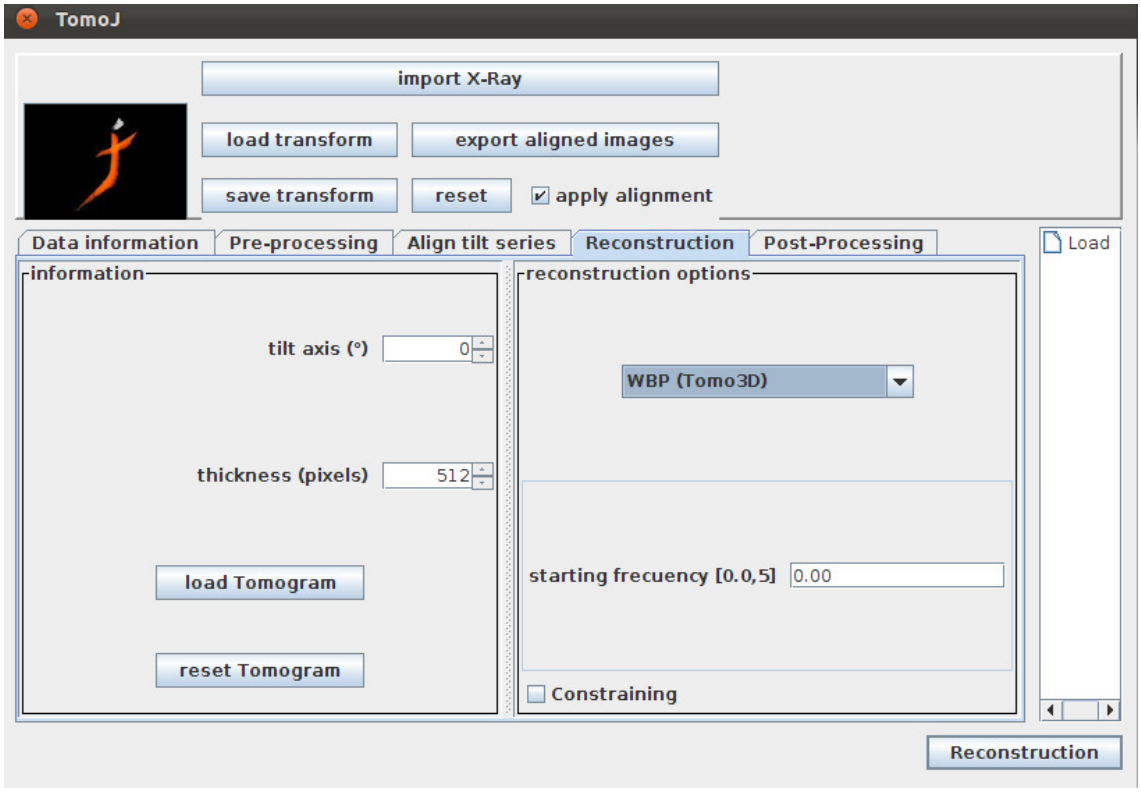


Figura 2.5.2.2. Visualización de los parámetros de entrada para el algoritmo WBP en el programa Tomo3D

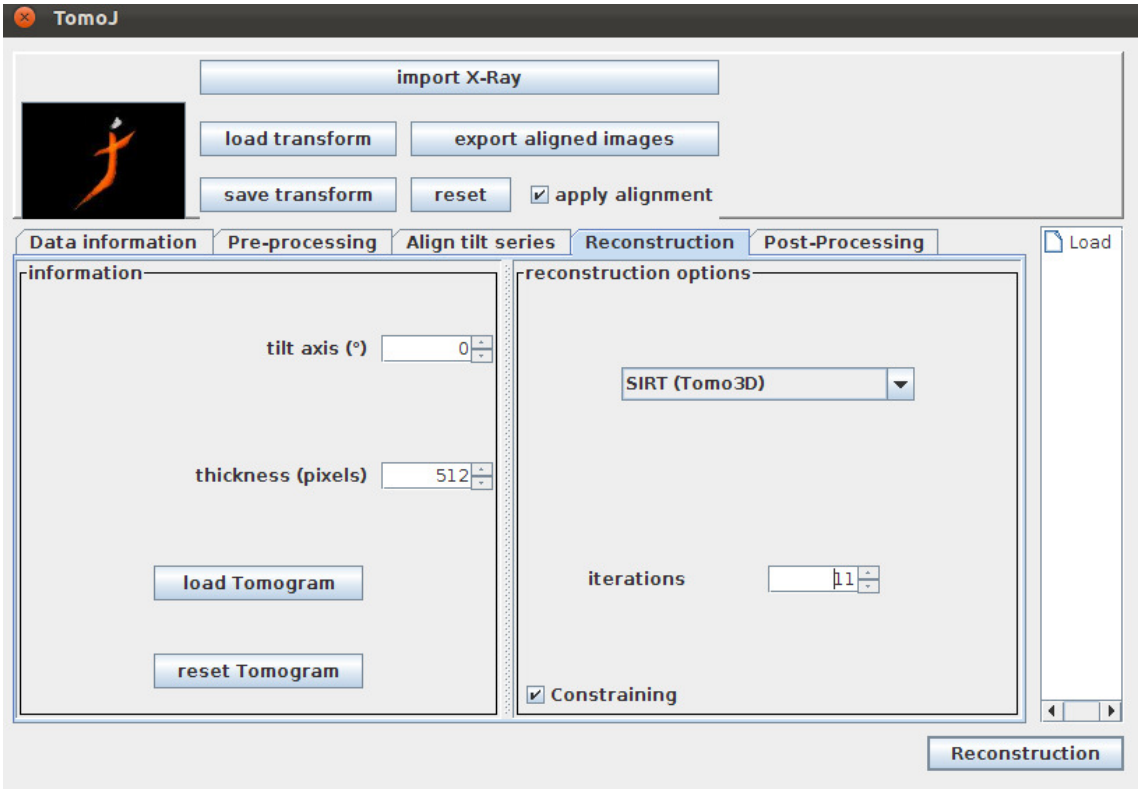


Fig. 2.5.2.3. Visualización de los parámetros de entrada para el algoritmo SIRT en el programa Tomo3D

Una vez introducidos los parámetros solicitados por la interfaz, a través del botón *Reconstruction* podemos obtener la reconstrucción de la serie. Como mencionamos anteriormente se tratará de un fichero en formato MRC que, será cargado en memoria y mostrado al usuario. En las figura 2.5.2.4 y 2.5.2.4 se muestra un ejemplo de la ejecución del programa para cada uno de los algoritmos de reconstrucción que permite utilizar Tomo3D. Debemos de tener en cuenta que, para el caso de este ejemplo la serie de proyecciones no ha sido alineada previamente, por lo que el resultado no será del todo correcto. En el capítulo [4] realizaremos una reconstrucción completa realizando los pasos de preprocesado y alineamiento.

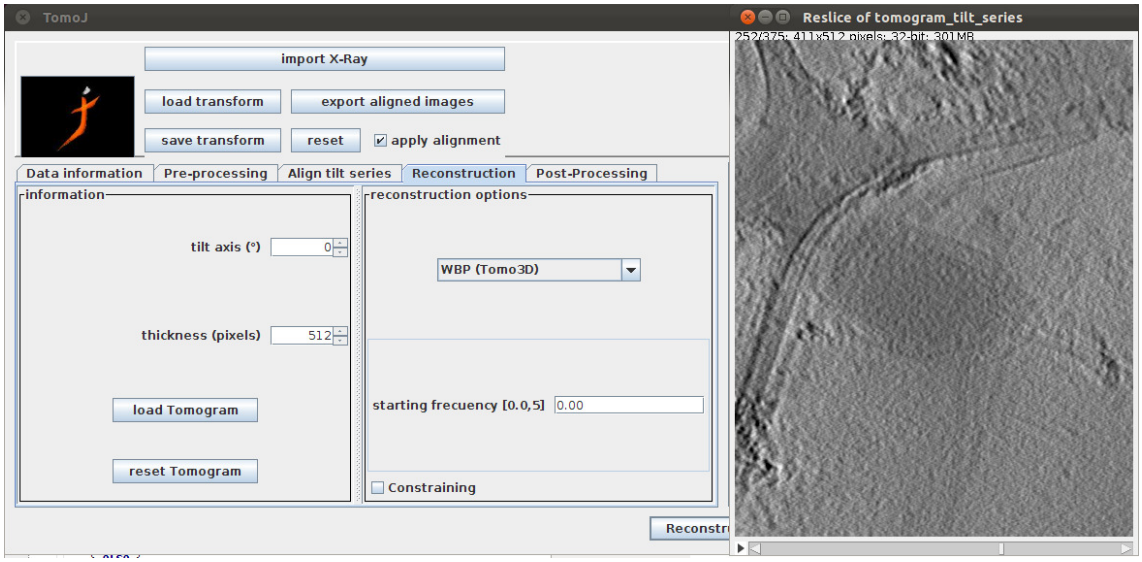


Fig. 2.5.2.4. Tomograma obtenido a través del programa Tomo3D para el algoritmo WBP con una frecuencia inicial de 0.00

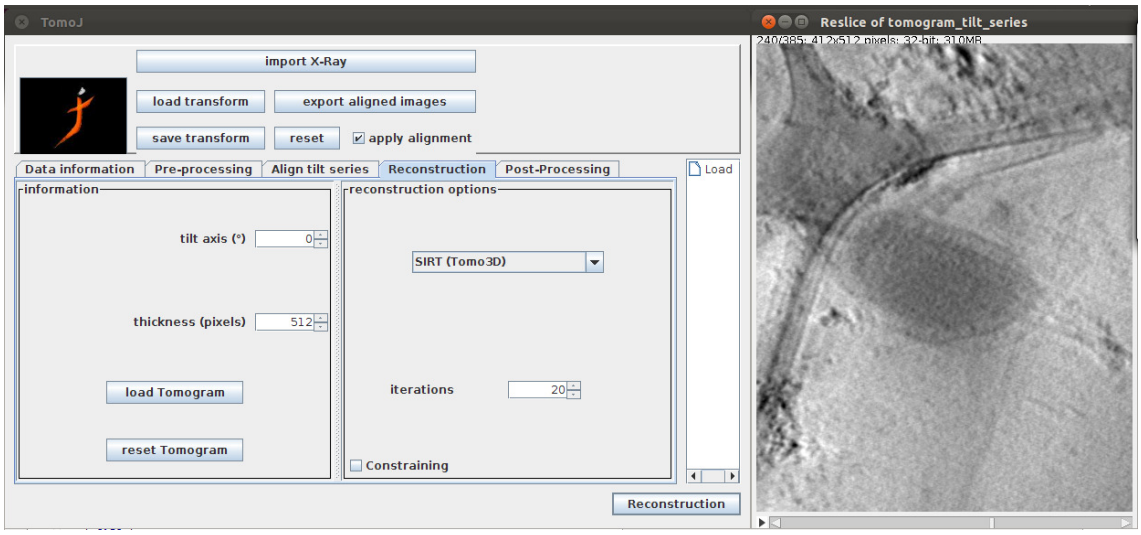


Fig. 2.5.2.4. Tomograma obtenido a través del programa Tomo3D para el algoritmo SIRT con 11 iteraciones

.

2.5.3 Importar imágenes de TomoX

Tal y como se presentó durante la enumeración de los objetivos que planteaba el proyecto, una de las funcionalidades con la que se iba a mejorar el software TomoJ era el de implementar la capacidad de importar imágenes relativas al campo de la tomografía de Rayos X, introducida en el capítulo [1.4], permitiendo al usuario tratarlas de la misma forma que en el caso de imágenes obtenidas por micrografía electrónica de transmisión para obtener su reconstrucción tridimensional.

El equipo de desarrollo de Xmipp ha trabajado en este campo durante años, desarrollando potentes algoritmos que permiten trabajar con imágenes de Rayos X desde su proceso de adquisición hasta el de su reconstrucción, por lo que el alumno integrará nuevamente estos algoritmos en el software TomoJ. Para ello se ha creado el botón import X-Ray en el panel superior de la interfaz de usuario de este programa, tal y como observamos en la figura [2.5.3.1]

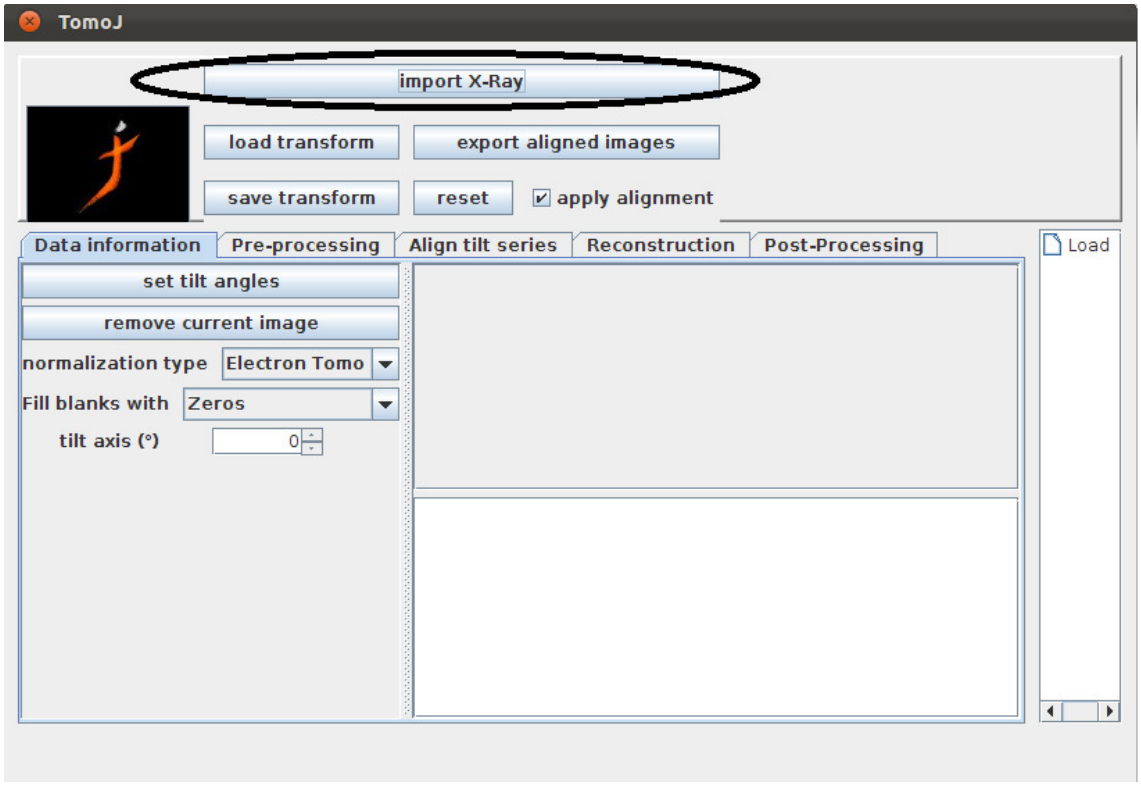


Figura 2.5.3.1: Interfaz de TomoJ, resaltado el botón que permite importar imágenes de TomoX.

Aunque en este apartado no nos centraremos en la implementación de esta funcionalidad es importante destacar que, como se ha mencionado anteriormente, se ha utilizado uno de los algoritmos de Xmipp, concretamente el programa *Xmipp_xray_import*, cuyo código fuente, escrito en C++, está ubicado en el directorio *Programs* de Xmipp. (Figura 2.1.2.1).

El funcionamiento básico de este programa, ejecutado por el usuario desde un terminal, es el de recibir como parámetros los directorios en los que se encuentran las imágenes obtenidas a partir de TomoX, así como una serie de flags opcionales que explicaremos a continuación, y un directorio de salida, en el que generará un fichero con extensión MRCS y un fichero formato SEL que apunte al anterior indicando además el ángulo de inclinación de cada una de las proyecciones. Un ejemplo de cómo ejecutar en un terminal este programa es el siguiente:

xmipp_xray_import --data "DirectorioDeDatos" --flat "DirectorioFlatField" --orrot "DirectorioDeSalida" -crop 7 -f -l

Donde *"DirectorioDeDatos"* es la ruta en la que se encuentran los datos, es decir, cada una de las proyecciones tomoadas con rayos X en el que la muestra está presente. *"DirectorioFlatField"* es el directorio en el que se encuentran los flatfields, correspondientes a proyecciones con el mismo ángulo de inclinación que las anteriores en las que no aparece la muestra. *"DirectorioDeSalida"* es el directorio en el que se va a generar la salida y los flags -crop, -thr, -f y -l permiten la opcionalidad de recortar las imágenes en el caso en que podamos tener problemas de memoria por una gran cantidad de datos, utilizar el número de threads elegido por el usuario para agilizar la ejecución del proceso, aplicar un filtro mediano para eliminar píxeles mal procesados en los bordes y aplicar una corrección log10 para normalizar las imágenes de salida respectivamente.

Para recoger cada uno de estos parámetros se ha creado la interfaz de usuario que podemos ver en la figura [2.5.3.2], que aparecerá como una ventana emergente en el momento en que el usuario seleccione el botón *import X-Ray*. Por razones de comodidad al usuario se ha decidido tomar como directorio de salida el directorio de

trabajo en el que se encuentran los datos obtenidos a partir de TomoX, por lo que el usuario no tiene la posibilidad de introducir este campo. Además, el número de threads que va a emplear la máquina para realizar el proceso será siempre de dos threads.

Como podemos observa, para los directorios de datos y de flatfield, la interfaz ofrece la posibilidad de introducir estas rutas por medio de una cadena de texto o seleccionando el directorio de entre los directorios del sistema de ficheros de la máquina en la que se esté ejecutando el proceso.

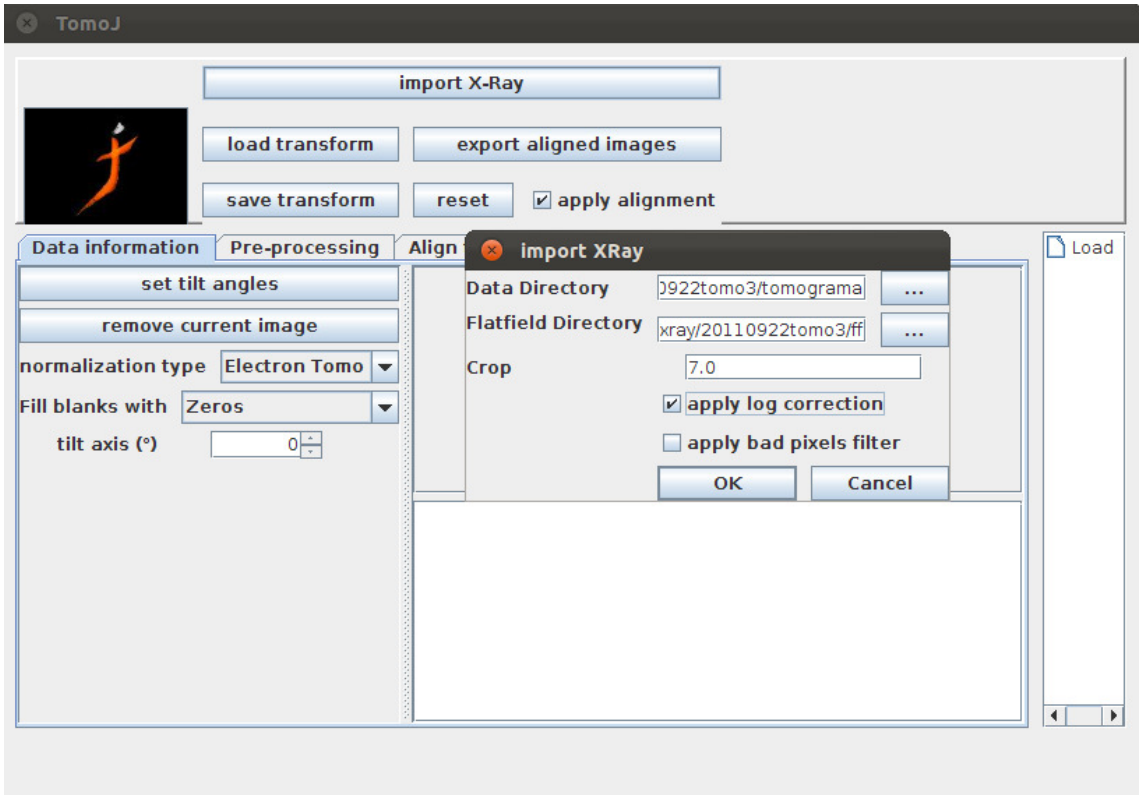


Figura 2.5.3.2: Interfaz de recogida de datos de usuario proporcionada por TomoJ para la ejecución de *Import X-Ray*

Una vez recogido los parámetros introducidos por el usuario se realizará una llamada al programa *xmipp_xray_import* desde el terminal, obteniendo un fichero en formato SEL con la serie de proyecciones equiparable a una serie obtenida a partir de tomografía electrónica por transmisión, por lo que esta serie será cargada en memoria

para poder reconstruirse posteriormente y mostrada al usuario, tal y como podemos apreciar en la figura [2.5.2.3].

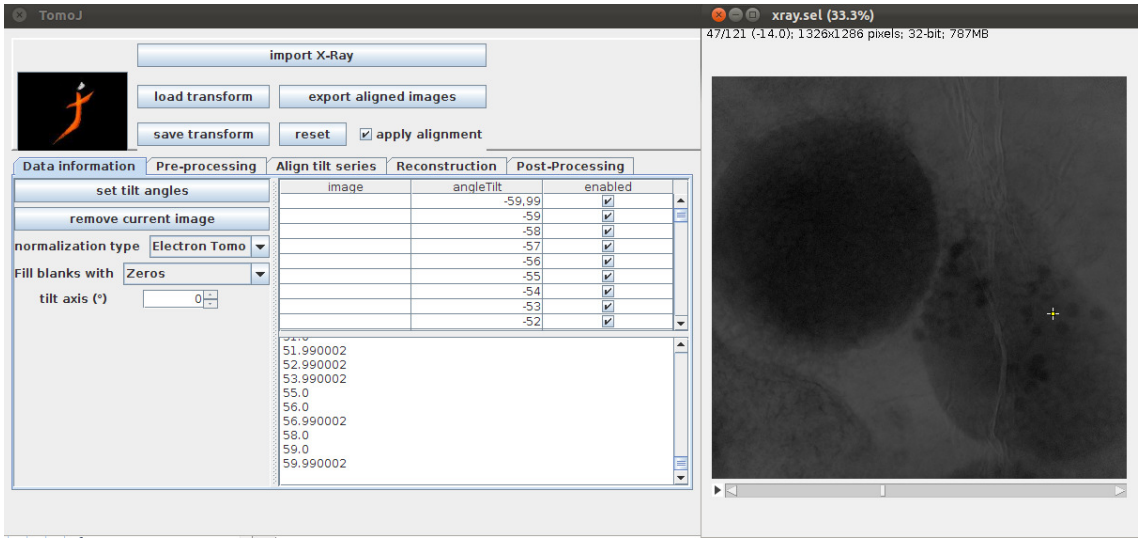


Figura 2.5.2.3: Resultado de la ejecución de importar imágenes de Rayos-X. En la imagen derecha observamos la serie y en el panel central podemos comprobar cómo ha sido cargada en memoria, con sus ángulos de inclinación incluidos

3 Implementación de la integración realizada

3.1 Estructura del software TomoJ

El programa *TomoJ* ha sido implementado, en su totalidad, a través del entorno de desarrollo *IntelliJ Idea*. La cual proporciona al desarrollador una interfaz gráfica de diseño que permite mantener separadas la interfaz gráfica de usuario y la lógica de negocio de una aplicación. Para ello, IntelliJ Idea permite crear interfaces gráficas de usuario utilizando los componentes Swing de la librería de Java de manera fácil y sencilla.

La interfaz de usuario se separa de su código Java y se almacena en ficheros de formulario basados en XML. Cada una de estas interfaces gráficas tendrá una clase Java asociada, la cual implementará cada uno los métodos que su interfaz haya definido.

Para implementar la interfaz gráfica de usuario *IntelliJ Idea* permite al desarrollador abstraerse del código Swing proporcionado un conjunto de herramientas con las que añadir, quitar y modificar los diferentes elementos de los que conste la interfaz gráfica de usuario, siendo el compilador Java el que genere todo el código correspondiente al formulario en su clase asociada.

Una vez creados formulario (interfaz gráfica de usuario) y clase Java asociada se seguirá un paradigma de programación dirigidas por eventos, en la que el flujo de la ejecución del programa será determinado por el usuario de *TomoJ*.

El software TomoJ está formado, por tanto, por una serie de pares formulario - clase asociada, donde el par más importante es el formado por la dupla *TomoJ_.form* y *TomoJ_.java* que muestran e implementan la interfaz principal de TomoJ que ya hemos mostrado a lo largo del proyecto.

La clase *TomoJ_.java* implementa cada uno de los métodos ofrecidos por la interfaz de usuario del programa, para ello, la herramienta de desarrollo crea

automáticamente un método por cada botón definido en el formulario, pasándole como argumento una instancia de la clase ActionListener, interfaz que utiliza Java para recibir eventos en este tipo de aplicaciones. La clase *TomoJ_.java* se apoya además en un conjunto de clases que representan, entre otras, toda la información relativa a la serie de proyecciones (clase *TiltSeries.java*) o los métodos utilizados para realizar las reconstrucciones tridimensionales (*TomoReconstruction.java*) entre otras.

Por el momento no vamos a centrarnos en profundidad sobre la estructura del proyecto TomoJ, ya que este no ha sido implementado por el alumno, por lo que nos centraremos en las integraciones realizadas a partir de dicha estructura.

3.2 Algoritmos de pre-procesado

Tal y como se presentó durante el desarrollo de la sección [2.5.1], los algoritmos de pre-procesado integrados por el alumno han sido el filtro gaussiano, el filtro mediano, la corrección gamma y la corrección de contraste (*Constrast Enhance*, así como la herramienta crop, que permite recortar imágenes. Recordaremos a su vez que estos algoritmos y herramientas están implementadas en el programa ImageJ y que, por lo tanto, para integrarlas, podemos utilizar la librería IJ de ImageJ, que proporciona los métodos necesarios para realizar la implementación de cada uno de estos métodos proporcionando una API para los desarrolladores de plugins de este programa. Para ello, únicamente hemos de importar la librería ij.jar a nuestro proyecto, aunque, como era de esperar, esta librería ya estaba siendo usada por los desarrolladores de TomoJ antes de la participación del alumno. Esta API, aparte de proporcionar métodos ofrece a su vez un conjunto de clases utilizadas, entre otras, para representar las series de proyecciones, como es el caso de la clase TiltSeries.java, que se trata de una extensión de la clase ImagePlus proporcionada por ImageJ.

Para implementar cada uno de los algoritmos descritos anteriormente la función de la librería de IJ que se ha utilizado ha sido la función *run(ImagePlus imp, String command, String options).* Este método recibe como parámetros de entrada una instancia de ImagePlus, en nuestro caso, recibirá una instancia de TiltSeries que, como mencionamos en el capítulo anterior se trata de la clase que representa nuestra serie de

proyecciones y que es, a su vez, una extensión de ImagePlus. La cadena de caracteres *command* representa el comando que se va a utilizar para la llamada. Cada algoritmo tiene su comando asociado. Por último, la cadena *options* define las opciones de macro, que para el caso de nuestra aplicación no es necesario utilizar. El resultado de ejecutar este método es el de modificar la instancia del objeto ImagePlus o TiltSeries aplicando sobre esta el comando de entrada.

La figura [3.2.1] muestra el diagrama de clases utilizado para realizar la integración de los algoritmos de pre-procesado. Para ello se ha utilizado la clase Plugin (siguiendo el modelo de *Xmipp_Tomo* de Xmipp), la cual define el método *run* comentado recientemente así como métodos de recogida de los parámetros que se utilizan para ejecutar este método. Se ha utilizado el principio de programación de herencia para caracterizar a cada uno de los algoritmos de pre-procesado, los cuales heredan e implementan cada uno de los métodos de recogida de parámetros que será los que utilice la clase Plugin en cada caso.

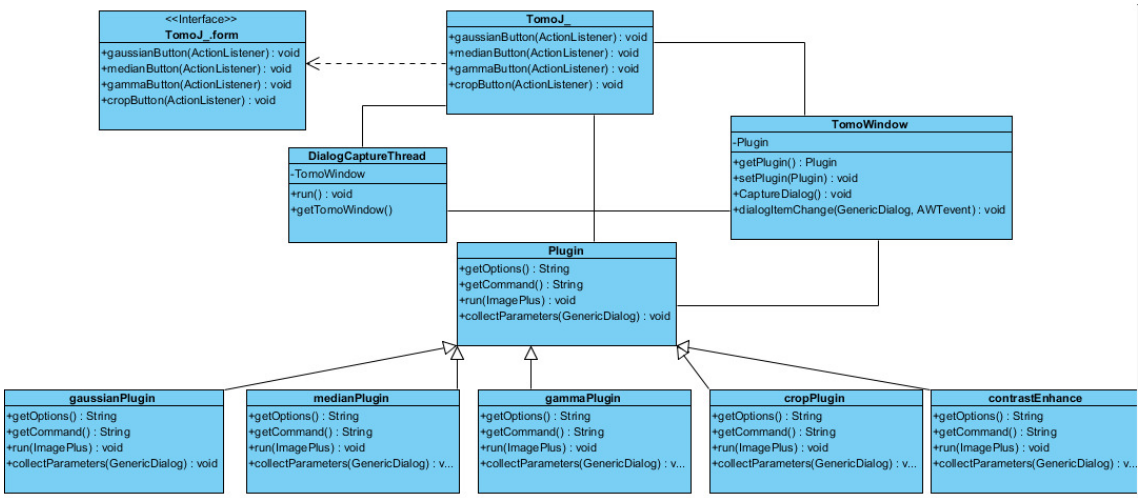


Fig. 3.2.1: Diagrama de clases utilizado para la implementación de los algoritmos de pre-procesado

Como ya hemos anticipado a lo largo del proyecto, una de las ideas de Xmipp_Tomo para su software era la del concepto de workflow, una estructura arbórea que mantuviese en cada nodo el estado por el que pasa el objeto de TiltSeries a medida que el usuario aplica diferentes algoritmos y métodos a través del programa para poder volver sus pasos en cualquier momento y sobre cualquier paso. Para llevar a cabo esta idea es totalmente indispensable conocer cada uno de los parámetros de entrada que utilizan las funciones de TomoJ para poder diferenciar los pasos realizados de manera inequívoca.

El problema que este concepto presenta en este contexto reside en que en el caso de utilizar la librería IJ para llevar a cabo la ejecución de los algoritmos de pre-procesado de imágenes reside en que el control del proceso pasa a ser de ImageJ, por lo que los diálogos de recogida de datos son proporcionados por el propio ImageJ en lugar de TomoJ. Esto implica que una vez devuelto el proceso a TomoJ después de aplicar el algoritmo seleccionado, el desarrollador de TomoJ desconoce cuáles han sido los valores de los parámetros de entrada.

Para solucionar este problema, se han creado las clases *TomoWindow.java* y *DialogCaptureThread.java* [3.2.1], que utilizan el método *captureDialog()* para capturar todos los diálogos que corren en cualquier terminal de la máquina en la que se esté ejecutando el proceso. Permitiendo seleccionar el tipo de diálogo que se esté tratando de capturar, en nuestro caso: "ij.gui.genericDialog", que se trata de la clase que permite mostrar los diálogos de ImageJ.

La clase *DialogCaptureThread.java* será una extensión de Thread, hilo de ejecución que lanzará el proceso para escuchar los diálogos abiertos e invocará en su método *run* al método de TomoWindow *captureDialog()*, el cual se subscribirá a *DialogListener()*. Este método se mantendrá inactivo hasta que sea capturado un diálogo, en cuyo caso se invoca el método *dialogItemChanged(GenericDialog gd, AWTEvent e)* de TomoWindow, que será el encargado de recoger los parámetros introducidos por el usuario. A continuación explicaremos la secuencia de pasos utilizada detallada para realizar la captura de parámetros de la ventana de ImageJ

- Durante la implementación del método relativo al botón de cada uno de los algoritmos de pre-procesado se lanza un hilo de ejecución de DialogCaptureThread:

  newThread(new DialogCaptureThread(TomoWindow window).start

- La clase DialogCaptureThread implementa el método *run(),* que es el que se ejecute cuando se lanza el proceso *start* del paso anterior. Este método invoca al método *TomoWindow.captureDialog()*

- El método *TomoWindow.captureDialog()* revisa todas las ventanas abierta por el terminal, y se subscribe a aquellas que coincidan con la clase *ij.gui.GenericDialog* a través del método *addDialogListener().*

- En el momento en el que se abre una ventana de las anteriores, el evento llega al método *TomoWindow.dialogChangedItem(GenericDialog gd, AWTevent e)* que en el objeto *gd* contiene todos los parámetros introducidos por el usuario.

- Por último, se utiliza el método *Plugin.collectParameters(GenericDialog gd).* A través de este método cada hijo de la jerarquía plugins recogerá el número de parámetros que le corresponda, ya que en el caso de haber sido el filtro gaussiano el utilizado el objeto *gd* únicamente contendrá el valor del parámetro sigma pero si el algoritmo que se ha utilizado ha sido el de mejora de contraste, *gd* contendrá todos los parámetros relativos a su utilización.

3.3 Integración de Tomo3D

Para realizar la integración del programa Tomo3D en TomoJ se ha seguido el diagrama de clases relativo a la figura [3.3.1]. Es importante destacar que este diagrama únicamente recoge los métodos y atributos implementados y utilizados por el alumno

para realizar la integración de estos programas. Cada uno de los métodos y atributos serán explicados detalladamente a lo largo de este capítulo.

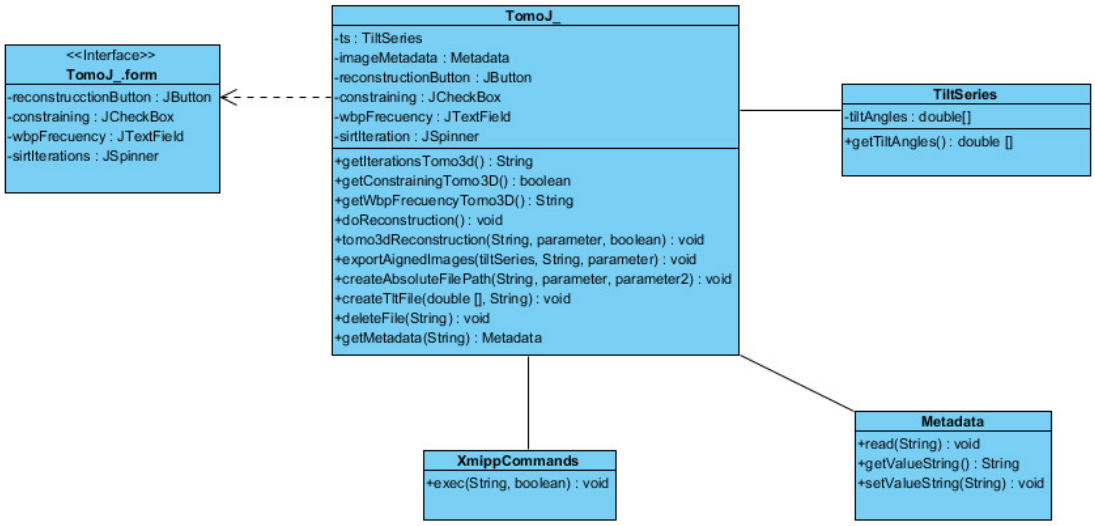


Fig. 3.3.1: Diagrama de clases utilizado para realizar la integración de *Tomo3D* en *TomoJ*

El primer paso que se ha realizado para realizar la implementación ha sido el de añadir la opción de utilizar el primer programa en la interfaz de usuario de *TomoJ*. Como mencionamos en el capítulo [3.1], la clase principal del proyecto es *TomoJ_.java*, que implementa las acciones a realizar una vez pulsados los botones de la interfaz de usuario, definida por *TomoJ_.form*. Por esta razón, el primer fichero recodificado ha sido este último, añadiendo a la lista de opciones del *ComboBox* las alternativas WBP (Tomo3D) y SIRT (Tomo3D), tal y como podemos observar en la figura 3.3.2.

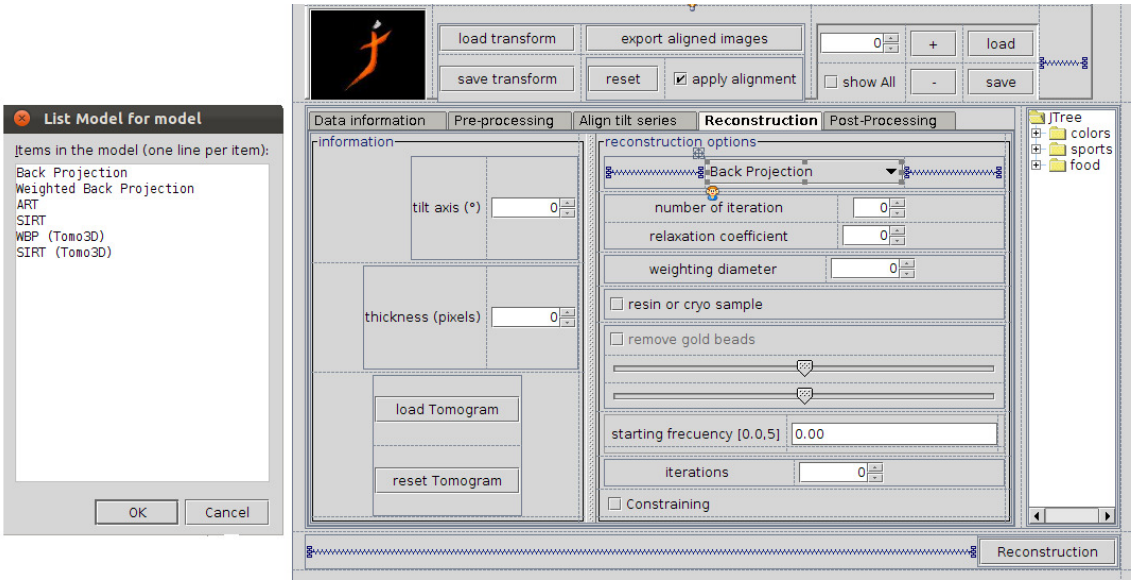


Fig. 3.3.2: Interfaz de diseño proporcionada por *TomoJ* para crear interfaces de usuario. La lista de la izquierda corresponde a las alternativas que presenta el *ComboBox* seleccionado, en el que en este momento está seleccionado la opción *Back Projection*

Además de añadir estas alternativas a la lista ha sido necesario añadir los paneles de recogida de datos para los dos algoritmos que se han sido implementados, recordaremos en este punto del capítulo [2.5.2] que, para el caso de utilizar el algoritmo *WBP* de *Tomo3D,* es necesario indicar la frecuencia inicial a partir de la cuál trabajará el filtro de *Hamming* que se utiliza para este algoritmo, mientras que para el caso del algoritmo SIRT de Tomo3D es necesario indicar el número de iteraciones que deseamos emplear para llevar a cabo la reconstrucción. Además, en ambos casos es posible habilitar la opción *constraining*, que asegura que las densidades de los voxels del volumen que se calcula sean positivas, dejando a cero las que sean negativas.

Como podemos observar en la figura [3.3.2] se han añadido tres paneles: el primero de ellos relativo al algoritmo *WBP*, el segundo relativo a *SIRT* y un último panel para habilitar o deshabilitar la opción *constraining*. Además, en la parte posterior nos encontramos a su vez con los paneles relativos al resto de métodos que ya estaban implementados por TomoJ. Por último, en la parte inferior derecha de la interfaz de

diseño podemos encontrar el botón de reconstrucción, que será el que lance el proceso una vez introducidos los parámetros solicitados por la interfaz de usuario.

En el momento en el que el usuario cambia el algoritmo de reconstrucción que va a utilizar, se lanza un evento que será recogido por el método de la clase TomoJ_.java para ejecutar el método asociado al JComoboBox. Este método habilita y deshabilita los paneles de recogida de parámetros de cada algoritmo en función del algoritmo elegido, por esta razón no visualizamos todos los paneles al mismo tiempo como en la interfaz de diseño, si no que únicamente vemos los parámetros asociados a cada algoritmo, tal y como podemos observar en las figuras [2.5.2.2] y [2.2.5.3].

Una vez seleccionado el algoritmo a utilizar así como sus parámetros asociados, el usuario utilizará en el botón *Reconstruction* para realizar la reconstrucción. Esto lanzará un evento recogido por el método de la clases Java asociada a dicho botón. Método que ya existía evidentemente antes de la llegada del alumno al proyecto ya que permite realizar la reconstrucción de la serie con los algoritmos implementados por TomoJ. Esta reconstrucción se realiza a través del método *doReconstruction() : void,* que podemos ver en el diagrama de clases de la figura [3.3.1]. Es importante recalcar en este punto una diferencia muy importante que existe entre las reconstrucciones ya implementadas y las que el alumno integra. Esta diferencia reside en que los algoritmos de reconstrucción implementados por TomoJ utilizan la clase *TomoReconstruction.java* para realizar la reconstrucción a partir de *TiltSeries.java.* Ambas clases son extensiones de la clase de la librería de ImageJ *ImagePlus.java*, por lo que para cada algoritmo, TomoJ crea una instancia de TomoReconstruction para obtener la tomografía en función del estado del objeto TiltSeries. (No vamos a centrarnos en cómo se realizan estas reconstrucciones al no haber sido implementadas por el alumno), aunque volveremos a la clase *TomoReconstruction.java* más adelante.

A diferencia de estos algoritmos, Tomo3D, como hemos introducido ya en capítulos anteriores, se limita a recibir una serie como parámetro de entrada en formato MRC y crear la reconstrucción en formato MRC. Esto implica que no se utiliza el estado de la serie en memoria, si no que se utiliza el estado de la serie que está almacenado en disco, mientras que los algoritmos de reconstrucción implementados por

TomoJ realizan todo el proceso en memoria. Este hecho va a provocar que, como veremos más adelante, sea necesario antes de realizar la reconstrucción almacenar el estado en el que se encuentra la serie en disco y, una vez realizada la reconstrucción cargar en memoria la tomografía a partir de la clase *TomoReconstruction.java.*

Como consecuencia de esta diferencia se ha decidido crear un método diferente para implementar los algoritmos de Tomo3D. Este método recibe el nombre de tomo3dReconstruction y recibe los parámetros del tipo de algoritmo que se va a utilizar (WBO ó SIRT), los parámetros del algoritmo (frecuencia inicial en el caso de WBP y número de iteraciones para el caso de SIRT) y si se desea utilizar la opción *constraining.*

tomo3dReconstruction(String type, String options, boolean constraining)

Además, se han implementados a su vez los métodos getIterationsTomo3D() : String, que permite obtener el valor introducido en el JSpinner "iterations", getWbpFrecuencyTomo3D() : String, que permite obtener el parámetro del JTextField "WBPfrecuency" y getConstrainingTomo3D() : boolean, que permite extraer el valor del JCheckBox "constraining". Por lo que las llamadas a este método son las siguientes en función del tipo de algoritmo que el usuario va a utilizar: *tomo3dReconstruction("wbp", getWbpFrecuencyTomo3D(),getConstrainingTomo3D()) tomo3dReconstruction("sirt", getIterationsTomo3D(),getConstrainingTomo3D())*

A través de este método se realizarán todos los pasos necesarios para ejecutar el programa tomo3D con TomoJ, teniendo en cuenta que la serie de proyecciones puede estar en diferentes formatos. En este punto es importante recordar uno de los objetivos del proyecto que, hasta ahora, no había vuelto a ser mencionado. Este objetivo era el de integrar la capacidad de leer y procesar ficheros en formato SEL a través de TomoJ.

Los ficheros SEL son un tipo especial de ficheros que representan la serie de proyecciones. Estos ficheros de texto contienen una línea para cada proyección de la serie, en la que aparecen el nombre de la imagen de la proyección, que puede estar en formato MRCS, STK o XMP entre otros, un valor booleano que representa el hecho de si esta proyección está habilitada o deshabilitada, y el ángulo de inclinación asociado a

dicha proyección. Este último parámetro es muy importante, ya que gracias a estos ficheros siempre tenemos información sobre el ángulo de inclinación de cada proyección. En el caso de otros formatos como MRC esta información no es conocida y, de hecho, al abrir el plugin de TomoJ con cualquier formato distinto de SEL el programa lo primero que realiza es preguntar al usuario cuáles son los ángulos de inclinación de cada proyección, permitiendo introducir el primer ángulo y el salto angular con respecto a la siguiente proyección, tal y como comentamos en el capítulo [1.3.5].

En este contexto, la clase *Metadata* que encontramos en el diagrama de clases de la figura [3.3.1] no es sino una representación de este fichero. Los objetos de esta clase mantienen toda la información relativa a cada proyección y para ello se utiliza el método *getMetadata(String filename) : Metadata* que crea una instancia de Metadata a partir del fichero SEL. Este método será utilizado y explicado en momentos posteriores de este capítulo.

La última clase del diagrama de clases de la figura [3.3.1] que aún no ha recibido mención es *XmippCommands.* Como recordaremos del capítulo [2.2] Xmipp contiene una librería de funciones que proporciona más de 150 programas distintos. que pueden ser ejecutados a través de la línea de comandos UNIX. Como veremos a continuación, la integración de Tomo3D en TomoJ va a requerir de la ejecución de alguno de estos programas de Xmipp, por lo que el objetivo de esta clase va a ser el de proporcionar una interfaz a la clase que la invoque (en este caso TomoJ_.java) para ejecutar comandos en el terminal de UNIX de forma transparente y sencilla. Para ello *XmippCommands* implementa el método *exec(String command)* : *ExitValue*, que ejecuta el comando *command* en el terminal UNIX devolviendo un valor de la enumeración ExitValue que permite conocer el resultado de la ejecución.

Volviendo al hilo de la ejecución del programa y los ficheros SEL, recordaremos que el programa Tomo3D se ejecuta desde la línea de comandos y que, además, únicamente es capaz de recibir ficheros MRC como entrada. Este hecho va a hacer que tengamos un especial cuidado a la hora de tratar ficheros SEL, ya que de algún modo tendremos que convertir este fichero a MRC. Por esta razón, y para explicar

el proceso a través del cual se lleva a cabo la reconstrucción vamos a explicar el flujo de procesos y de métodos que se sigue para lograr este objetivo teniendo en cuenta que la serie de proyecciones se encuentra en formato SEL, indicando qué pasos no serán necesarios realizar en caso en que la serie se encuentre en formato MRC.

El primer paso consiste en guardar el estado de la serie en disco. A diferencia del resto de algoritmos de reconstrucción implementados por TomoJ, Tomo3D necesita guardar en disco el estado en el que se encuentra la serie antes de ser reconstruida ya que, en la mayoría de los casos, la serie ha sido pre-procesada y alineada antes de ser reconstruida. Este estado de la serie guardado en disco será el que utilizaremos posteriormente para realizar la reconstrucción, ya que el fichero original no cambia al aplicarse las técnicas de alineamiento y pre-procesado, si no que estos cambios se realizan en memoria a través del objeto TiltSeries. En realidad, esto ocurre únicamente para el caso de los algoritmos de pre-procesado, que sí modifican el objeto TiltSeries, mientras que los algoritmos de alineamiento que implementa TomoJ crean una matriz con el valor de cada uno de los píxeles alineados, sin modificar el objeto TiltSeries. Estos ficheros que guardarán el estado de la serie de proyecciones en el momento previo de la reconstrucción serán borrados al finalizar la ejecución del programa, y serán creados en una carpeta ubicada en el mismo directorio que la serie original denominada *temporal*, en la que, como veremos posteriormente será necesario crear otros ficheros temporales. Para implementar esta funcionalidad se ha utilizado la función *exportAlignedImages(ImagePlus, String, String)*, que ya había sido implementado previamente por los desarrolladores de TomoJ, y que permite guardar el estado de la serie en el mismo formato que la serie original, utilizando para ello la función de la librería IJ *runPlugin(ImagePlus ts, String command, String filename)*, donde *command* será el nombre del plugin de Tomo3D que permita escribir un fichero a partir de una instancia de ImagePlus ("sel_writer, "mrc_writer", "spider_writer" entre otros).

Una vez guardados los cambios sufridos por la serie de proyecciones y, únicamente en el caso de que el formato de la serie original sea SEL, es necesario convertir la serie de proyecciones a formato MRC. Es importante recordar que la serie con la que estamos tratando en este punto no es la serie original, si no que es la obtenida

en el paso anterior, que contiene el estado en el que se encuentra la serie antes de realizar la reconstrucción. Para convertir una serie de proyecciones SEL en formato MRC se ha utilizado una de las funciones de Xmipp, por lo que para ejecutarla se llamará al método *XmippCommands.exec(String command)* explicado anteriormente. Este programa de Xmipp llamado *xmipp_image_convert* convierte una serie de proyecciones en formato SEL a la misma serie formato MRC. Para ello se ejecuta desde un terminal UNIX con los siguiente parámetros: *xmipp_image_convert -i selFile.sel -o mrcFile.mrc :mrcs*, aunque en nuestro caso esté será el comando que le pasaremos al método *XmippCommands.exec(String command).*

En este punto debemos de tener en cuenta un hecho del que hasta el momento no nos habíamos percatado. El directorio en el que se encuentra el terminal cuando se ejecuta ImageJ es el propio directorio /ImageJ que, dependiendo del usuario, estará ubicado por el directorio /home del usuario. Sin embargo, la serie de proyecciones que estamos cargando puede estar ubicada en un directorio diferente por lo que hemos de tener sumo cuidado con las rutas que utilicemos a la hora de programar. Este factor hace que en este caso sea imprescindible que la referencia a cada una de las proyecciones de la serie que se encuentren en el fichero SEL se encuentren como ruta absoluta. En caso contrario será necesario reescribir el fichero SEL con las rutas absolutas de cada proyección. Para ello se ha implementado la función *createAbsolutePathSelFile (String, String, String)*, que convierte el fichero SEL anterior en otro igual poniendo todas las referencias a las proyecciones como ruta absoluta. Para ello, este método utiliza la clase Metadata de Xmipp, disponible a través de la librería Xmipp.JNI.jar, que, como introdujimos anteriormente representa la estructura del fichero SEL. A continuación se explica paso por paso qué tareas realiza esta función:

- Obtiene el objeto Metadata a partir del fichero SEL anterior a través de la función *getMetadata (String filePath) : Metadata*, implementada por el alumno. La cuál realiza una llamada a la función de la clase Metadata (implementada por los desarrolladores de Xmipp*) Metadata.read(String filepath) : void*.

- Para cada proyección representada en el objeto Metadata obtiene el nombre del fichero al que hace referencia, a través de la función *Metadata.getValueString(MDLable MDL_IMAGE, long proyection) : String,* modifica este valor fijando la ruta absoluta y modifica el objeto a través de la función *Metadata.setValueString(MDLable MDL_IMAGE, String absolutePath, long proyection).*

- Por último, utiliza la función *Metadata.write(String file)* para crear un nuevo fichero SEL con el objeto Metadata modificado en el paso anterior.

Una vez realizado este paso ya tendremos la serie, en cualquier formato, preparada para ejecutar el programa Tomo3D. Es importante destacar que todos los ficheros creados en este paso serán ficheros temporales, por lo que serán creados en el directorio /temporal creado en el paso anterior y una vez terminada la ejecución del programa serán borrados.

Además de la serie de proyecciones en formato MRC, Tomo3D necesita a su vez un fichero de texto IMOD con extensión TLT que contenga el ángulo de inclinación con el que ha sido tomada cada proyección. Para ello, se ha implementado la función *createTltFile(double[] angles, String file),* donde el vector de ángulos es obtenido fácilmente a parir del objeto instanciado de TiltSeries a través de la función *TiltSerie.getTiltAngles() : double[]* , implementada por los miembros de TomoJ.

Una vez obtenidos todos los ficheros a partir de los cuales realizar la reconstrucción construiremos el comando que permita ejecutar el programa Tomo3D desde el terminal UNIX. Para construir este comando utilizaremos los parámetros de entrada de esta función que, como recordaremos indicaban el tipo de algoritmo que se desea utilizar así como los parámetros asociados a este (la frecuencia inicial para aplicar el filtro de Hamming en el caso de WBP y el número de iteraciones en el caso de SIRT) así como el uso de la opción constraining. Una vez construido el comando utilizaremos el método *XmippCommands.exec(String command) : void* , que, como vimos anteriormente permite ejecutar comandos de forma transparente en la línea de comandos

UNIX. Es importante comentar en este punto que este comando realizará una llamada al fichero binario, relacionado con el tipo de procesador de la máquina, de Tomo3D, por lo que es necesario que este fichero binario se encuentre ubicado en el directorio de ImageJ. Estos requisitos de instalación los veremos detalladamente en el manual de instalación del capítulo [5].

Una vez ejecutado el comando en el terminal, obtendremos la reconstrucción tridimensional almacenada en disco y en formato MRC. Sin embargo, lo realmente interesante sería mantener el estado de este volumen en memoria, ya que de este modo, podremos aplicar a este volumen los algoritmos de post-procesamiento implementados por TomoJ que estudiamos en el capítulo [2.3.2] y, haciendo además que el programa Tomo3D sea totalmente equivalente al resto de algoritmos de TomoJ, ya que, a partir de una serie de de proyecciones genera una instancia de la clase TomoReconstruction.

La clase TomoJ_.java contiene un atributo del tipo TomoRecostruction, que se instancia cada vez que el usuario realiza una reconstrucción independientemente del tipo de algoritmo que utilice. Recordemos que en este punto nuestro volumen computado se encuentra almacenado en disco en formato MRC en el directorio */temporal*. Para obtener una instancia de TomoReconstruction a partir del fichero MRC se han seguido los siguientes pasos:

- Generar una instancia de IJ.ImagePlus a partir del volumen en formato MRC. Para ello se ha utilizado la función *IJ.load(String filePath) : ImagePlus* de la librería de ImageJ.

- Construir el objeto de TomoReconstruction a través de su constructor: *TomoReconstruction (ImagePlus)*. Recordaremos en este punto que la clase TomoReconstruction es una extensión de la clase ImagePlus, por lo que sus tipos son totalmente compatibles.

- Abrir el volumen computado a través del objeto TomoReconstruction. Para ello se ha utilizado la función *ImagePlus.show( ).*

Por último, se borrará el directorio */temporal* en el que se han creado todos los ficheros auxiliares necesarios para llevar a cabo la ejecución del programa. En este directorio podemos encontrar el fichero TLT con los ángulos de inclinación de cada serie, la serie de proyección en el mismo formato de la serie original creado en el paso previo a la reconstrucción para guardar el estado en el que se encontraba la serie antes de realizar la reconstrucción, el fichero SEL creado a partir del original con las referencias hacia cada proyección como ruta absoluta y la tomografía obtenida y cargada en el paso anterior en memoria.

3.4 Integración de import X-ray

Como se ha ido comentando a lo largo del proyecto, el software TomoJ ha sido implementado a través del entorno de desarrollo idea (IntelliJ), el cuál proporciona una interfaz de diseño a través de la cual crear interfaces gráficas de usuario utilizando los componentes Swing de la librería de Java de manera fácil y sencilla.

Para realizar la implementación de esta funcionalidad se ha utilizado esta interfaz de diseño, a través de la cual se ha obtenido una interfaz de usuario que permite la recogida de los parámetros necesarios para ejecutar la importación de imágenes tomadas a partir de TomoX. Estos parámetros, de los que ya hablamos en el capítulo [2.5.3] servirán como entrada para el programa *xmipp_xray_convert* de Xmipp.

En la figura 3.1.1 se muestra la interfaz de diseño del entorno de desarrollo en la que se han introducido componentes JSwing para diseñar la interfaz de usuario. Como se puede observar, los componentes JSwing conforman una estructura arbórea con todos los componentes del formulario. Se ha dividido este formulario en distintos paneles, distinguibles por la funcionalidad que ofrecen. Los dos primeros son relativos al directorio de datos y el directorio de flatfields, en los que se ofrece la posibilidad de introducir estas ruta a través de un JTextField, en la que el usuario introducirá la ruta del directorio a mano y, además se permite la posibilidad de escoger un directorio del sistema de ficheros del usuario pulsando el botón contiguo. El panel de *Crop* permite recoger por medio de un JTextField el número de píxeles que se quiere recortar a cada lado, mientras que los dos últimos paneles permiten habilitar por medio de un

JCheckBox si se desea habilitar la corrección logarítmica y el uso del filtro mediano para eliminar píxeles erróneos.

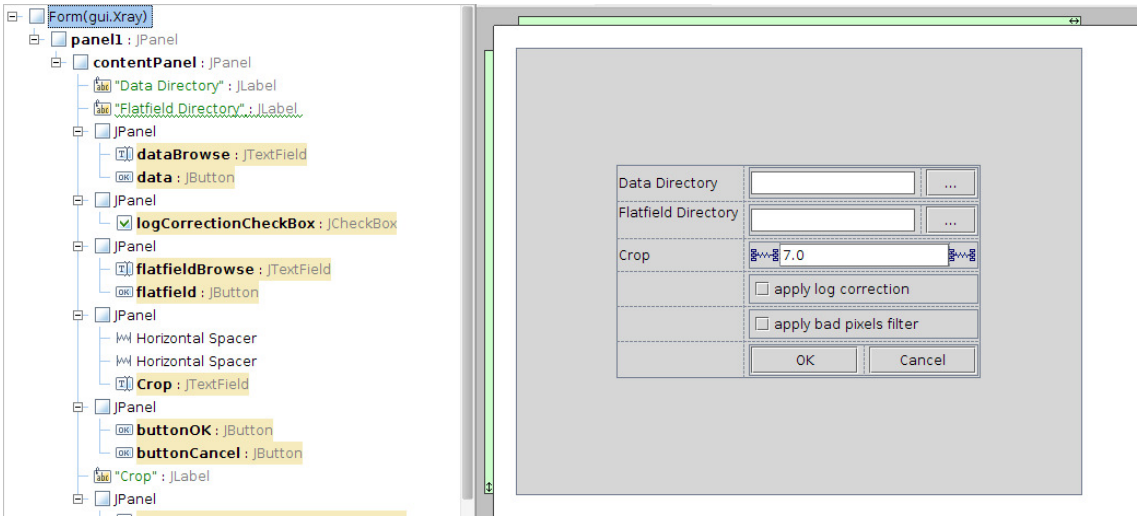


Figura 3.1.1: Interfaz de diseño proporcionada por el entorno de desarrollo idea (IntelliJ) utilizada para crear el fichero *xray.form*

Cada uno de los botones de la interfaz tiene un método *Listener* asociado en la clase Xray.java que será invocado cada vez que el usuario pulsa cada uno de ellos. Los dos primeros deberán mostrar al usuario su sistema de directorios para escoger los directorios de datos y de *flatfield* con los cuales ejecutar el proceso. Para ello se ha utilizado la clase JFileChooser, una clase java que nos permite mostrar fácilmente una ventana para la selección de un fichero o un directorio. Por defecto, JFileChooser se abre en el directorio home del usuario, tal y como observamos en la figura [3.1.2] y permite seccionar cualquier tipo de fichero o directorio, aunque podemos realizar una restricción sobre el tipo que queramos abrir y, en nuestro caso, nos interesa únicamente mostrar los directorios por lo que utilizaremos el filtro JFileChooser.DIRECTORIES_ONLY. Una vez seleccionado el directorio se escribirá en el JTextField la ruta seleccionada. (Figura 3.1.3)

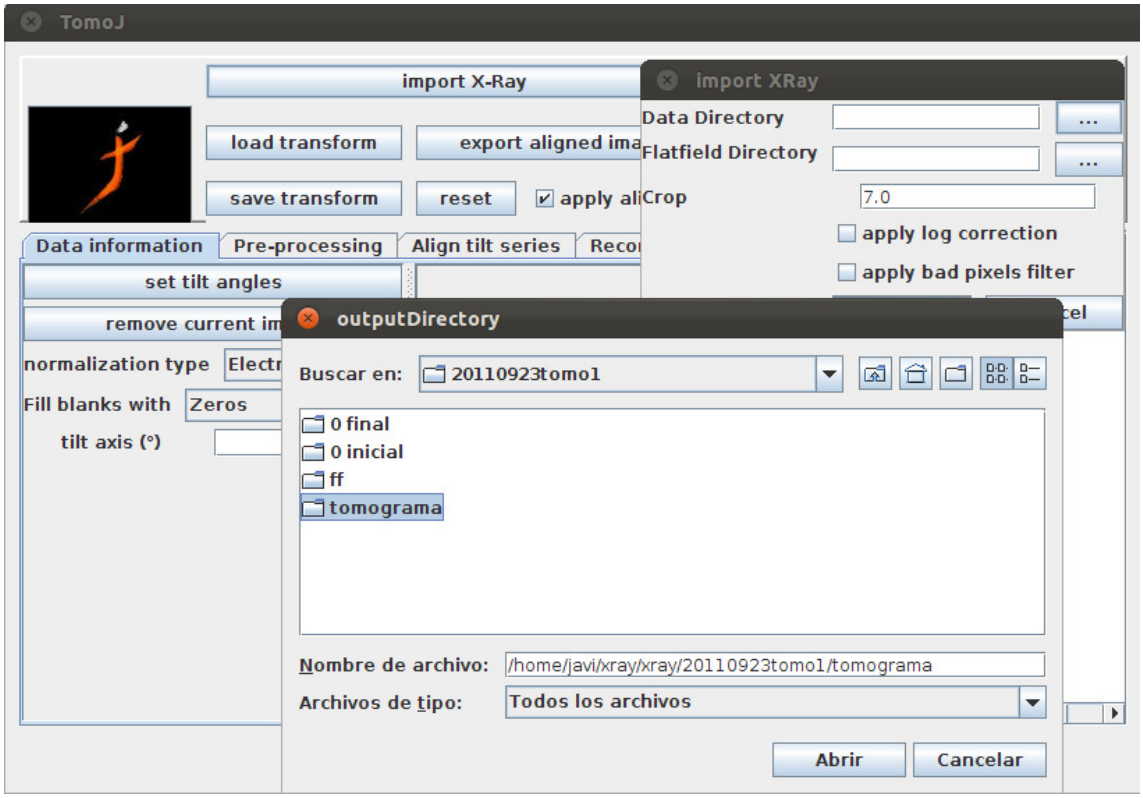


Fig. 3.1.2: Ventana emergente para seleccionar un directorio del sistema de ficheros del usuario. Por defecto JFileChooser se abre en el directorio home del usuario

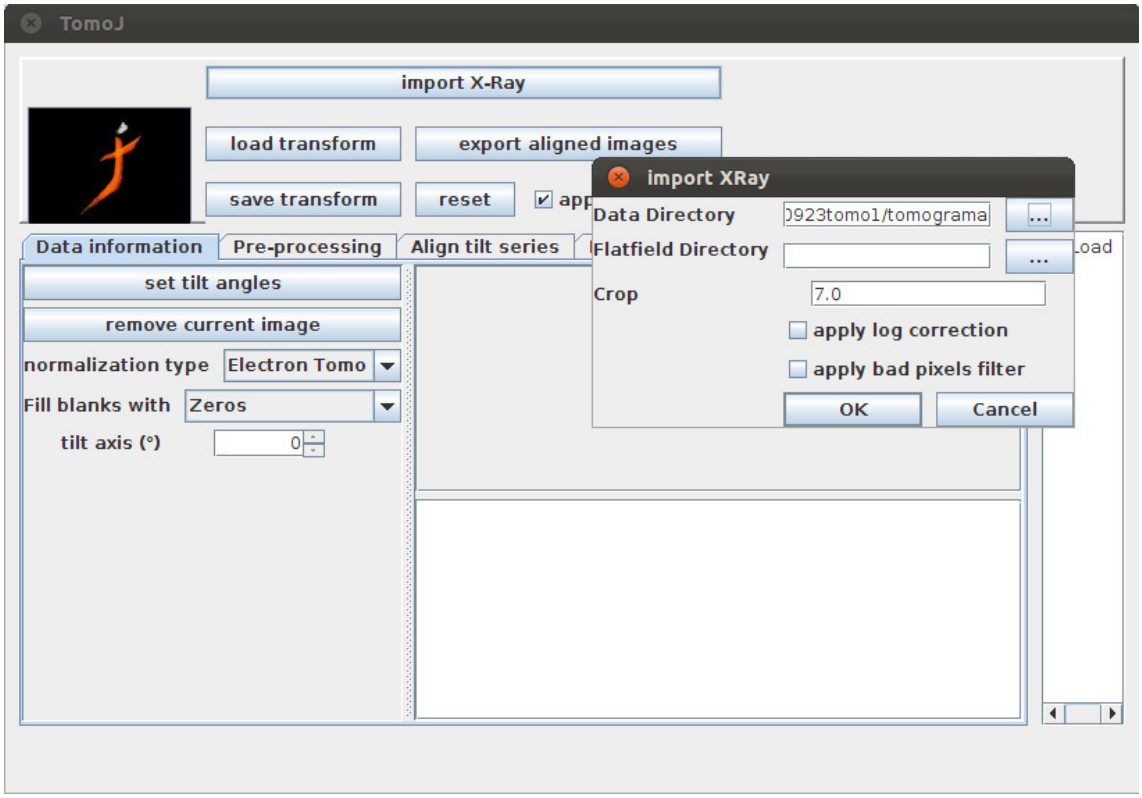


Fig. 3.1.3 Interfaz de X-Ray una vez seleccionado el directorio de la figura anterior

El botón *Cancel* cancela el proceso de importación de Rayos-X dejando la ejecución del programa en el mismo estado que se encontraba previamente utilizando el método *dispose( )*, mientras que, por su parte, el botón *OK* es el que desencadena la ejecución del programa accediendo a cada uno de los JTextField y de los JCheckBox para obtener los parámetros que requiere el programa, construyendo el comando que permita invocar al programa *xmipp_xray_convert* y ejecutándola a partir del método *XmippCommands.exec(String command)* que ya vimos en capítulos anteriores.

El resultado de ejecutar este programa es un fichero con extensión MRCS y otro fichero en formato SEL que apunte al anterior indicando además el ángulo de inclinación de cada una de las proyecciones, por lo que el siguiente paso que debemos realizar es el de cargar en memoria esta serie generada. Al igual que ocurría en el caso de la integración de Tomo3D se ha creado un directorio temporal dónde se almacenan

los ficheros generados, que podrán ser descartados y el usuario lo cree conveniente. Se ha decidido además que el directorio /temporal estará ubicado en el directorio padre del directorio de datos que haya introducido el usuario.

Recordaremos en este punto que el software TomoJ trabaja con la clase *TiltSerie*s, cuya instancia mantiene toda la información sobre el estado de la serie de proyecciones. Podemos darnos cuenta de el hecho de que cargar una imagen en memoria a partir del una serie de proyecciones almacenadas en disco es precisamente lo que realiza el programa al arrancar, comprueba qué imagen está abierta en ImageJ, a partir de ella obtiene un objeto *ImagePlus* a través del método *WindowManager.getCurrentImage()* y genera el objeto TiltSeries utilizando para ello el método *init(ImagePlus, boolean).* El caso en el que nos encontramos en este instante es muy similar al anterior con la salvedad de el fichero que hemos de cargar en memoria no es el fichero al que apunte la ventana de ImageJ abierta, sino el fichero SEL creado tras la ejecución del programa ubicado en el directorio padre del directorio de datos seleccionado por el alumno, por lo que reutilizaremos este método para cargar en memoria la serie de proyecciones obtenida a partir de *Import X-Ray.*

Tras la llamada al método *init* obtendremos un objeto TiltSeries que mostraremos a través de una nueva ventana a partir del método *TiltSeries.show()*

4 Manual de instalación

Para utilizar el plugin TomoJ a través de ImageJ utilizando las funciones del paquete Xmipp es necesario seguir una serie de pasos que se muestran a continuación:

1. Descargar el paquete Xmipp. Como ya hemos mencionado a lo largo del proyecto, Xmipp es un paquete portable que puede ser integrado en cualquier máquina que tenga instalado el compilador C++ de GNU y las librerías gráficas Qt. En la práctica esto incluye cualquier máquina UNIX y Windows vía Cygwin. Se trata de un software abierto, por lo que puede ser descargado desde el repositorio de sourceforge.net fácilmente. A su vez, el equipo de Xmipp ofrece la posibilidad de descargar su propia máquina virtual con sistema operativo Ubuntu 10.11. Esta máquina virtual puede ser descargada desde la web oficial de Xmipp. Esta última opción ha sido la empleada por el alumno.

2. Compilar Xmipp. Una vez descargado el paquete Xmipp se procederá a su compilación situándose con un terminal en el directorio raíz de Xmipp y ejecutando *-xcompile.* Este proceso, que puede llevar uno minutos compilará todas las clases ubicadas dentro del paquete e instalará en la máquina todo el software necesario para utilizar las funciones de Xmipp.

3. Ejecutar el comando *source .xmipp.bashrc* en el directorio raíz de Xmipp.

4. Descargar el programa ImageJ (versión 1.44o). El propio paquete de Xmipp ya cuenta con este programa, ubicado en el directorio /xmipp/external/imagej. Por lo que se recomienda su uso. No obstante este software se puede descargar fácilmente desde la red.

5. Copiar las librerías de Xmipp utilizadas por ImageJ para la visualización de imágenes en formato no estándar, como es el caso de los ficheros

SEL. Estas librerías, ubicadas en /xmipp/java/lib son las siguientes: XmippViewer.jar, XmippUtils.jar, XmippJNI.jar, y XmippPPicker.jar.

6. Copiar los ficheros binarios de tomo3d en el directorio de ImageJ. Tal y como comentamos en los capítulos [2.4] y [3.3], el programa tomo3D se ejecuta partir de la llamada a unos ficheros binarios que debemos ubicar en el directorio de Xmipp, ya que éste es el directorio raíz del proceso.

7. Por último, es necesario copiar la librería de clases compiladas de TomoJ en el directorio /plugins de ImageJ, esta librería puede ser descargada desde el repositorio sourceforge.net

En el caso en el que, además de utilizar el plugin de TomoJ, se desee mejorar la aplicabilidad de TomoJ es posible descargarse el código fuente de TomoJ, modificarlo y compilarlo. Para ello se han de seguir los siguientes pasos.

8. Descargar e instalar el entorno de desarrollo idea (IntelliJ).

9. Descargar el código de TomoJ y abrirlo en idea.

10. Importar las librerías de Xmipp que ya están copiadas en el directorio /imageJ/plugins así como la librería IJ.jar que proporciona los métodos y clases de ImageJ que, como hemos visto durante el proyecto utilizamos a la hora de programar, al proyecto. Para ello utilizaremos la interfaz de file->Project Structure->Dependencies y los añadimos manualmente.

11. Indicar que, una vez compilado el proyecto, el directorio en el que se genere el JAR de salida sea el propio /imagej/plugins, lo que reemplazará el JAR anterior y lo reemplazará por el nuevo. Este paso lo realizaremos en la pestaña file->Project Structure->Artifacts

5 Resultados

Una vez explicadas cada una de las funcionalidades que el alumno ha integrado en el software TomoJ, vamos a realizar un proceso que ponga a prueba cada uno de estos algoritmos para realizar el proceso de reconstrucción completo de una serie de proyecciones bidimensionales.

Tal y como hemos visto a lo largo del proyecto, el alumno ha integrado la posibilidad de trabajar con datos procedentes del campo de tomografía de Rayos-X, algoritmos de preprocesado que permiten al usuario aumentar la calidad de la relación señal a ruido de la serie, la posibilidad de trabajar con series de datos en formato SEL, así como el algoritmo de reconstrucción Tomo3D. Además, ha colaborado en la integración del algoritmo de alineamiento de Xmipp.

Con todo ello, durante este capítulo, realizaremos la reconstrucción de una serie de datos de Rayos-X obtenida en el Sincrotrón Alemán de Electrones de Berlín. Para ello seguiremos cada uno de los pasos que hemos desarrollado a lo largo del proyecto: Importaremos las imágenes de Rayos-X para cargarlas en memoria, preprocesaremos la serie de proyecciones obtenida para aumentar la relación señal a ruido con alguno de los algoritmos implementados por el alumno, alinearemos la serie con el algoritmo de alineamiento de Xmipp integrado y, finalmente, la reconstruiremos a partir del algoritmo Tomo3D.

5.1 Importación de imágenes de TomoX

Para realizar este ejemplo, se ha utilizado una colección de datos procedentes del Sincrotrón Alemán de Electrones de Berlín. Esta colección de datos, para poder ser tratada por el programa TomoJ tiene que estar necesariamente dividida en dos directorios. El primero de ellos será aquél que contenga las imágenes relativas a las proyecciones de la muestra expuesta a Rayos-X desde diferentes ángulos de inclinación (Imagen izquierda de la figura 4.1.1). Cada una de ellas debe ir acompañada de un fichero TXT en el que se encuentren todas las características en las que se encuentre el

microscopio en el momento en el que se toma la proyección, como la temperatura, la inclinación de la muestra y la energía con la que ha sido tomada entre otros.

Además del directorio anterior es necesario contar con otro en el cuál se encuentren las mismas proyecciones que en el anterior tomadas sin la muestra, es decir, únicamente se refleja el haz de electrones en el detector (Imagen derecha de la figura 4.1.1).. Este directorio de *flatfields*, nombre con el que se conoce a este tipo de proyecciones, también deberá contener un fichero TXT con las características del microscopio en el momento de tomar la imagen.

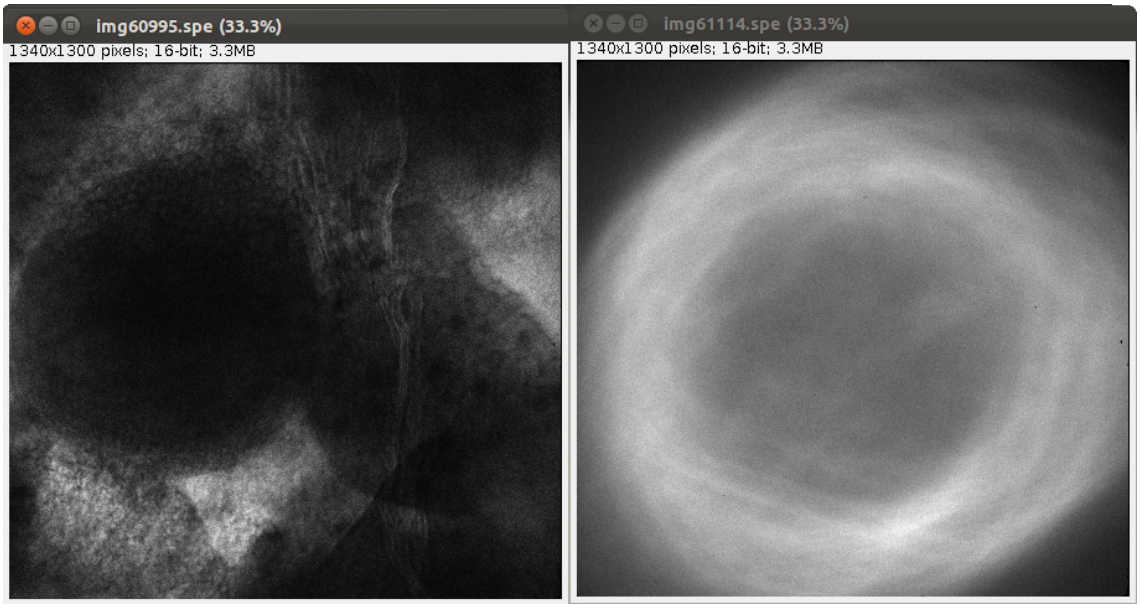


Fig. 4.1.1 A la izquierda se muestra la imagen obtenida por el microscopio de Sincrotrón con la muestra, mientras que a la derecha se muestra la misma proyección sin la muestra expuesta

Durante este paso, obtendremos una serie de proyecciones como Metadata. Para ello, el programa creará un fichero SEL con cada una de las proyecciones con su ángulo de inclinación. Para obtener este resultado el usuario, previa apertura de ImageJ y de TomoJ, deberá pulsar el botón *Import X-Ray* e indicar en qué directorios de su

sistema de ficheros se encuentran los directorios que contienen las proyecciones anteriores (figura 4.1.2)

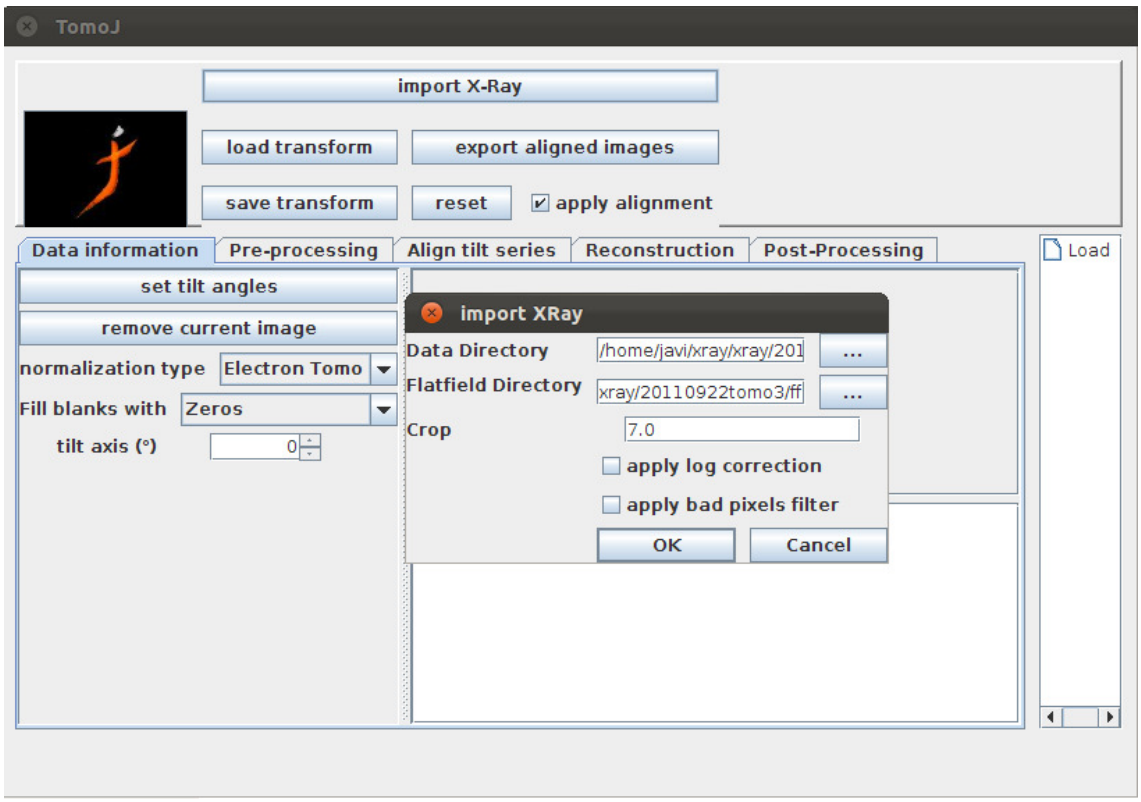


Fig. 4.1.2 Selección de los directorios de datos y flatfields por el usuario

El resultado de realizar esta opción es el de obtener un fichero SEL en el directorio /temporal ubicado en el directorio padre de /datagrama, que, en mi caso era el directorio en el que se encontraban los las proyecciones en las que la muestra estaba expuesta. Este Metadata será cargado en memoria y mostrado al usuario tal y como observamos en la figura 4.1.3

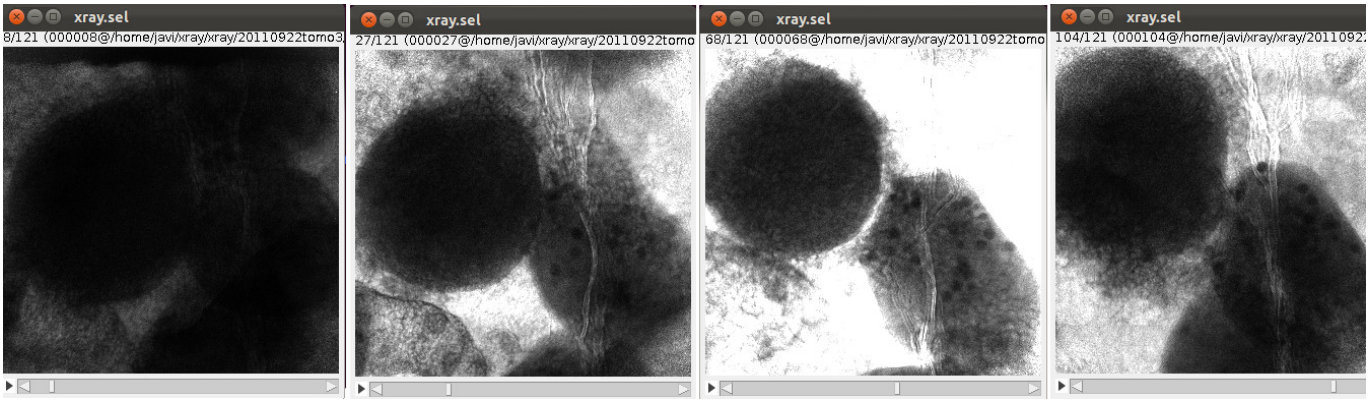


Fig. 4.1.3 Proyecciones 8, 27, 58 y 104 de la serie obtenida a partir de la colección de datos obtenida a partir de TomoX

5.2 Preprocesado de la serie

Una vez obtenida la serie de proyecciones, el siguiente paso a la hora de realizar la reconstrucción tridimensional es el de preprocesar la serie. Tal y como adelantamos en el capítulo [1.4] las imágenes procedentes de TomoX tienen la particularidad de tener una buena relación señal a ruido, por lo que no será necesario un preprocesado agresivo. El filtro que se ha decidido utilizar para llevar a cabo esta reconstrucción ha sido un filtro Gaussiano de Sigma 2.0. Podemos observar los cambios producidos en la serie en la figura 4.2.1 en la que se compara las proyecciones de la serie original con la serie filtrada. Recordaremos en este punto que este filtrado añadir detalles de baja frecuencia, por lo que puede producir un efecto nebuloso en la imagen.

Este tipo de acciones sobre la serie no conlleva cambios en disco, sino que la aplicación del filtro se mantiene en memoria.

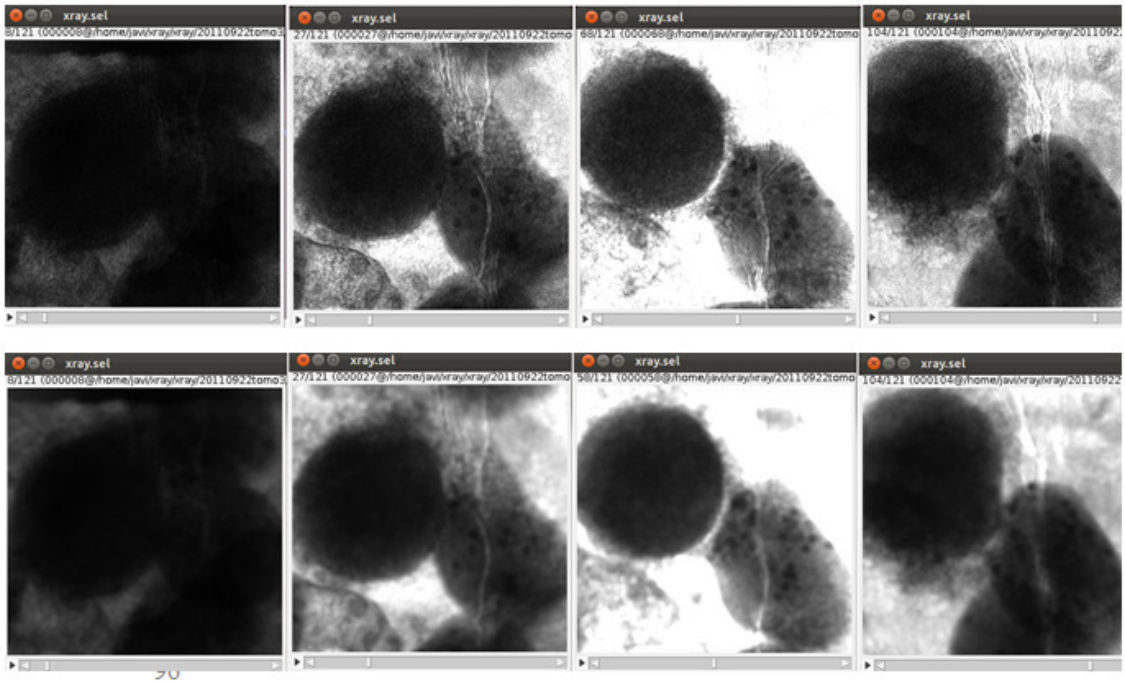


Fig. 4.2.1 En la parte superior la serie original, en la inferior las mismas proyecciones después de aplicar filtro Gaussiano con Sigma (Radius) 2.00

5.3 Alineamiento de la serie

Una vez preprocesada la serie, procederemos a alinearla a través del alineamiento de Xmipp integrado. Este proceso puede llevar a costes temporales muy altos, por esta razón, este ejemplo está siendo realizado con una serie de proyecciones submuestreada con un factor de escala de 0.5.

El resultado de la ejecución de este proceso es el de obtener en disco una serie alineada rotacional y traslacionalmente que será cargada posteriormente en memoria automáticamente. En la parte inferior de la figura 4.3.1 podemos observar la imagen alineada.

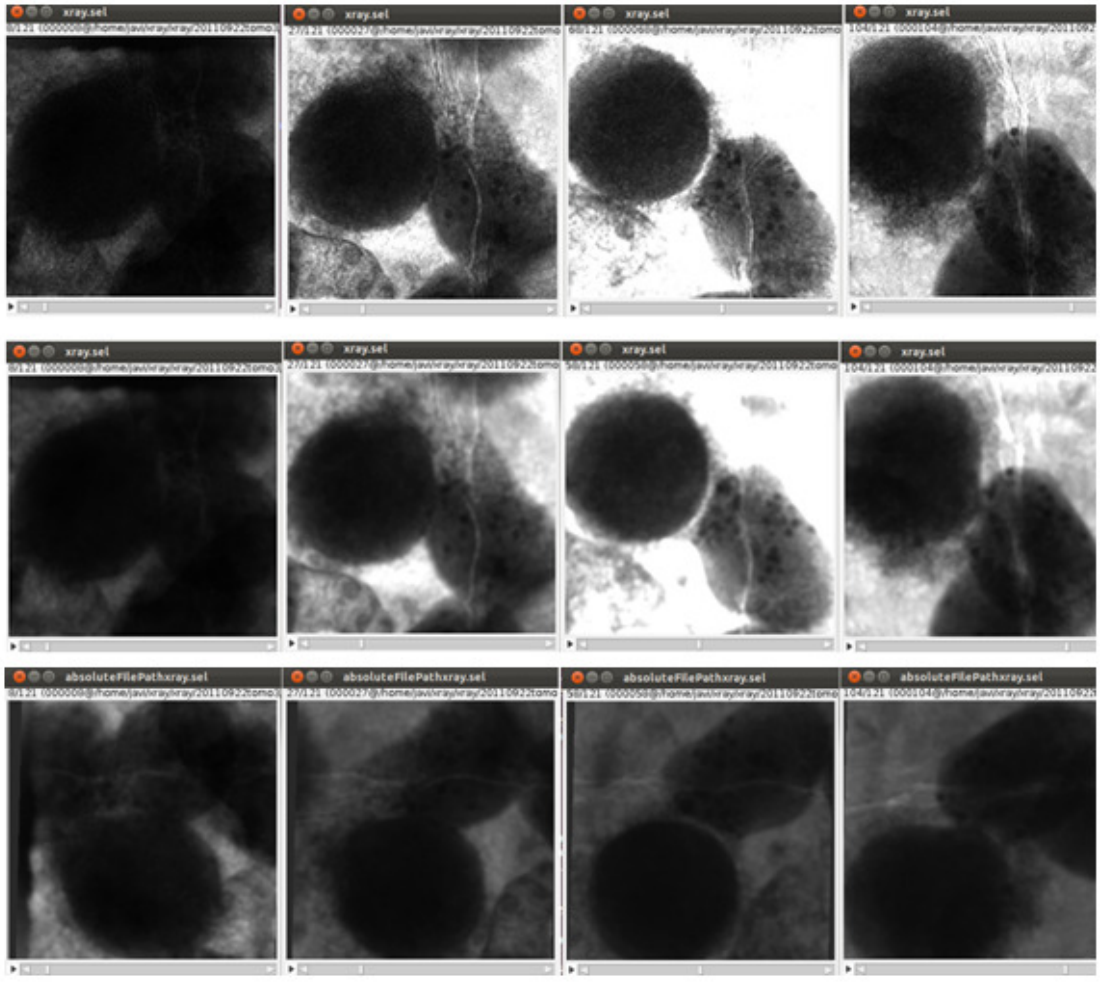


Figura 4.3.1 Imágenes relativas a las mismas proyecciones de la serie cargada (parte superior), preprocesada (parte media de la imagen) y alineada (parte inferior)

5.4 Reconstrucción de la serie con Tomo3D

Una vez alineada la serie realizaremos el proceso de reconstrucción a partir del programa Tomo3D integrado por el alumno. Como ya presentamos en el capítulo [2.5.2], este software aprovecha la tecnología multicore para realizar reconstrucciones con un tiempo computacional muy inferior al de los algoritmos convencionales, permitiendo al usuario, además utilizar el algo WBP o el algoritmo iterativo SIRT. A continuación se va a mostrar el resultado de aplicar el algoritmo *WBP* con una frecuencia inicial para el filtro de Hamming de 0,25 (Figura 4.4.1) y la reconstrucción obtenida a través del algoritmo SIRT empleando para ello 40 iteraciones.

Figura 4.4.1 Reconstrucción obtenida a partir de la serie alineada con el algoritmo *WBP* de Tomo3D

Figura 4.4.2 Reconstrucción obtenida a partir de la serie alineada con el algoritmo *SIRT* de Tomo3D

6 Conclusiones

A través de este proyecto fin de carrera, el alumno ha realizado la integración de una serie de algoritmos de preprocesado y reconstrucción de series de proyecciones bidimensionales tomadas a partir de las técnicas de Tomografía Electrónica de Transmisión y Tomografía de Rayos-X en el programa TomoJ.

Los algoritmos de preprocesado integrados están implementados y disponibles para su uso a través de las librerías de ImageJ y, a su vez, el programa de reconstrucción tomo3D, al igual que las funciones utilizadas para importar imágenes procedentes de Tomo-X, están igualmente disponible en el paquete Xmipp. La dificultad del proyecto radica pues en el estudio previo de cada uno de los programas, métodos y funciones programados previamente de las que el alumno ha hecho uso durante la integración de estos algoritmos y encajar cada una de estas piezas en el software TomoJ.

A través de los algoritmos integrados, el usuario de TomoJ es capaz de realizar el proceso de reconstrucción completo de colecciones de datos procedentes de TEM ó Tomo-X, además, gracias a la integración de Tomo3D se ha reducido considerablemente el coste temporal del paso de reconstrucción, permitiendo obtener reconstrucciones del orden de segundos en el caso del algoritmo *WBP* y del orden de muy pocos minutos en el caso de *SIRT*.

Pese a la integración de Tomo3D, el proceso de reconstrucción sigue acarreando un coste temporal elevado, causado principalmente por el paso de alineamiento de la serie, que actúa como cuello de botella. En la actualidad, el equipo de desarrollo de Xmipp está trabajando para aprovechar la tecnología multicore de los procesadores al igual que realiza el programa Tomo3D para reducir el tiempo de ejecución del proceso.

7 Bibliografía

i. C. Messaoudi, T. Boudier, C.O.S. Sorzano, S. Marco. *TomoJ: tomography software for 3D reconstruction in transmission electron tomography*. BMC Bioinformatics 8, 288 (2007)

ii. C.O.S. Sorzano, S. Jonic, M. Cottevieille, E. Larquet, N. Boisset, S. Marco. *3D Electron microscopy of biological nanomachines: principles and applications*. European Biophysics Journal, 36: 995-1013 (2007)

iii. C.O.S.Sorzano, R. Marabini, J. Velázquez-Muriel, J.R. Bilbao-Castro, S.H.W. Scheres, J.M. Carazo, A. Pascual-Montano. *XMIPP: a new generation of an open-source image processing package for Electron Microscopy*. Journal of Structural Biology 148: 194-204 (2004)

iv. C.O.S.Sorzano, *Algoritmos Iterativos de Tomografía Tridimensional en Microscopía Electrónica de Trnasmisión*. (2002)

v. J. Otón, C.O.S. Sorzano, E. Pereiro, J. Cuenca, R. Navarro, R. Marabini, J.M. Carazo. *Image formation in cellular X-ray microscopy*. J. Structural Biology, 178: 29-37 (2012)

vi. S. Jonic, C.O.S. Sorzano, N. Boisset. *Comparison of single-particle analysis and electron tomography approaches: an overview*. Journal of Microscopy, 232: 562-579 (2008)

vii. C.O.S. Sorzano, C. Messaoudi, M. Eibauer, J.R. Bilbao-Castro, R. Hegerl, S. Nickell, S. Marco, J.M. Carazo. *Marker-free image registration of electron tomography tilt-series*. BMC Bioinformatics, 10: 124 (2009)