Universidad Rey Juan Carlos

OPTIMIZACIÓN DE PROCESOS DE AJUSTE EN MICROSCOPÍA ELECTRÓNICA Y CRIBADO VIRTUAL DE PROTEÍNAS MEDIANTE ARQUITECTURAS GRÁFICAS

TESIS DOCTORAL

Santiago García Sánchez

2015

Universidad Rey Juan Carlos

Escuela Técnica Superior de Ingeniería Informática

Departamento de Ciencias de la Computación, Arquitectura de la Computación, Lenguajes y Sistemas Informáticos y Estadística e Investigación Operativa

OPTIMIZACIÓN DE PROCESOS DE AJUSTE EN MICROSCOPÍA ELECTRÓNICA Y CRIBADO VIRTUAL DE PROTEÍNAS MEDIANTE ARQUITECTURAS GRÁFICAS

TESIS DOCTORAL

Directores:

Dr. D. Pablo Chacón Montes

Dr. D. Raúl Cabido Valladolid

Tutor:

Dr. D. Antonio Sanz Montemayor

Doctorando:

D. Santiago García Sánchez

2015

El Dr. D. Pablo Chacón Montes, Científico Titular del Consejo Superior de Investigaciones Científicas y el Dr. D. Raúl Cabido Valladolid, Profesor Colaborador Honorífico del departamento de Ciencias de la Computación, Arquitectura de la Computación, Lenguajes y Sistemas Informáticos y Estadística e Investigación Operativa de la Universidad Rey Juan Carlos, codirectores de la Tesis Doctoral *"Optimización de procesos de ajuste en microscopía electrónica y cribado virtual de proteínas mediante arquitecturas gráficas"* realizada por el doctorando D. Santiago García Sánchez,

HACEN CONSTAR:

que esta Tesis Doctoral reúne los requisitos necesarios para su defensa y aprobación.

En Móstoles, a 12 de Marzo de 2015,

Dr. D. Pablo Chacón Montes Dr. D. Raúl Cabido Valladolid

Dr. D. Antonio Sanz Montemayor
(Tutor)

Índice general

Índice de figuras

Índice de tablas

Acrónimos

ADN Ácido DesoxirriboNucleico

AUC Area Under the Curve

AVX Advanced Vector Extensions

CCD Charge Coupled Device

CPU Central Processing Unit

CTF Contrast Transfer Function

CUDA Compute Unified Device Architecture

DUD Directory of Useful Decoys

DUD-E Directory of Useful Decoys - Enhanced

FBM Fast Bessel Matching

FFT Fast Fourier Transform

FPM Fast Projection Matching

FPR False Positive Rate

FRM2D Fast Rotational Matching 2D

FRM3D Fast Rotational Matching 3D

FRODRUG Fast Rotational DRUG

GPGPU General-Purpose computing on the Graphics Processing Unit

GPU Graphics Processing Unit

HPC High Performance Computing

IPC Instructions Per Cicle

MPI Message Passing Interface

PDB Protein Data Bank

PFT Polar Fourier Transform

RAM Random Access Memory

RMSD Root Mean Square Deviation

RNA RiboNucleic Acid

RPC Resampling to Polar Coordinates

ROC Receiver Operating Characteristic

SCF Self Correlation Function

SNR Signal Noise Ratio

SSE Streaming SIMD Extensions

TPR True Positive Rate

VRAM Video Random Access Memory

A mis padres

Agradecimientos

En primer lugar me gustaría dar las gracias a mis directores de tesis: Pablo y Raúl. Este trabajo no habría sido lo que es sin vuestra ilimitada paciencia.

Me gustaría dar las gracias a todas esas buenas gentes que me han acompañado día a día a lo largo de este viaje. Ellos son Erney, Mon, Pieter, y durante un breve pero intenso momento, Nacho. Así mismo me gustaría ofrecer mi agradecimiento a los compañeros que me dieron una cálida acogida en el CIB: Benet, Eli, David, Chiara, Laura, Ruth, Gloria y Claudia. También a los compañeros que nos recibieron cuando nos trasladamos al IQFR: Yasmina, Diana, Santi, Nerea, Soraya, Flor y Miguel. Querría ofrecer un agradecimiento especial a Manu, Bego, Ioanna, Lara y Lupe por el tiempo que habéis compartido conmigo. Un saludo también a todo el grupo de cristalografía, en especial a María Ángela, Mercedes, Antonio, María y Yanaisis. Gracias también a Adi, Chiara, Luis, Elisa, Dani y Roberta por todas las horas de comidas, charlas y cafés.

Quisiera agradecer a J.J. Pantrigo, A. Sanz, P. de las Heras, J. Centeno y F.J. Ballesteros la excelente educación universitaria que me han proporcionado. No habría llegado hasta aquí de no ser por vosotros.

Esta página no podría estar completa sin nombrar a Marisol. Gracias a ti todo esto ha sido posible. Gracias por animarme a seguir, por todo el apoyo que me has dado, y por tantas otras cosas que no hay página de agradecimientos que pueda cubrirlo. También quiero darte las gracias a ti, César, porque siempre has estado ahí.

Gracias a ti, María, por toda tu ayuda, tu tiempo y tu cariño. Tu apoyo y comprensión han sido inestimables durante la escritura de este libro.

Por último quiero ofrecer un agradecimiento muy especial a todos lo que han estado apoyándome durante la escritura de este documento y en general durante la vida. Gracias a mis padres, hermano, abuelos y tíos.

Resumen

En el campo de la bioinformática existe un elevado número de procesos que requieren de una alta capacidad computacional. Entre ellos se encuentran el alineamiento de imágenes de microscopía electrónica y el cribado virtual de proteínas. El alineamiento de imágenes es un subproceso empleado en la reconstrucción de volúmenes tridimensionales de moléculas observadas con el microscopio electrónico. Su propósito es el de ajustar las imágenes experimentales en una misma posición y orientación para luego promediarlas y así reducir el ruido que poseen y mejorar la cantidad de señal. Normalmente este proceso implica realizar millones de alineamientos entre imágenes. Por otra parte, el cribado virtual de proteínas es un proceso clave en el descubrimiento de nuevos fármacos. Su objetivo es filtrar los compuestos químicos que interaccionan con cierta proteína objetivo de los que no. Cada compuesto puede evaluarse en millones de posiciones y orientaciones diferentes. Además, sobre una misma proteína objetivo se prueban típicamente del orden de millones de compuestos químicos diferentes.

Actualmente se dispone de una gran variedad de dispositivos capaces de hacer cómputo de altas prestaciones mediante los cuales se pueden resolver de forma eficiente los problemas planteados. En concreto las tarjetas gráficas actuales ofrecen un alto rendimiento a un coste reducido en comparación con los tradicionales supercomputadores.

Este trabajo de tesis se centra en la adaptación de los dos procesos expuestos a arquitecturas gráficas. Con respecto al alineamiento de imágenes de microscopía electrónica se propone la adaptación de un nuevo método denominado *Fast Bessel Matching*. Por otra parte, para realizar el cribado virtual de proteínas de un modo más eficiente se propone la adaptación de un nuevo método denominado FRODRUG. De cada procedimiento se han realizado tres adaptaciones: una secuencial para CPUs, otra paralela para GPUs y por último una para sistemas de memoria distribuida.

De la investigación realizada en esta tesis se ha podido concluir que *Fast Bessel Matching* es capaz de realizar alineamientos correctos con un error subpixel. La versión GPU de este método alinea casi 50 veces más rápido que su correspondiente versión secuencial. Por otra parte, FRODRUG es capaz de realizar el cribado virtual de proteínas con éxito, identificando muchos de los ligandos que se unen a una proteína dada y desechando gran cantidad de los que no lo hacen. La versión paralela en GPU de FRODRUG lleva a cabo el cribado hasta 30 veces más rápido con respecto a su adaptación secuencial.

Abstract

In bioinformatics there are several processes that require high computing power. Alignment of electron microscopy images and virtual screening are among them. The alignment of electron microscopy images is a subprocess used in the reconstruction of three-dimensional volumes of molecules. Its purpose is to place the experimental images in the same position and orientation and then average them to reduce its noise and improve the amount of signal. Usually this process involves performing millions of alignments between images. On the other hand, virtual screening is a key process in drug discovery. Its purpose is to filter the chemical compounds that interact with certain target proteins from those not. Each compound can be tested in millions of different positions and orientations. Furthermore, on the same target protein are tested typically millions of different chemical compounds.

Nowadays there is a variety of devices capable of doing high performance computing by which this problems can be efficiently addressed. Specifically, current graphic cards offer high performance at a reduced cost compared to traditional supercomputers.

This thesis focuses on the adaptation of the two exposed processes to graphic architectures. Regarding to the alignment of electron microscopy images, the adaptation of a new method named Fast Bessel Matching is proposed. On the other hand, to perform a more efficient virtual screening, an adaptation of a new method named FRODRUG is proposed. For each procedure, three adaptations have been done: a secuential one for CPUs, a parallel one for GPUs and a last one for distributed memory systems.

From the research conducted in this thesis has been concluded that FBM is able to perform accurate alignments with subpixel error. The GPU version of this method aligns up to 50 times faster than the corresponding sequential version. On the other hand, FRODRUG is capable of performing virtual screening successfully identifying many of the ligands which bind to a given protein and discarding many of which do not. The parallel version of FRODRUG performs up to 30 times faster compared to its sequential adaptation.

Capítulo 1

Motivación

La bioinformática es la aplicación de tecnologías de computación sobre datos pertenecientes al campo de la biología. En este ámbito existen una gran cantidad de procesos que precisan de herramientas informáticas para poder ser llevados a cabo. Más en concreto, dentro del campo de la biología se encuentra la biología estructural cuyo propósito es la investigación de la estructura molecular. Esta rama alberga estudios relacionados con la determinación de la forma de las moléculas a través de imágenes de microscopía electrónica ó la interacción de estructuras moleculares. Dentro de estas dos aplicaciones existen problemas concretos que requieren de una alta capacidad computacional para ser resueltos: el alineamiento de imágenes de microscopía [48] y el cribado virtual de proteínas [233].

El alineamiento de imágenes es un subproceso empleado en la reconstrucción de volúmenes tridimensionales de moléculas observadas con el microscopio electrónico. El propósito del alineamiento es que las imágenes puedan ser luego promediadas para reducir la gran cantidad de ruido que poseen. La reconstrucción del volumen tridimensional se lleva a cabo a través de un proceso probabilístico iterativo: se alinean las imágenes de las distintas proyecciones de la molécula, se promedian y se utilizan para contribuir a la mejora del modelo 3D de la molécula. Esto se repite la cantidad de veces que sea necesario hasta alcanzar un modelo 3D adecuado. Normalmente se necesita realizar millones de alineamientos entre imágenes. Esto convierte al proceso de alineamiento en uno de los cuellos de botella de la reconstrucción.

El cribado de proteínas es un proceso clave en el descubrimiento de nuevos fármacos. Su objetivo es filtrar los compuestos químicos que interaccionan con cierta proteína objetivo de los que no. Tradicionalmente, el cribado se realiza con grandes equipos robotizados que prueban experimentalmente compuestos químicos sobre la proteína objetivo. Este es un proceso costoso tanto en tiempo como económicamente (equipos caros, mantenimiento, reactivos, etc...). Como alternativa a este proceso surge el cribado virtual, que es un proceso computacional que persigue el mismo objetivo. Las simulaciones de ajuste molecular pueden llegar a probar millones de compuestos químicos. Además cada compuesto puede evaluarse en más de 2.500 posiciones y más de 15.000 orientaciones diferentes, lo que da lugar a la evaluación de más de 37 millones de posiciones relativas por compuesto químico.

Actualmente se dispone de una gran variedad de dispositivos capaces de hacer cómputo

de altas prestaciones. En concreto las tarjetas gráficas de consumo cotidiano ofrecen un alto rendimiento a un precio muy bajo en comparación con los tradicionales supercomputadores. Este trabajo de tesis se centra en la aplicación de tecnologías de cómputo de altas prestaciones sobre estos dos procesos del campo de la biología estructural, y más en concreto en su adaptación a arquitecturas gráficas. Ambos procedimientos muestran las características apropiadas para ser adaptados sobre dispositivos gráficos: la cantidad de datos que necesitan para trabajar es bajo y son procesos que pueden dividirse en pequeñas partes individuales. El alineamiento de una pareja de imágenes es independiente del de otra, así como una proteína puede ajustarse simultáneamente con varios compuestos químicos de forma independiente. Con el fin de reducir el cuello de botella relativo al alineamiento de millones de imágenes de microscopía se busca adaptar el modelo matemático de un nuevo método de alineamiento, denominado *Fast Bessel Matching*, propuesto por Kovacs *et al.* [109]. Por otro lado, para mejorar la actual eficiencia y precisión en el campo del cribado virtual de proteínas se busca adaptar un nuevo procedimiento, denominado FRODRUG, basado en método de ajuste molecular realizado por Garzón *et al.* [61, 62].

Capítulo 2

Hipótesis y objetivos

La hipótesis que se establece como base de la investigación desarrollada en este trabajo de tesis es que la exigencia computacional de los procesos de ajuste de imagen de microscopía electrónica y de cribado virtual de proteínas se puede reducir sustancialmente mediante la aplicación de sistemas de computación de altas prestaciones, y en particular con tarjetas gráficas. Por tanto, el objetivo principal de esta tesis es:

La adaptación y optimización de los procesos de ajuste de imagen de microscopía electrónica y de cribado virtual de proteínas en arquitecturas gráficas de altas prestaciones y bajo coste.

Para lograr este objetivo principal, es necesario alcanzar los siguientes objetivos operativos:

1. Proponer la adaptación de un nuevo método de ajuste 2D para el alineamiento de imágenes de microscopía electrónica:
   1.1. Adaptar, validar y optimizar sobre CPU y GPU el nuevo método de ajuste 2D *Fast Bessel Matching*, basado en transformadas de Fourier-Bessel.
   1.2. Comparar el rendimiento y la precisión de *Fast Bessel Matching* sobre dispositivos gráficos con la versión de CPU equivalente y con otros métodos de ajuste.
2. Proponer la adaptación de un nuevo método de ajuste 3D para el cribado virtual de proteínas:
   2.1. Adaptar, validar y optimizar sobre CPU y GPU el nuevo método de ajuste 3D FRODRUG, basado en armónicos esféricos.
   2.2. Comparar el rendimiento y la precisión de FRODRUG sobre dispositivos gráficos con la versión de CPU equivalente y con otros métodos.

Capítulo 3

Introducción

Este trabajo de tesis se centra en la aplicación de arquitecturas de cómputo de altas prestaciones sobre dos técnicas concretas del campo de la bioinformática: microscopía electrónica [48] y cribado virtual de proteínas (*virtual screening*) [233]. A continuación se introducen brevemente los fundamentos básicos de estas técnicas.

3.1. Problemas de ajuste 2D y 3D en bioinformática

La microscopía electrónica permite caracterizar la estructura tridimensional de las macromoléculas biológicas a media/baja resolución a partir de proyecciones 2D de densidad electrónica (ver Figura 3.1). Para ello se hace uso de métodos de procesamiento de imagen. En ese contexto se realizan varios procesos críticos como el alineamiento de imágenes 2D y la reconstrucción 3D a partir de dichas imágenes [56]. El gran número de proyecciones 2D (millones en algunos casos) y el carácter iterativo de estos métodos justifica el empleo de aceleración mediante GPU.

Por otra parte, existe la necesidad de desarrollar fármacos seguros e innovadores. La selección por medios computacionales de posibles candidatos a fármaco de entre las extensas librerías de compuestos representa una rápida y efectiva aproximación. En particular, el *virtual screening* basado en receptor se está convirtiendo en una parte clave de este proceso. Este procedimiento

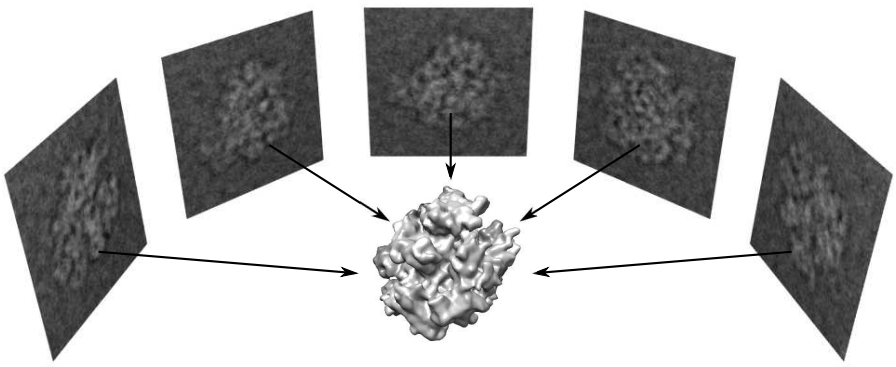


Figura 3.1: Reconstrucción 3D de una macromolécula a partir de proyecciones 2D de densidad electrónica.

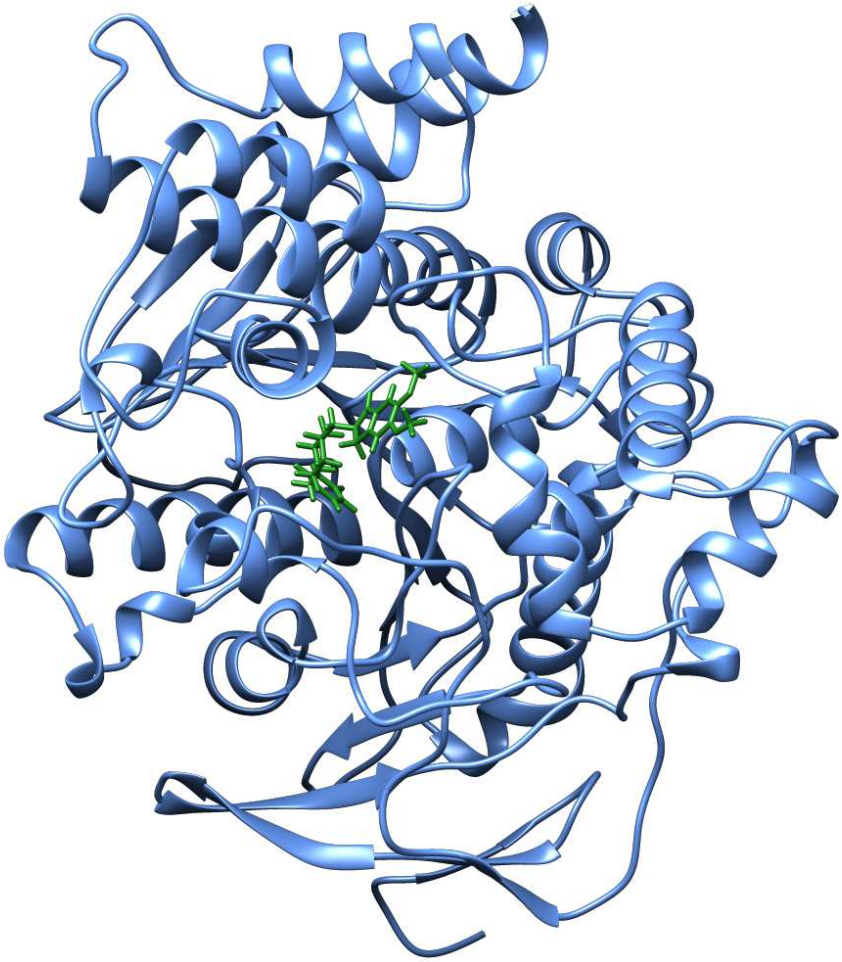


Figura 3.2: Ajuste molecular de un ligando (verde) en el sitio de unión de una proteína (azul).

depende de métodos de *docking* para realizar el cribado computacional. Los métodos de *docking* proteína-ligando intentan identificar la posición, orientación y conformación óptimas de un pequeño compuesto (ligando) con respecto a la proteína objetivo con una estructura 3D conocida (receptor) (ver Figura 3.2). Por cada ligando se pueden llegar a probar más de 2.500 posiciones espaciales y más de 15.000 rotaciones, lo que hacen un total de más de 37 millones de posiciones. Además en un proceso típico de cribado virtual se pueden llegar a probar más de 10^6 compuestos. Esto convierte el cribado virtual en un proceso costoso y en un objetivo perfecto para estudios en GPU.

A continuación se comentarán brevemente los principios básicos de estos problemas de ajuste 2D y 3D.

3.2. Microscopía electrónica y análisis de partículas individuales

La microscopía electrónica de partícula única se ha convertido en la técnica más potente en biología estructural para el estudio de grandes macromoléculas y sus complejos. En condiciones cercanas a las fisiológicas, la estructura 3D de complejos biológicos relevantes puede ser determinada desde una resolución subnanómetro hasta atómica [56, 133]. Usando una pequeña

cantidad de muestra purificada, los investigadores procesan y promedian las proyecciones 2D de las macromoléculas recolectadas en el microscopio electrónico para obtener una reconstrucción 3D. A continuación se describe el proceso general empleado para obtener el modelo 3D de una partícula.

3.2.1. Procesamiento de imagen y reconstrucción 3D

El proceso de reconstrucción de partícula individual se divide, en general, en varios pasos [164]. En un primer paso, numerosas imágenes de una única molécula son obtenidas de múltiples medidas de microscopía electrónica. Estas imágenes corresponden a proyecciones 2D de densidad electrónica de la molécula en diferentes orientaciones aleatorias. Al ser estas imágenes extremadamente ruidosas, son alineadas y clasificadas por similitud para reducir el ruido. De hecho, la reconstrucción 3D de partículas individuales es un procedimiento iterativo de alineamiento y clasificación, donde las imágenes promediadas, producidas mediante clasificación, son usadas como imágenes de referencia para subsiguientes pasos de refinamiento. Por lo tanto, el alineamiento, que determina la calidad de la reconstrucción 3D, es repetido en múltiples ocasiones. Tal alineamiento típicamente maximiza una sencilla función de correlación (un producto escalar de los valores de densidad electrónica almacenados en las imágenes 2D) entre las ruidosas imágenes experimentales y las imágenes de referencia. Además, el tiempo de cálculo para la reconstrucción 3D aumenta con el número de imágenes, convirtiéndose en un cuello de botella para estudios de alta resolución. En un microscopio convencional usualmente se necesitan más de 10^6 proyecciones para obtener alta resolución [68]. Recientemente se han investigado técnicas para mejorar la calidad de los detectores de los microscopios. Esto deriva en una mejora en el contraste de la imagen, haciendo que se necesiten menos imágenes para una reconstrucción de alta resolución [9]. Aún así, el proceso completo puede requerir muchas horas de cálculo en un clúster. En resumen, el alineamiento es un paso relativamente costoso y crítico que controla la eficiencia y precisión de la reconstrucción 3D de microscopía electrónica.

En la actualidad existen numerosos paquetes de programas de microscopía que realizan el proceso completo de reconstrucción 3D [244]: IMIRS [113], EMAN [119], SPIDER [57, 242], FREALIGN [67, 112], XMIPP [121, 204] ó AUTO3DEM [241] entre otros.

A continuación de detallan cada uno de los tres pasos en los que se puede dividir de forma clara el proceso de reconstrucción de partícula individual: obtención de datos, alineamiento y reconstrucción.

3.2.1.1. Obtención de imágenes

La obtención de los datos de las partículas se realiza a través de un microscopio. Los microscopios pueden clasificarse en tres tipos: ópticos, de transmisión y de sonda. Los que se emplean para tomar las imágenes de complejos moleculares son los de transmisión de electrones [180]. Esta clase de microscopios utiliza un haz de electrones para formar las imágenes junto con lentes electromagnéticas para conseguir el enfoque; pueden llegar a lograr más de cuatro millones de aumentos.

Tradicionalmente las imágenes de los microscopios se tomaban en película fotográfica, pero

este proceso resultaba muy lento. El tiempo de obtención y tratamiento de las imágenes se vio mejorado en gran medida gracias a la inclusión de detectores CCD. Estos permiten registrar gran cantidad de datos muy rápidamente [205], llegando a obtener imágenes de más de 11.000 partículas por hora. En la actualidad existen sensores, como el FEI Falcon, que detectan directamente los electrones y mediante los cuales es posible obtener imágenes de mejor calidad e incluso vídeo [9]. Así mismo, el proceso de toma de datos ha sido automatizado en gran medida.

La calidad de las imágenes obtenidas por los microscopios electrónicos es un factor fundamental para una reconstrucción 3D fidedigna. Estas imágenes tienen normalmente un contraste muy bajo y presentan mucho ruido, lo que dificulta la identificación de la partícula. Principalmente esto se debe a la baja dosis de electrones lanzada sobre la muestra [13]. Una dosis más elevada haría que las partículas se desintegraran. No obstante, es posible realizar un filtrado previo de las imágenes con el fin de reducir parte de su ruido. Este proceso puede llevarse a cabo empleando diferentes métodos. Algunos de los filtros más utilizados son el bilateral [96], el de difusión anisotrópica no lineal [55, 50], el iterativo de medianas [221] ó el de promedios no locales [21, 237].

El microscopio proporciona una imagen de gran tamaño que incluye todas las partículas observadas. Antes de poder hacer uso de estas es necesario extraerlas en imágenes individuales. El proceso de identificación y selección de partículas puede ser automatizado empleando paquetes de *software* especialmente destinados a llevar a cabo esta función [170, 230]. Una vez se dispone de imágenes individuales de las partículas es posible continuar al siguiente paso: alineamiento y clasificación.

3.2.1.2. Alineamiento y clasificación de imágenes

Como se ha expuesto anteriormente, las proyecciones 2D de las partículas obtenidas en el microscopio tienen gran cantidad de ruido y es necesario promediarlas para mejorar su señal. Antes de realizar el promediado es necesario alinearlas. El alineamiento de imágenes 2D en análisis de partícula individual puede considerarse como un problema modelo de registro de imágenes mediante un patrón. Las referencias son imágenes promediadas obtenidas utilizando clusterización, o bien generadas como proyecciones 2D a partir de una estructura 3D de referencia [191, 190, 199]. Las proyecciones experimentales de microscopía electrónica son comparadas con las imágenes de referencia para encontrar los parámetros de alineamiento que producen un ajuste más preciso. Una vez alineadas todas las imágenes correspondientes a una misma proyección, se promedian para reducir el ruido. Existen diversos procedimientos para realizar este paso que serán presentados más adelante. Una vez las imágenes se han alineado y promediado se emplean para realizar la reconstrucción 3D de la partícula.

3.2.1.3. Reconstrucción 3D

El proceso de reconstrucción 3D [38] consiste en generar el volumen correspondiente a la partícula observada a partir de las proyecciones 2D que se poseen de esta. Esto se realiza a través de un procedimiento iterativo en el que las proyecciones promediadas de múltiples orientaciones contribuyen a mejorar el volumen de la partícula (ver Figura 3.3). El paso de alineamiento y

clasificación de las proyecciones 2D se repite antes de cada contribución al volumen. Esto se debe a que las proyecciones 2D pueden ser asignadas incorrectamente al inicio del proceso y ser reasignadas a una orientación correcta según el modelo de la partícula evoluciona.

Existen gran variedad de métodos para llevar a cabo la reconstrucción del modelo 3D a partir de las proyecciones 2D. Algunos de los más importantes son el método de líneas comunes de Crowther [35, 34], que emplea el dominio de frecuencias para acelerar los cálculos, el de retroproyección con pesos de Radermacher [171] ó el de expectación-maximización [213]. Recientemente, Estrozi y Navaza han desarrollado un método [45] complementando un método anterior basado en armónicos esféricos [140] con otro llamado *Fast Projection Matching* [46]. Existen también métodos de reconstrucción 3D que hacen uso de funciones de simetría icosaédrica para reducir el efecto del ruido en las imágenes [118]. Sin embargo, esta clase de métodos sólo pueden utilizarse con imágenes de microscopía que presenten simetría, como las de virus icosaédricos.

3.2.2. Métodos de ajuste 2D

Los actuales paquetes de procesamiento de imagen de microscopía electrónica usan diferentes métodos de alineamiento para llevar a cabo eficientemente el registro 2D [100, 178]. Las aproximaciones más populares son *Self Correlation Function* (SCF) [222], *Resampling to Polar Coordinates* (RPC) y *Fast Rotational Matching 2D* (FRM2D). Aunque SCF es mucho más eficiente que RPC, es menos robusto contra el ruido [100, 29]. Para mejorar su precisión, los investigadores han añadido un paso de refinamiento local. SCF con el protocolo de refinamiento es el procedimiento de alineamiento estándar más rápido. Alternativamente, el método *Fast Rotational Matching 2D* (FRM2D) mantiene la precisión de RPC y aún es competitivo en tiempo con SCF [29, 30] siendo tan sólo dos veces más lento. Existen también otros métodos que no realizan el ajuste de las imágenes de microscopía contra un conjunto de referencia sino utilizando directamente proyecciones 3D de la partícula; los métodos más conocidos que siguen este modelo de trabajo son *Fast Projection Matching* (FPM) y *Polar Fourier Transform* (PFT). Recientemente, Kovacs y colaboradores [109] han propuesto el modelo matemático de un nuevo método, conocido como *Fast Bessel Matching* (FBM), basado en correlación en el dominio de frecuencias para llevar a cabo el paso de alineamiento 2D. En la primera parte de este trabajo de tesis se investiga sobre la adaptación de este novedoso método, que se sugiere como más eficiente que los utilizados actualmente, a arquitecturas de cómputo secuencial. Además, para aumentar su rendimiento más aún, se propone también su adaptación a sistemas de cómputo de altas prestaciones. En este contexto se han investigado dos aproximaciones: una para sistemas de memoria distribuida que emplea *Message Passing Interface* (MPI) y otra para dispositivos gráficos empleando la arquitectura CUDA [151]. A continuación se ilustra brevemente el modo en que trabajan los métodos de ajuste 2D expuestos.

3.2.2.1. *Resampling to Polar Coordinates* (RPC)

Este método muestrea una imagen fija en el espacio de coordenadas polares con respecto a algunas posiciones de la otra imagen. Por medio del teorema de convolución de Fourier, el ángulo rotacional entre proyecciones se determina a partir de una transformada de Fourier 1D,

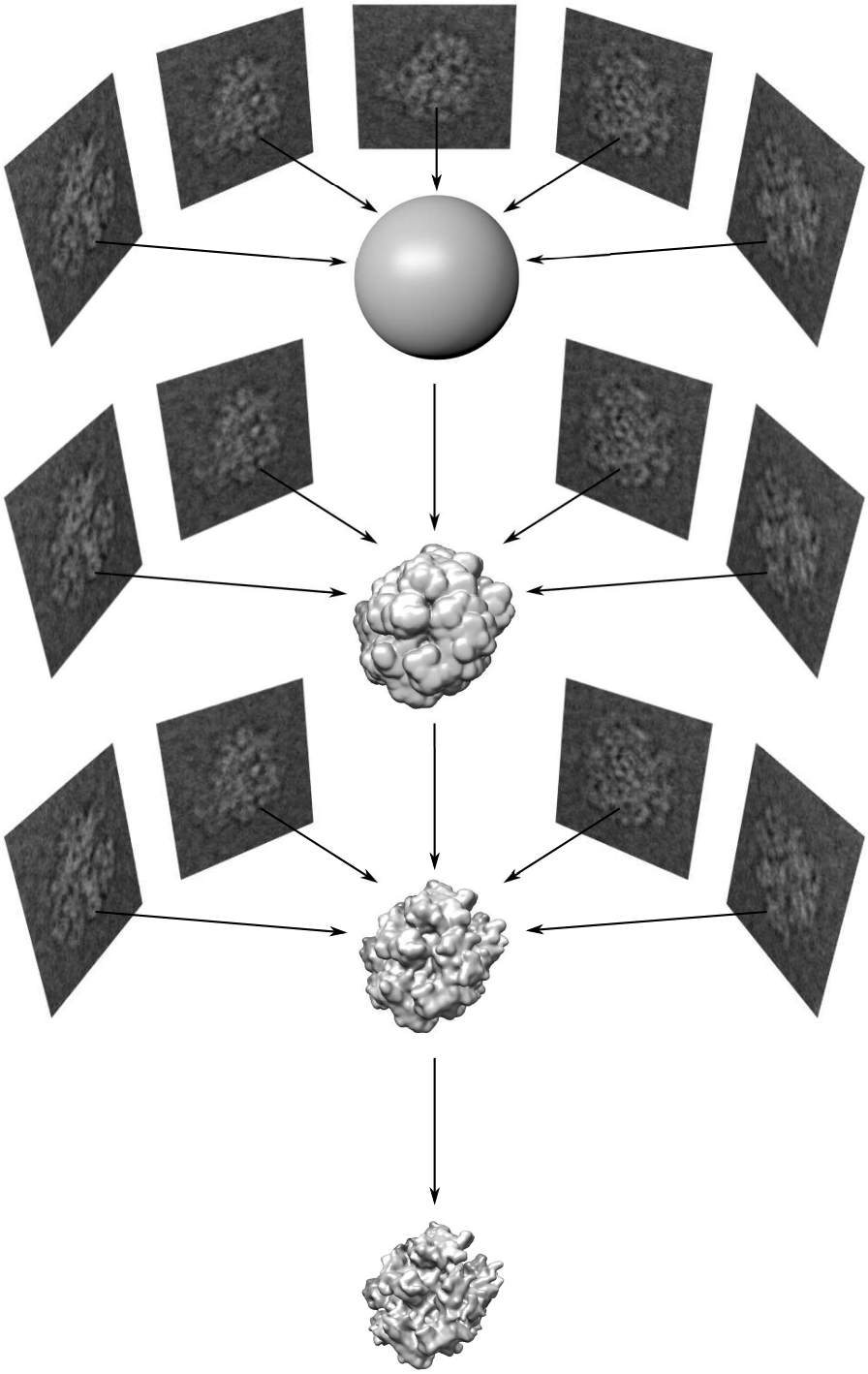


Figura 3.3: Reconstrucción 3D de una macromolécula a partir de proyecciones de microscopía electrónica mediante un proceso iterativo.

donde se escanean dos parámetros traslacionales (desplazamientos x e y).

3.2.2.2. *Self Correlation Function* (SCF)

SCF es un método que trabaja en el espacio de frecuencias y separa rotación y traslación. El ángulo rotacional es computado del mismo modo que el método RPC, pero utilizando una función de correlación mutua en lugar de la estándar cross-correlación de densidad. Los parámetros traslacionales se obtienen mediante una transformada de Fourier 2D que acelera en gran medida el alineamiento.

3.2.2.3. *Fast Projection Matching* (FPM)

Fast Projection Matching [46] es un método de alineamiento que tiene una aproximación muy distinta a los expuestos anteriormente. Este método no alinea un conjunto de imágenes experimentales con un conjunto de imágenes de referencia sino que las alinea directamente con proyecciones de un modelo 3D. Por cada imagen experimental alineada se obtienen cinco parámetros; tres corresponden con los ángulos que orientan el modelo 3D de la partícula y los otros dos proporcionan el centro de la imagen experimental.

3.2.2.4. *Polar Fourier Transform* (PFT)

El método *Polar Fourier Transform* [11] es similar a FPM en el aspecto de que ambos emplean un volumen 3D para generar las imágenes de referencia contra las que se alinea. Este otro método sin embargo difiere en que requiere que las partículas alineadas posean simetría (como los virus icosaédricos). Al contrario que otros métodos, PFT es muy poco sensible al ruido.

3.2.2.5. *Fast Rotational Matching 2D* (FRM2D)

En FRM2D el problema de alineamiento se reformula en un grado de libertad traslacional y dos rotacionales. En lugar de fijar una imagen mientras se traslada y rota la otra, ambas imágenes son rotadas y una de ellas es trasladada a través del eje formado entre los centros de ambas. Este método acelera la estimación de las dos rotaciones mediante transformadas de Fourier, mientras que el restante parámetro traslacional es explorado sistemáticamente [29, 30]. FRM2D ha sido implementado en el paquete EMAN y ha sido usado satisfactoriamente en reconstrucciones 3D de alta resolución [31]. Este método puede también ser considerado como una versión 2D de *Fast Rotational Matching 3D*, que se ha empleado con éxito para predecir interacciones de proteínas [62] y para ajustar estructuras atómicas en mapas de densidad de baja resolución de microscopía electrónica [61].

3.2.2.6. *Fast Bessel Matching* (FBM)

En FBM el problema de emparejamiento se reformula en tres parámetros angulares que se pueden estimar mediante una única transformada 3D de Fourier. Para acelerar los tres grados de libertad rotacionales, la función de correlación se expresa en términos de la transformada de Fourier-Bessel de las dos imágenes de proyección. Las estimaciones teóricas de FBM muestran

una complejidad inferior a los actuales métodos de alineamiento 2D [109]. Estos resultados sugieren FBM como el método óptimo para alineamiento de imágenes de microscopía electrónica muy ruidosas.

3.3. Cribado virtual para el descubrimiento de nuevos fármacos

La necesidad de desarrollar fármacos seguros e innovadores apunta hacia la mejora en las fases tempranas del descubrimiento de fármacos. Tradicionalmente, el cribado de proteínas se ha realizado experimentalmente empleando robots especializados [4]. Aunque este proceso ha evolucionado a lo largo del tiempo, sigue teniendo inconvenientes: la baja velocidad de cribado y el alto coste de los compuestos. En este contexto, la selección por medios computacionales de posibles candidatos a fármaco de las extensas librerías de compuestos [90, 8, 14, 69, 70], representa una rápida y efectiva aproximación [10]. Existen dos aproximaciones para realizar el cribado virtual; una está basada en ligando [92, 226] y otra en receptor. En la primera sólo se tiene en cuenta la estructura de los ligandos activos y en la segunda se considera también la estructura del sitio de interacción de la proteína. En particular, el cribado virtual basado en receptor se está convirtiendo en una parte clave del proceso de descubrimiento de fármacos. Esta clase de cribado virtual depende de métodos de ajuste molecular para llevar a cabo su tarea. Los métodos de ajuste molecular proteína-ligando intentan identificar las posiciones, orientación y conformación óptimas de un pequeño compuesto (ligando) con respecto a la proteína objetivo con una estructura 3D conocida (receptor). Normalmente las aproximaciones de cribado virtual se hacen en un proceso de dos pasos. Primero, toda la base de datos de ligandos es filtrada usando una eficiente pero poco discriminatoria aproximación de ajuste molecular. Después, los mejores compuestos son pasados a un método de cribado más preciso y exigente. Al final del proceso, un limitado número de compuestos que potencialmente se unen (desde cientos a unos pocos miles), pueden ser probados experimentalmente. En la actualidad hay muchos programas de cribado virtual disponibles [36, 236]; algunos de los más utilizados son Glide [58, 75], GOLD [97, 98], Lead Finder [209, 145], Autodock Vina [220], rDock [182], Surflex [94], CRDOCK [24], ICM [1, 218, 219], FlexX [172], PhDOCK [99], DOCK [110], MS-DOCK [184], VSDMIP [64], PLANTS [106], FRED [130], LigandFit [225] y VSDOCKER [169] entre otros. Todos ellos implementan este protocolo de dos pasos, pero aún no ha surgido ningún programa que sobrepase a los demás en todos los casos. A pesar del significante progreso en tecnologías de ajuste molecular, el mayor reto es combinar una predicción precisa de unión con velocidad y potencia de muestreo. Actualmente las aproximaciones de muestreo y clasificación toman desde segundos a minutos en procesar un único compuesto, incluso en el primer paso de cribado. Por consiguiente, la eficiencia y precisión de las aproximaciones actuales debe ser mejorada para poder manejar la avalancha de nuevos objetivos de fármacos y la actual necesidad de realizar un cribado totalmente automatizado de bases de datos que contienen millones de compuestos.

3.3.1. Ajuste molecular: *Docking*

La segunda parte de este trabajo de tesis se centra en la primera y computacionalmente más desafiante etapa del *docking*, que consiste en el muestreo traslacional y rotacional de cuerpo rígido de un pequeño ligando con respecto a una molécula receptora fija, mientras la función

de puntuación de la interacción entre ambos es maximizada. El espacio 6D muestreable de las orientaciones relativas entre el ligando y el receptor es muy grande. De hecho, se pueden evaluar más de 37 millones de posiciones relativas. Además, el número de moléculas en las librerías químicas que se usan actualmente para cribado virtual es del orden de millones. Por lo tanto, se necesita una herramienta de *docking* altamente eficiente. Para aliviar este cuello de botella computacional se ha adaptado en este trabajo de tesis un procedimiento al problema de ajuste proteína-ligando que Garzón *et al.* [61, 62] ha desarrollado previamente para resolver problemas relacionados con el ajuste proteína-proteína. Este método, basado en una parametrización del grupo rotacional tridimensional con armónicos esféricos, aumenta de forma significativa la eficiencia del ajuste molecular mediante la aceleración de la búsqueda de la parte rotacional [107, 108]. La adaptación propuesta en esta tesis para ajuste proteína-ligando involucra derivar nuevas expresiones matemáticas y por lo tanto es una nueva metodología. Además, para mejorar la eficiencia del proceso de cribado virtual más aún, se ha portado esta metodología a sistemas de computación de altas prestaciones. Se han propuesto dos estrategias de paralización: una para sistemas de memoria distribuida, basada en MPI, y otra para GPUs que utiliza CUDA. Este nuevo procedimiento se denomina FRODRUG.

Los diferentes paquetes de *software* de cribado virtual utilizan distintas técnicas de ajuste molecular 3D para encontrar la mejor posición del principio activo sobre la proteína. Normalmente un paquete de *software* no utiliza una sola técnica sino una mezcla de estas. A continuación se detallan algunas de las técnicas existentes.

3.3.2. Técnicas de ajuste 3D

Existe una diversa cantidad de técnicas que se utilizan para realizar el ajuste 3D de complejos moleculares [198, 132]. Entre los métodos existentes que realizan este proceso se encuentran los que se basan en afinidad de unión y los que utilizan complementación de las superficies de las moléculas [95, 102, 232, 80]; otros utilizan la geometría de las moléculas de un modo más avanzado teniendo en cuenta incluso los vectores normales de las superficies [197, 144]. De entre las técnicas basadas en afinidad se pueden encontrar métodos que utilizan diversos tipos de campos de fuerza [59, 65] y métodos basados en conocimientos empíricos [238, 106]. Además también es posible tener en cuenta la flexibilidad de los complejos moleculares a la hora de buscar el mejor ajuste [243]. Típicamente los procedimientos de ajuste 3D no utilizan sólo uno de estos métodos sino una combinación de varios para obtener la mejor solución posible. Sin embargo la gran carga computacional de estas técnicas requiere una cuidadosa selección para lograr un compromiso entre calidad y tiempo de cómputo. Algunos programas de ajuste realizan en primer lugar una aproximación poco precisa pero rápida para a continuación llevar a cabo un refinamiento de las mejores soluciones con métodos más precisos que producen una carga computacional más elevada. La eficiencia en los cálculos depende también en gran medida de la aproximación al problema de ajuste. En este aspecto se ha demostrado que el empleo del dominio de frecuencias para acelerar la búsqueda en varios grados de libertad de forma simultánea es una aproximación precisa y eficiente [176].

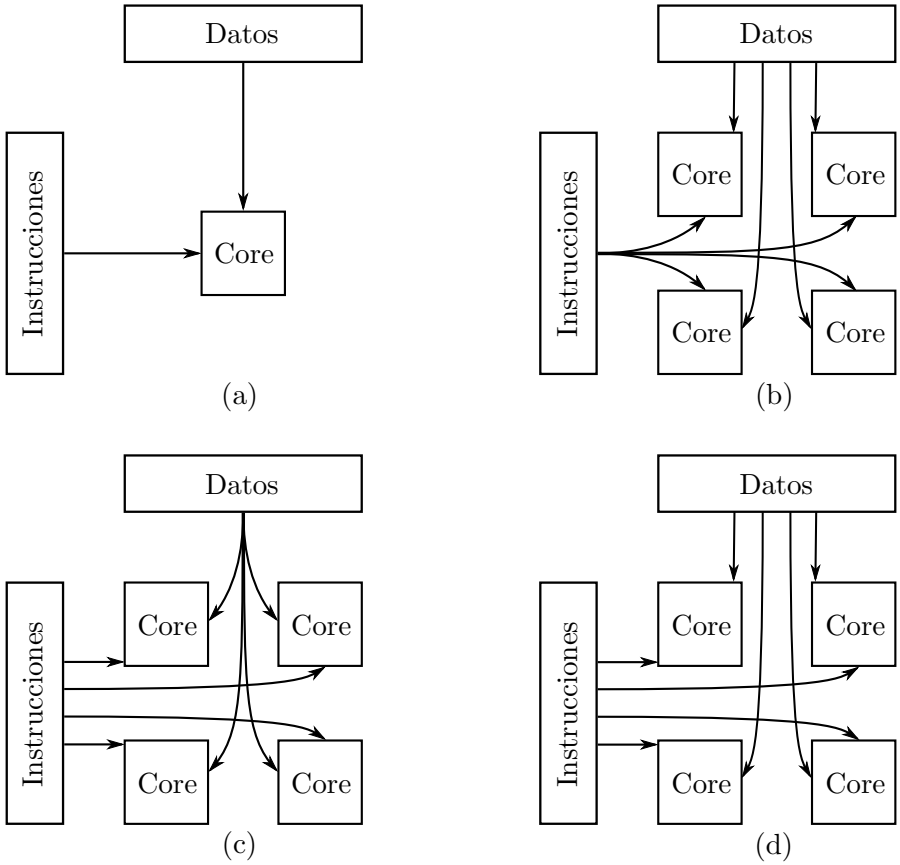


Figura 3.4: Diferentes arquitecturas de cómputo de la taxonomía de Flynn: (a) SISD, (b) SIMD, (c) MISD, (d) MIMD.

3.4. Arquitecturas de cómputo de altas prestaciones

La computación de altas prestaciones (*High Performance Computing* (HPC)) aparece para dar solución a problemas que presentan una alta carga computacional. Su objetivo es el de realizar una determinada tarea en la menor cantidad de tiempo posible. Estos sistemas de cómputo pueden estar compuestos por procesadores de diferentes arquitecturas. La taxonomía de Flynn [52] define los cuatro tipos diferentes de arquitecturas en las que pueden enmarcarse los distintos procesadores (ver Figura 3.4). La descripción de cada una se indica a continuación:

A. SISD (*Single Instruction Single Data*): La arquitectura tradicional de los procesadores de un solo núcleo. Una instrucción es ejecutada sobre un único dato.

B. SIMD (*Single Instruction Multiple Data*): La arquitectura de los actuales dispositivos gráficos. Se aplica una única instrucción sobre un amplio conjunto de datos.

C. MISD (*Multiple Instruction Single Data*): En el caso menos común se aplican diferentes instrucciones sobre un mismo dato.

D. MIMD (*Multiple Instruction Multiple Data*): La arquitectura más común, utilizada en la actualidad en los procesadores de más de un núcleo. Cada núcleo ejecuta diferentes instrucciones sobre diferentes conjuntos de datos.

La arquitectura SISD ha sido la más utilizada en los procesadores hasta la pasada década. Ahora que estos poseen más de un núcleo, la arquitectura SISD ha quedado relegada a pequeños sistemas embebidos y otros sistemas específicos. Los procesadores multinúcleo actuales se encuadran dentro de la arquitectura MIMD. Este tipo de procesadores son los componentes principales de los actuales supercomputadores. Tradicionalmente estos sistemas son los más empleados para realizar cálculo de altas prestaciones [227]. Por otra parte, los dispositivos gráficos modernos disponen de procesadores vectoriales, por lo que quedan enmarcados en la arquitectura SIMD. Debido a la gran potencia de cómputo que estos dispositivos son capaces de proporcionar, actualmente se emplean también como sistemas de cómputo de altas prestaciones. A continuación se realiza una introducción tanto de los supercomputadores como de los dispositivos gráficos en el marco de la computación de altas prestaciones.

3.4.1. Supercomputadores

A lo largo de las últimas décadas se han empleado sistemas supercomputadores distribuidos para llevar a cabo muchas tareas científicas que implican un alto coste computacional. Esta clase de sistemas pueden estar configurados de dos maneras: clúster ó grid. Un clúster consiste en un conjunto de ordenadores interconectados en la misma red de area local a través de interfaces de red de alta velocidad. Por otra parte, los ordenadores de un grid no están conectados a través de la misma red de area local, sino que pueden estar en diferentes dominios. Incluso los diferentes ordenadores que componen el grid pueden tener distintos sistemas operativos. En ambas plataformas la descomposición del paralelismo se realiza comúnmente a nivel de tarea, pero con cada una de ellas se trabaja de un modo diferente [126, 39]. La computación a través de grid [54, 239] requiere del uso de una capa intermedia, llamada *middleware*, para la gestión de la alta heterogeneidad del sistema. Entre los *middleware* más conocidos se encuentran gLite [27] o Globus [53]. Por otro lado, el cómputo paralelo en clúster se lleva a cabo empleando otras herramientas. Algunas de las más utilizadas son PVM [210], MapReduce [37, 5] ó MPI [134]. MPI define tan sólo una interfaz de comunicación entre procesos. Es necesario por lo tanto emplear una de las muchas librerías que implementan esta interfaz. Algunas de las más conocidas son MPICH [71], CHIMP [6], LAM-MPI [122] ó OpenMPI [60].

Los grandes supercomputadores de altas prestaciones actuales disponen del orden de millones de procesadores [217]. Esto hace que la probabilidad de que un nodo falle durante la ejecución de un proceso sea elevada, pudiendo perder el trabajo realizado. MPI, al definir tan sólo una interfaz, no es tolerante a fallos por si mismo. Esta tarea recae por lo tanto en el sistema de gestión de procesos utilizado. En este aspecto se han llevado a cabo diversos desarrollos [18, 88, 234, 235] a fin de generar implementaciones MPI más fiables.

Recientemente, Intel ha desarrollado una nueva arquitectura llamada MIC (*Many Integrated Core*) [41]. Esta se basa en varias tecnologías: la arquitectura Larrabee [194] y los procesadores Teraflops Research Chip [223] y Single-Chip Cloud Computer [79]. La arquitectura MIC se utiliza en la familia de coprocesadores Intel Xeon Phi [28]. Los dispositivos más modernos de esta familia incluyen 61 procesadores y 16 GB de memoria RAM dedicada. Estos coprocesadores están comenzando a utilizarse para cómputo de altas prestaciones en algunos sistemas

supercomputadores, como el Stampede [212]. Este cuenta con 6.880 coprocesadores Xeon Phi SE10P, además de 13.088 procesadores Intel Xeon y 128 GPUs NVIDIA Tesla K20.

3.4.2. Arquitecturas gráficas y paralelización

Tradicionalmente los dispositivos gráficos surgieron como procesadores de apoyo para cálculo de gráficos en 3D [240]. Estas arquitecturas *hardware* estaban diseñadas para realizar operaciones sobre vértices de un modo más eficiente que las CPUs. Esto se lograba a través de la aplicación en paralelo de una misma operación de transformación sobre un conjunto de vértices. Este diseño se aprovechó posteriormente para realizar cálculo de propósito general ó GPGPU (*General-Purpose computing on the Graphics Processing Unit*) [160]. Las tarjetas gráficas actuales contienen una serie de procesadores vectoriales que ejecutan de forma paralela la misma instrucción sobre un amplio conjunto de datos [117]. Estos procesadores vectoriales usualmente funcionan a una frecuencia mucho inferior a la de las CPUs actuales y no disponen de la tecnología destinada a aumentar la eficiencia de una ejecución secuencial que estos tienen [84]. Sin embargo, al disponer de una cantidad masiva de núcleos programables son capaces de procesar miles de datos simultáneamente. La programación de las tarjetas para realizar cómputo general requiere del empleo APIs específicas para acceder al *hardware* gráfico. Un estudio sobre estas interfaces de programación se detalla más adelante, en la sección 3.4.4.

La descomposición del paralelismo de un procedimiento sobre dispositivos gráficos se puede realizar tanto a nivel de datos como de tarea, incluso de ambos modos a la vez. Sin embargo, no todos los procedimientos son aptos para ser paralelizados sobre este tipo de dispositivos [161]. Uno de los factores decisivos en el éxito de la adaptación de un procedimiento a arquitecturas gráficas es la cantidad de datos a los que se necesite acceder con respecto a la cantidad de cómputo que se realiza sobre los mismos. Esto se debe a las actuales velocidades de transmisión de los datos desde RAM principal a VRAM y viceversa [23]. Si la cantidad de datos a transferir hacia ó desde el dispositivo gráfico es demasiado elevada, se corre el riesgo de invertir más tiempo en la copia de datos que en el cómputo de esos mismos datos en una CPU. Otro de los factores clave es que el procedimiento a adaptar debe mostrar un alto grado de paralelizabilidad. Un procedimiento que no pueda separarse en diferentes partes independientes ó bien que si pueda pero que unas partes dependan del resultado de otras, por lo general no mostrará una mejoría en rendimiento al adaptarlo a arquitecturas gráficas.

Desde la aparición de los primeros aceleradores 3D, que no disponían de salida de vídeo sino que se utilizaban exclusivamente como coprocesadores, hasta las actuales arquitecturas gráficas masivamente paralelas, las GPUs han ido progresando rápidamente a lo largo del tiempo. A continuación se ilustra brevemente su evolución.

3.4.3. Evolución de los dispositivos gráficos

Originariamente los cálculos en las tarjetas gráficas se realizaban empleando un *pipeline* fijo, sin etapas programables [74]. A través de los diversos tramos del *pipeline* se calculaban las posiciones de los vértices en el espacio, y a partir de estos se generaban triángulos. A continuación, estos triángulos eran asignados a píxeles de la imagen, formando un fragmento

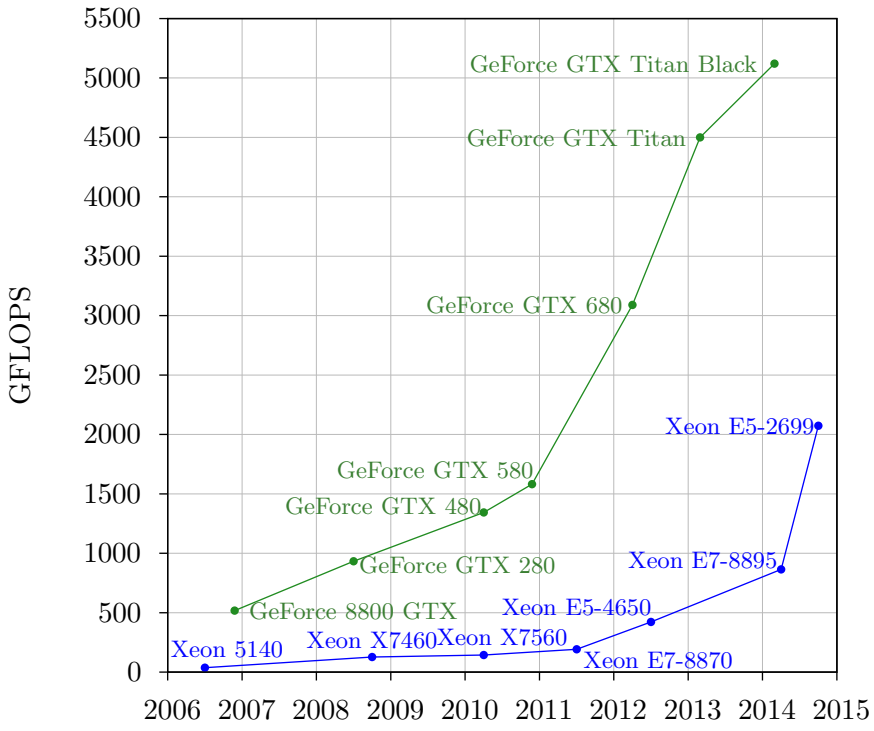


Figura 3.5: Comparativa de la evolución de la capacidad de cómputo entre las tarjetas gráficas de NVIDIA y los procesadores de Intel.

por cada pixel. Finalmente se aplicaba a los fragmentos información relativa a color y texturas para dar el color final a cada pixel de la imagen. El punto clave en la evolución de los dispositivos gráficos fue la sustitución de las etapas fijas de vértices y fragmentos por etapas programables [116, 105, 136]. De este modo se hizo posible obtener efectos visuales avanzados además de dar comienzo a la GPGPU. Estas etapas programables presentaban originariamente conjuntos de instrucciones diferentes; esto quedó solventado con la inclusión del *Unified Shader Model* [16]. La evolución de los dispositivos gráficos continuó con la eliminación del *pipeline* fijo y la inclusión de procesadores programables (*streaming multiprocessors*) que pueden realizar todas las tareas de las diferentes etapas del antiguo *pipeline* [117]. Esta es la arquitectura de los dispositivos gráficos actuales. Gracias a ella se dispone de gran flexibilidad a la hora de realizar cómputo de propósito general en esta clase de dispositivos.

A lo largo de esta última década la capacidad computacional de los dispositivos gráficos ha crecido en gran medida. En la Figura 3.5 se puede observar una comparativa de la evolución del poder de cómputo de los dispositivos gráficos con respecto a las CPUs. Esta evolución ha sido principalmente motivada por la industria del videojuego, en la búsqueda de gráficos cada vez más realistas.

Desde que estuvo disponible la posibilidad de realizar cómputo de propósito general sobre tarjetas gráficas, el modo en que estas se programan también ha ido evolucionando. Las APIs utilizadas para ello a lo largo de las últimas décadas se detallan a continuación.

3.4.4. Interfaces de programación en dispositivos gráficos

A lo largo de la historia de los dispositivos gráficos programables han aparecido diversas interfaces de programación que han sido utilizadas para llevar a cabo cálculo de propósito general sobre estos [160, 161, 195, 139]. Algunas de las que más impacto han tenido son HLSL (*High Level Shading Language*) [155] sobre DirectX 9, Cg [123, 49] sobre OpenGL ó GLSL (*OpenGL Shading Language*) [124, 103] también sobre OpenGL. El principal inconveniente de estas interfaces es que su propósito nunca ha sido el de proveer un modo de realizar cálculo general sobre las tarjetas sino el de ofrecer la capacidad de programar elementos gráficos con fines exclusivamente visuales. Esto dificulta en gran medida el uso de las tarjetas gráficas como procesador de cómputo general forzando a los desarrolladores a utilizar elementos de las librerías gráficas, como texturas, para almacenar los datos. En un intento por abstraer la capa gráfica y disponer de una interfaz más sencilla y de más alto nivel surgieron en el año 2004 algunos proyectos de investigación como Sh (*Shader Algebra*) [128, 129] o BrookGPU [22]. Estas interfaces ocultaban de forma efectiva la capa gráfica, pero seguían dependiendo de ella para su funcionamiento. Se programaban empleando un subconjunto del lenguaje C que se traduce posteriormente a otros lenguajes que emplean OpenGL o DirectX. En 2006, Microsoft desarrolló otra interfaz llamada Accelerator [215] cuyas diferencias más importantes con respecto a otras anteriores es que el lenguaje empleado para desarrollar en ella es C# y la compilación se realiza en tiempo de ejecución. RapidMind [127] también utilizaba compilación en tiempo de ejecución y era capaz de generar código tanto para GPUs como para sistemas CPU multinúcleo. En el año 2011 apareció OpenACC (Open Accelerators) [158]. Esta interfaz está basada en OpenMP [159], pero su objetivo de paralelización son los dispositivos gráficos. Tanto OpenACC como OpenMP abstraen la paralelización de las tareas a través de directivas para el compilador, el cual se encarga de generar el código paralelo de forma automática. Esto hace que paralelizar un procedimiento secuencial sea relativamente sencillo a costa de obtener una implementación general, menos eficiente. En la actualidad, las plataformas más novedosas e importantes, cuyo propósito exclusivo es el de cómputo de propósito general son CUDA (*Compute Unified Device Architecture*) [151] y OpenCL (*Open Compute Library*) [104]. OpenCL apareció en el año 2009 como una iniciativa de agrupar distintos dispositivos de cómputo bajo una sola API. Mediante OpenCL se puede generar código tanto para tarjetas de NVIDIA ó ATI como para sistemas de CPU multinúcleo sin necesidad de realizar modificaciones en el código. La portabilidad supone una gran ventaja, pero también existen grandes inconvenientes como la imposibilidad de obtener un alto rendimiento al emplear código genérico. Otro importante problema de unificar los dispositivos de cómputo es que cada uno de ellos evoluciona a un ritmo diferente y puede incluir mejoras que otros dispositivos no contemplen; disponer de una interfaz genérica hace imposible programar un código portable que utilice todas las novedades de nuevos dispositivos *hardware*. Con CUDA, por el contrario, sólo es posible generar código para tarjetas de NVIDIA. Al enfocar una arquitectura concreta es posible diseñar código más específico capaz de alcanzar un rendimiento más elevado. Por otro lado, la plataforma CUDA evoluciona al mismo tiempo que las tarjetas NVIDIA, ofreciendo siempre las últimas características de los dispositivos gráficos. A continuación se detalla brevemente esta plataforma de cómputo de altas prestaciones por ser la empleada durante el desarrollo de esta tesis doctoral.

C.C.	1.0	2.0	3.0	5.0
Dimensiones del *grid*	2	3	3	3
Bloques máximos por multiprocesador	8	8	16	32
Warps máximos por multiprocesador	24	48	64	64
Registros por multiprocesador	8K	32K	64K	64K
Registros máximos por *thread*	128	63	255	255
Memoria compartida máxima por multiprocesador	16 KB	48 KB	48 KB	64 KB
Bancos de memoria por multiprocesador	16	32	32	32

Tabla 3.1: Resumen de las principales características de las diferentes capacidades computacionales.

3.4.4.1. CUDA

CUDA (*Compute Unified Device Architecture*) es una arquitectura *hardware / software*, publicada en 2006, que permite usar las tarjetas gráficas de NVIDIA para realizar cómputo de propósito general en paralelo. Esta tecnología consiste en una parte *hardware* (sus tarjetas gráficas de la serie 8 en adelante) y una parte *software* (*drivers*, librerías y utilidades) que permiten trabajar con la parte *hardware*. El código para dispositivos gráficos se programa usando lenguaje C y la pequeña extensión que CUDA aporta a C para el control del *hardware* gráfico. Aunque en principio la programación con CUDA fue sólo ideada para la ejecución en tarjetas gráficas, recientemente se han desarrollado un conjunto de herramientas llamadas MCUDA que permiten compilar código CUDA para ser ejecutado sobre procesadores multinúcleo [208].

CUDA ha ido evolucionado desde su lanzamiento y se han ido añadiendo nuevas características y opciones según ha ido avanzando la tecnología y arquitectura de las tarjetas de NVIDIA. Esto ha dado pie a las diferentes capacidades computacionales [153]. Cada capacidad computacional describe las características de una versión de la arquitectura de las tarjetas de NVIDIA. Las capacidades computacionales básicas son la 1.0, correspondiente a la arquitectura Tesla [117], la 2.0 correspondiente a la arquitectura Fermi [148], la 3.0 correspondiente a la arquitectura Kepler [149] y la 5.0 correspondiente a la arquitectura Maxwell [152]. Existen subversiones de algunas de las capacidades computaciones que son pequeñas revisiones de estas. En la Tabla 3.1 se resumen las características de las principales capacidades computacionales.

Con respecto a la estructura lógica, en CUDA se definen los elementos *thread*, *block*, *grid* y *kernel* (ver Figura 3.6). El *thread* de CUDA es la unidad de ejecución más pequeña. Los *threads* se ejecutan en grupos de 32 unidades. Esta agrupación de *threads* es llamada *warp*. Un bloque es la manera que existe en CUDA de agrupar un conjunto de *threads*. La principal característica de un bloque es que los *threads* que este alberga son capaces de compartir datos y sincronizarse entre ellos. Los bloques tienen una cantidad máxima de *threads* que pueden albergar. Este valor depende de la capacidad computacional del dispositivo empleado. Un bloque puede tener una, dos o tres dimensiones para adecuar su forma a la de los datos a procesar. El tamaño del bloque se selecciona antes de comenzar la ejecución de una función en el dispositivo gráfico. Un *grid* es una agrupación de bloques. Este engloba los bloques necesarios para cubrir todos los datos que

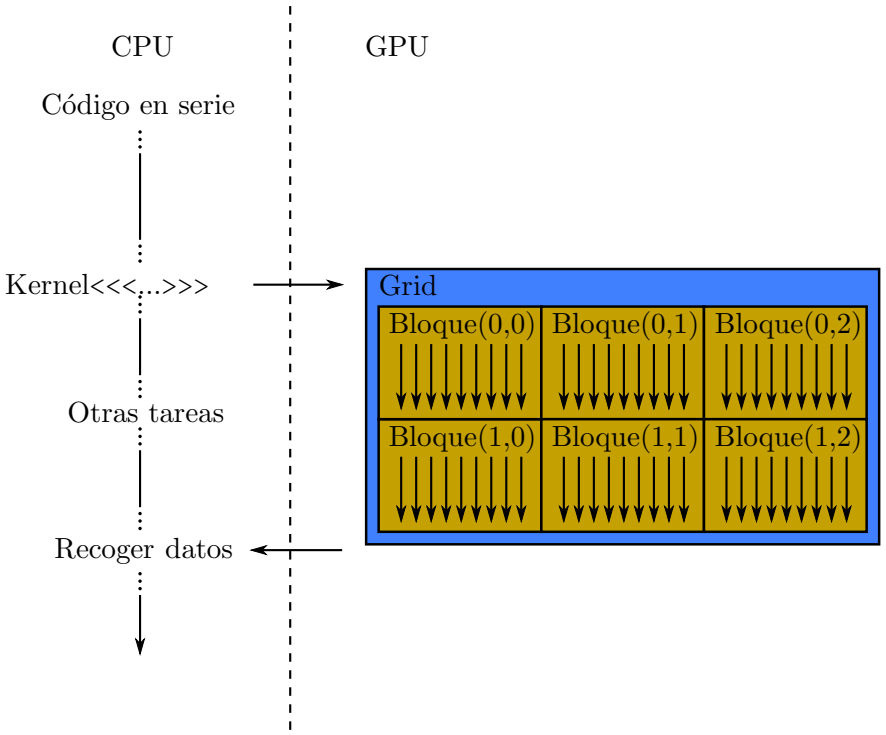


Figura 3.6: Lanzamiento de un *kernel* de CUDA. Las llamadas a *kernels* son asíncronas. La CPU puede realizar otras tareas mientras que la GPU realiza el cálculo solicitado. Al ejecutar un *kernel* se genera un *grid* cuyas dimensiones en *bloques* se indican en la llamada. Cada bloque contiene la cantidad de *threads* correspondiente a su tamaño, indicado también en la llamada al *kernel*.

necesitan ser procesados. Al igual que el bloque, el *grid* dispone de tres dimensiones. Estas son también seleccionadas antes de comenzar la ejecución en la tarjeta. El *kernel* es el equivalente a la función en C. Al ejecutar un *kernel* se le está indicando a la tarjeta que ejecute un algoritmo sobre un conjunto de datos, indicando al mismo tiempo los tamaños de bloque y *grid* deseados. Los lanzamientos de *kernels* en los dispositivos gráficos se realizan siempre de forma asíncrona. Existen no obstante un conjunto de funciones síncronas en las librerías de CUDA como las utilizadas para configurar los *kernels* o la utilizada para copiar memoria desde el dispositivo gráfico hasta la RAM principal.

Desde el punto de vista físico, en las tarjetas de NVIDIA los procesadores están distribuidos en paquetes llamados multiprocesadores. El contenido interno de estos multiprocesadores depende de cada capacidad computacional. Los bloques de un *grid* son distribuidos entre los multiprocesadores de la tarjeta de forma aleatoria por el *driver*. Cuando un bloque es asignado a un multiprocesador, todos los *threads* de ese bloque son procesados exclusivamente por ese multiprocesador. Gracias a la estructuración del cómputo de los datos mediante los bloques, esta tecnología es altamente escalable entre las distintas generaciones de tarjetas. Una vez un bloque es asignado a un multiprocesador, las instrucciones para los *threads* de ese bloque son lanzadas por un *scheduler*. Los dispositivos de arquitectura Tesla poseen un sólo *scheduler* por multiprocesador, los Fermi dos *schedulers* y los Kepler y Maxwell poseen cuatro.

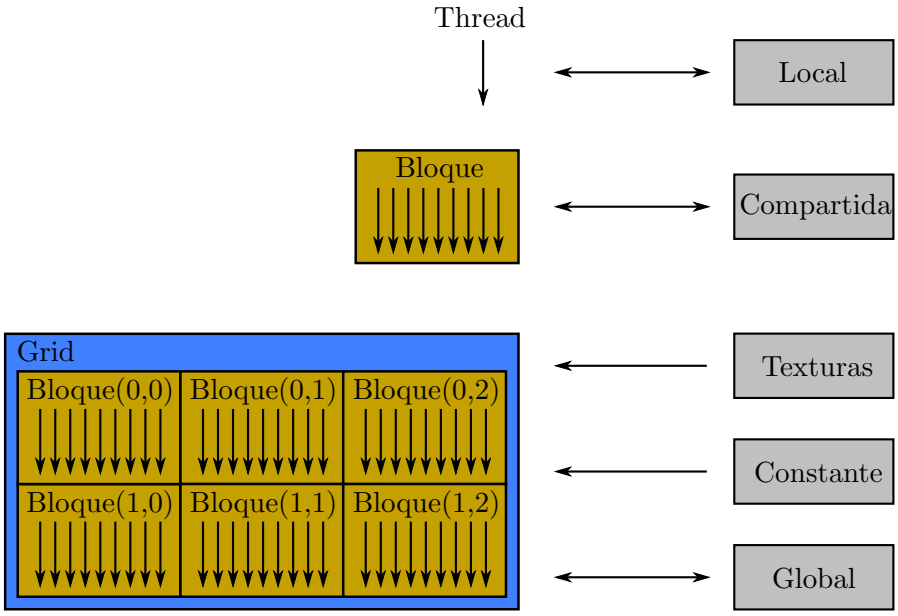


Figura 3.7: Tipos de memoria en CUDA. Las memorias global, constante y de texturas tienen visibilidad para todo el *grid*. La memoria compartida puede ser accedida por los *threads* del mismo bloque. La memoria local es privada para cada *thread*.

En la Figura 3.7 se muestran los diferentes tipos de memoria a la que los *threads* pueden acceder: global, de texturas, constante, compartida y local. La mayor de todas es la memoria global, que se encuentra en la RAM de vídeo. Los *threads* pueden realizar lectura y escritura sobre ella, y su latencia es elevada. Las memorias de texturas y constante residen también en la RAM de vídeo, pero disponen de una pequeña caché a diferencia de la memoria global. Estas dos memorias no pueden ser escritas por los *threads*, tan sólo leídas. Los datos almacenados en estas tres memorias tienen el tiempo de vida de la aplicación. La memoria compartida reside dentro de los multiprocesadores y su acceso es casi igual de rápido que el de un registro. Todos los *threads* de un mismo bloque pueden leer y escribir en el espacio de memoria compartida reservado para ese bloque. Esta memoria tiene el tiempo de vida del bloque. Por último se encuentra la memoria local. Esta memoria tiene acceso de lectura y escritura por *thread*, pero el programador no dispone de acceso a ella. El compilador se encarga de utilizarla cuando la considera necesaria. La memoria local tiene el tiempo de vida del *thread*.

Desde los dispositivos de arquitectura Fermi en adelante, los multiprocesadores disponen además de dos niveles de caché: una caché L1 por multiprocesador y una caché L2 compartida entre todos los multiprocesadores. La caché L1 reside en el mismo chip que la memoria compartida. Este chip tiene una capacidad de 64 KB en las arquitecturas Fermi y Kepler. Es posible configurar los tamaños de la caché L1 y la memoria compartida. Existen dos posibles configuraciones para los 64 KB disponibles: 16 KB de memoria compartida y 48 KB de caché L1 ó 48 KB de memoria compartida y 16 KB de caché L1. Además, a partir de la arquitectura Kepler existe un tercer modo adicional de configuración: 32 KB de memoria compartida y 32 KB de caché L1. Las tarjetas gráficas de arquitectura Fermi son las únicas que utilizan la caché L1 para cachear accesos a memoria global. Desde la arquitectura Kepler en adelante, esta caché se utiliza por el compilador para almacenar *buffers* y variables locales entre otras cosas.

En cualquier caso, es siempre conveniente configurar el tamaño de esta caché lo más grande que se pueda respetando la cantidad de memoria compartida que cada *kernel* requiera para su funcionamiento. De este modo se aprovecha al máximo la memoria de este chip.

3.4.5. Justificación de los desarrollos paralelos

Existe un factor fundamental que justifica el desarrollo de algoritmos paralelos: aumentar el rendimiento de código que se ejecuta de forma secuencial es cada vez más complicado debido a que las CPUs tienen grandes problemas de consumo de energía y generación de calor cuando sobrepasan frecuencias más allá de los 4 GHz [179]. Debido a esto, los fabricantes de chips han optado por reducir las frecuencias de sus procesadores y empaquetar más núcleos que trabajen en paralelo. Para poder aprovechar toda la potencia de este nuevo diseño *hardware* es indispensable que los actuales procedimientos secuenciales sean rediseñados empleando aproximaciones paralelas. Por otra parte, el hecho de investigar la paralelización preferentemente en arquitecturas gráficas sobre los tradicionales procesadores está justificado no sólo por la mayor capacidad computacional de estas sino también por su reducido precio. A fecha de Marzo de 2015, uno de los más potentes procesadores de Intel (Xeon E5-2699) tiene un precio cercano a los 5.000 € mientras que una GPU de NVIDIA disponible en el mercado unos meses antes (GeForce GTX Titan Black) puede encontrarse por un precio cercano a los 1.000 €. Además de esta diferencia en el precio inicial del *hardware*, también existe la diferencia en el coste derivado de la energía necesaria para alimentarlo. El consumo energético de los dispositivos gráficos es sustancialmente menor que el de los supercomputadores [12, 179]. Por último cabe destacar que este *hardware* gráfico de consumo puede encontrarse en la actualidad en casi cualquier hogar, lo que otorga la posibilidad de realizar cómputo de altas prestaciones al público general. Esto puede ser aprovechado fácilmente a través de iniciativas como el Folding@Home [44], en la que se puede utilizar cualquier ordenador conectado a internet para predecir el plegamiento de una proteína, formando una arquitectura de nube de cómputo. La técnica de diseño basada en la paralelización de procesos ya se ha utilizado anteriormente con éxito en muchos campos científicos. En biología, un ejemplo es el *software* NAMD de dinámica molecular [167, 207]. Este dispone de una versión para dispositivos gráficos que funciona 20 veces más rápido que la versión CPU. GROMACS [115], también de dinámica molecular, dispone de igual modo de una versión paralela para dispositivos gráficos que utiliza la plataforma CUDA. VMD (*Visual Molecular Dynamics*) [86] incluye también parte de su implementación en CUDA para realizar funciones con una alta carga computacional como por ejemplo el cálculo de mapas de potencial electrostático. Esta función en concreto realiza su tarea sobre una tarjeta NVIDIA 8800 GTX 44 veces más rápido que en un Intel QX6700 a 2,6 GHz. En el campo de la imagen médica se ha hecho también un extenso uso de la computación paralela [168, 125]. La precisión y eficiencia en el análisis de los datos recopilados por dispositivos médicos es un factor fundamental para una correcta y rápida diagnosis. En este aspecto se han llevado a cabo recientemente desarrollos paralelos con dispositivos gráficos en campos como la segmentación de imagen médica [193, 162, 181, 183] ó la reconstrucción de tomografía por emisión de positrones [81], obteniendo una notable mejora en la eficiencia con respecto a sus correspondientes versiones secuenciales. En el campo de la microscopía, tanto electrónica como óptica, se han realizado también desarrollos paralelos en

GPU. Las imágenes obtenidas con los microscopios ópticos en ocasiones sufren problemas de distorsión por difracción de la luz y aberraciones de las lentes. Existen estudios cuyo objetivo es el de reducir estas distorsiones a través del modelado del comportamiento del microscopio [40]. Existen también desarrollos en GPU en el campo de la microscopía electrónica [192], y más en concreto relacionados con el alineamiento de imágenes [25], como el *software* Alignator [26]. El método que usa este programa para realizar los alineamientos es una simple correlación cruzada en el dominio de frecuencias. Prácticamente todos los paquetes de *software* de procesamiento de imágenes contienen procedimientos que han sido optimizados para GPU, como por ejemplo EMAN2 [214]. Así mismo también existen desarrollos GPU en el campo del ajuste molecular 3D y cribado virtual de proteínas [165, 175]. Muchos métodos en el campo de la bioinformática aceleran sus procesos trabajando en el dominio de frecuencias. Para ello necesitan realizar la transformada de Fourier sobre los datos a tratar. Uno de los algoritmos más utilizados es el Cooley-Tukey [32] y es altamente paralelizable. Gracias a esto, y para acelerar aún más el proceso de esta transformada, se han realizado diversos desarrollos de este algoritmo sobre dispositivos gráficos [147, 146, 72, 150]. Otros muchos procedimientos científicos de naturaleza tridimensional, o aquellos que requieren de una representación esférica en dos dimensiones, emplean transformadas de armónicos esféricos [135, 120] para la representación de sus datos y aceleración de sus cálculos. Estas transformadas son comúnmente empleadas en la representación de las partículas por los programas de ajuste molecular. En el tratamiento de estas transformadas se han realizado gran cantidad de desarrollos tratando de optimizar al máximo su eficiencia. Algunos de los paquetes más importantes que realizan esta función son LIBPSHT [173], S^2HAT [206], SpharmonicKit [78] ó SPHEREPACK [3]. Cada uno de ellos propone una aproximación diferente para la resolución del problema. Los que se consideran más eficientes son LIBPSHT en entornos de memoria compartida y S^2HAT en entornos de memoria distribuida [211]. De algunos de estos paquetes de *software* existen también desarrollos en GPU [203, 20, 87, 211].

Capítulo 4

Nuevos métodos de ajuste 2D y 3D

Para cumplir los objetivos de tesis se ha llevado a cabo la adaptación de varios métodos de ajuste 2D y 3D de aplicación en el campo de la bioinformática. El primero está orientado a la microscopía electrónica y el segundo al ajuste molecular de proteína-ligando. Para ambos métodos se han realizado optimizaciones sobre arquitecturas de cómputo de altas prestaciones. La implementación en dispositivos gráficos se ha llevado a cabo empleando la plataforma CUDA y para sistemas de cómputo de memoria distribuida se ha empleado MPI (*Message Passing Interface*). En primer lugar se detalla la adaptación del método de alineamiento 2D para microscopía electrónica.

4.1. Ajuste de imágenes 2D para microscopía electrónica

El propósito del nuevo método de ajuste 2D adaptado es el de alinear imágenes de microscopía electrónica. El alineamiento de varias imágenes de la misma proyección de una partícula permite reducir el ruido mediante un promediado de las mismas. Este método, denominado *Fast Bessel Matching* (FBM) [109], emplea transformadas de Bessel para llevar a cabo el alineamiento. En esta sección se procederá a detallar tanto dicho método como la adaptación secuencial realizada. Después, en la sección 4.1.3, se comenta una optimización realizada al método más eficaz empleado actualmente en el paquete de herramientas de microscopía electrónica EMAN: *Fast Rotational Matching 2D*. FBM se comparará con la versión optimizada de *Fast Rotational Matching 2D*.

4.1.1. *Fast Bessel Matching*

El alineamiento de una imagen 2D se define a través de tres grados de libertad. Estos parámetros típicamente consisten en dos grados traslacionales para desplazar la imagen en los ejes x e y, y un grado rotacional para orientarla. Sin embargo, el alineamiento en FBM se define a través de tres grados de libertad rotacionales. Estos dos tipos de configuración se muestran en la Figura 4.1. FBM permite la estimación directa de todos los valores de correlación entre dos imágenes como una función de los ángulos ξ, η y ω. La traslación de la imagen experimental se obtiene utilizando la constante b y los ángulos ξ y η mientras que el ángulo ω determina su orientación.

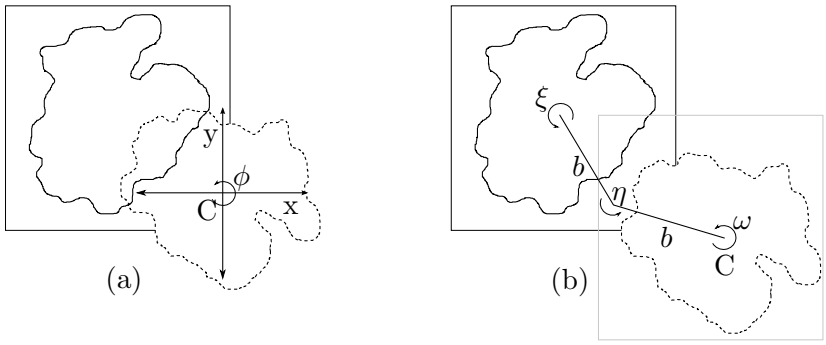


Figura 4.1: Configuración de alineamiento 2D. La imagen (a) muestra la configuración tradicional de ajuste de imágenes 2D, con dos grados de libertad traslacionales (x e y) y uno rotacional (ϕ). La imagen (b) muestra la configuración de FBM, con tres grados de libertad rotacionales. La imagen de la izquierda está fija, y la imagen de la derecha se mueve mediante las rotaciones de ξ, η y ω para coincidir con la imagen de la izquierda. b es una constante que se utiliza para desplazar el centro de rotación de la imagen experimental. El punto C indica el centro de la imagen experimental.

El alineamiento se realiza maximizando la correlación entre los píxeles de una imagen de referencia que permanece fija (f) y una móvil que se denomina experimental (g).

En primer lugar es necesario definir algunos conceptos básicos del método: la definición de las transformadas de Bessel, el modo en que se representan las imágenes de microscopía y el proceso de la aplicación de un movimiento a una imagen.

La transformada de Fourier-Bessel [73] de una imagen $f(r,\lambda)$ se define como:

$$F_m(x) = \int_0^\infty \hat{f}_m(r) J_m(rx) r dr \tag{4.1}$$

donde $\hat{f}_m(r)$ se corresponde con la transformada de Fourier de la imagen $f(r,\lambda)$ en coordenadas polares donde r es el radio y λ es el ángulo de la coordenada polar, y $J_m(rx)$ corresponde a la función de Bessel del primer tipo con orden m. La transformada de Fourier de $f(r,\lambda)$ se expresa como:

$$\hat{f}_m(r) = \frac{1}{2\pi} \int_0^{2\pi} f(r,\lambda) e^{-im\lambda} d\lambda \tag{4.2}$$

mientras que las funciones de Bessel del primer tipo de orden m se definen como:

$$J_m(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!\Gamma(k+m+1)} \left(\frac{x}{2}\right)^{2k+m} \tag{4.3}$$

siendo $\Gamma(n)$ una extensión de la función factorial para argumentos reales y complejos.

Un movimiento aplicado a una imagen experimental g se define a través del operador $\Lambda_M(g)$. Este movimiento M se determina mediante dos parámetros rotacionales (ϕ y ψ) y uno traslacional (ρ). El operador de movimiento se factoriza posteriormente en una concatenación de dos movimientos independientes para dar lugar a los tres ángulos mostrados en la Figura 4.1. La aplicación de M a g se lleva a cabo en tres etapas. En primer lugar se realiza una rotación ϕ de la imagen. A continuación se realiza una traslación de la imagen de ρ píxeles a través de

su eje x, que se encuentra rotado por ϕ. Por último se realiza una nueva rotación de la imagen utilizando el ángulo ψ.

La ecuación para calcular la correlación entre la imagen de referencia y la experimental se puede expresar como:

$$c(M) = \int_0^{\infty} \int_0^{2\pi} f(r,\lambda)\, \overline{\Lambda_M(g)(r,\lambda)}\, r \, \mathrm{d}\lambda \mathrm{d}r \tag{4.4}$$

Sustituyendo $f(r,\lambda)$ y $\Lambda_M(g)(r,\lambda)$ en la Ecuación 4.4 por sus expansiones de Fourier:

$$\begin{aligned} f(r,\lambda) &= \sum_m e^{im\lambda} \hat{f}_m(r) \\ \Lambda_M(g)(r,\lambda) &= \sum_m e^{im\lambda} \widehat{\Lambda_M(g)}_m(r) \end{aligned} \tag{4.5}$$

se obtiene la siguiente ecuación:

$$c(M) = 2\pi \sum_m \int_0^{\infty} \hat{f}_m(r) \overline{\widehat{\Lambda_M(g)}_m(r)} r \, \mathrm{d}r \tag{4.6}$$

Mediante el teorema de Plancherel [228] para transformadas de Fourier-Bessel, la Ecuación 4.6 se convierte en

$$c(M) = 2\pi \sum_m \int_0^{\infty} F_m(x) \overline{(G^M)_m(x)} x \mathrm{d}x \tag{4.7}$$

siendo F_m la transformada de Fourier-Bessel de la imagen de referencia y G^M la correspondiente transformada de Fourier-Bessel de la imagen experimental con el movimiento M aplicado. Esta última se define como

$$(G^M)_m(x) = \sum_{m'} G_{m'}(x) J_{m-m'}(\rho x) e^{-i(m\phi + m'\psi)} \tag{4.8}$$

Integrando la Ecuación 4.8 en la 4.7, la correlación entre dos imágenes queda definida por:

$$c(M) = 2\pi \sum_{m,m'} e^{i(m\phi + m'\psi)} \int_0^{\infty} F_m(x) \overline{G_{m'}(x)} J_{m-m'}(\rho x) x \mathrm{d}x \tag{4.9}$$

El movimiento $M(\phi, \rho, \psi)$ se factoriza como una sucesión dos movimientos concatenados: $M_1(\phi_1, \rho_1, \psi_1)$ y $M_2(\phi_2, \rho_2, \psi_2)$. Esto permite realizar el alineamiento a través de los tres ángulos mostrados en la Figura 4.1. Los parámetros (ϕ, ρ, ψ) que componen cada uno de los movimientos son:

$$\begin{aligned} M_1(\phi_1, \rho_1, \psi_1) &= M_1(\xi, b, 0) \\ M_2(\phi_2, \rho_2, \psi_2) &= M_2(\eta + \epsilon, b, \omega) \end{aligned} \tag{4.10}$$

donde $b = \rho/2$ y ϵ es un desplazamiento angular que se añade a η para evitar valores duplicados en las soluciones. Los ángulos ϕ y ψ del movimiento original están ahora representados por ξ y ω. Estos se corresponden en la Ecuación 4.9 con las variables m y m' respectivamente. Al factorizar el movimiento M se agrega un nuevo ángulo: $\eta + \epsilon$, que se representa mediante la nueva variable h. Integrando la factorización del movimiento en la Ecuación 4.9 se produce la

siguiente ecuación:

$$C(M) = 2\pi \sum_{m,h,m'} e^{i(m\xi+h(\eta+\epsilon)+m'\omega)} \times \int_0^\infty F_m(x)\overline{G_{m'}(x)}J_{m-h}(bx)J_{h-m'}(bx)x\,dx \tag{4.11}$$

Dado que $J_n(x)$ se comporta como $(1/\sqrt{2\pi n})\cdot(ex/2n)^n$ para valores grandes de n y pequeños de x [2], se puede realizar el siguiente cambio de variables:

$$\begin{aligned} h &= h_1 + m' \\ m &= m_1 + h_1 + m' \\ \eta' &= \xi + \eta \\ \omega' &= \xi + \eta + \omega \end{aligned} \tag{4.12}$$

Ahora $C(M)$ se convierte en una función $T(\xi,\eta',\omega')$ cuya transformada inversa de Fourier es:

$$\hat{T}(m_1,h_1,m') = 2\pi e^{i(h_1+m')\epsilon} \times \int_0^\infty F_{m_1+h_1+m'}(x) \times \overline{G_{m'}(x)}J_{m_1}(bx)J_{h_1}(bx)x\,dx \tag{4.13}$$

De esta forma, calculando una única transformada inversa de esta función es posible obtener directamente los valores de correlación de ajuste para todos los tripletes angulares (ξ,η,ω) de un alineamiento.

4.1.2. FBM para alineamiento de múltiples imágenes

En el apartado anterior se ha detallado cómo alinear una imagen de referencia con una experimental, sin embargo el procesamiento de imágenes en microscopía electrónica requiere el alineamiento de miles e incluso millones de imágenes experimentales contra una de referencia.

El diagrama de flujo del método de ajuste FBM para múltiples imágenes está reflejado en la Figura 4.2. En primer lugar se realiza el precálculo para los conjuntos de imágenes de referencia y experimentales. Este paso previo implica dos operaciones: muestreo de las imágenes y aplicación de las transformadas de Fourier-Bessel. A continuación el método carga una a una las imágenes experimentales y por cada una de ellas itera sobre todas las de referencia. Por cada pareja de imágenes referencia/experimental se calcula su correlación y se guarda el triplete angular (ξ,η,ω) con la correlación más alta. Por cada imagen experimental se guardan n soluciones. El valor de n es proporcionado como parámetro de entrada al método. Cada una de estas soluciones se corresponde con el mejor ajuste la imagen experimental con una referencia diferente.

A continuación se detalla el modo en que se ha realizado la adaptación secuencial del método a través de la discretización de su modelo matemático. En primer lugar se comienza por el cálculo de las transformadas de Fourier-Bessel y a continuación se explica cómo se calculan los parámetros de ajuste 2D entre dos imágenes.

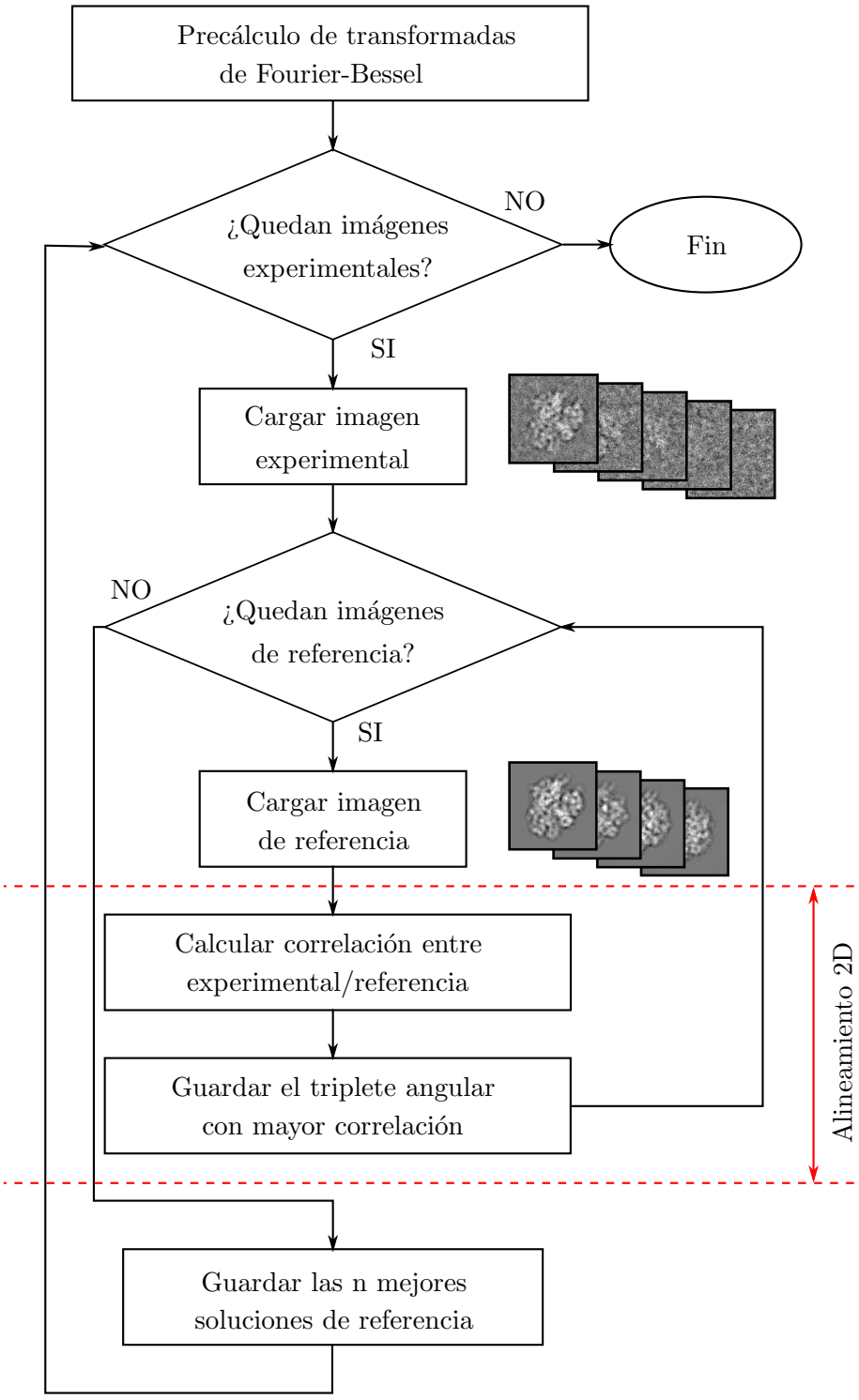


Figura 4.2: Flujo de ejecución de *Fast Bessel Matching*. El proceso comienza realizando el precálculo de las transformadas de Fourier-Bessel. A continuación se cargan las imágenes experimentales. Por cada una de ellas se itera sobre todas las de referencia. Para cada imagen experimental se guardan los n mejores ajustes.

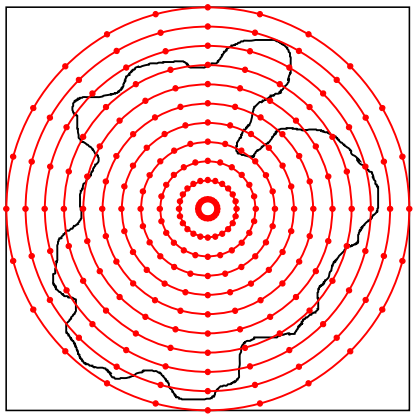


Figura 4.3: La cantidad de muestras radiales que se toman por imagen es de $2B/\pi$. Por cada radio se realiza un muestreo angular de $2B$ puntos. En rojo se indica un ejemplo de diferentes radios muestreados. La silueta negra corresponde a una partícula observada al microscopio.

4.1.2.1. Transformada de Fourier-Bessel de las imágenes

El paso inicial es el muestreo en coordenadas polares de las imágenes de microscopía electrónica. Se fija p_s como el número de píxeles promedio por cada intervalo angular:

$$p_s = \frac{\pi A}{s} \tag{4.14}$$

siendo s el número de intervalos angulares y A el radio de la imagen en píxeles. πA es por lo tanto la longitud del perímetro a mitad del radio. El número de intervalos angulares se fija al doble del ancho de banda ($2B$). El valor del ancho de banda (B) es proporcionado como parámetro de entrada al método. El número de intervalos angulares se selecciona de esta manera ya que la transformada de Fourier en la Ecuación 4.1, que se calculará en el siguiente paso, se muestrea sólo para $|m| < B$. Habiendo seleccionado el número de intervalos angulares, el número de intervalos radiales se calcula como A/p_s que equivale a $2B/\pi$. En la Figura 4.3 se ilustra un ejemplo de los puntos muestreados en una imagen.

El muestreo en coordenadas polares de cada imagen se almacena en memoria en una matriz 2D de $2B$ x $(2B/\pi + 1)$ valores. Para cada uno de los valores muestreados, se toman los cuatro píxeles más próximos al punto y se realiza una interpolación bilineal. En la Figura 4.4 se muestra el resultado del muestreo de una imagen de microscopía electrónica.

Para calcular la transformada de Bessel de las imágenes (Ecuación 4.1) se realiza en primer lugar la transformada de Fourier del muestreo realizado anteriormente sobre la imagen original, obteniendo el término $\hat{f}_m(r)$. A continuación se aplican las funciones de Bessel y se calcula la integral.

Por cada una de las circunferencias muestreadas en la imagen original se realiza una transformada de Fourier 1D. Para el cómputo de las transformadas de Fourier se hace uso de la librería FFTW (Fast Fourier Transform in the West) [47]. Esta librería permite realizar las transformadas de una forma eficiente mediante una única llamada a función. La Figura 4.5 muestra el resultado de la aplicación de la transformada de Fourier a una imagen muestreada

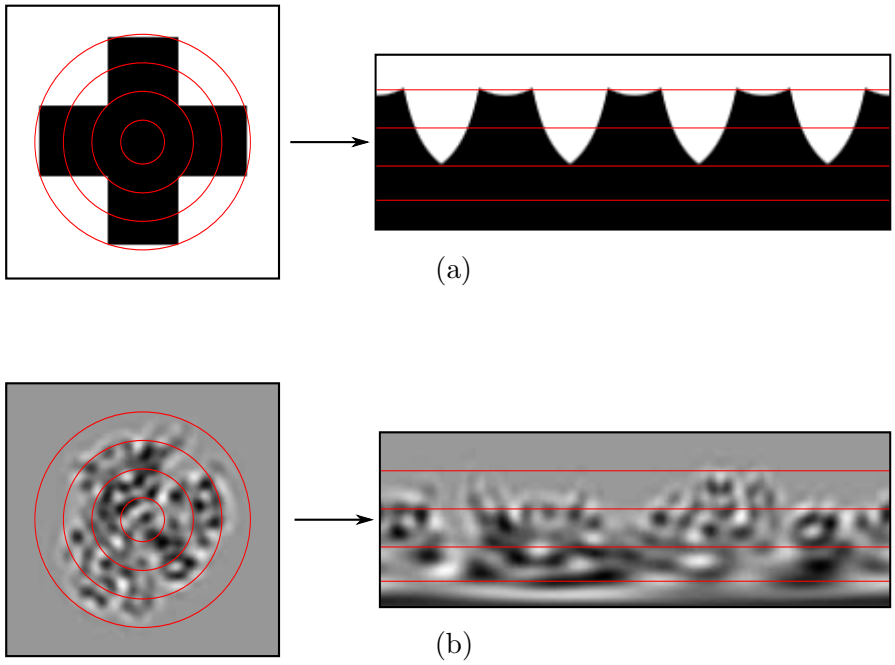


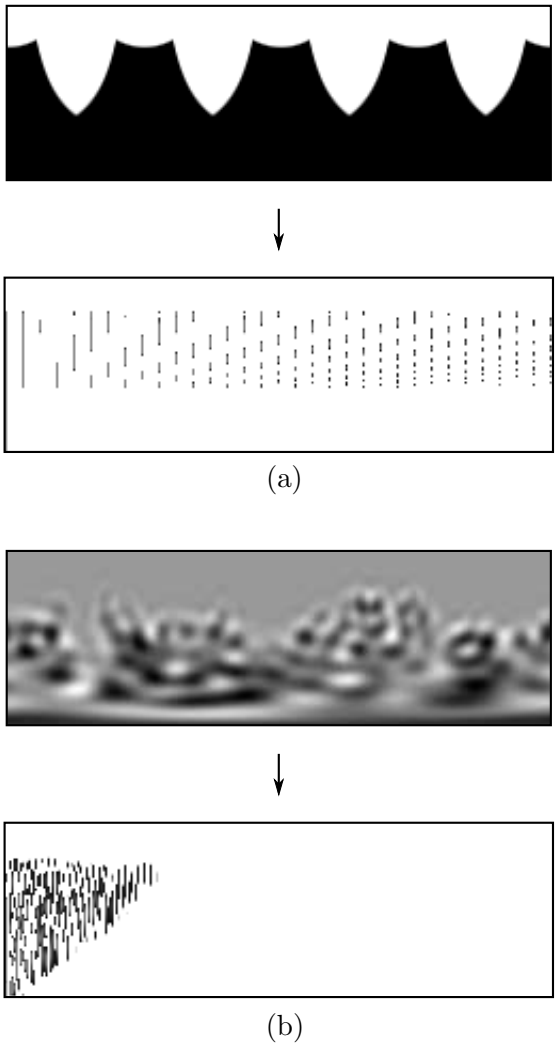
Figura 4.4: En la Figura (a) se puede ver el muestreo radial de una imagen (derecha) y su correspondiente original (izquierda) en forma de cruz. En la (b) se puede ver el muestreo radial y su correspondiente original de una imagen típica de microscopía electrónica. Las líneas rojas de las imágenes están colocadas exclusivamente a modo de guía visual.

en coordenadas polares. El tamaño de la matriz 2D de almacenamiento de las transformadas de Fourier es de $B+1$ x $(2B/\pi+1)$ datos de tipo complejo en coma flotante. Este cambio en una de las dimensiones, de $2B$ en el vector que contiene el muestreo de la imagen a $B+1$ en el que contiene su correspondiente transformada de Fourier, responde a dos factores. En primer lugar, la reducción de $2B$ a B se debe a una característica de las transformadas de Fourier. Para toda función $f(x)$ real, su transformada de Fourier es una función Hermítica, que cumple la propiedad de simetría hermítica: $f(-x) = \overline{f(x)}$, requiriendo sólo el almacenaje de la mitad de los datos. En segundo lugar, el valor complejo adicional en la dimensión modificada $(B+1)$ se debe a un requisito del método que realiza la transformada de Fourier.

A continuación se procede a aplicar las funciones de Bessel de primer tipo $(J_n(x))$. En la Figura 4.6 se puede ver un ejemplo de las funciones de Bessel de diferentes órdenes.

Estas funciones se invocan de forma repetitiva a lo largo de la aplicación de la transformada de Bessel sobre la imagen muestreada. Con el propósito de optimizar esta parte, se han modificado las funciones de Bessel disponibles en la librería C de GNU [15]. Para ello se ha definido una caché donde se almacenan los resultados obtenidos al invocar las funciones Bessel con diferentes parámetros. De este modo sólo es necesario calcular el valor para cada pareja de índices n y x una sola vez.

Antes de calcular la transformada es necesario conocer el límite superior de la variable x en la Ecuación 4.1, que luego se utilizará también para integrar la Ecuación 4.13. El valor máximo de x tiene que ser compatible con el número de puntos tomados a lo largo del radio ($2B/\pi$ muestras). Según el teorema de muestreo de Nyquist-Shannon, el número de puntos tiene que



(a)

(b)

Figura 4.5: En la imagen (a) se puede observar el muestreo radial de la imagen con forma de cruz de la Figura 4.4. Este muestreo tiene un tamaño de $2B$ x $(2B/\pi + 1)$. A continuación se encuentra la transformada de Fourier 1D para cada una de sus circunferencias. En la imagen (b) se muestra el mismo proceso para la imagen de microscopía electrónica.

ser el doble de la frecuencia más alta, indicado en este caso por la oscilación de la función de Bessel $J_m(rx)$. El rango de la variable r va desde 0 hasta A, y esto debe ser multiplicado por 2 ya que cada radio r tiene información sobre todo el círculo. Teniendo en cuenta que la función $J_m(rx)$ se comporta como $\sqrt{2/\pi rx}\cos(rx)$ [2], se puede estimar el valor máximo de x como $xA/2\pi$. Considerando todo esto, el valor máximo de x se puede definir como:

$$\frac{2 \cdot B}{\pi} = \frac{2 \cdot 2 \cdot x \cdot A}{2 \cdot \pi} \tag{4.15}$$

resultando finalmente

$$x = \frac{B}{A} \tag{4.16}$$

Una vez que se ha definido el rango de x, se procede al cálculo de la integral de la Ecuación 4.1. Esto se realiza acumulando los valores generados por la integral para todo el rango de r en cada combinación de m y x. El cálculo de la transformada de Fourier-Bessel de una imagen se

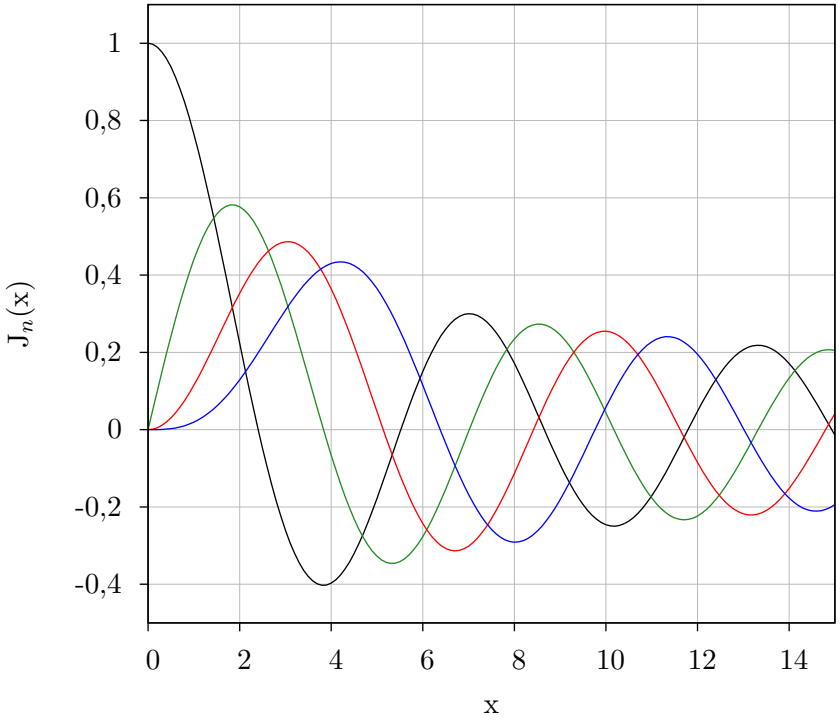


Figura 4.6: Funciones de Bessel $J_n(x)$ de órdenes 0 (negro), 1 (verde), 2 (rojo) y 3 (azul).

realiza sólo una vez y los datos se reutilizan posteriormente las veces que sea necesario. En la Figura 4.7 se puede ver un ejemplo de la aplicación de la transformada de Bessel a la anterior transformada de Fourier.

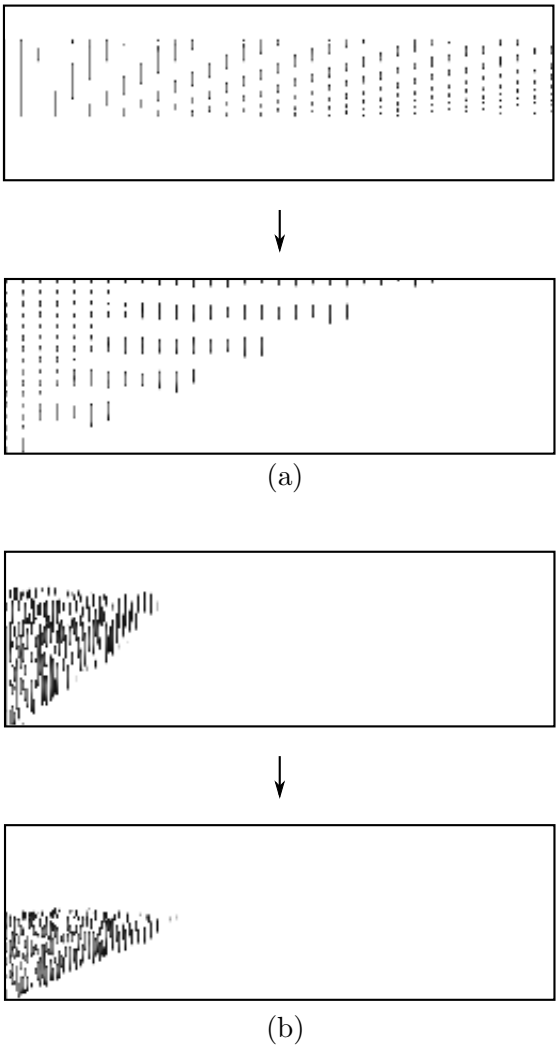
4.1.2.2. Obtención del ajuste 2D

El siguiente paso del método es la obtención de los tres ángulos de ajuste (ξ, η, ω) para la pareja de imágenes. Estos ángulos se calculan a partir de los máximos encontrados en el volumen de correlación obtenido mediante la transformada inversa de Fourier de la Ecuación 4.13. Las tres coordenadas de un máximo de este volumen (m_1, h_1, m') se emplean para deshacer el cambio de variables indicado en la Ecuación 4.12, obteniendo directamente los ángulos de ajuste.

Para calcular la integral es necesario conocer los rangos de las variables m_1, h_1, m' y el valor de ϵ. Se define k como la cantidad de puntos que se muestrean en la línea que hay entre los centros de ambas imágenes.

$$k = \frac{\rho_{max}}{p_s} \tag{4.17}$$

siendo ρ_{max} la distancia de un centro a otro en píxeles y p_s la cantidad de píxeles que hay entre puntos muestreados. Como se puede observar en la Figura 4.1, el muestreo del ángulo ξ define el muestreo angular del punto central de la imagen experimental mientras que el muestreo del ángulo η define el muestreo radial. El número de intervalos de estos ángulos debe ajustarse por



(a)

(b)

Figura 4.7: En la parte superior de la Figura (a) se puede observar la transformada de Bessel aplicada a la transformada de Fourier de la imagen (a) de la Figura 4.5. En la parte inferior (b) se encuentra la transformada de Bessel aplicada a la transformada de Fourier correspondiente a la imagen de microscopía electrónica.

lo tanto como:

$$\begin{aligned} s_\xi &= \pi k \\ s_\eta &= k \end{aligned} \tag{4.18}$$

De este modo, el punto C puede colocarse en una cantidad de puntos k muestreados entre los centros de las imágenes. Es posible modificar libremente los rangos de estas variables siempre que se mantenga su producto constante. Se define una variable a que se multiplica por el muestreo angular de η y se divide entre el muestreo angular de ξ. Los valores de muestreo quedan definidos del siguiente modo:

$$\begin{aligned} s_\xi &= \frac{\pi k}{a} \\ s_\eta &= ak \end{aligned} \tag{4.19}$$

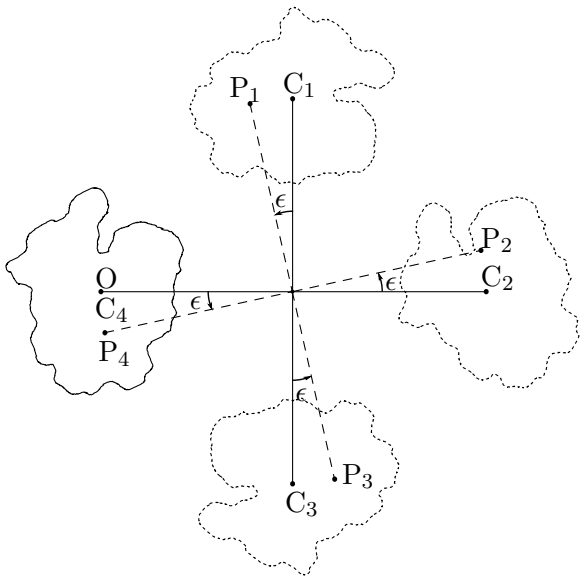


Figura 4.8: Efecto del desplazamiento de ϵ. Este diagrama corresponde a los puntos muestreados de η con $k = 4$ (C_1, C_2, C_3, C_4). Las distancias $\overline{OC_1}$ y $\overline{OC_3}$ son iguales. Si se aplica un desplazamiento ϵ al ángulo η, se obtienen los nuevos puntos (P_1, P_2, P_3, P_4). Todas las distancias desde O hasta cualquier punto P son ahora diferentes.

Para obtener una configuración de rangos simétrica igualamos los valores de muestreo de ambos ángulos. El resultado obtenido para conseguir la simetría es de $a = \sqrt{\pi}$. Teniendo en cuenta que anteriormente se ha definido $k = \rho_{max}/p_s$ y $p_s = \pi A/2B$, el número de intervalos de los ángulos ξ y η se calculan como:

$$\begin{aligned} s_\xi &= \frac{2B\rho_{max}}{\sqrt{\pi}A} \\ s_\eta &= \frac{2B\rho_{max}\sqrt{\pi}}{\pi A} \end{aligned} \tag{4.20}$$

siendo estos los tamaños utilizados en el rango de las variables m_1 y h_1 de la integral.

El ángulo ω proporciona la rotación sobre el centro de la imagen experimental, como se puede observar en la Figura 4.1. El muestreo radial de la imagen se hace con un tamaño de $2B$, por lo que se utilizará este tamaño para definir el rango de la variable m'.

El ángulo η se desplaza ligeramente en función del valor de ϵ evitando de este modo que se obtengan soluciones duplicadas. El valor para ϵ que produce los mejores resultados es $\pi/(4k\sqrt{\pi})$, que equivale a 1/8 de paso del muestreo angular $2\pi/s_\eta$. Este valor se ha definido experimentalmente a través del estudio realizado en la sección 5.1.3. En la Figura 4.8 se puede ver el resultado de la aplicación del desplazamiento producido por ϵ.

Una vez establecidos los límites de las variables de la integral se procede a su cálculo y a continuación se aplica la transformada inversa de Fourier, obteniéndose la matriz de correlación. En esta matriz tridimensional se busca el valor más alto, que se corresponde con el mejor alineamiento. Cuando se encuentra el mejor valor de correlación, se opera con los índices de la matriz de acuerdo al cambio de variables realizado en la Ecuación 4.10 para obtener el triplete

angular final (ξ, η, ω).

4.1.3. Optimización de FRM2D para su comparación con FBM

Junto al modelo matemático de *Fast Bessel Matching*, se propuso una mejora de diseño para *Fast Rotational Matching* 2D [109]. Se ha decido implementar esta mejora en FRM2D con el fin de comparar *Fast Bessel Matching* contra una versión más eficiente de uno de los métodos de ajuste más rápidos.

El método de ajuste *Fast Rotational Matching* 2D [107, 108] utiliza dos parámetros rotaciones y uno traslacional para proporcionar el alineamiento entre dos imágenes. Tanto la imagen de referencia como la experimental rotan en sus centros, pero sólo esta última es trasladada a través de un eje. Este eje está formado por la línea que une los centros de ambas imágenes. La configuración de imágenes en FRM2D se puede ver en la Figura 4.9. Los ángulos mediante los que se ajustan las imágenes se denominan ϕ y ϕ' y la distancia entre los centros de las imágenes está definida mediante ρ. Según la ecuación principal desarrollada en [29], los parámetros de ajuste se obtienen del siguiente modo:

$$c(\phi, \phi'; \rho) = \sum_{m,n} e^{i(m\phi + n\phi')} I_{mn}(\rho) \tag{4.21}$$

donde

$$I_{mn}(\rho) = 2\pi \int_0^A (\hat{h}^n_{r,\rho})_m \cdot \overline{\hat{g}_m(r)} \cdot r \, dr \tag{4.22}$$

siendo

$$h^n_{r,\rho}(\beta) = e^{-in\beta'} \overline{\hat{f}_n(r')} \tag{4.23}$$

Para cada distancia entre los centros de las imágenes (ρ), se calcula la correlación entre los dos ángulos mediante una transformada 2D de Fourier. De esta manera se acelera la búsqueda rotacional de dos grados de libertad.

Teniendo en cuenta que la distancia ρ es usualmente pequeña comparada con el radio de las imágenes, las distancias r y r' serán parecidas para la mayoría de la imagen y los ángulos β y β' serán también muy similares para la mayoría de los puntos. Debido a esto, la transformada de Fourier de $(\hat{h}^n_{r,\rho})_m$ será pequeña salvo para los valores de m tales que $m+n$ sea un valor pequeño. Realizando un cambio de variables es posible aplicar de un modo más cómodo las restricciones necesarias en los bucles para evitar calcular partes de las integrales cuyo valor es muy próximo a cero. El cambio de variables es el siguiente:

$$\begin{aligned} q &= m + n \\ \psi &= \phi' - \phi \end{aligned} \tag{4.24}$$

quedando ecuación de correlación del siguiente modo:

$$c(\phi, \psi; \rho) = \sum_{q,n} e^{i(q\phi + n\psi)} I_{q-n,n}(\rho) \tag{4.25}$$

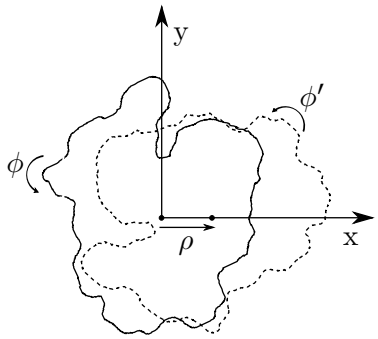


Figura 4.9: La partícula en línea continua permanece fija mientras que la de la linea discontinua se mueve traslacionalmente una distancia ρ a lo largo del eje x. Las imágenes rotan mediante los ángulos ϕ y ϕ' respectivamente.

y su transformada de Fourier es:

$$\hat{c}(q,n;\rho) = 2\pi \int_0^A (\hat{h}^n_{r,\rho})_{q-n} \cdot \overline{\hat{g}_{q-n}(r)} \cdot r\, dr \tag{4.26}$$

Asumiendo que la transformada de Fourier de las imágenes experimentales tiene valor cero para $|m| \geq (r/A)B$, es posible restringir el cálculo a $|(q-n)| < (B/A)r$. Del mismo modo se tiene $|n| < (B/A)r$ en $h^n_{r,p}$. Juntando estas dos condiciones se tiene:

$$r_0(q,n) = (A/B)max\{|n|,|(q-n)|\} \tag{4.27}$$

quedando la transformada de Fourier de la función de correlación del siguiente modo:

$$\hat{c}(q,n;\rho) = 2\pi \int_{r_0(q,n)}^A (\hat{h}^n_{r,\rho})_{q-n} \cdot \overline{\hat{g}_{q-n}(r)} \cdot r\, dr \tag{4.28}$$

Mediante esta restricción en el rango de los índices se evita computar los valores donde las integrales no tienen información. La mejora aplicada consigue por lo tanto una reducción en la cantidad de operaciones del método. La estimación de operaciones que se ejecutan en el método original es de $7,64B^3\rho + 7,64B^3$, que queda reducida a $6B^2\rho^2 + 4,09B^2\rho$ mediante la aplicación de esta optimización.

4.2. Adaptación de FBM a sistemas de computación de altas prestaciones

En esta tesis doctoral, el método de ajuste 2D *Fast Bessel Matching* ha sido portado a sistemas de computación de altas prestaciones, tanto para tarjetas gráficas como para sistemas de memoria distribuida. La adaptación para CUDA requiere tarjetas gráficas con una capacidad computacional 2.0 o superior. La adaptación sobre sistemas de memoria distribuida se ha llevado a cabo utilizando el estándar de comunicaciones MPI. La librería que se ha utilizado para las comunicaciones es Open MPI 1.6.2 [157]. A continuación se detallan las distintas estrategias de paralelización empleadas para estos sistemas de cómputo de altas prestaciones.

4.2.1. Adaptación sobre dispositivos gráficos

La aproximación que se toma para adaptar el método en dispositivos gráficos se basa en una paralelización a dos niveles. En primer lugar se paraleliza a nivel de alineamiento, utilizando un elevado número de *threads* para procesar los datos de un único ajuste. En segundo lugar se paraleliza a un nivel superior, realizando ajustes simultáneos por lotes. La cantidad de ajustes que se realizan al mismo tiempo está limitado por la cantidad de memoria disponible en la tarjeta gráfica.

Los alineamientos de *Fast Bessel Matching* en el dispositivo gráfico se llevan a cabo en cinco pasos:

1. Transferir al dispositivo gráfico las imágenes de referencia y experimentales.
2. Calcular la integral de la correlación (Ecuación 4.13).
3. Calcular la transformada inversa de Fourier correspondiente a la integral de la correlación.
4. Buscar la correlación más alta y calcular sus ángulos de alineamiento.
5. Transferir a memoria principal los valores de alineamiento calculados.

Para calcular la transformada 3D de Fourier, se ha empleado la librería cuFFT de NVIDIA [150]. Se ha empleado la interfaz *cufftPlanMany* con la configuración nativa de compatibilidad para obtener el mejor rendimiento. Mediante esta interfaz se pueden realizar múltiples transformadas de Fourier a través de una sola llamada a función. Prácticamente la totalidad de la memoria requerida por el método se destina al cálculo de las transformadas de Fourier mediante cuFFT. Por lo tanto, el número de transformadas de Fourier y por consiguiente la cantidad de alineamientos 2D procesados en lote, están limitados por la memoria de GPU disponible.

Restringidos por las necesidades de memoria de las transformadas de Fourier y por el ancho de banda, se han diseñado dos *kernels*: uno para computar las integrales y otro para buscar el mejor resultado de ajuste. Una vez que el primer *kernel* ha calculado todas las integrales, las transformadas inversas de Fourier se calculan a través de cuFFT. Las matrices resultantes conteniendo todos los valores de correlación se examinan para encontrar los mejores valores de ajuste. Después de que todas las imágenes del lote han sido procesadas, los mejores emparejamientos son seleccionados y transferidos a memoria principal.

Debido a que el ancho de banda utilizado en el método es variable, se han escrito dos versiones de cada *kernel* diseñado. Estas se diferencian en la cantidad de memoria compartida reservada por cada bloque. Una de las versiones está optimizada para anchos de banda menores o iguales a 128 mientras que la otra admite anchos de banda desde 128 hasta 256. De este modo, el uso de anchos de banda muy pequeños tiene un impacto menor tanto en la tasa de ocupación de los multiprocesadores como el en tamaño disponible para la caché L1.

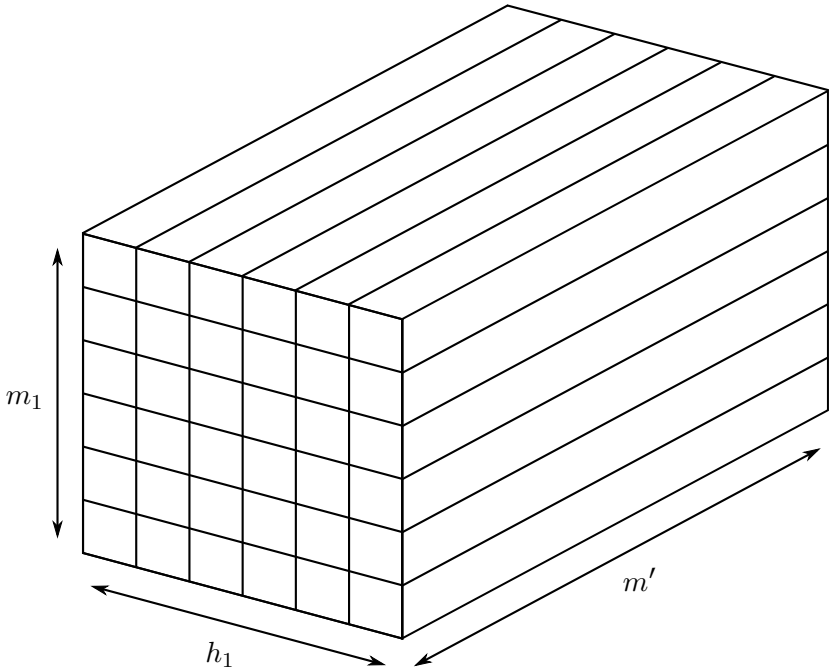


Figura 4.10: Volumen de una integral de correlación. Las dimensiones m_1 y h_1 tienen un tamaño de $2B\rho_{max}/\sqrt{\pi}A$. La dimensión m' tiene un tamaño de $2B$. Las divisiones que aparecen en el volumen indican el reparto de los datos de la integral entre los bloques del *grid*. El tamaño de bloque empleado es de B x 1 x 1. La cantidad de bloques utilizados para calcular una integral es de $(2B\rho_{max}/\sqrt{\pi}A)^2$.

4.2.1.1. *Kernel* para el cálculo de la integral de la correlación

Este *kernel* realiza el cálculo de la integral de la Ecuación 4.13:

$$\hat{T}(m_1,h_1,m') = 2\pi e^{i(h_1+m')\epsilon} \times \int_0^\infty F_{m_1+h_1+m'}(x) \times \overline{G_{m'}(x)} J_{m_1}(bx) J_{h_1}(bx) x\,dx$$

Esta integral genera un volumen cuyo tamaño se corresponde con los rangos de las variables m_1, h_1 y m' definidos en la sección 4.1.2.2. En la Figura 4.10 se puede ver un volumen de correlación y la división que se hace de este para asignar cada parte de la integral a los diferentes bloques de *threads*. Las dos dimensiones del volumen correspondientes a m_1 y h_1 tienen un tamaño de $2B\rho_{max}/\sqrt{\pi}A$. La última dimensión, correspondiente a m', tiene un tamaño de $2B$. La dimensión seleccionada para los bloques es de B x 1 x 1, por lo que cada *thread* procesa dos valores para cubrir la dimensión completa de m'. En cada bloque se fijan m_1 y h_1 al mismo valor para todos los *threads* y se calcula la integral para todos los valores de m'. A través de esta configuración de bloque es sencillo obtener estructuras con dimensiones que sean múltiplo del tamaño de *warp*.

Para obtener un mayor rendimiento se calculan varias integrales simultáneamente. Esto se logra a través del ajuste de las dimensiones del *grid*. En la jerarquía de alto nivel, este *grid* está organizado como un vector 2D. La disposición del *grid* para este *kernel* está esquematizada en la Figura 4.11. El número de imágenes de referencia y experimentales alineados simultáneamente

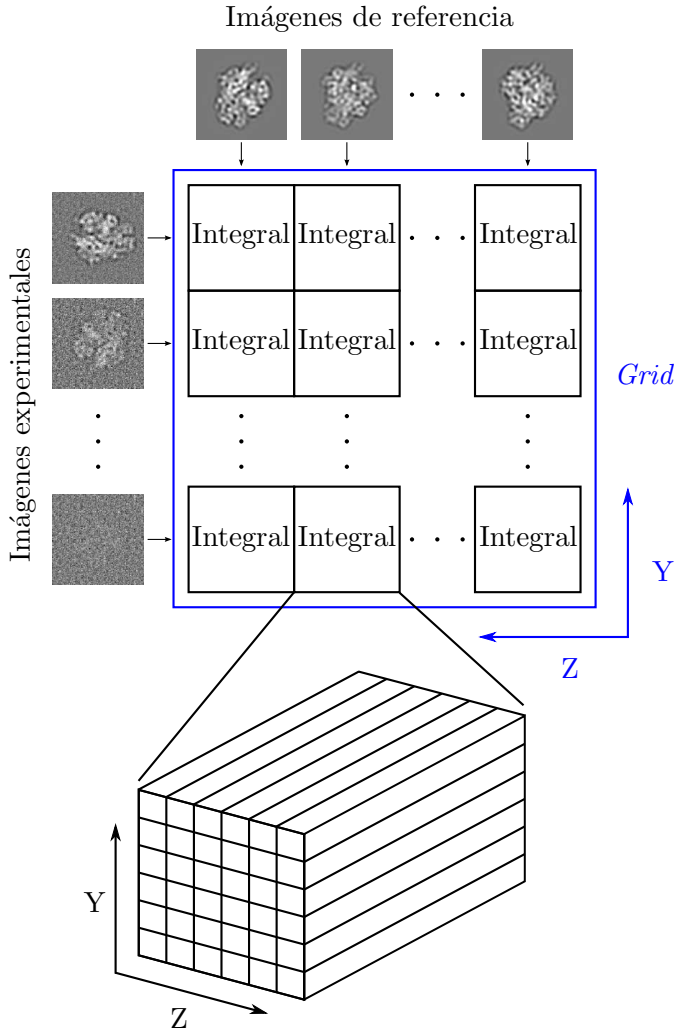


Figura 4.11: Disposición del *grid* para el *kernel* que genera la integral. Nótese que es posible modificar de forma independiente las dos dimensiones del *grid* para calcular las integrales necesarias dependiendo de la cantidad de imágenes de referencia y experimentales de las que se disponga.

está determinado por las dimensiones z e y de dicho *grid*.

4.2.1.2. *Kernel* para la búsqueda de los mejores ángulos de ajuste

La búsqueda del valor más alto en un conjunto de datos puede llevarse a cabo mediante una operación de reducción. La ejecución de esta clase de operaciones resulta poco eficiente sobre dispositivos gráficos debido a la baja intensidad aritmética y el decreciente uso de los procesadores a medida que el procedimiento avanza. Sin embargo, este proceso se lleva a cabo en la GPU debido a que es menos costoso en tiempo que transferir a memoria principal los volúmenes de correlación.

Se ha diseñado un proceso iterativo en el que un sólo bloque se dedica a encontrar la correlación más alta dentro de un único volumen de correlación. Como se puede ver en la Figura 4.12, el bloque tiene un tamaño de B x 1 x 1, coincidiendo con la mitad del tamaño

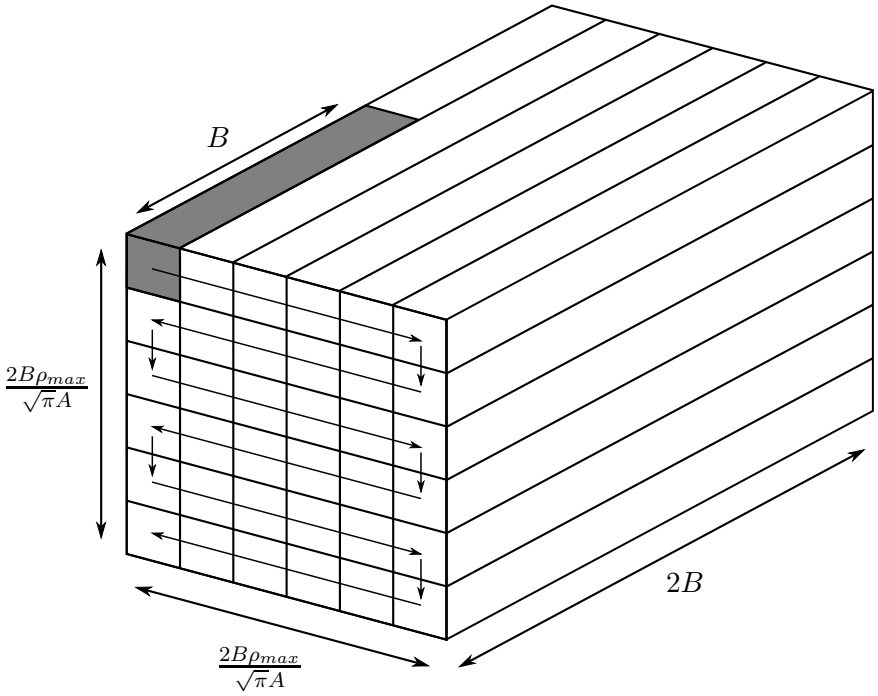


Figura 4.12: Búsqueda en un volumen de correlación. El volumen de correlación está indicado en color blanco. El bloque de *threads* está indicado en color gris. El bloque itera a través de todas las filas del volumen obteniendo el máximo de cada una de ellas. Cuando todas las filas se han procesado, el valor resultante es el máximo del volumen.

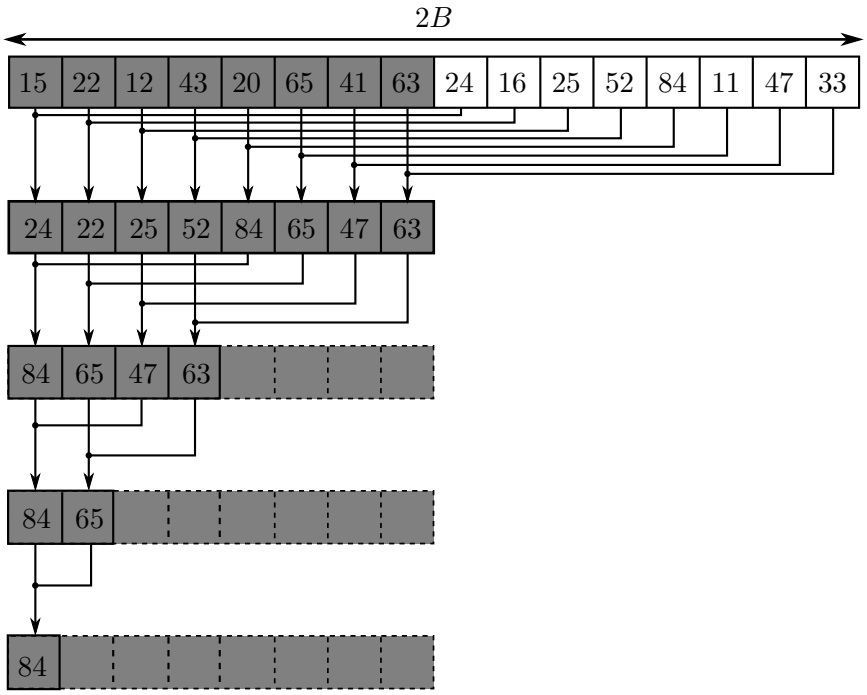


Figura 4.13: Reducción de una fila para la obtención del valor máximo. En gris se muestra un bloque de B *threads*. El tamaño inicial de la fila es de $2B$ datos. El proceso de reducción se realiza en varias iteraciones. En cada una de ellas se reduce la fila a la mitad de su tamaño comparando la mitad de sus valores con la otra mitad y conservando los más altos. Las lineas discontinuas del bloque indican los *threads* que han dejado de contribuir a la reducción.

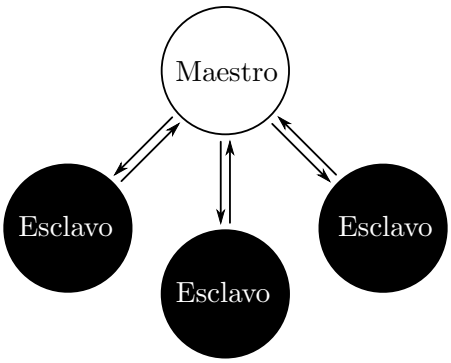


Figura 4.14: Modelo de comunicación Maestro/Esclavo. Los procesos esclavos no se comunican entre ellos. La cantidad de procesos esclavos es variable dependiendo de las limitaciones del método y del sistema en el que se ejecute, siendo requisito indispensable tener al menos un nodo de este tipo. Es necesario que exista un único proceso maestro.

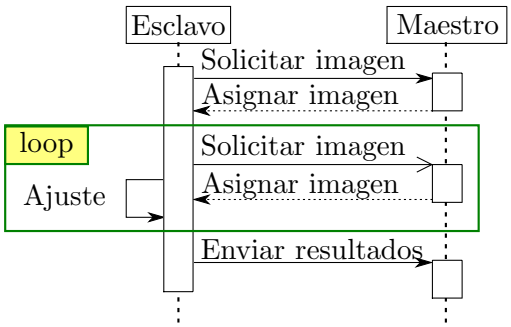


Figura 4.15: Secuencia de comunicación entre procesos de *Fast Bessel Matching* mediante MPI.

de la dimensión más grande del volumen. El bloque itera a través de cada una de las filas del volumen de correlación buscando su valor máximo. Cuando todas las filas se han procesado, el valor obtenido es el máximo del volumen. El valor más alto de cada fila se obtiene realizando una reducción de sus datos tal como se muestra en la Figura 4.13. Para realizar el cálculo del modo más eficiente posible, los datos de la fila se cargan en memoria compartida antes de realizar la reducción. Los valores de la primera mitad de la fila se comparan con los de la segunda mitad, conservando los más altos. Este paso se repite reduciendo en cada iteración a la mitad el tamaño de la fila hasta que sólo queda el valor más alto. A partir de los tres índices del valor máximo obtenido en el volumen de correlación se obtienen los valores para el triplete angular (ξ, η, ω). Estos ángulos definen la rotación y traslación del ajuste entre las dos imágenes. Para realizar la búsqueda del máximo en varios volúmenes de correlación simultáneamente el *grid* se expande en dos dimensiones: una para las imágenes de referencia y otra para las experimentales. Esta configuración de *grid* es análoga a la del *kernel* anterior.

4.2.2. Adaptación sobre sistemas de memoria distribuida

La adaptación sobre sistemas de memoria distribuida se ha llevado a cabo utilizando MPI para realizar la comunicación entre los diferentes procesos dentro del conjunto de ordenadores. Se utiliza un modelo Maestro/Esclavo para la asignación de tareas entre procesos. Un ejemplo

de este modelo de comunicación puede verse en la Figura 4.14. Mediante esta configuración se dispone de un único proceso llamado maestro que controla al resto de procesos del sistema, llamados esclavos. Cada esclavo sigue el esquema de flujo presentado en la versión secuencial. La secuencia de comunicación con MPI que se lleva a cabo entre maestro y esclavos se ilustra en la Figura 4.15. Se ha hecho uso del conjunto de funciones asíncronas de MPI en los procesos esclavos con el fin de solapar cómputo y comunicación, aumentando el rendimiento. En el proceso maestro se emplean las funciones síncronas, ya que su única tarea es esperar a recibir peticiones y atenderlas. Las definiciones de estas funciones se encuentran en el apéndice A. A continuación se detallan los diferentes tipos de mensajes definidos para la comunicación y coordinación entre procesos en la red.

- Mensajes de asignación de imágenes:

  Estos mensajes son utilizados por el proceso maestro para asignar a los procesos esclavos las imágenes experimentales que cada uno de ellos debe alinear. El contenido del mensaje es un número entero positivo que indica el índice de la imagen que el proceso esclavo debe leer del fichero de imágenes experimentales. Cuando no queden más imágenes por alinear, el proceso maestro enviará un valor negativo como índice de imagen. Esto indicará al proceso esclavo que debe enviar al maestro todos los resultados de los alineamientos que ha realizado.

- Mensajes de resultados:

  Estos mensajes contienen los resultados de los alineamientos. Esta clase de mensajes son exclusivamente enviados por los procesos esclavos al maestro cuando reciban la señal de finalización. Los resultados de los todos alineamientos se envían a la vez al final del proceso para hacer un uso más eficiente del ancho de banda de red. Los resultados de los alineamientos se almacenan en cinco vectores diferentes. Dos de estos vectores almacenan en formato entero el índice de la imagen alineada y el índice de la imagen de referencia más semejante a la alineada. Los tres vectores restantes almacenan los datos correspondientes a los parámetros de ajuste de la imagen alineada (ángulo de rotación y desplazamientos en x e y), en formato de coma flotante. El envío de resultados se realiza en cinco pasos. En cada uno de los pasos se realiza el envío de uno de los vectores de resultados. El proceso maestro recibe los resultados de cada esclavo, los ordena y los guarda.

4.3. Ajuste 3D para cribado virtual: FRODRUG

La segunda parte de este trabajo de tesis es la investigación sobre nuevo un método para realizar *docking* proteína-ligando y cribado virtual de proteínas. Este método ha sido denominado FRODRUG. El procedimiento de ajuste proteína-ligando desarrollado está basado en armónicos esféricos [82] y en el método de ajuste *Fast Rotational Matching 3D* (FRM3D) [107]. FRM3D ha sido empleado anteriormente con éxito en el diseño de otros métodos como *docking* proteína-proteína [62] o *fitting* multi-resolución [61]. El procedimiento de *docking* proteína-ligando consiste en una búsqueda en seis dimensiones (tres traslacionales y tres rotacionales). Mediante FRM3D

se acelera la búsqueda en las tres dimensiones rotacionales, mientras que las tres traslacionales son sistemáticamente exploradas.

4.3.1. *Fast Rotational Matching 3D*

Tanto la proteína (receptor) como el ligando se definen en el espacio $\mathbb{R}^3$ como funciones de densidad f y g respectivamente.

$f : \mathbb{R}^3 \to \mathbb{R}$,

$g : \mathbb{R}^3 \to \mathbb{R}$

El criterio de ajuste consiste en la maximización de una función de puntuación basada en distintos potenciales de interacción de energía. Cada término se calcula como una función de correlación definida del siguiente modo:

$$E(R) = \int_{\mathbb{R}^3} f \cdot \overline{\Lambda_R g} \tag{4.29}$$

donde el término Λ_R denota la operación de rotación sobre g, definida por los ángulos de Euler (ϕ, θ, ψ). La posición final de g se obtiene componiendo una serie de rotaciones. En primer lugar se realiza la rotación de un ángulo ϕ en el eje Z, seguida de una rotación de ángulo θ en el eje Y, para finalmente terminar aplicando una rotación de ψ en el eje Z.

La energía de interacción de las moléculas del receptor y del ligando pueden ser expresadas en la unidad esfera en términos de funciones de armónicos esféricos Y_{lm} y sus correspondientes coeficientes, $\hat{f}_{lm}(r)$ y $\hat{g}_{lm}(r)$:

$$\begin{aligned} f(r\mathbf{u}) &\approx \sum_{l=0}^{B-1} \sum_{m=-1}^{l} \hat{f}_{lm}(r) Y_{lm}(\mathbf{u}) \\ g(r\mathbf{u}) &\approx \sum_{l=0}^{B-1} \sum_{m=-1}^{l} \hat{g}_{lm}(r) Y_{lm}(\mathbf{u}) \end{aligned} \tag{4.30}$$

donde r es el radio de la esfera y $\mathbf{u}$ es el vector unidad en la dirección y sentido de r. Las variables l y m son el grado y el orden del armónico esférico respectivamente. La generación de los coeficientes $\hat{f}_{lm}(r)$ y $\hat{g}_{lm}(r)$ correspondientes al muestreo de la proteína y el ligando se detalla en la sección 4.3.2.2. La constante B se corresponde con el ancho de banda; esta define el muestreo angular que se seleccione para los tres ángulos de Euler y el rango de frecuencias empleadas en los armónicos esféricos. Un ancho de banda de 16 proporciona un muestreo angular de 11,25º, lo que implica la prueba de más de 15.000 rotaciones diferentes.

Las funciones de armónicos esféricos Y_{lm} se pueden escribir en términos de funciones asociadas de Legrende P_l^m del siguiente modo:

$$Y_{lm}(\beta, \lambda) = (-1)^m \sqrt{\frac{(2l+1)(l-m)}{4\pi(l+m)!}} P_l^m(\cos\beta) e^{im\lambda} \tag{4.31}$$

donde las variables β y λ expresan la co-latitud y longitud en la esfera. Para cualquier rotación

se tiene:

$$(\Lambda_R g)(r\mathbf{u}) = \sum_{l,m,n} \hat{g}_{lm}(r) D^l_{nm}(R) Y_{ln}(\mathbf{u}) \tag{4.32}$$

donde D^l_{nm} son los coeficientes que definen la matriz de elementos de la representación irreducible del grupo rotacional tridimensional [19]. Estos coeficientes se pueden escribir del siguiente modo mediante ángulos de Euler:

$$D^l_{mn}(\phi, \theta, \psi) = e^{-im\phi} d^l_{mn}(\theta) e^{-in\psi} \tag{4.33}$$

donde las funciones d^l_{mn} son reales. Estas funciones y el modo en que se calculan pueden encontrarse en [19, 174].

Integrando las Ecuaciones 4.30 y 4.32 en la 4.29 se obtiene la siguiente ecuación:

$$E(R) = \sum_{ll'mm'n} \overline{D^{l'}_{nm'}(R)} \int_{\mathbb{R}^3} \hat{f}_{lm}(r) \overline{\hat{g}_{l'm'}(r)} Y_{lm}(\mathbf{u}) \overline{Y_{l'n}(\mathbf{u})} \tag{4.34}$$

Debido a la ortogonalidad de los armónicos esféricos, la integral en la Ecuación 4.34 se puede reducir a:

$$\delta_{ll'} \delta_{mn} I^l_{mm'} \tag{4.35}$$

siendo δ una función delta de Kronecker y

$$I^l_{mm'} = \int_0^\infty \hat{f}_{lm}(r) \cdot \overline{\hat{g}_{lm'}(r)} \cdot r^2 \cdot dr \tag{4.36}$$

A continuación se factoriza la rotación R en la Ecuación 4.34 y se aplica un cambio de variables $(\phi, \theta, \psi \to \xi, \eta, \omega)$. En términos de ángulos de Euler, tenemos la rotación $R = (\phi, \theta, \psi)$. Se cumple que $R = R_1 \cdot R_2$ siendo estas rotaciones:

$$\begin{aligned} R_1 &= (\xi, \frac{\pi}{2}, 0) \\ R_2 &= (\eta, \frac{\pi}{2}, \omega) \end{aligned} \tag{4.37}$$

el valor de estos nuevos parámetros es:

$$\begin{aligned} \xi &= \phi - \frac{\pi}{2} \\ \eta &= \pi - \theta \\ \omega &= \psi - \frac{\pi}{2} \end{aligned} \tag{4.38}$$

Teniendo en cuenta que

$$D^l_{nm}(R_1 \cdot R_2) = \sum_h D^l_{nh}(R_1) D^l_{hm}(R_2) \tag{4.39}$$

se puede utilizar la Ecuación 4.33 para expresar $D^l_{nm}(R)$ como:

$$D^l_{nm}(R) = \sum_h d^l_{nh}\left(\frac{\pi}{2}\right) d^l_{hm}\left(\frac{\pi}{2}\right) e^{-i(n\xi + h\eta + m\omega)} \tag{4.40}$$

De este modo, mediante la sustitución de la Ecuación 4.40 en la 4.34 obtenemos la ecuación de correlación:

$$E(R) = \sum_{lmhm'} d^l_{mh}\, d^l_{hm'}\, I^l_{mm'} e^{i(m\xi + h\eta + m'\omega)} \tag{4.41}$$

Calculando la transformada inversa de Fourier de esta ecuación se obtiene la ecuación principal del método:

$$E(R) = FT^{-1}_{m,h,m'} \left[\sum_l d^l_{mh}\left(\frac{\pi}{2}\right) d^l_{hm'}\left(\frac{\pi}{2}\right) I^l_{mm'} \right] \tag{4.42}$$

Así, para una cierta traslación, la correlación para todas las rotaciones puede ser calculada muy rápidamente con una sola transformada inversa de Fourier.

En el caso especial de ajuste proteína-ligando, en este trabajo de tesis se ha realizado una modificación en la representación de este último para aumentar la eficiencia. El ligando (g) se expresa ahora mediante un conjunto de propiedades atómicas como un sumatorio de la forma $\sum_i^N L_i \cdot \delta_{p_i}$ siendo L_i propiedades atómicas para cada uno de los átomos i, y δ_{p_i} una función delta de Kronecker del átomo i centrado en su posición $p = \{r_i, \beta_i, \lambda_i\}$. De este modo, sus coeficientes esféricos se pueden representar siguiendo esta ecuación:

$$\begin{aligned} \hat{g}_{lm}(r) &= \sum_{i=0}^{N} L_i \int_{s^2} \delta_{p_i}(r\mathbf{u})\overline{Y_{lm}(\mathbf{u})}\, \mathrm{d}\sigma \\ &= \sum_{i=0}^{N} \frac{L_i}{r^2} \delta_{r_i}(r)\overline{Y_{lm}(\mathbf{u}_i)} \end{aligned} \tag{4.43}$$

siendo σ los elementos del volumen en coordenadas polares.

Sustituyendo esta ecuación en 4.36 se obtiene la siguiente integral:

$$\begin{aligned} I^l_{mm'} &= \int_0^{\infty} \hat{f}_{lm}(r) \cdot \sum_{i=0}^{N} L_i \cdot \delta_{r_i}(r) \cdot Y_{lm'}(\mathbf{u}_i) \cdot \mathrm{d}r \\ &= \sum_{i=0}^{N} L_i \cdot \hat{f}_{lm}(r_i) \cdot Y_{lm'}(\mathbf{u}_i) \end{aligned} \tag{4.44}$$

Iterando a través de cada uno de los átomos del ligando se evita tener que calcular los coeficientes de armónicos esféricos a través de un mapa de potenciales.

Integrando la Ecuación 4.44 en la 4.42 se obtiene la ecuación principal del método:

$$E(R) = FT^{-1}_{m,h,m'} \left[\sum_l d^l_{mh}\left(\frac{\pi}{2}\right) d^l_{hm'}\left(\frac{\pi}{2}\right) \sum_{i=0}^{N} L_i \cdot \hat{f}_{lm}(r_i) \cdot Y_{lm'}(\mathbf{u}_i) \right] \tag{4.45}$$

Los coeficientes d pueden ser precalculados, así como las correspondientes funciones armónicas del receptor ya que permanece fijo en el espacio. Esto hace que el cálculo de la ecuación principal del método sea muy eficiente.

4.3.2. Adaptación secuencial de FRODRUG

Se han realizado varias adaptaciones de FRODRUG. En primer lugar se detalla la versión secuencial. Más adelante, en la sección 4.4, se muestran las adaptaciones y optimizaciones para

sistemas de memoria distribuida y arquitecturas gráficas.

El diagrama de flujo del proceso general de *docking* para varios ligandos puede verse en la Figura 4.16. En primer lugar se definen los puntos que componen el sitio de unión sobre la proteína y se calculan los coeficientes energéticos para cada uno de ellos. Luego se aplica el método para cada ligando iterando a través de todos los puntos del sitio de unión de la proteína. Por último se guardan las mejores soluciones para cada ligando.

4.3.2.1. Definición del sitio de unión

El primer paso de proceso de cribado es la generación del conjunto de puntos traslacionales que van a componer el sitio de unión del ligando sobre la proteína. Este conjunto de posiciones espaciales se genera a través de la estructura tridimensional de la proteína, obtenida a partir de su fichero PDB [163]. El fichero en formato PDB contiene la información sobre los átomos que componen la proteína y sus posiciones en el espacio. Una vez leídos los átomos, se genera un volumen en el que albergarlos en sus respectivas posiciones. Este tiene un muestreo de 1 Å por vóxel y su tamaño se introduce como parámetro de entrada al método. El volumen está centrado en el punto correspondiente al centro de masas del ligando de referencia. Los valores almacenados en los vóxeles de este volumen indicarán únicamente si en ese espacio existe algún átomo del receptor o no. Para ello se emplea el radio de Van der Waals de cada átomo [17]. Desde cada uno de los vóxeles del volumen se realiza una traza de rayos. De este modo es posible conocer, por cada uno, su distancia máxima y mínima a la superficie de la proteína. Teniendo en cuenta estas distancias es posible omitir, en la generación del sitio de unión, las posiciones traslacionales que hagan que el ligando quede demasiado lejos de la superficie de la proteína o bien quede en el interior de esta. Tras aplicar estas restricciones se obtiene el listado final con las posiciones traslacionales en las que se van a evaluar los ligandos a lo largo del proceso de cribado. La Figura 4.17 muestra en su parte izquierda la estructura de una proteína en color azul y su ligando de referencia incluido en el PDB en color verde. En la parte derecha se ilustra en color rojo los diferentes puntos seleccionados para la generación del sitio de unión de esta proteína teniendo en cuenta las restricciones aplicadas.

4.3.2.2. Muestreo de la proteína y el ligando

A continuación se realiza el cálculo de los coeficientes energéticos del receptor ($\hat{f}_{lm}(r)$) para todos los puntos del sitio de unión. En primer lugar se genera un cubo que envuelve la estructura completa de la proteína. Este cubo tiene un muestreo de 1 Å por vóxel. El total de energía para cada posición traslacional se calcula sumando los distintos potenciales energéticos de cada uno de los átomos de la proteína. Se han considerado los dos tipos de potencial más sencillos: Van der Waals [43] y electrostático. La energía se calcula según se indica en la siguiente ecuación:

$$P(p) = \sum_{i}^{N} P_W^{(i)}(p) + P_E^{(i)}(p) \tag{4.46}$$

siendo N el total de átomos de la proteína, $P_W^{(i)}(p)$ la contribución del potencial de Van der Waals del átomo i en la posición p y $P_E^{(i)}(p)$ la correspondiente contribución de energía electrostática.

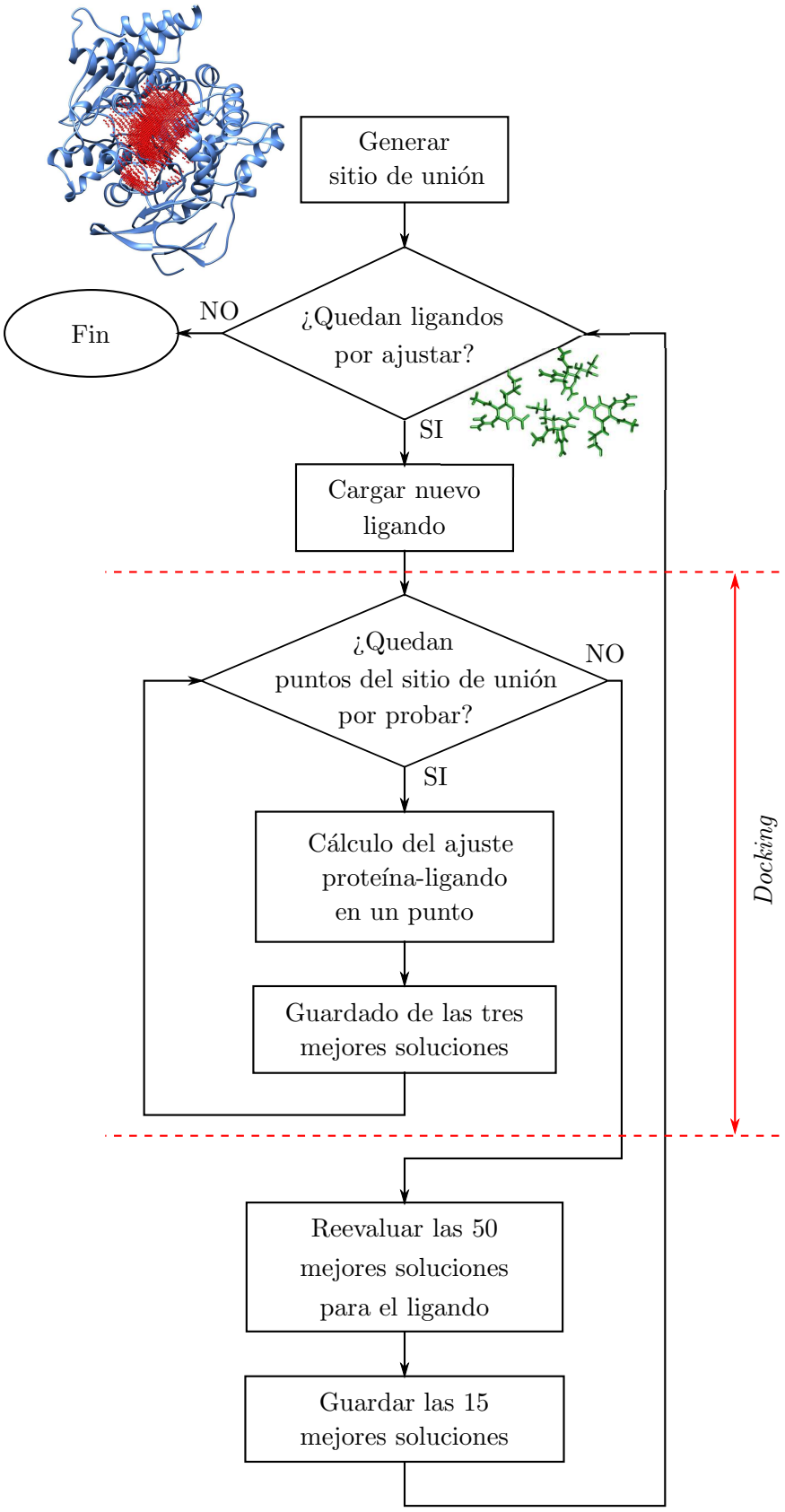


Figura 4.16: Flujo de ejecución de FRODRUG para múltiples ligandos. Primero se generan los puntos del sitio de unión. A continuación se prueban los ligandos uno a uno en todos los puntos del sitio de unión. Se guardan las 15 mejores soluciones obtenidas para cada ligando.

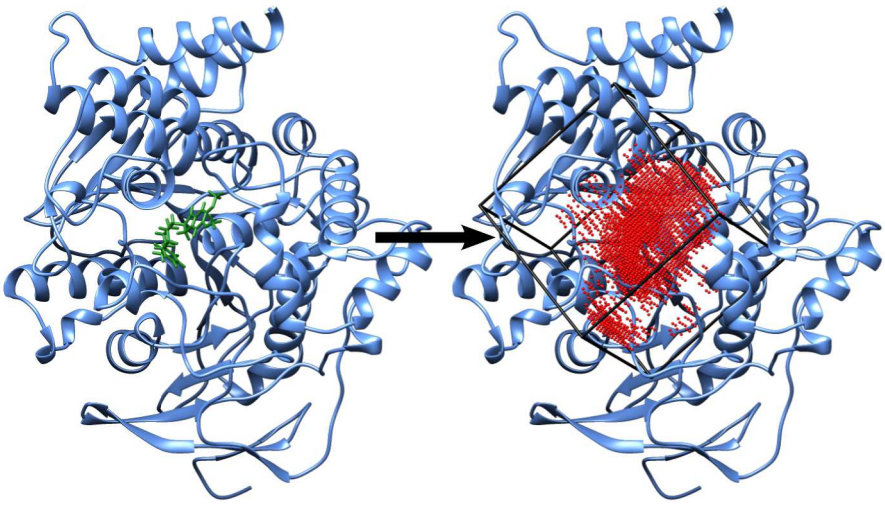


Figura 4.17: En la izquierda se muestra una proteína (azul) y su ligando de referencia incluido en el PDB (verde). En la derecha se muestra en rojo los puntos seleccionados para el sitio de unión de este complejo.

El cálculo de la energía de Van der Waals se realiza mediante un potencial de Lennard-Jones 6-12 [111] empleado anteriormente con éxito en [51, 66]. El potencial se calcula del siguiente modo:

$$P_W^{o(i)}(p) = \frac{A_i}{r_{pi}^{12}} - \frac{B_i}{r_{pi}^{6}} \tag{4.47}$$

donde la variable r_{pi} indica la distancia entre el centro del átomo y el punto del volumen para el que se está calculando el potencial, y A y B son constantes que dependen de cada uno de los átomos. Estas se calculan a partir del número de electrones, la polarizabilidad y el radio de los átomos en cuestión [83]. Para poder calcular el potencial en un punto del espacio mediante esta ecuación, se simula que en ese punto hay un átomo genérico de carbono con un radio de 1,7 Å. La alta sensibilidad del potencial de Van der Waals a pequeños cambios conformacionales puede generar una gran cantidad de ruido en el cálculo de la energía. Con el fin de reducir este ruido la parte repulsiva del potencial es truncada; el cálculo se realiza según la Ecuación 4.48:

$$P_W^{(i)}(p) = \begin{cases} P_W^{o(i)}(p) & \text{si} \quad P_W^{o(i)}(p) \leq 0 \\ \dfrac{P_W^{o(i)}(p) \cdot P_{max}}{P_W^{o(i)}(p) + P_{max}} & \text{si} \quad P_W^{o(i)}(p) > 0 \end{cases} \tag{4.48}$$

siendo P_{max} la constante que limita el potencial. Esta constante se fija a un valor de 1,0 Kcal/mol [51].

El potencial energético correspondiente a la parte electrostática ($P_E^{(i)}(p)$) se calcula a partir de la ley de Coulomb modificada:

$$P_E^{(i)}(p) = \frac{q_i}{4r_{pi}^2} \tag{4.49}$$

donde q_i indica la carga del átomo i y r_{pi} es la distancia entre el centro del átomo y el punto del volumen para el que se está calculando el potencial.

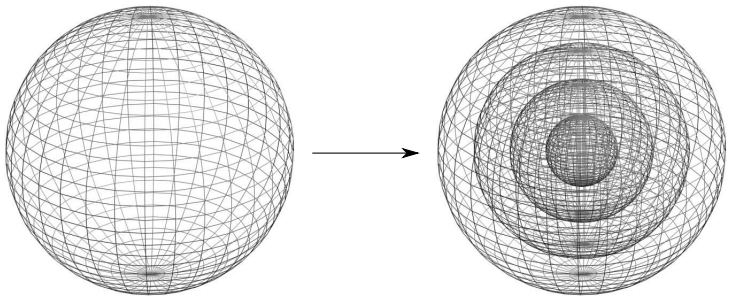


Figura 4.18: Mapeado esférico de los coeficientes del receptor. En la parte izquierda se puede ver una esfera con sus ejes latitudinal y longitudinal divididos en 32 segmentos. Cada esfera almacena 1.024 valores. En la parte derecha se pueden observar varias esferas concéntricas de diferentes radios. Cada una de ellas corresponde a una capa de la representación esférica.

Tanto el potencial de Van der Waals como el electrostático del receptor se pueden pre-calcular, por lo que estos términos energéticos se leen desde ficheros almacenados en disco antes de comenzar el proceso de cribado.

Una vez realizados estos pre-cálculos se dispone de dos volúmenes cuyos vóxeles contienen la contribución de todos los potenciales energéticos en cada punto traslacional. A continuación, en las coordenadas traslacionales pertenecientes a cada punto del sitio de unión se realizan varios muestreos esféricos. Estos se hacen con un paso angular de $5{,}6^{\text{o}}$ en latitud y de $11{,}25^{\text{o}}$ en longitud, que se corresponde con dividir ambas dimensiones en 32 segmentos almacenando así 1.024 valores por esfera. Estas dimensiones están producidas por el empleo de un ancho de banda fijo de 16. Los puntos definidos en cada esfera pueden observarse en la Figura 4.18. Cada uno de estos valores se calcula mediante la interpolación trilineal de los vóxeles más próximos al punto en cuestión en el volumen de potenciales. La cantidad de muestreos esféricos que se realizan por cada punto del sitio de unión depende del radio de los ligandos que se van a evaluar. Se hace un muestreo esférico por cada 0,5 Å a partir del centro del vóxel hasta una distancia igual al radio del ligando más grande. A continuación se aplica la transformada directa de Fourier a estos datos y se multiplican por las correspondientes funciones armónicas ($Y_{lm}(\beta, \lambda)$) para obtener finalmente los coeficientes armónicos del receptor. Este proceso se realiza sólo una vez antes del comienzo del cribado.

El muestreo de cada uno de los ligandos a cribar se hace justo tras su carga desde disco, antes de ser utilizados. El primer paso para calcular los coeficientes esféricos del ligando consiste en obtener sus propiedades atómicas (L_i), que van a ser empleadas para el cálculo de la correlación. Por ejemplo, para el potencial de Van der Waals, esas propiedades son la masa de cada uno de los átomos. A continuación se calculan los coeficientes de armónicos esféricos correspondientes a la latitud y longitud de cada uno de sus átomos. Estos valores de latitud y longitud se obtienen a partir de una esfera con el origen coincidiendo con el centro de masas del ligando.

Tras estas operaciones, los coeficientes de proteína y ligando están preparados para ser empleados en el cálculo de la correlación definida en la Ecuación 4.45.

4.3.2.3. Obtención de ángulos de ajuste 3D

La obtención de los ángulos que proporcionan la posición del ligando sobre la proteína se lleva a cabo a través de un proceso iterativo. Por cada ligando de la base de datos se itera a través de cada uno de los puntos del sitio de unión. Para cada pareja de ligando-punto, el cálculo de la puntuación de *docking* se realiza en cuatro pasos: en primer lugar se calcula el sumatorio definido en la Ecuación 4.44. A continuación se multiplican estos datos por los coeficientes que definen la matriz de elementos de la representación del grupo tridimensional rotacional como está indicado en los corchetes dentro de la Ecuación 4.45. En tercer lugar, para el volumen de correlación se calcula una transformada inversa de Fourier utilizando para ello la librería FFTW [47]. Por último, una vez que se ha obtenido la transformada inversa, se realiza una búsqueda sobre este volumen tratando de encontrar máximos. Se define un máximo como un vóxel cuyo valor es superior al de todos sus vecinos. Por cada máximo local encontrado se realiza una interpolación de su valor con el de sus vecinos para mejorar la precisión. Por cada volumen de correlación se almacenan los tres máximos con valores más altos. De entre los máximos más altos para todos los puntos del sitio de unión se seleccionan y almacenan sólo los 50 mejores. Los ángulos de ajuste para una solución se obtienen a partir de los tres índices almacenados para el máximo.

4.3.2.4. Revaluación de las soluciones

Las soluciones obtenidas a través de la búsqueda de máximos en el volumen son a continuación revaluadas. Esto se lleva a cabo a través de una función probabilística basada en estructuras de complejos proteína-ligando ya conocidas. Esta función es parecida a DSX [142] y a DrugScore [224].

Por cada una de las 50 mejores soluciones obtenidas para el ligando, se leen los datos de rotación y traslación y se aplican a cada uno de sus átomos para situarlo en la posición indicada. A continuación se calcula la nueva puntuación para cada solución según la Ecuación 4.50:

$$p_{total} = \sum_{a_p} \sum_{a_l} p(a_p, a_l, r_{a_p a_l}) \tag{4.50}$$

siendo a_p y a_l los tipos de átomo de la proteína y el ligando respectivamente, y $r_{a_p a_l}$ la distancia entre cada par de átomos. Para reducir el tamaño del problema, se generan diferentes conjuntos con las parejas de tipos de átomos a_p y a_l que producen potenciales similares. El potencial probabilístico para cada conjunto de tipos de átomos se calcula como:

$$p(c, r) = -\ln\left(\frac{P(c|r)}{\overline{P}(c)}\right) \tag{4.51}$$

siendo c cada uno de los conjuntos de tipos de átomos con potenciales similares, $P(c|r)$ la probabilidad de encontrar los átomos del conjunto c a la distancia r, y $\overline{P}(c)$ la probabilidad de encontrarlos en un estado de referencia con fuerzas promediadas [201]. Las probabilidades se

calculan como:

$$\overline{P}(c) = \frac{\sum_{r'} P(c|r')}{n_r} \tag{4.52}$$

$$P(c|r) = \frac{N(c,r)}{F(c)\sum_{c'} \frac{N(c',r)}{F(c')}} \tag{4.53}$$

siendo n_r el número de divisiones espaciales realizadas en la distancia r, $N(c,r)$ el número de ocurrencias de los tipos de átomos del conjunto c a la distancia r, y $F(c)$ un factor de normalización que indica el número total de ocurrencias de los tipos de átomos del conjunto c en toda la base de conocimiento.

Una vez que este proceso se ha realizado para cada una de las 50 mejores soluciones generadas, se realiza una reordenación de estas empleando como clave el valor de la función basada en conocimiento y se almacenan finalmente las 15 mejores soluciones por cada ligando evaluado.

4.4. Adaptación de FRODRUG a sistemas de cómputo de altas prestaciones

Se han explorado dos estrategias de paralelización para la adaptación de FRODRUG a sistemas de cómputo de altas prestaciones. Por una parte se ha adaptado una versión empleando la plataforma CUDA para cómputo sobre dispositivos gráficos de NVIDIA (sección 4.4.1). Por otra parte se ha realizado una extensión empleando MPI para llevar a cabo el proceso de *docking* tanto en CPUs como en GPUs sobre sistemas de memoria distribuida (sección 4.4.2).

4.4.1. Adaptación sobre dispositivos gráficos

La adaptación sobre dispositivos gráficos de FRODRUG se divide en seis etapas. Salvo el paso de la transformada de Fourier y las transferencias de datos, cada una de las restantes etapas se procesa mediante un *kernel*. La división del proceso se ha hecho del siguiente modo:

- Transferencia de la proteína y los ligandos a memoria de vídeo.
- Cálculo de la integral de los coeficientes de proteína y ligando definida en la Ecuación 4.45.
- Generación de los volúmenes de correlación a través de la multiplicación de las constantes d, definidas a través de la Ecuación 4.40 con los valores de integral generados en el paso anterior.
- Cálculo de la transformada inversa de Fourier de los volúmenes de correlación. Para obtener las transformadas se hace uso de la librería cuFFT [150] proporcionada por NVIDIA.
- Búsqueda de las orientaciones de ligandos con valor de correlación más alto en los volúmenes generados en el paso anterior.
- Transferencia de los resultados a memoria principal.

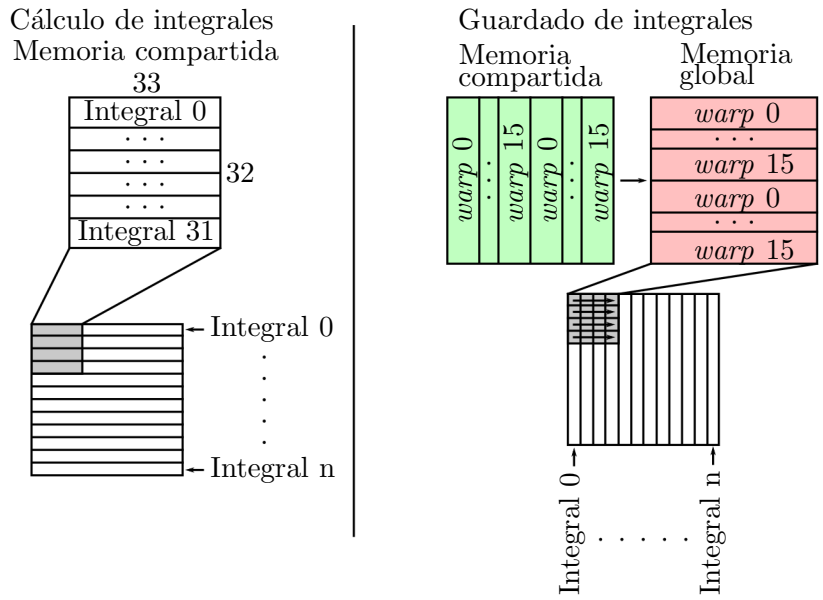


Figura 4.19: La figura ilustra el proceso de generación de la integral para cada punto traslacional. Los datos para cada integral se calculan en dos pasos escribiéndose por filas en memoria compartida, luego son transpuestos antes de ser escritos en memoria global, también en dos pasos. La zona verde indica lecturas de memoria. La zona roja indica escrituras.

Cada uno de los tres *kernels* ha sido optimizado para aprovechar al máximo las capacidades de la caché disponible. Algunas de las optimizaciones de estos *kernels* están basadas en el tamaño de *warp*. El ancho de banda de la versión CUDA del método de *docking* tiene un valor constante de 16. Esto se ha designado así ya que con este tamaño se producen estructuras de datos que se ajustan al tamaño de *warp*. La adaptación realizada requiere el uso de tarjetas gráficas con capacidad computacional 2.0 o mayor (arquitecturas de Fermi en adelante) debido al número de registros y *threads* por bloque requeridos por los *kernels*. A continuación se detallan cada uno de los tres *kernels*.

4.4.1.1. *Kernel* para el cálculo de la integral

Este *kernel* se encarga de calcular una integral (Ecuación 4.44) para cada punto traslacional de un sitio de unión en una proteína para un ligando dado.

El tamaño de las integrales generadas depende del ancho de banda seleccionado según:

$$tamaño = \frac{B(4B^2 - 1)}{3}$$

Para el ancho de banda utilizado (16), se generan unas integrales que contienen 5.456 valores complejos. Los datos complejos se almacenan en memoria separados en dos vectores diferentes: uno para la parte real y otro para la parte imaginaria. Esto se hace para acelerar las lecturas y escrituras en los accesos a la memoria global del dispositivo. El proceso del cómputo de la integral está ilustrado en la Figura 4.19. En la parte izquierda de la figura se muestra un bloque de memoria compartida de 32 x 33 valores conteniendo datos de 32 integrales diferentes. La columna extra que se agrega al bloque de memoria compartida sirve para distribuir adecuadamente los

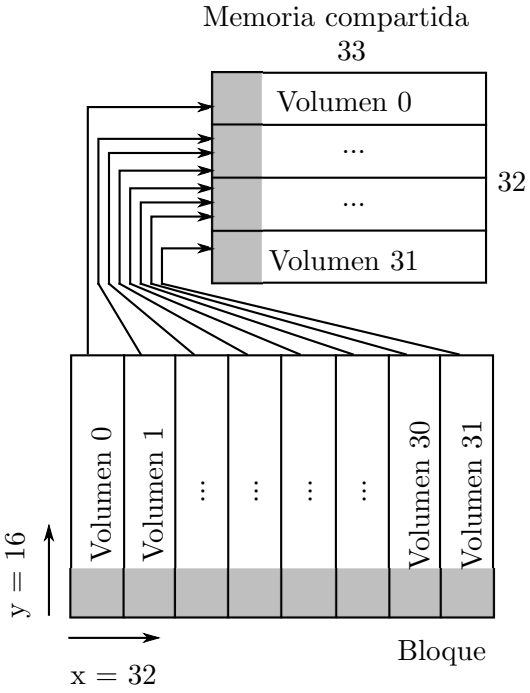


Figura 4.20: Configuración de bloque para el *kernel* que genera los volúmenes de correlación. Cada bloque de *threads* procesa 16 valores complejos de 32 volúmenes de correlación diferentes y almacena la parte real e imaginaria en columnas consecutivas del bloque de memoria compartida. Los rectángulos grises indican los datos tratados por los 32 *threads* de un mismo *warp*.

datos entre los diferentes bancos de memoria, evitando de este modo conflictos de acceso a los mismos. La forma del bloque de *threads* es rectangular, con 32 *threads* en el eje X, ajustándose al tamaño de *warp*. La dimensión Y del bloque se ajusta a 16. Debido a esta configuración de *threads*, el bloque de memoria compartida tiene que ser computado en dos iteraciones al tener el doble de datos a calcular que *threads* disponibles. El tamaño de bloque se ha fijado de este modo por motivos de eficiencia, ya que el *hardware* se comportaba mejor con bloques de menor tamaño debido a limitaciones en los recursos físicos. En la parte derecha de la figura se muestra el proceso de la escritura de los datos a memoria global. Los *warps* leen los datos de la memoria compartida por columnas y los almacenan de forma lineal en memoria global. Como en la operación de cómputo de datos, este proceso se realiza en dos iteraciones. Los datos de las integrales se escriben traspuestos en memoria global para optimizar su lectura por el siguiente *kernel*, que necesita leer los datos por columnas.

Para formar el *grid* de bloques de este *kernel*, se extienden las dimensiones X e Y cuanto sea necesario para cubrir todas las integrales.

4.4.1.2. *Kernel* para la generación de la matriz de correlación

Este *kernel* multiplica los datos de las integrales por los coeficientes definidos en la Ecuación 4.33. La operación que el *kernel* lleva a cabo se encuentra definida en la Ecuación 4.42:

$$\sum_l d^l_{mh}\left(\frac{\pi}{2}\right)\cdot d^l_{hm'}\left(\frac{\pi}{2}\right)\cdot I^l_{mm'}$$

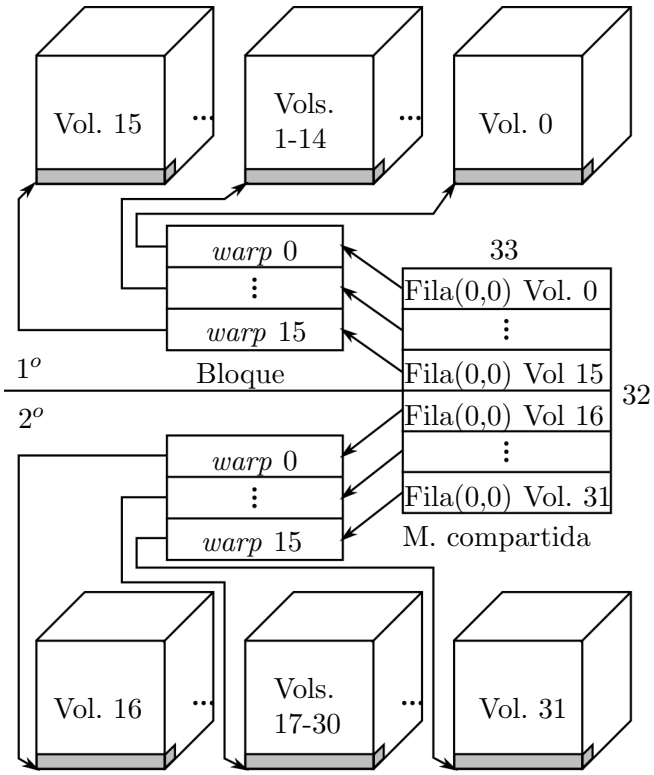


Figura 4.21: Guardado de los volúmenes de correlación. Cada bloque de *threads* almacena sus datos en dos pasos, guardando la mitad de los datos en cada uno. El primer paso se ilustra en la parte superior y el segundo en la parte inferior. En este ejemplo, las filas almacenadas en memoria global corresponden a los índices Z=0 e Y=0 de cada volumen de correlación.

Cada integral produce un volumen de correlación de 32 x 32 x 34 valores flotantes. Estos volúmenes de correlación tienen una forma cúbica cuyas dimensiones están definidas por el doble del ancho de banda, con dos columnas adicionales sin datos en la última dimensión que son requeridas por la transformada de Fourier. La parte real e imaginaria está intercalada en el volumen. Su tamaño por lo tanto es de 32 x 32 x 17 valores complejos. Sólo dos de los cuadrantes del volumen son computados; los otros dos son generados por simetría a partir de los ya calculados. Teniendo en cuenta todo, por cada volumen se calculan en total 16 x 32 x 16 datos complejos.

En la Figura 4.20 se puede observar la configuración seleccionada para los bloques de *threads*. Se ha fijado una longitud de 32 *threads* en el eje X, ajustándose al tamaño de *warp*, y 16 *threads* en el eje Y. La tarea de un bloque individual es calcular una fila de 16 valores complejos correspondientes a 32 volúmenes de correlación diferentes. Cada bloque tiene una sección de memoria compartida de 32 x 33 valores flotantes para almacenar los valores de correlación. Al igual que antes, esta columna sirve para evitar conflictos de bancos. La zona marcada en gris en la Figura 4.20 corresponde a *threads* del mismo *warp*, indicando que los datos se escriben por columnas en memoria compartida. Cada *thread* dentro del *grid* calcula un único valor complejo dentro del volumen que le corresponde. El volumen asignado a cada *thread* se define a través del cálculo de blockIdx.x · blockDim.x + threadIdx.x. Dentro de este volumen, las coordenadas del valor complejo que calcula cada *thread* se definen mediante (X=threadIdx.y, Y=blockIdx.y,

Z=blockIdx.z). Ambas partes del valor complejo son calculadas en serie por el mismo *thread*. Esta configuración de bloques se ha tomado para reducir la divergencia de *threads*. Así, todos los valores complejos dentro de un mismo volumen se calculan de diferente modo atendiendo a una configuración de bucles definida por sus índices (X,Y,Z) dentro del volumen. Es muy conveniente que todos los *threads* de un mismo *warp* calculen el valor complejo con el mismo índice (X,Y,Z) de diferentes volúmenes, ya que la configuración de bucles es la misma, evitando de este modo la divergencia de *threads*.

Una vez calculado el volumen de correlación, se procede a escribirlo en memoria global. En la Figura 4.21 se muestra un bloque de *threads* escribiendo la fila X correspondiente a los índices Z=0 e Y=0 de 32 volúmenes diferentes. El resto de filas del volumen (Z=[1-15] e Y=[1-31]) se calculan del mismo modo extendiendo el *grid* en las dimensiones Z e Y con unos tamaños de 16 y 32 respectivamente. El proceso de escritura se realiza en dos pasos. La parte superior de la Figura 4.21 muestra el primer paso, donde el bloque escribe los datos correspondientes a los 16 primeros volúmenes. El resto de los valores son escritos en la segunda iteración, como se muestra en la parte inferior de la Figura 4.21. Los volúmenes de correlación para el resto de puntos del sitio de unión se calculan ampliando la cantidad de bloques a través de la extensión de la dimensión X del *grid*.

4.4.1.3. *Kernel* para la búsqueda de parámetros de ajuste

Los volúmenes de correlación generados a partir de la trasformada inversa de Fourier proporcionan las puntuaciones de *docking* correspondientes a un muestreo rotacional del ligando de 11,25º (360º/2B). Estos nuevos volúmenes tienen un tamaño de 32 x 32 x 32 valores flotantes. Los volúmenes se dividen virtualmente en 32 láminas de 32 x 32 valores. De estas 32 láminas, sólo 17 se leen ya que las otras 15 contienen valores repetidos por simetría. Se ha fijado un tamaño de bloque de *threads* de 32 x 32, cubriendo el espacio de una lámina del volumen. Cada bloque de *threads* explora su volumen lámina a lámina en busca de máximos tal como se ilustra en la Figura 4.22. El valor de cada máximo encontrado es interpolado con los vóxeles vecinos para obtener así un valor más preciso.

El último paso de la búsqueda se encuentra ilustrado en la parte inferior de la Figura 4.22. Una vez que se ha obtenido una lámina con los mejores máximos del volumen, se realiza una reducción paralela sobre esta. Al final se obtiene por cada bloque el máximo absoluto que corresponde con la mejor orientación del ligando evaluado en su correspondiente punto traslacional. Finalmente, las soluciones se transfieren a memoria principal y son ordenadas por la CPU.

4.4.2. Adaptación sobre sistemas de memoria distribuida

La adaptación sobre sistemas de memoria distribuida se ha realizado con MPI siguiendo un modelo de comunicación maestro/esclavo similar a la empleada para FBM. El proceso maestro se encarga de distribuir los ligandos entre los procesos esclavos y de recoger posteriormente los resultados. Los procesos esclavos se encargan exclusivamente de realizar el proceso de *docking*. El maestro envía a los esclavos el identificador del próximo ligando que deben ajustar mientras

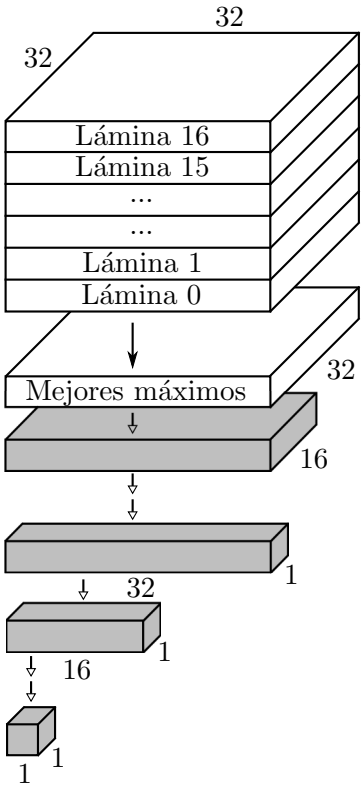


Figura 4.22: Búsqueda en el volumen de correlación. En primer lugar se escanean las láminas del volumen de una en una para obtener una única lámina de 32 x 32 con los mejores máximos. Luego se realiza una reducción paralela sobre la lámina resultante para obtener el mejor valor.

están ya computando el ajuste del actual. De este modo se evita que los procesos esclavos se detengan tras terminar un ajuste mientras esperan a que se les asigne un nuevo ligando. Al final, el proceso maestro ordena los resultados y los ligandos son clasificados mediante la puntuación obtenida tras la revaluación. Las mejores soluciones para cada ligando, junto con su posición traslacional y rotacional son almacenadas en disco.

Se ha definido un protocolo para la comunicación entre los procesos MPI de FRODRUG. Existen tres tipos diferentes de mensajes. Los dos primeros son similares a los utilizados en FBM. Se definen del siguiente modo:

- Mensajes de solicitud de ligando:

  Estos mensajes son enviados sin contenido por los procesos esclavos al proceso maestro. Este responde al esclavo con una variable que contiene el identificador único del próximo ligando que debe procesar, o un valor negativo si no quedan ligandos por procesar.

- Mensajes envío de ajustes:

  Esta clase de mensajes son enviados por los procesos esclavos al maestro tras computar el *docking* para un ligando. En primer lugar el proceso esclavo envía el identificador único del ligando que ha procesado. A continuación envía un paquete de datos con las 50 mejores soluciones para ese ligando.

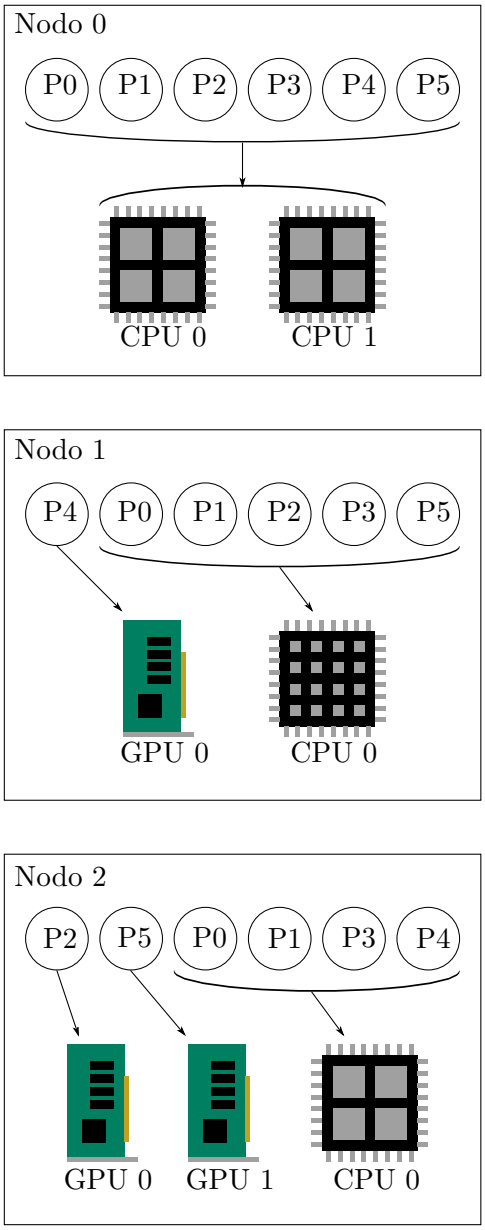


Figura 4.23: Diagrama de ejemplo de distribución de procesos de FRODRUG en sistemas de memoria distribuida mediante MPI. En este ejemplo se lanzan 18 procesos repartidos entre tres nodos. El sistema de gestión de carga del sistema distribuido asigna seis procesos a cada nodo. El nodo 0 cuenta con ocho núcleos distribuidos entre dos procesadores. Los seis procesos del nodo 0 son distribuidos por el sistema operativo entre los núcleos disponibles. En el nodo 1 existe un dispositivo CUDA y un procesador de 16 núcleos. La GPU es utilizada por el proceso 4, que ha sido el primero en enviar el mensaje de información *hardware* al proceso maestro. El resto de procesos emplean núcleos de CPU. En el nodo 2 existen dos dispositivos CUDA y un procesador de cuatro núcleos. Las GPUs serán utilizadas por los procesos 2 y 5, que han sido los dos primeros en enviar los mensajes de información *hardware*. Los cuatro procesos restantes de este nodo emplearán núcleos de CPU.

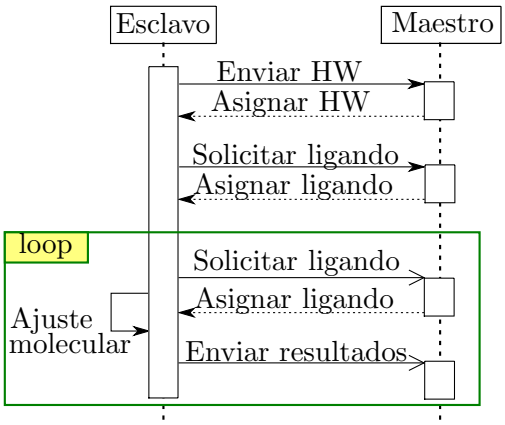


Figura 4.24: Secuencia comunicaciones entre procesos para FRODRUG mediante MPI.

Se han extendido un poco más la adaptación MPI con respecto a la versión utilizada en FBM, introduciendo una importante mejora que permite un notable incremento en el rendimiento. Esta optimización consiste en la inclusión de un sistema de gestión de dispositivos CUDA. Un nodo individual de un clúster puede estar compuesto por varias CPUs y varios dispositivos CUDA. A través de esta mejora, los procesos esclavos que hayan sido lanzados en nodos que tengan tarjetas gráficas de NVIDIA podrán hacer uso de ellas a través de la adaptación paralela del método. Una tarjeta gráfica sólo podrá ser utilizada por un único proceso. Los procesos excedentes del nodo, para los que no queden tarjetas gráficas disponibles, ejecutarán la versión secuencial del método como se hacía en FBM. El *hardware* que cada proceso esclavo empleará para el ajuste molecular será indicado por el proceso maestro mediante un nuevo tipo de mensaje: mensaje de información *hardware*. Cuando un proceso esclavo es lanzado en un nodo, recopila información sobre la cantidad de dispositivos CUDA que ese nodo posee. A continuación envía dos mensajes de información *hardware* al proceso maestro. Uno de ellos contiene un nombre identificador único para el nodo, y el otro contiene la cantidad de dispositivos CUDA disponibles en dicho nodo. El proceso maestro recibe la información y a continuación puede realizar dos acciones:

- Si el proceso maestro no tiene información sobre ese nodo, almacena el nombre y su cantidad de dispositivos CUDA. Si hay al menos una GPU disponible, le indica al proceso esclavo que la utilice. En caso contrario se le indica que emplee un núcleo de CPU.

- Si el proceso maestro ya tiene información sobre ese nodo, comprueba si queda disponible en él al menos un dispositivo CUDA. En caso afirmativo se le indica al proceso esclavo que haga uso de él. En otro caso se le indica que utilice un núcleo de CPU.

La asignación de *hardware* entre los distintos procesos dentro un mismo nodo depende exclusivamente del orden de llegada de sus mensajes de información *hardware* al proceso maestro. El modo en que se distribuyen los procesos sobre los diferentes nodos de un clúster depende del sistema de gestión de recursos que se esté ejecutando en dicho clúster. Normalmente, ejecuciones consecutivas de FRODRUG generan diferentes patrones de distribución de *hardware* sobre el clúster. Un ejemplo de asignación de los recursos disponibles puede observarse en la Figura 4.23.

Teniendo en cuenta el nuevo tipo de mensaje de información *hardware* agregado en esta adaptación MPI, se puede observar un ejemplo de la secuencia de comunicación de FRODRUG en la Figura 4.24.

Capítulo 5

Resultados y análisis de rendimiento de los métodos de ajuste

En este capítulo se presentan las diferentes pruebas de validación sobre *Fast Bessel Matching* y FRODRUG para determinar su rendimiento y precisión. También se han hecho comparativas con otros procedimientos presentes en la literatura para determinar la mejora que los métodos presentados en esta tesis doctoral proporcionan.

En primer lugar se muestran los resultados obtenidos para FBM, seguidos por los registrados para FRODRUG.

5.1. Validación de *Fast Bessel Matching*

La validación de FBM se ha llevado a cabo empleando imágenes simuladas de microscopía electrónica. En primer lugar se detalla el procedimiento que se ha empleado para generar estas imágenes. A continuación se procede a mostrar los resultados obtenidos en el proceso de validación del método.

La prueba de validación consiste en un test ciego que enfrenta un conjunto de 76.000 imágenes simuladas contra 80 de referencia. La Figura 5.1 muestra este esquema de ejecución. Las 76.000 imágenes se generan rotando y trasladando las de referencia y añadiendo ruido para simular imágenes experimentales como las que se pueden obtener en condiciones reales. En total se realizan más de seis millones de alineamientos 2D. A continuación se contrastan los datos de rotación y traslación correctos con los obtenidos en los alineamientos de FBM para cada una de las imágenes simuladas. El resultado deseable es que el método de ajuste identifique correctamente para cada imagen simulada la imagen de referencia a partir de la cual se ha generado, así como la rotación y traslación que le ha sido aplicada.

Durante todas las pruebas realizadas el error en el ajuste entre dos imágenes alineadas se mide en píxeles como:

$$e = d\left|\sin\left(\frac{\Delta\phi}{2}\right)\right| + \sqrt{\Delta x^2 + \Delta y^2} \tag{5.1}$$

donde d es el diámetro de la partícula en píxeles, $\Delta\phi$ es la diferencia en grados entre la rotación

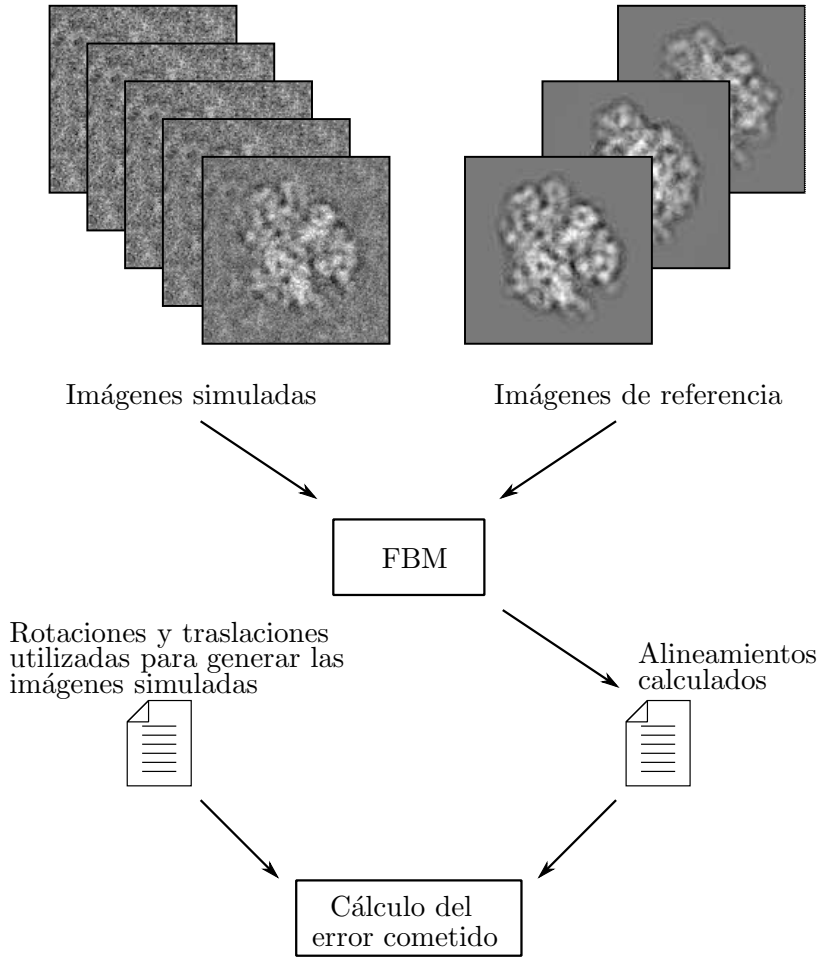


Figura 5.1: Esquema de ejecución en la validación de FBM. El método toma como entrada un conjunto de 76.000 imágenes simuladas y 80 de referencia. A continuación se contrastan los alineamientos obtenidos con la rotación y traslación correctas para cada imagen simulada.

obtenida en el ajuste y la correcta. La diferencia entre los parámetros traslacionales obtenidos y los correctos se describe a través de Δx y Δy. Esta ecuación para calcular el error cometido ha sido formulada y empleada con anterioridad en [100]. El primer término corresponde al error radial, y el segundo al error traslacional entre las imágenes alineada y de referencia. El diámetro de la partícula es difícil de determinar de forma precisa debido a las distorsiones introducidas. Por este motivo, el valor de d simplemente se ha fijado a 2/3 del tamaño del lado de las imágenes. El error máximo que se admite para considerar útil un ajuste es una cantidad inferior al 5 % del tamaño del lado de imagen utilizado.

5.1.1. Generación de un conjunto de pruebas de imágenes simuladas de microscopía electrónica para validación del ajuste 2D

Las imágenes simuladas se han generado a partir de proyecciones de la estructura de la RNA polimerasa II. Esta macromolécula, que cataliza la transcripción de ADN, ha sido caracterizada por microscopía electrónica en varios estados conformacionales [156]. Las imágenes simuladas han sido obtenidas a partir de su estructura atómica contenida en un fichero PDB (3M3Y.pdb)

usando el software de análisis de partícula individual Xmipp 3.0 [121, 187, 204]. En primer lugar se generan las imágenes de referencia. Este proceso se realiza en tres etapas a través de tres herramientas de Xmipp. El primero consiste en centrar y escalar la estructura atómica de la macromolécula. Esto se realiza con el fin de que, al hacer las proyecciones 2D, la macromolécula aparezca en el centro de la imagen ocupando aproximadamente 2/3 de la misma. El comando ejecutado es:

```
xmipp_phantom_transform -i 3M3Y.pdb -o 3M3YC.pdb
                        --operation scale 0.7 0.7 0.7
                        --center_pdb
```

A partir de la estructura PDB centrada (`3M3YC.pdb`) se realizan las proyecciones 2D de la densidad electrónica de la macromolécula:

```
xmipp_phantom_project -i 3M3YC.pdb
                      -o r.stk
                      --params ref.par
                      --sampling_rate 3
```

El fichero de configuración (`ref.par`) indica la cantidad de proyecciones que deben realizarse, el tamaño de la proyección 2D y los ángulos desde los cuales se debe proyectar. En total se hacen 80 proyecciones para el volumen completo. Se realizan en el eje latitudinal ocho proyecciones desde los 20^{o} hasta los 160^{o}. Esto produce un paso angular de 20^{o}. Por cada ángulo muestreado en este eje, se realizan diez proyecciones a través del eje longitudinal desde los 0^{o} hasta los 342^{o}. En este caso se produce un paso angular de 38^{o}. Se han seleccionado estos pasos angulares de muestreo para evitar simetrías, con el fin de obtener un conjunto de proyecciones que se diferencien lo máximo posible entre ellas.

Por último se utiliza una herramienta que permite añadir desenfoque, astigmatismo, aberración y la CTF (*Contrast Transfer Function*) de la lente a estas 80 proyecciones ideales con el fin de hacerlas más parecidas a las obtenidas por un microscopio electrónico real. El comando que aplica estas distorsiones es:

```
xmipp_phantom_simulate_microscope -i r.stk
                                  -o ref.spi
                                  --ctf ctfdesc.txt
                                  --noNoise
```

Este comando toma como entrada el fichero de 80 proyecciones en formato SPIDER (`r.stk`) y escribe como salida otro fichero (`r.spi`) también en formato SPIDER con la CTF aplicada. El comando toma como entrada un fichero de configuración (`ctfdesc.txt`) que contiene los parámetros que modifican la función de transferencia de contraste. El contenido de este fichero de configuración se puede ver en el apéndice B. En la Figura 5.2 se pueden ver algunas de las proyecciones realizadas a la macromolécula tras el proceso descrito.

Una vez obtenidas las imágenes de referencia se generan las simuladas, las cuales van a ser alineadas con las de referencia. Para ello, primero se rotan y trasladan las de referencia y posteriormente se les añade ruido. Las proyecciones de las imágenes simuladas se generan a partir de este comando:

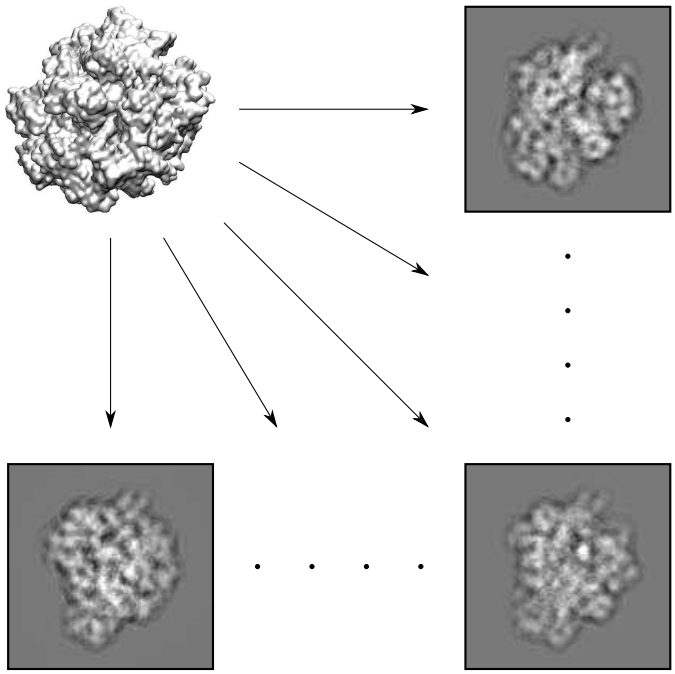


Figura 5.2: Mapa de microscopía electrónica de RNA polimerasa II (superficie gris) y algunas de las proyecciones 2D de su densidad electrónica empleadas como imágenes patrón.

```
xmipp_phantom_project -i 3M3YC.pdb
                      -o rand.stk
                      --params rand.par
                      --sampling_rate 3
```

Este comando es el mismo que se emplea en el segundo paso de las imágenes de referencia, pero se introduce un fichero diferente para indicar la configuración de las proyecciones. La configuración de los ángulos para las proyecciones es igual a la proporcionada en el fichero de configuración de las imágenes de referencia excepto que por cada una de las 80 proyecciones 2D realizadas se generan 50 copias con diferentes rotaciones y traslaciones. La rotación aplicada varía entre 7,2º y 352,8º, con un paso angular de 7,2º. Esto produce 49 rotaciones para cada proyección. A continuación se aplica una traslación aleatoria en ambos ejes a cada una de las 50 imágenes rotadas. La cantidad de píxeles que se desplaza la imagen desde el centro en cada uno de sus ejes se calcula como un ruido gaussiano con una desviación estándar de 1,5 y media 0. Esto produce desplazamientos de hasta seis píxeles como máximo desde el centro de la imagen en cualquier dirección. Teniendo en cuenta todas las proyecciones realizadas con sus rotaciones y traslaciones, en total se genera un conjunto de 4.000 imágenes.

El siguiente y último paso consiste en agregar la CTF y la cantidad de ruido deseado al conjunto de 4.000 imágenes. Para ello se emplea el siguiente comando:

```
xmipp_phantom_simulate_microscope -i rand.stk
                                  -o exp#.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR #
                                  --after_ctf_noise --defocus_change 50
```

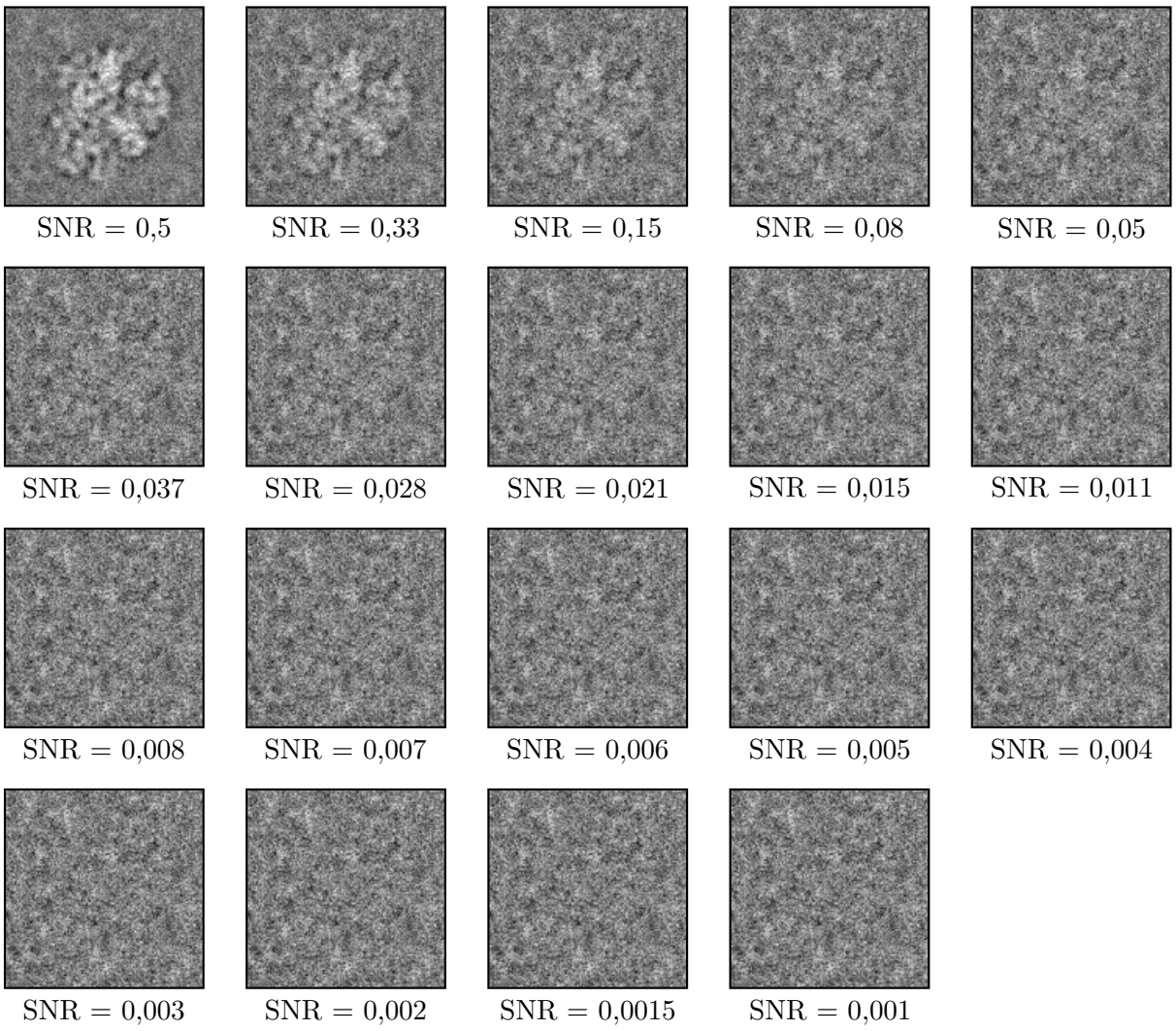


Figura 5.3: Ejemplo ilustrativo de los 19 diferentes niveles de ruido añadidos a proyecciones aleatorias y sus SNR estimados.

La almohadilla del fichero de salida (`exp#.spi`) se sustituye manualmente por un número entero que servirá para identificar posteriormente la cantidad de ruido aplicado. Esto se hace para generar y almacenar simultáneamente conjuntos de imágenes con diferentes niveles de ruido. Mediante el parámetro `---targetSNR` se indica el SNR (*Signal Noise Ratio*) que se desea que las imágenes producidas tengan. Con este comando se agrega ruido a la imagen de tal manera que el SNR se acerque lo más posible al solicitado. El SNR se define como $\sigma^2_{signal}/\sigma^2_{noise}$. Para realizar la validación y las pruebas de rendimiento, se generan conjuntos de imágenes con 19 diferentes niveles de SNR producidos a partir de distintas cantidades de ruido, desde 0,5 hasta 0,001. Estos 19 ficheros generados componen el conjunto de pruebas de validación. En la Tabla 5.1 se puede consultar el SNR estimado para las imágenes de cada uno de los ficheros generados. Un ejemplo de estas imágenes con diferentes niveles de SNR se puede ver en la Figura 5.3.

El *script* completo que realiza la generación de este conjunto de imágenes de microscopía electrónica, así como los ficheros de configuración utilizados, se encuentran en el apéndice B.

FICHERO	SNR
exp1.spi	0,5
exp2.spi	0,25
exp3.spi	0,13
exp4.spi	0,074
exp5.spi	0,048
exp6.spi	0,037
exp7.spi	0,028
exp8.spi	0,021
exp9.spi	0,015
exp10.spi	0,011
exp11.spi	0,008
exp12.spi	0,007
exp13.spi	0,006
exp14.spi	0,005
exp15.spi	0,004
exp16.spi	0,003
exp17.spi	0,002
exp18.spi	0,0015
exp19.spi	0,001

Tabla 5.1: Valores de SNR estimados para cada uno de los 19 ficheros de imágenes experimentales generados.

5.1.2. Análisis de ρ_{max} y B

Las pruebas de validación se han realizado recorriendo el rango de dos parámetros: el ancho de banda (denominado como B ó *bandwidth*) y ρ_{max}. El ancho de banda determina el muestreo rotacional de las imágenes mientras que ρ_{max} indica la distancia máxima en píxeles entre los centros de las mismas. Para el ancho de banda se han utilizado los valores de 64 y 128. Estos se corresponden con realizar un muestreo rotacional de 2,8º y 1,4º respectivamente. El ancho de banda interviene directamente en el muestreo de la imagen original. La cantidad promedio de píxeles por punto muestreado se calcula como:

$$p_s = \frac{\pi \cdot A}{2 \cdot B} \tag{5.2}$$

siendo A el tamaño de la mitad del lado de la imagen y B el ancho de banda seleccionado. Para esta prueba en concreto se emplean imágenes de 128 x 128 píxeles; la variable A toma el valor 64. Empleando el ancho de banda de 128 se obtiene un $p_s = 0,78$ mientras que para $B = 64$ se tiene un $p_s = 1,57$. Estos valores indican un promedio de píxeles por punto muestreados. La cantidad de píxeles utilizados es mayor a la media en la zona central de la imagen y menor en las zonas exteriores. Los puntos finales empleados en el muestreo de las imágenes para ambos anchos de banda se pueden observar en la Figura 5.4. Seleccionar un ancho de banda de 64 produce un

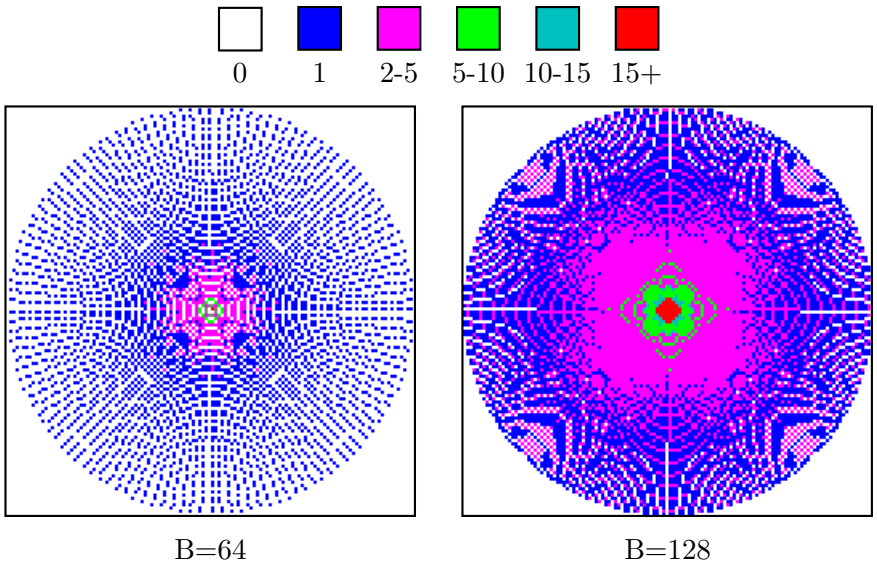


Figura 5.4: Puntos de una imagen genérica de 128 x 128 píxeles que se emplean para su muestreo en los anchos de banda de 64 y 128. Cada uno de los píxeles tiene un color que indica la cantidad de veces que ha sido empleado para realizar el muestreo.

	B = 128	B = 64
ρ_{max} = 7	256 puntos	64 puntos
ρ_{max} = 12	784 puntos	196 puntos
ρ_{max} = 16	1.296 puntos	324 puntos

Tabla 5.2: Cantidad de puntos traslacionales para las diferentes configuraciones de ρ_{max} y B mostradas en la Figura 5.5.

ligero submuestreo de la imagen. Aproximadamente la mitad de sus píxeles no se han utilizado. Un muestreo más equilibrado se consigue con un ancho de banda de 128 donde casi todos los píxeles de la imagen se han leído.

Los valores empleados para ρ_{max} en las pruebas de validación han sido 7, 12 y 16. El valor de 7 ha sido elegido porque se conoce que la máxima traslación que se ha aplicado a las imágenes de prueba es de 6 píxeles. El valor más grande de ρ_{max}, 16, es seleccionado por cubrir algo más del doble del rango mínimo de escaneo. Entre estos dos valores se ha seleccionado un punto intermedio en ρ_{max} = 12.

Debido al diseño del método, tanto el ancho de banda como el valor de ρ_{max} modifican la cantidad y posición de los puntos traslacionales que pueden ser empleados para trasladar las imágenes. Los diferentes patrones generados para las posibles combinaciones de ρ_{max} y B se pueden ver en la Figura 5.5. Por último, en la Tabla 5.2 se puede ver la cantidad de puntos traslacionales generados para cada combinación de variables.

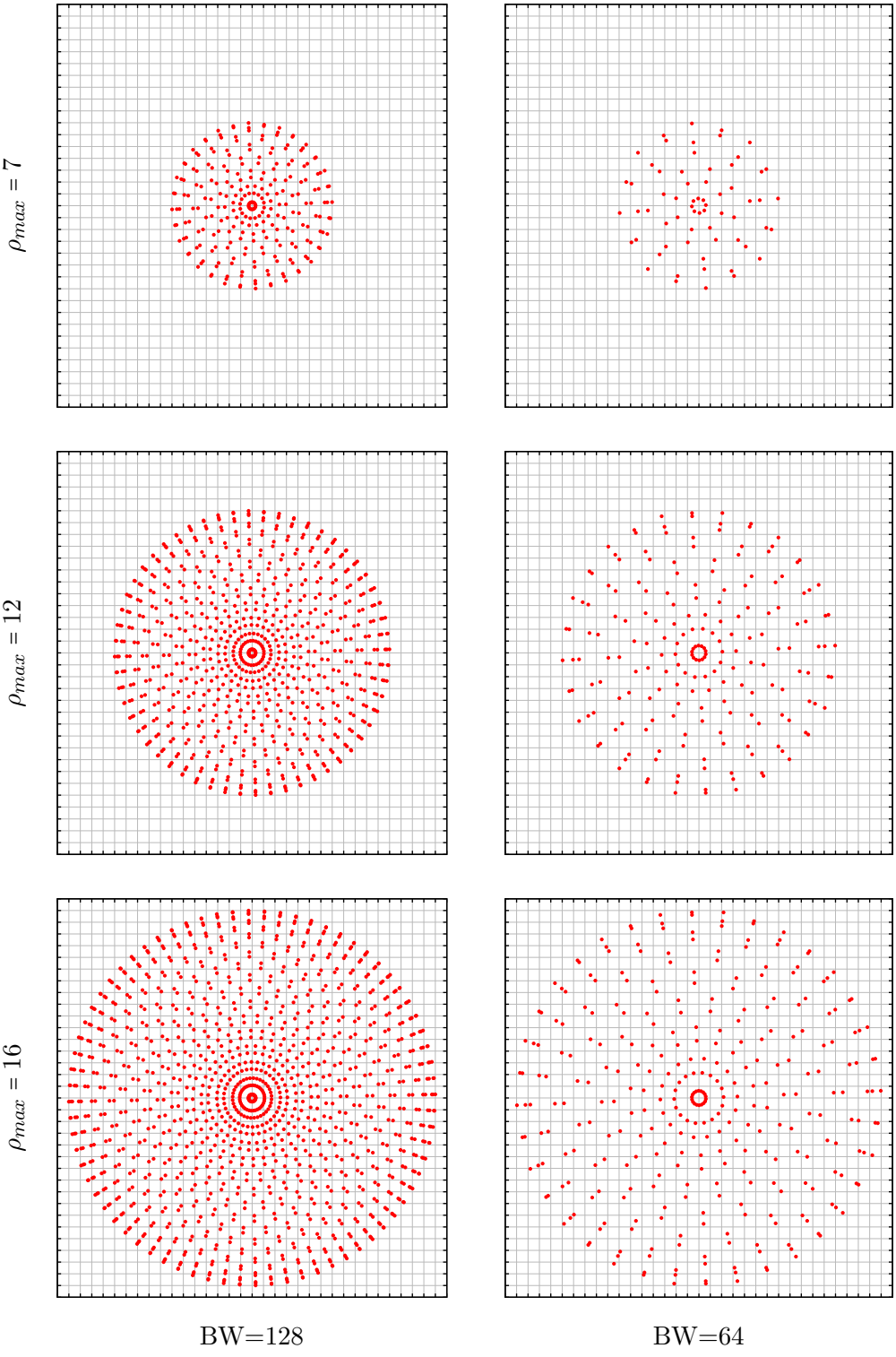


Figura 5.5: Patrones de puntos traslacionales para diferentes ρ_{max} y anchos de banda. El tamaño de imagen empleado es 128 x 128 píxeles. Cada cuadrado muestra una región de la imagen con un área de 34 x 34 píxeles. El punto central de cada uno de los cuadrados se corresponde con la coordenada [64,64]. ρ_{max} toma los valores 7, 12 y 16. Se emplean los anchos de banda 128 y 64. La rejilla de cada uno de los cuadrados muestra los píxeles de la imagen.

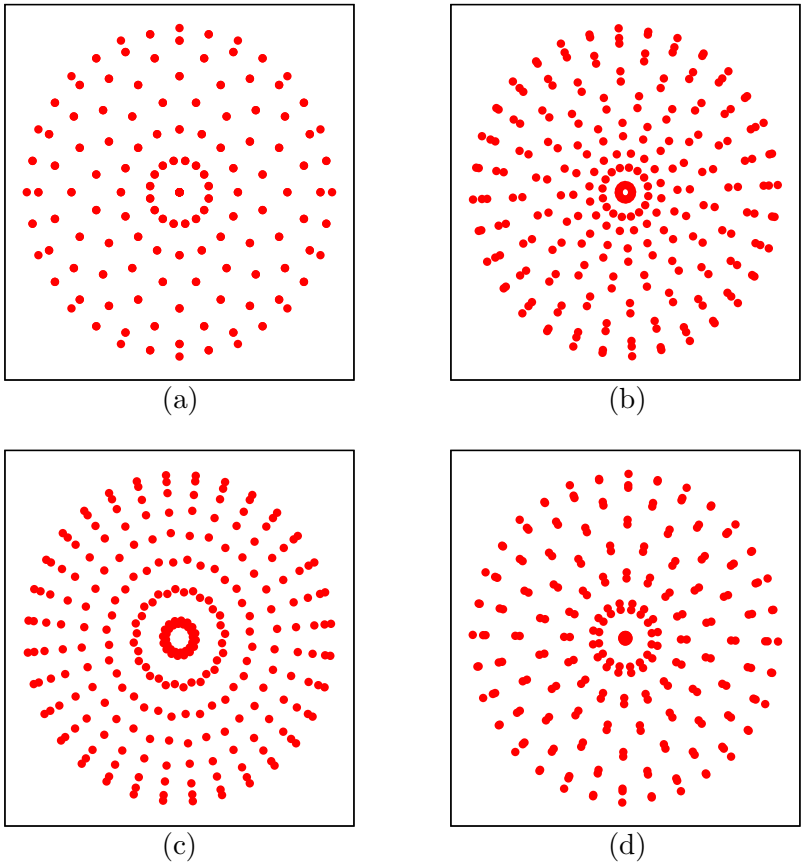


Figura 5.6: Diferentes patrones de muestreo obtenidos al emplear diversos valores para el ángulo ϵ. (a) No se aplica ϵ, ilustrando el patrón de muestreo traslacional original. (b) Patrón obtenido utilizando ϵ = 1/8 del paso angular de η. (c) Patrón obtenido utilizando ϵ = 1/4 del paso angular de η. (d) Patrón obtenido utilizando ϵ = 1/2 del paso angular de η. Se ha utilizado un tamaño de imagen de 128 x 128, un ancho de banda de 128 y un ρ_{max} = 7.

5.1.3. Análisis de ϵ

El ángulo ϵ desplaza ligeramente al ángulo η con el fin de evitar obtener soluciones traslacionales duplicadas, según se indica en la sección 4.1.2.2. El valor final de este ángulo se ha determinado experimentalmente tras realizar varias pruebas con diversos valores y observar los diferentes patrones de muestreo generados.

El patrón de muestreo traslacional varía en relación a varios parámetros. Estos son: tamaño de la imagen, ancho de banda y ρ_{max}. Como se pudo ver anteriormente en la Figura 5.4, el patrón de muestreo más homogéneo se obtiene haciendo que el ancho de banda sea igual al tamaño de una de las dimensiones de la imagen. Para analizar los diferentes patrones de muestreo traslacionales se han fijado las dimensiones de la imagen a 128, al igual que el ancho de banda. El valor de ρ_{max} empleado es de 7 píxeles. Con estos valores se genera un círculo cuyo centro está situado en el punto (64,64). Este círculo engloba un total de 172 píxeles.

Los patrones de muestreo obtenidos para los valores más significativos de ϵ con esta configuración de variables se muestran en la Figura 5.6. Estos valores se corresponden con no

	PUNTOS TRASLACIONALES	ÁREA CUBIERTA
$\epsilon = 0$	108	62,79 %
$\epsilon = 1/8$ de η	138	80,23 %
$\epsilon = 1/4$ de η	133	77,32 %
$\epsilon = 1/2$ de η	122	70,93 %

Tabla 5.3: Número de puntos traslacionales distintos en cada una de las configuraciones de ϵ probadas. Se muestra también el área del círculo cubierta en porcentaje. Como referencia se toma un círculo de radio 7 píxeles, con un área de 172 píxeles.

aplicar ϵ (Figura 5.6.a), y con aplicar ϵ con un valor igual a 1/8 (Figura 5.6.b), 1/4 (Figura 5.6.c) y 1/2 (Figura 5.6.d) del paso angular de η, cuyo valor se calcula como $2\pi/k$. Los dos patrones que tienen los puntos traslacionales mejor distribuidos son los obtenidos con ϵ = 1/8 y ϵ = 1/2. De estos dos patrones, el primero es el que cubre de una forma más homogénea la zona central de la imagen. Así mismo puede observarse en la Tabla 5.3 que el número de puntos traslacionales distintos en el círculo es mayor para ϵ = 1/8. Por estas razones se ha seleccionado finalmente el valor de 1/8 para ϵ.

5.1.4. Validación con imágenes simuladas de microscopía electrónica

A continuación se detallan las pruebas realizadas con el conjunto de imágenes simuladas generadas en la sección 5.1.1. Los resultados obtenidos para el test ciego utilizando distintos valores de SNR y fijando el ancho de banda a 128 se pueden observar en la Figura 5.7. Como se puede ver, para algunos valores de SNR la variación entre errores obtenidos es elevada. Esta diferencia se debe a que cuando una imagen se empareja con su correspondiente referencia el error en el alineamiento suele ser bajo, pero cuando se empareja con la referencia errónea el error suele ser alto. Esto termina traduciéndose en desviaciones muy grandes. Por motivos de claridad en las figuras se presenta sólo el valor de error promedio y se omiten las desviaciones.

La Figura 5.8 muestra el promedio de error que se ha obtenido para el test ciego al alinear las imágenes simuladas a través de los diferentes valores de ρ_{max} y ancho de banda 128. Cabe destacar que para ρ_{max} = 12 y ρ_{max} = 16 la línea de error prácticamente se superpone. Esto indica que aumentar por encima de 12 píxeles el rango de exploración para este tamaño de imagen (128 x 128) y ancho de banda (128) no aumenta la precisión del ajuste. Un valor de ρ_{max} = 7 produce una línea con un error ligeramente más alto que la de ρ_{max} = 12. El método alinea correctamente con un error inferior a 1 pixel las imágenes con un SNR de 0,03 o superior para todos los valores de ρ_{max}. Las imágenes con SNR comprendido entre 0,02 y 0,03 se alinean con un error de 4 píxeles o menos para todos los ρ_{max}. Esta cantidad de error equivale a menos del 3 % del tamaño de imagen, por lo que estos alineamientos se consideran correctos. A partir de un SNR inferior a 0,021 el error aumenta rápidamente desde los 10 píxeles hasta un máximo de 66. Esto indica que la cantidad de ruido en las imágenes es tan elevado que los ajustes calculados son aleatorios. La diferencia de error obtenido entre ρ_{max} = 7 y ρ_{max} = 12 para el rango de SNR desde 0,028 hasta 0,5 es muy pequeña y siempre inferior a 0,2 píxeles. Esta diferencia aumenta a 1 pixel para las imágenes con SNR = 0,021. A medida que el ruido de las imágenes se intensifica,

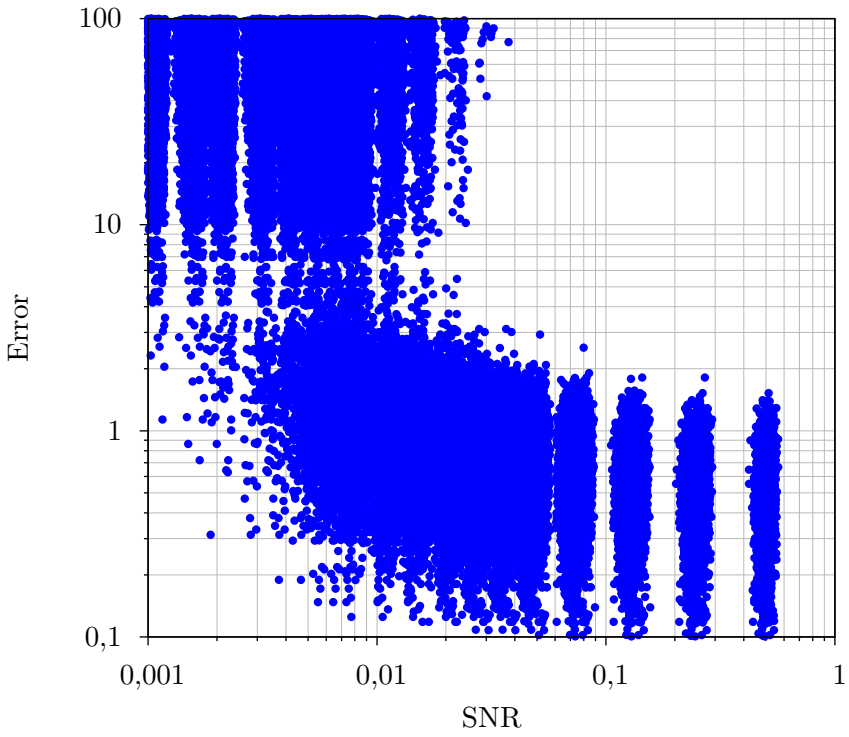


Figura 5.7: Error cometido en el ajuste para cada una de las 76.000 imágenes del conjunto de pruebas simuladas a través de los diferentes niveles de SNR para un ancho de banda de 128. El valor de ρ_{max} es 12.

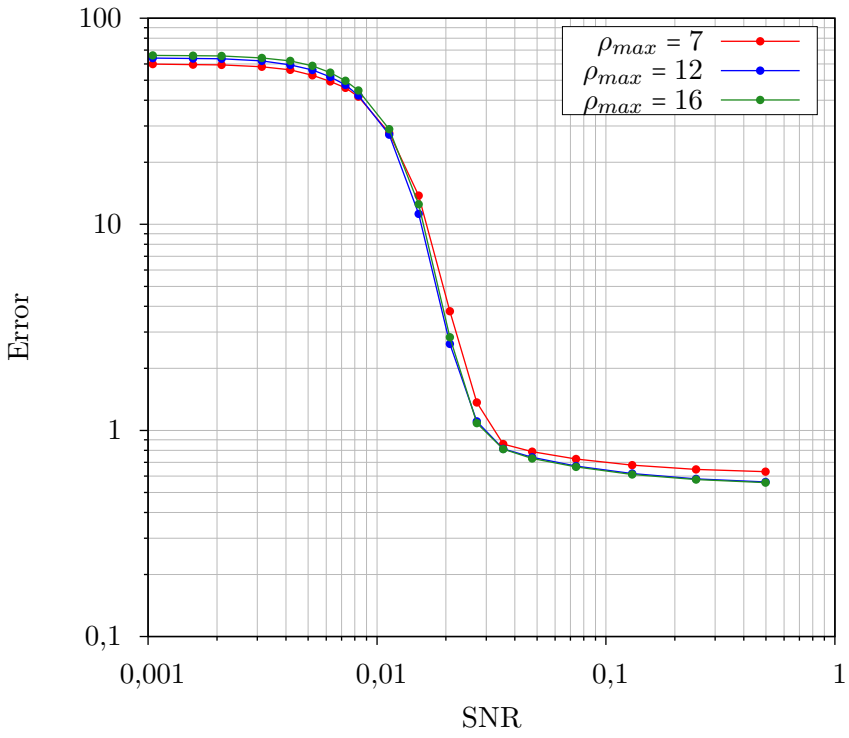


Figura 5.8: Promedio de error cometido en el ajuste del conjunto de pruebas de imágenes simuladas a través de los diferentes niveles de SNR para un ancho de banda de 128. La gráfica muestra el error para los valores de ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16.

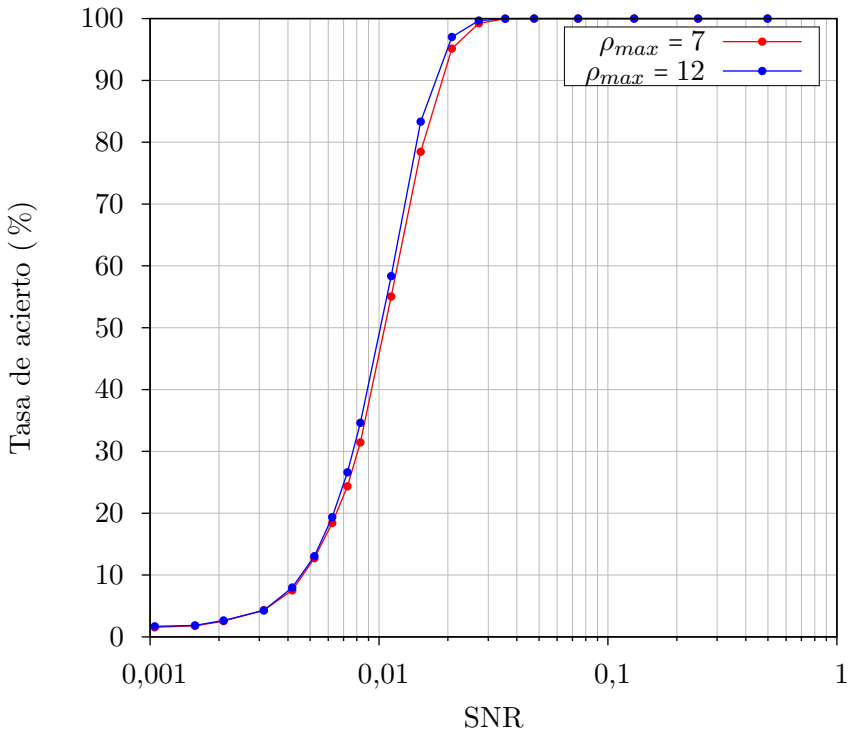


Figura 5.9: Tasa de emparejamientos con la referencia correcta para los valores de ρ_{max} = 7 y ρ_{max} = 12 utilizando el ancho de banda 128.

la diferencia de error aumenta lentamente hasta un máximo de 4 píxeles para el SNR = 0,001.

A continuación se analiza el porcentaje de alineamientos en los que se empareja correctamente una imagen simulada con su correspondiente imagen de referencia. Esta tasa de acierto es un indicador de la fiabilidad del método, sin embargo no existe un umbral fijo a partir del cual se pueda certificar que un método es fiable o no. A efectos de validación se considera que el método es fiable cuando su tasa de acierto supera el 85 % de imágenes correctamente emparejadas. En la Figura 5.9 se muestra la tasa de acierto para ρ_{max} = 7 y ρ_{max} = 12. El valor de ρ_{max} = 16 se omite ya que produce los mismos resultados que ρ_{max} = 12. La tasa de aciertos es del 100 % para ambos ρ_{max} hasta un SNR de 0,035. A partir de este SNR, la tasa de acierto desciende ligeramente hasta el 95 % para ρ_{max} = 7 y hasta el 97 % para ρ_{max} = 12. Las imágenes con un SNR de 0,015 se emparejan correctamente con su referencia en el 78 % de los casos para ρ_{max} = 7 y en el 83 % de los casos para ρ_{max} = 12. Con estas tasas de acierto, el método de alineamiento ya no se considera fiable. A partir de un SNR de 0,01 la cantidad de imágenes correctamente emparejadas desciende rápidamente, con una tasa de acierto del 58 % para ρ_{max} = 12. Esto indica que el método no es fiable para imágenes con una cantidad de ruido tan elevada.

La Figura 5.10 muestra el error promedio cometido en el ajuste sólo para aquellas imágenes cuya referencia ha sido correctamente identificada. Al eliminar los ajustes de imágenes erróneamente emparejadas se proporciona una visión más concreta de la precisión del método. El rango de la gráfica se ha adecuado para mostrar mejor los ajustes dentro del rango de SNR en el que el método se puede considerar efectivo. Ambos valores de ρ_{max} producen líneas de error muy parecidas con una diferencia de error despreciable entre ellas (menor a 0,1 píxeles).

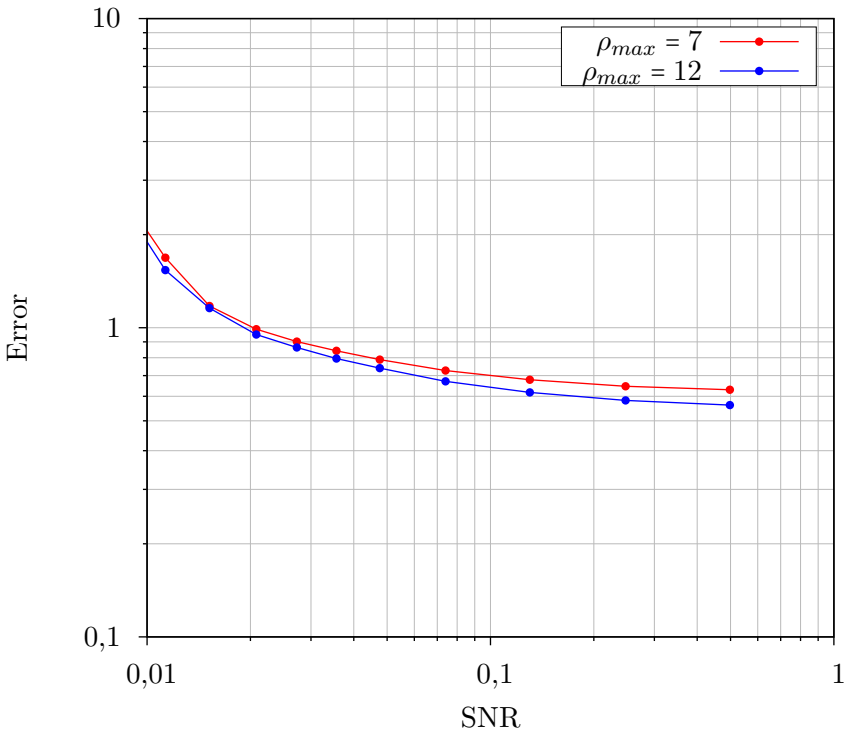


Figura 5.10: Promedio de error cometido en el ajuste sólo para las imágenes experimentales cuya referencia ha sido adecuadamente identificada. Los datos corresponden al ancho de banda 128 y a los valores ρ_{max} = 7 y ρ_{max} = 12.

En prácticamente todo el intervalo muestreado el error promedio es inferior a 1 pixel. El error se acerca a 1 pixel a partir de un SNR de 0,021, donde la tasa de acierto es del 95 % para ρ_{max} = 7 y de 97 % para ρ_{max} = 12. En promedio se obtiene un error de 1,18 píxeles para imágenes con un SNR de 0,015 con ρ_{max} = 7; la tasa de acierto lograda es de 78 %. La línea de ρ_{max} = 12 muestra un error promedio de 1,16 píxeles para el mismo SNR, con una tasa de acierto del 83 %.

A continuación se detallan los resultados obtenidos al variar el ancho de banda a 64. Emplear este ancho de banda supone un ligero submuestreo de las imágenes y una ligera pérdida de resolución rotacional y traslacional. A cambio se obtienen unos tiempos de ejecución menores.

En primer lugar se muestra en la Figura 5.11 el error obtenido para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16. Al igual que con el ancho de banda 128, la línea de ρ_{max} = 16 se superpone con la de ρ_{max} = 12, indicando que no se puede obtener mejor precisión para este rango de exploración con este tamaño de imagen. Para este ancho de banda, todas las líneas están situadas por encima de 1 pixel de error. Esta falta de precisión se debe tanto al elevado paso angular de muestreo (2,8º) como al reducido número de puntos traslacionales que el método puede proporcionar como resultado del ajuste. Las imágenes con SNR de 0,035 o superior se alinean con un error inferior a 2 píxeles. Las imágenes con SNR = 0,028 se ajustan con un promedio de error de entre 2 y 3 píxeles para todos los valores de ρ_{max}. Esta cantidad de error está dentro del margen aceptable siendo inferior al 2 % del tamaño de imagen. Las imágenes con un SNR de 0,021 tienen un error promedio inferior a 6 píxeles empleando ρ_{max} = 12 y ρ_{max} = 16. Sin embargo, el error es superior al umbral marcado de 6,4 píxeles empleando ρ_{max} = 7 no pudiendo considerarse

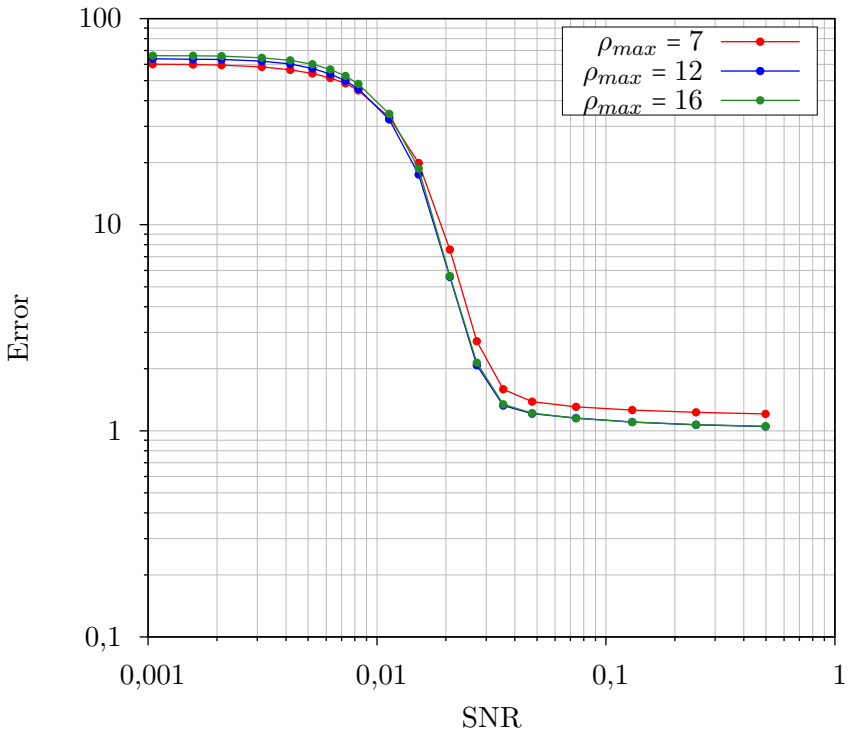


Figura 5.11: Promedio de error cometido en el ajuste del conjunto de pruebas de imágenes de microscopía simuladas a través de diferentes niveles de ruido para un ancho de banda de 64. La gráfica muestra el error para los valores ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16.

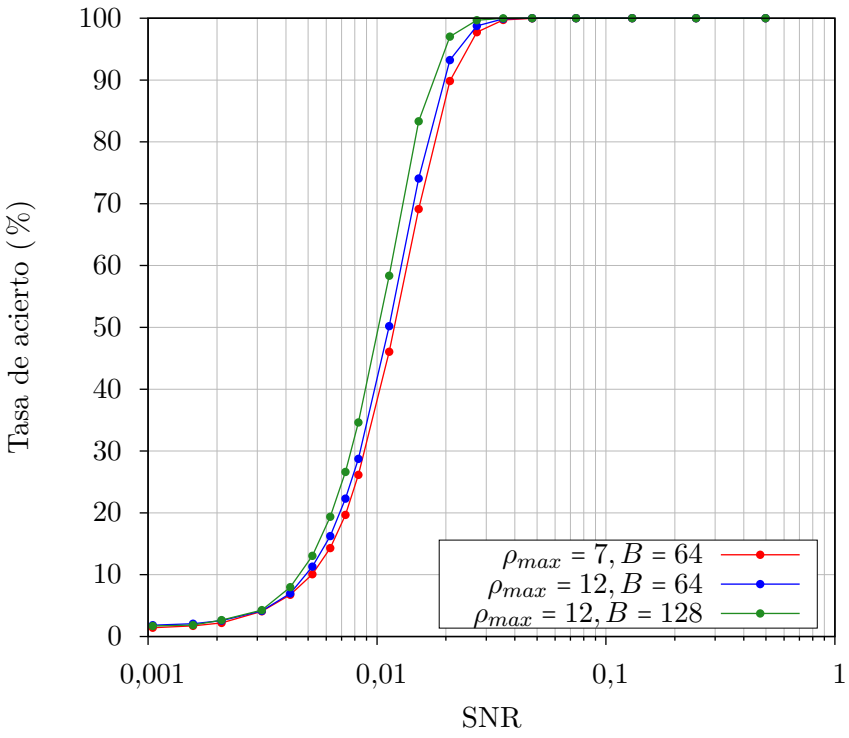


Figura 5.12: Tasa de emparejamientos con la referencia correcta para el ancho de banda de 64 con ρ_{max} = 7 y ρ_{max} = 12. Se muestra también la tasa de acierto para el ancho de banda de 128 con ρ_{max} = 12.

	B = 128	B = 64
ρ_{max} = 7	10,15 ms	0,75 ms
ρ_{max} = 12	32,09 ms	2,29 ms
ρ_{max} = 16	57,11 ms	4,04 ms

Tabla 5.4: Tiempo promedio que lleva realizar un único ajuste entre una pareja de imágenes.

válidos estos alineamientos. Las imágenes alineadas por debajo del SNR = 0,021 producen un error demasiado grande como para que puedan considerarse útiles. Dentro del rango de SNR para el que se obtiene una cantidad de error aceptable (0,028-0,5), emplear ρ_{max} = 12 produce respectivamente un error promedio de entre 0,6 y 0,2 píxeles menos que ρ_{max} = 7.

A continuación se muestra en la Figura 5.12 la tasa de aciertos obtenida para ρ_{max} = 7 y ρ_{max} = 12 en el ancho de banda 64. A modo de comparativa, también se incluye en la gráfica la tasa de aciertos correspondiente a ρ_{max} = 12 y un ancho de banda de 128. La tasa de acierto del ancho de banda 64 es ligeramente inferior a la de 128 debido a un peor muestreo de las imágenes y una precisión angular menor. Con respecto a las dos líneas correspondientes al ancho de banda de 64, la tasa de acierto es del 100 % hasta un SNR de 0,035. La cantidad de fallos se incrementa rápidamente a partir de este punto. El punto de SNR = 0,021 es el último en el que el método puede considerarse fiable, con una tasa de acierto del 90 % para ρ_{max} = 7 y del 93 % para ρ_{max} = 12. El siguiente conjunto de imágenes, con un SNR de 0,015, se ajusta con una tasa de acierto del 69 % y del 74 % para ρ_{max} = 7 y ρ_{max} = 12 respectivamente. Estas tasas de acierto no son lo suficientemente altas como para considerar fiable el resultado. Por otro lado, para este mismo SNR, empleando el ancho de banda 128 se obtiene una tasa de acierto del 83 % con ρ_{max} = 12.

En la Figura 5.13 se muestra el promedio de error obtenido sólo para emparejamientos con la referencia adecuada, empleando un ρ_{max} = 12. Los anchos de banda empleados son 64 y 128. El último SNR para el que el método se considera fiable empleando el ancho de banda de 64 es de 0,021, con una tasa de acierto del 93 %. Como se puede ver, la línea de error comprendida entre los SNR 0,021 y 0,5 nunca supera el nivel de 2 píxeles de error. Como era de esperar, la línea correspondiente al ancho de banda de 128 sigue una trazado similar a 64 pero con un error menor. En este caso, el último SNR fiable es 0,015 con una tasa de acierto del 83 %. Para este punto de SNR, el error promedio obtenido entre ambas líneas es inferior a 1 pixel. En el rango de SNR superiores a 0,015 la diferencia entre ambas líneas es siempre inferior a 0,7 píxeles.

Las diversas configuraciones de ancho de banda y ρ_{max} probadas ofrecen distintos patrones de muestreo y resolución, tanto traslacional como rotacional. La cantidad de puntos calculados al realizar los ajustes de las imágenes tiene un impacto directo en el tiempo que se requiere para calcular dichos ajustes. La Tabla 5.4 muestra el tiempo promedio que se tarda en realizar el emparejamiento entre dos imágenes para cada una de las configuraciones empleadas anteriormente. Estos tiempos han sido registrados empleando la versión secuencial del método sobre un Intel i7-950 a 3,3 GHz.

Dividiendo este tiempo entre la cantidad de puntos traslacionales empleados en cada una de

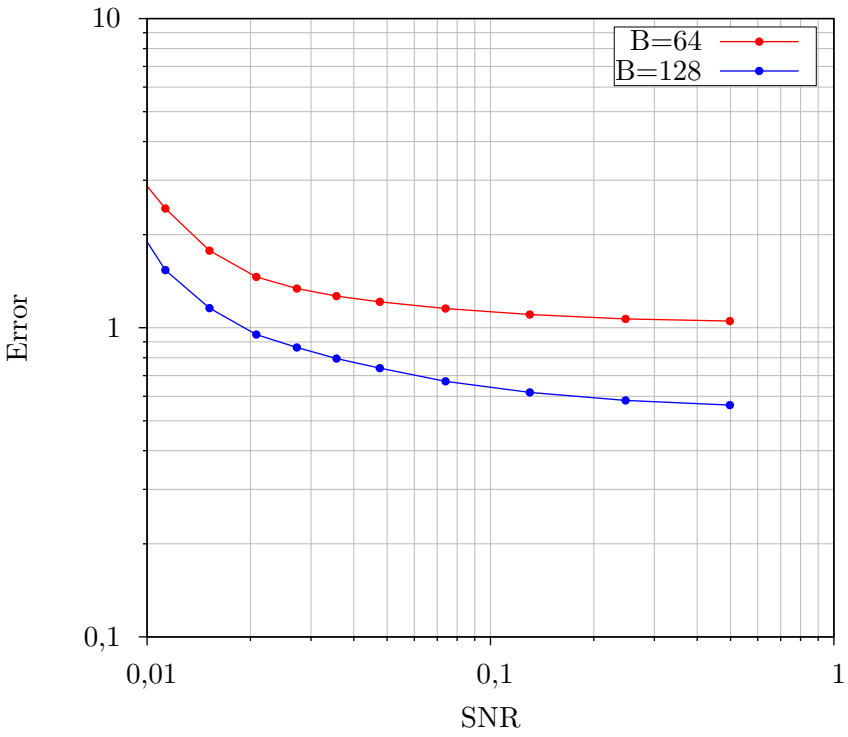


Figura 5.13: Promedio de error obtenido a través del emparejamiento de imágenes experimentales cuya imagen de referencia ha sido identificada de forma correcta para ρ_{max} = 12. Los anchos de banda empleados son 64 y 128.

las configuraciones, para el ancho de banda de 128 se registran unos tiempos de 39,65 μs, 40,93 μs y 44,07 μs por cada punto traslacional para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16 respectivamente. Con respecto al ancho de banda de 64 se obtienen unos tiempos de 11,72 μs, 11,68 μs y 12,47 μs respectivamente. Si se fija el ancho de banda y se hace variar el valor de ρ_{max}, el tiempo requerido por ajuste escala de un modo prácticamente lineal con la cantidad de puntos traslacionales generados.

La conclusión a la que se llega al analizar las pruebas de validación es que FBM cumple adecuadamente la función para la que ha sido diseñado; es capaz de realizar alineamientos correctos de imágenes similares a las obtenidas por los actuales microscopios electrónicos con una reducida cantidad de error. Utilizando el ancho de banda 128 se obtiene un error promedio menor en los alineamientos correctos, de hasta 1 pixel de diferencia, que empleando el ancho de banda 64. Esto se produce por dos motivos: por una parte el muestreo angular es más fino empleando un ancho de banda mayor. Por otro lado se ha demostrado que el muestreo óptimo de los píxeles de una imagen se obtiene fijando el ancho de banda a un valor igual al tamaño del lado de esta. Utilizar un ancho de banda de 128 tiene como contrapartida un tiempo de ejecución 14 veces más lento en comparación a utilizar el ancho de banda 64.

5.2. Análisis de rendimiento de *Fast Bessel Matching*

En esta sección se presentan en primer lugar análisis de rendimiento y precisión realizados sobre FBM en comparación con otros sistemas de cómputo de altas prestaciones. Las pruebas

de rendimiento en estos sistemas se han llevado a cabo empleando el mismo conjunto de imágenes simuladas utilizado en la validación (sección 5.1.1). En segundo lugar se realiza un análisis de los *kernels* desarrollados para la versión paralela de FBM sobre GPU. Por último se compara FBM con una versión optimizada de otro método de ajuste: FRM2D. Como se comentó en la introducción, se ha demostrado que es uno de los métodos más eficaces [29]. En la comparativa con FRM2D se utilizan, además de las imágenes simuladas, varios conjuntos de pruebas compuestos por imágenes experimentales obtenidas a partir de microscopios electrónicos reales.

5.2.1. Comparativa de CPU con diferentes sistemas

En esta sección se evalúa el rendimiento de FBM en sistemas de memoria distribuida (clúster) y tarjetas gráficas (GPUs). Para ello se llevan a cabo las mismas pruebas realizadas en la validación registrando el tiempo de ejecución sobre los diferentes sistemas. En primer lugar se detallan los resultados obtenidos sobre sistemas de memoria distribuida y a continuación los correspondientes en GPU.

5.2.1.1. Comparativa con sistemas de memoria distribuida

Las pruebas detalladas en esta sección se realizan sobre un clúster heterogéneo y dos procesadores multinúcleo. El objetivo de las pruebas en estas CPUs individuales es el análisis del escalado sobre sistemas controlados que no se ven afectados por factores externos. La versión de FBM para sistemas distribuidos utiliza MPI. La librería utilizada para el paso de mensajes es OpenMPI 1.6.2 [157]. La pruebas realizadas consisten en medir el tiempo que se toma para alinear un conjunto de 4.000 imágenes simuladas con las 80 imágenes de referencia utilizando diferente número de procesos MPI. Este número varía en cada sistema probado, por lo que para cada uno se especifica en su correspondiente sección. Para todas las pruebas se ha utilizado un ancho de banda de 128 y unos límites de escaneo traslacional de ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16. En primer lugar se muestran los resultados para el clúster, seguidos por los registrados para los dos procesadores individuales.

El clúster utilizado es el del Instituto de Química-Física Rocasolano del Consejo Superior de Investigaciones Científicas. Este clúster, denominado Ladón, es un sistema heterogéneo compuesto por los procesadores indicados en la Tabla 5.5. Los procesadores E5-2650 cuentan con 20 MB de caché y la extensión del conjunto de instrucciones AVX. Los procesadores X5650, algo más antiguos que los E5-2650, cuentan con 12 MB de memoria caché y la extensión del conjunto de instrucciones SSE4.2. El clúster está gestionado por el sistema de distribución de recursos *Sun Grid Engine*. Mediante este sistema no es posible seleccionar sobre qué procesadores se desea ejecutar el trabajo sino que el gestor distribuye los procesos entre los nodos del clúster según la carga actual.

Las pruebas de rendimiento llevadas a cabo en Ladón se han realizado utilizando 20, 40, 60 y 80 procesos. Los datos de tiempos registrados se muestran en la Figura 5.14. El tiempo medido corresponde al promedio que se necesita para completar un único ajuste entre dos imágenes. La Figura 5.15 muestra las correspondientes aceleraciones tomando como base el

PROCESADOR	NÚCLEOS	FRECUENCIA
Intel Xeon X5650	48	3,06 GHz
Intel Xeon E5-2650	256	2,8 GHz

Tabla 5.5: Procesadores pertenecientes al clúster Ladón. La cantidad indicada es el total de núcleos disponibles para cada tipo de procesador.

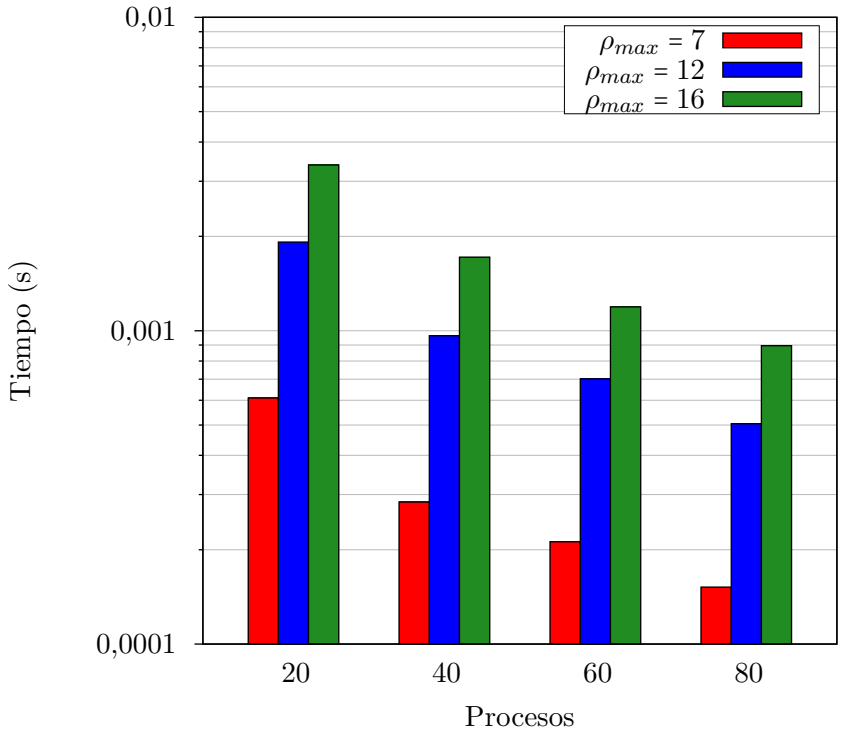


Figura 5.14: Tiempo promedio que lleva realizar el ajuste entre una pareja de imágenes empleando MPI con varias configuraciones de ρ_{max} en el clúster Ladón. Los tiempos se han medido empleando desde 20 hasta 80 procesos para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16. El ancho de banda empleado es de 128.

tiempo correspondiente a 20 procesos. Como se puede observar en la figura, para cualquier valor de ρ_{max}, el método escala adecuadamente con cualquier número de procesos empleados. Las aceleraciones obtenidas con 40 procesos son de 2,15x, 1,99x y 1,97x para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16 respectivamente. Estos valores de aceleración, cercanos a 2, son los esperados ya que se multiplica por 2 el número de procesadores empleados. Cabe destacar que para ρ_{max} = 7 la aceleración sea ligeramente superior a 2. Esto se debe a que el clúster está formado por procesadores de diferentes características que operan con distinto rendimiento. El sistema de gestión *SGE* trata de equilibrar la carga computacional del mejor modo entre los diferentes nodos. Esto se traduce en que cada ejecución de FBM se va a llevar a cabo con una configuración de *hardware* diferente. Lo que está sucediendo en este caso es que en la ejecución con 40 procesos el conjunto de procesadores asignado por *SGE* proporciona globalmente un rendimiento superior al utilizado en la ejecución con 20 procesos. Las aceleraciones empleando 60 procesos son de 2,88x, 2,84x y 2,84x para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16 respectivamente. Estos valores

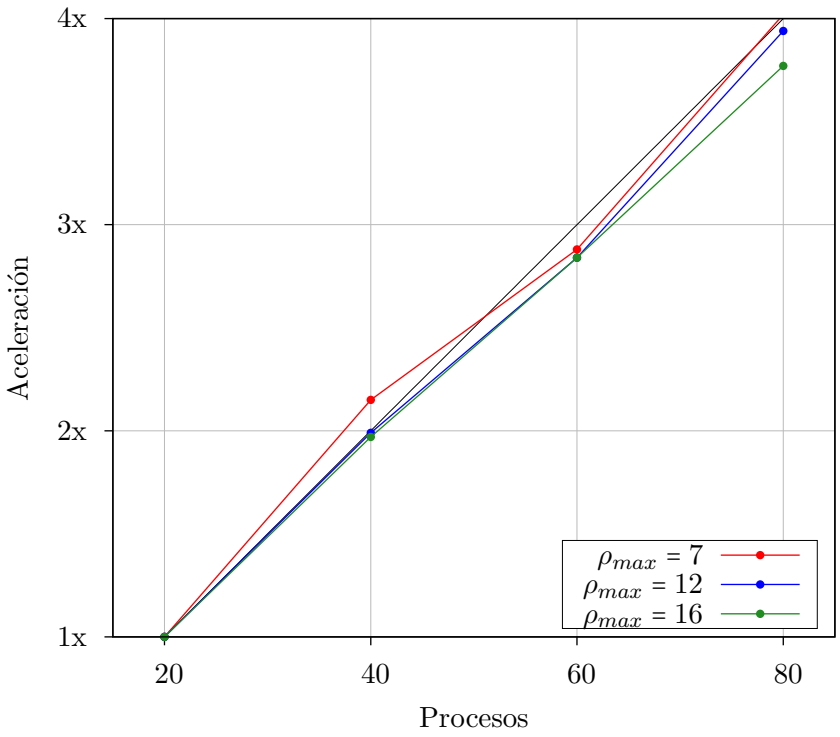


Figura 5.15: Aceleración obtenida en FBM con MPI sobre el clúster Ladón utilizando diferente número de procesos. Se muestra la aceleración empleando ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16. En negro se muestra una aceleración lineal de referencia. El ancho de banda empleado es 128.

PROCESADOR	FRECUENCIA	NÚCLEOS	CACHÉ L3
Intel i7 950	3,3 GHz	4	8 MB
Intel Xeon E5-2650	2,8 GHz	8	20 MB

Tabla 5.6: CPUs individuales empleadas en las pruebas de rendimiento.

coinciden con los esperados al triplicar el número de procesos. Finalmente, las aceleraciones registradas empleando 80 procesos son de 4,02x, 3,94x y 3,77x para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16 respectivamente. Estos valores cercanos a 4 son los esperados al cuadriplicar el número de procesos. Se puede afirmar que FBM escala de un modo lineal en sistemas de memoria distribuida.

A continuación se detallan los resultados obtenidos sobre los dos procesadores multinúcleo. Las características de los mismos se muestran en la Tabla 5.6. El número mínimo de procesos que se muestra en las gráficas para estos procesadores es de dos. El motivo de esto es que la versión para sistemas de memoria distribuida de FBM emplea el modelo de comunicación Maestro/Esclavo para gestionar la distribución de cómputo entre los diferentes recursos disponibles. Es indispensable que al menos se utilicen dos procesos al realizar los ajustes, siendo uno de estos el proceso maestro. Este no realiza cómputo, dedicándose exclusivamente a la gestión de recursos. El resultado es que el tiempo necesario para completar una prueba en MPI con dos procesos es el equivalente a llevarla a cabo con un sólo hilo en la versión secuencial. La cantidad máxima de procesos que se utilizan en las pruebas de cada CPU se fija al doble de la

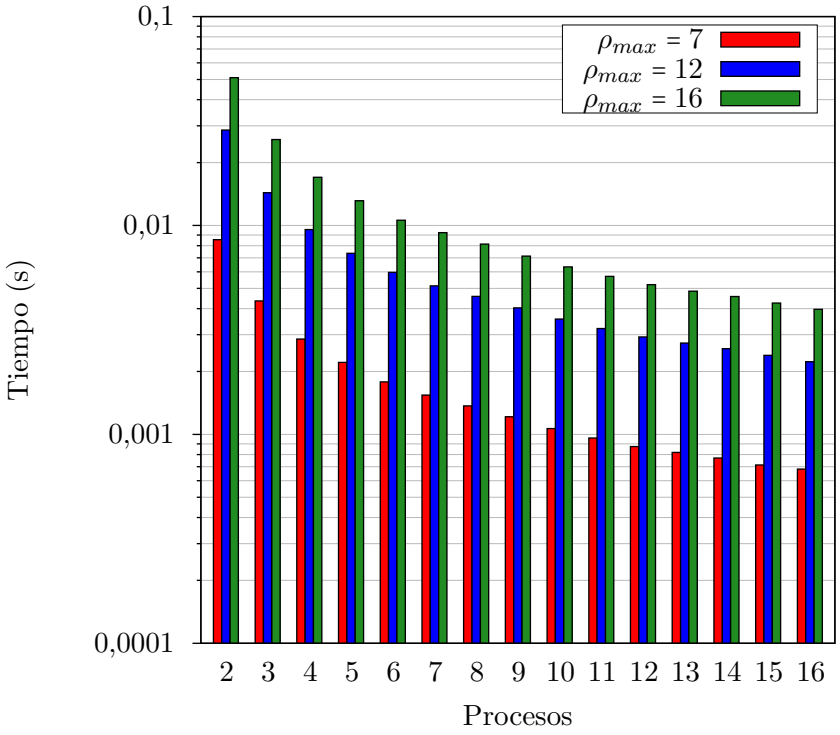


Figura 5.16: Tiempo promedio que lleva realizar el ajuste entre una pareja de imágenes empleando MPI con ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16 sobre el procesador Xeon E5-2650. El ancho de banda empleado es de 128.

cantidad de núcleos de los que dispone.

En primer lugar se realiza la prueba con el procesador Intel E5-2650. Este dispone de ocho núcleos físicos y de la tecnología *HyperThreading*. Mediante esta se duplican ciertas unidades computacionales del procesador. De este modo se dispone del doble de núcleos de forma virtual. Por lo tanto la prueba se realiza desde los dos procesos hasta un máximo de 16. La Figura 5.16 muestra el tiempo promedio que toma completar un único emparejamiento entre dos imágenes utilizando diferente número de procesos con esta CPU. Las correspondiente aceleración para ρ_{max} = 12 se muestra en la Figura 5.17 tomando como tiempo base el registrado para dos procesos. Las aceleraciones de ρ_{max} = 7 y ρ_{max} = 16 no se muestran ya que se superponen con la de ρ_{max} = 12. La aceleración obtenida empleando ocho procesos es de 6,24x, 6,25x y 6,26x para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16 respectivamente. Este procesador posee ocho núcleos físicos de modo que utilizando ocho procesos se están usando la totalidad de sus núcleos. Sólo siete de los procesos realizan cómputo, por lo que la aceleración no puede superar los 7x con este número de procesos. Sin embargo, la aceleración de 6,25x obtenida indica que el método escala muy bien. Empleando 16 procesos se obtienen unas aceleraciones de 12,54x, 12,85x y 12,84x respectivamente. Estos valores se alejan ligeramente de la aceleración cercana a 15x esperada en un escalado lineal. La aceleración crece más lentamente a partir de los ocho procesos debido a que la característica *HyperThreading* sólo duplica ciertas unidades dentro de cada núcleo. Los diferentes procesos ubicados en el mismo núcleo tienen que compartir recursos, compitiendo por ellos y degradando el rendimiento general.

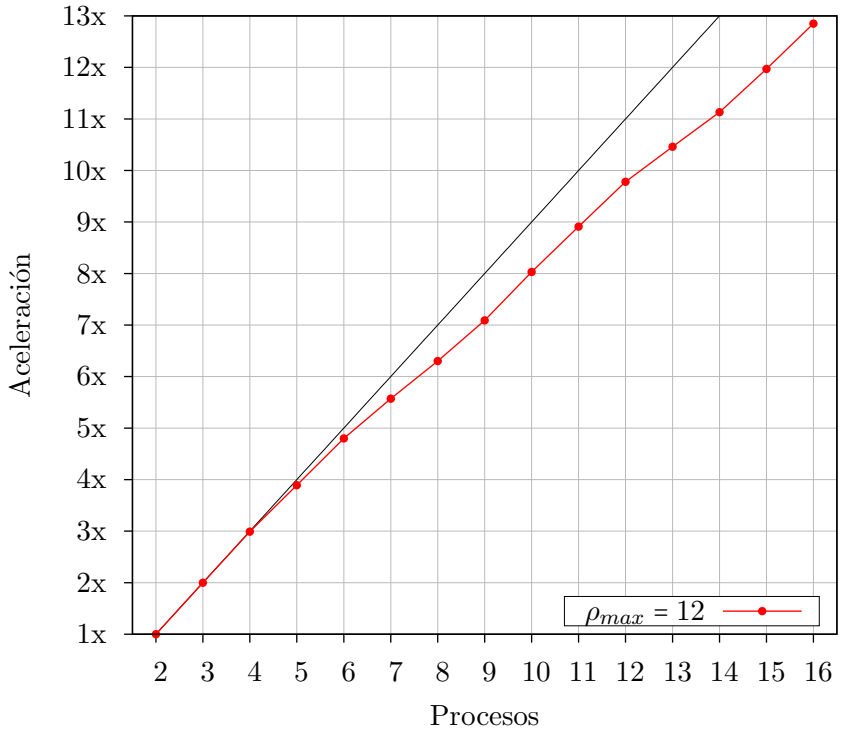


Figura 5.17: Aceleración obtenida en FBM con MPI sobre el Xeon E5-2650 utilizando diferente número de procesos. Se muestra la aceleración empleando $\rho_{max} = 12$ y una aceleración lineal de referencia. El ancho de banda utilizado es 128.

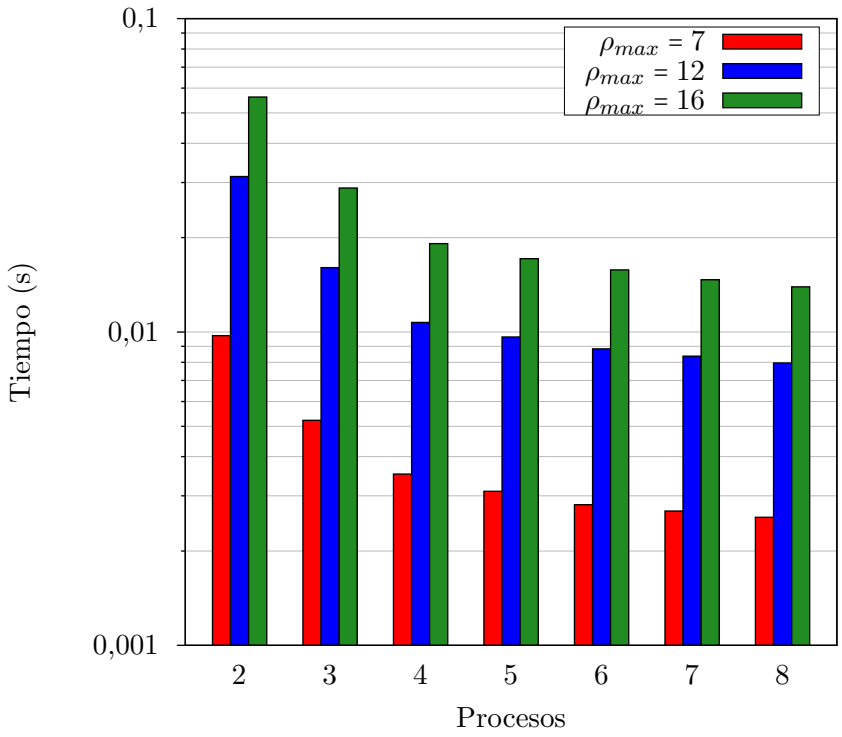


Figura 5.18: Tiempo promedio que lleva realizar el ajuste entre una pareja de imágenes empleando MPI con $\rho_{max} = 7$, $\rho_{max} = 12$ y $\rho_{max} = 16$ sobre el procesador Intel i7-950. El ancho de banda empleado es de 128.

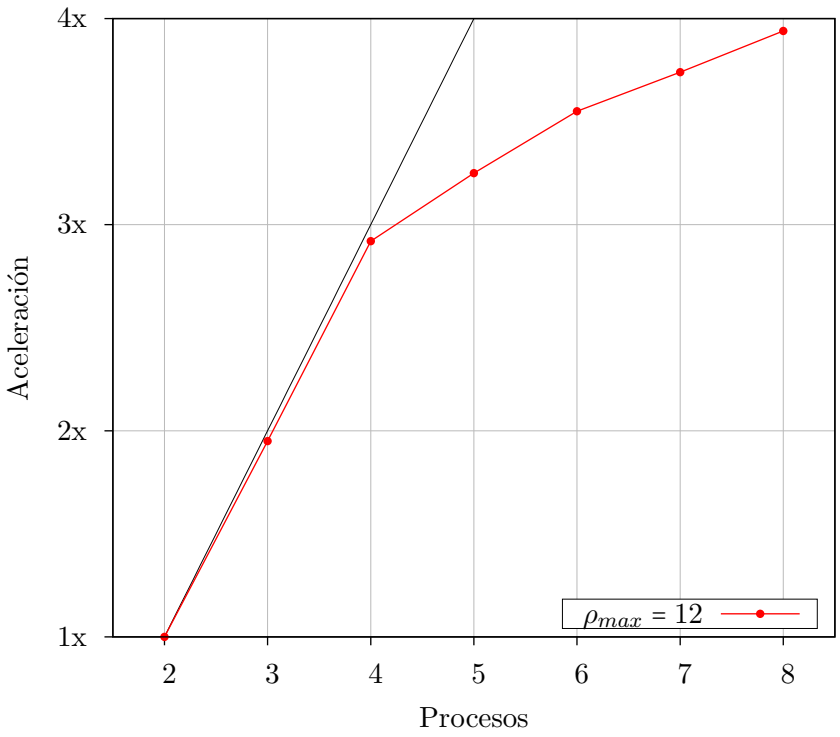


Figura 5.19: Aceleración obtenida en FBM con MPI sobre el i7-950 utilizando diferente número de procesos. Se muestra la aceleración empleando ρ_{max} = 12 y una aceleración lineal de referencia. El ancho de banda utilizado es 128.

Los resultados de tiempo promedio obtenidos con el procesador Intel i7-950 en las mismas condiciones que con el Intel Xeon se muestran en la Figura 5.18. Esta prueba se ha realizado desde los dos hasta los ocho procesos ya que este procesador posee cuatro núcleos físicos. Mediante la tecnología *HyperThreading* se dispone de ocho núcleos virtuales. Las tres configuraciones de ρ_{max} producen una curva de tiempos similar pero a diferentes escalas. Los datos correspondientes a la aceleración utilizando ρ_{max} = 12 se muestran en la Figura 5.19. La información correspondiente a los otros dos rangos de exploración no se muestra ya que prácticamente se superponen con la mostrada. Los tiempos base utilizados para calcular las aceleraciones son los registrados para dos procesos. En este procesador, el método escala de forma adecuada usando hasta cuatro procesos. Con este número se obtiene una aceleración de 2,94x, 2,92x y 2,77x para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16 respectivamente. Estas aceleraciones son adecuadas ya que se emplean tres

	ρ_{max} = 7	ρ_{max} = 12	ρ_{max} = 16
Intel i7-950	2,56 ms	7,96 ms	13,93 ms
Intel Xeon E5-2650	0,682 ms	2,228 ms	3,969 ms
Clúster Ladón	0,152 ms	0,505 ms	0,896 ms

Tabla 5.7: Resumen de tiempos de alineamiento en FBM sobre diferentes plataformas CPU. Los tiempos mostrados se corresponden con el empleo del máximo número de procesos disponibles en cada sistema: ocho procesos en el i7, 16 procesos en el Xeon y 80 procesos en Ladón. El ancho de banda es 128.

GPU (CODE NAME)	FRECUENCIA	NÚCLEOS	ARQUITECTURA
NVIDIA GTX 680 (GK104)	1,06 GHz	1536	Kepler
NVIDIA GTX 470 (GF100)	1,22 GHz	448	Fermi

Tabla 5.8: GPUs empleadas en las pruebas de rendimiento y precisión.

de los cuatro procesos para realizar cómputo. A partir de los cinco procesos, hasta el máximo de ocho, el método no escala bien. Esto se debe a las limitaciones del procesador con respecto a la tecnología *HyperThreading*. Empleando ocho procesos las aceleraciones obtenidas son 3,8x, 3,94x y 4,03x para ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16 respectivamente. En general, utilizar más de cuatro procesos en este procesador implica consumir más recursos sin obtener la correspondiente mejora en el rendimiento.

En resumen, FBM escala de forma lineal en todas las plataformas probadas mientras no se saturen los recursos *hardware*. La Tabla 5.7 muestra un resumen de los mejores tiempos promedio obtenidos en cada una de las plataformas realizando el alineamiento de una única pareja de imágenes.

5.2.1.2. Comparativa con dispositivos gráficos

A continuación se muestran los análisis de rendimiento y precisión realizados sobre la versión paralela de FBM para GPU y se comparan con los resultados obtenidos en la versión secuencial. Las pruebas realizadas son las mismas que las detalladas en el apartado de validación (sección 5.1.4). Estas consisten en el ajuste del conjunto completo de 76.000 imágenes simuladas con las 80 de referencia. En las comparativas de GPU con CPU se analizan los resultados obtenidos para ρ_{max} = 12. Este rango de exploración es el que mejores resultados proporciona con respecto a la precisión según se ha demostrado en la validación del método. Los anchos de banda utilizados en estas pruebas son 64 y 128. Los dispositivos gráficos empleados para llevar a cabo estas pruebas se muestran en la Tabla 5.8. La tarjeta NVIDIA GTX 470, con arquitectura Fermi y chip GF100, cuenta con una capacidad computacional 2.0. La tarjeta NVIDIA GTX 680, con la nueva arquitectura Kepler y chip GK104, tiene una capacidad computacional 3.0. Las pruebas se han realizado empleando la versión 302.59 del *driver* de vídeo y la versión 5.0 de CUDA [151].

En primer lugar se muestran los resultados relativos a la precisión. En la Tabla 5.9 se muestra el error promedio cometido en píxeles para los 19 niveles de SNR del conjunto de pruebas tanto en GPU como en CPU. La mayor diferencia se produce para SNR = 0,011 utilizando un ancho de banda de 64. Esta diferencia, de tan solo 0,4 píxeles, se considera una cantidad despreciable. Por otro lado, el ajuste de imágenes con este nivel de SNR no se considera válido según se concluyó en la sección de validación. El SNR más reducido para el que se han obtenido ajustes válidos con este conjunto de imágenes es 0,021. La diferencia entre los resultados de CPU y GPU para este SNR es de 0,02 píxeles. Esta mínima disparidad en los resultados responde a diversos factores. Por una parte el compilador de CUDA dispone de una opción (*fast math*) que permite explícitamente sustituir algunas operaciones con datos flotantes por versiones más rápidas implementadas mediante *hardware* en los dispositivos gráficos. Esto

SNR	B=64 (CPU)	B=64 (GPU)	B=128 (CPU)	B=128 (GPU)
0,5	1,05	1,05	0,56	0,56
0,25	1,07	1,07	0,58	0,58
0,13	1,10	1,10	0,62	0,62
0,074	1,15	1,15	0,67	0,67
0,048	1,21	1,21	0,74	0,74
0,037	1,32	1,33	0,81	0,81
0,028	2,08	2,10	1,11	1,11
0,021	5,58	5,60	2,63	2,63
0,015	17,52	17,62	11,24	11,24
0,011	32,42	32,82	27,22	27,22
0,008	45,57	45,67	42,21	42,21
0,007	49,95	50,01	47,47	47,47
0,006	53,87	53,60	51,94	51,94
0,005	57,28	56,98	56,10	56,10
0,004	60,21	60,17	59,37	59,37
0,003	62,17	62,16	62,06	62,06
0,002	63,34	63,16	63,55	63,55
0,0015	63,51	63,34	63,74	63,74
0,001	63,82	63,90	64,03	64,03

Tabla 5.9: Error promedio cometido en píxeles para CPU y GPU al realizar el ajuste mediante FBM del conjunto de pruebas de imágenes de microscopía electrónica simuladas. El valor de ρ_{max} se fija a 12.

produce un rendimiento mayor en los cálculos a costa de una reducción en la precisión de los resultados. Por otro lado, la librería que realiza el cálculo de las transformadas de Fourier en GPU (cuFFT) no siempre produce los mismos resultados que la librería que realiza esta misma tarea en CPU (FFTW). cuFFT dispone de un modo de compatibilidad con FFTW que produce los mismos resultados que esta en perjuicio de un rendimiento inferior. Dispone también de un modo nativo que trabaja de forma más eficiente a costa de obtener unos resultados similares pero no exactamente iguales. En la versión paralela de FBM se han activado tanto la opción *fast math* como el modo nativo de cuFFT ya que se ha demostrado que la diferencia de error entre la versión paralela con estas opciones activadas y la versión secuencial es despreciable.

A continuación se realiza una comparativa de la diferencia en la tasa de acierto obtenida entre ambas versiones. La Tabla 5.10 muestra las tasas de acierto obtenidas en CPU y GPU para ρ_{max} = 12 y los anchos de banda 64 y 128. Al igual que en la tabla anterior, la diferencia más alta se encuentra para SNR = 0,011. Ajustando imágenes con esta cantidad de ruido, la diferencia entre las versiones paralela y secuencial es del 0,45 %. Con respecto a las imágenes más ruidosas con las que los ajustes del método se consideran válidos (SNR = 0,021), la diferencia entre versiones es del 0,08 %. Estas disparidades en la tasa de acierto son tan reducidas que se consideran despreciables.

SNR	B=64 (CPU)	B=64 (GPU)	B=128 (CPU)	B=128 (GPU)
0,5	100 %	100 %	100 %	100 %
0,25	100 %	100 %	100 %	100 %
0,13	100 %	100 %	100 %	100 %
0,074	100 %	100 %	100 %	100 %
0,048	100 %	100 %	100 %	100 %
0,037	99,90 %	99,90 %	99,97 %	99,97 %
0,028	98,75 %	98,77 %	99,67 %	99,67 %
0,021	93,22 %	93,30 %	97,02 %	97,02 %
0,015	74,07 %	74,20 %	83,32 %	83,32 %
0,011	50,20 %	49,75 %	58,35 %	58,35 %
0,008	28,72 %	28,75 %	34,62 %	34,62 %
0,007	22,30 %	22,25 %	26,62 %	26,62 %
0,006	16,25 %	16,37 %	19,37 %	19,37 %
0,005	11,30 %	11,27 %	13,05 %	13,05 %
0,004	6,97 %	6,85 %	7,97 %	7,97 %
0,003	4,15 %	4,22 %	4,27 %	4,27 %
0,002	2,57 %	2,57 %	2,65 %	2,65 %
0,0015	2,07 %	2,15 %	1,85 %	1,85 %
0,001	1,82 %	1,80 %	1,7 %	1,7 %

Tabla 5.10: Porcentaje de aciertos para el conjunto de pruebas de imágenes de microscopía electrónica simuladas a través de FBM para CPU y GPU. Se fija el valor de ρ_{max} a 12.

GF100	ρ_{max} = 7	ρ_{max} = 12	ρ_{max} = 16
B=64	0,037 ms	0,113 ms	0,190 ms
B=128	0,476 ms	1,514 ms	2,596 ms
GK104	ρ_{max} = 7	ρ_{max} = 12	ρ_{max} = 16
B=64	0,022 ms	0,066 ms	0,111 ms
B=128	0,204 ms	0,659 ms	1,121 ms
i7-950	ρ_{max} = 7	ρ_{max} = 12	ρ_{max} = 16
B=64	0,747 ms	2,289 ms	4,037 ms
B=128	10,150 ms	32,088 ms	57,108 ms

Tabla 5.11: Tiempos promedio invertidos en el ajuste de una única pareja de imágenes en FBM. Se han registrado los tiempos para las arquitecturas gráficas GF100 y GK104. A modo de comparativa se presentan los tiempos correspondientes al procesador Intel i7-950. La tabla se divide en tres secciones, una para cada pieza de *hardware* empleado.

Por último se analiza el rendimiento de la adaptación de FBM a GPU. En la Tabla 5.11 se muestran los tiempos promedio para realizar un único alineamiento tanto con las arquitecturas gráficas GF100 y GK104 como con la CPU Intel i7-950. La tarjeta GF100 tiene un comportamiento estable con todos los anchos de banda y configuraciones de ρ_{max}. La aceleración promedio obtenida este chip es del 21,03x con respecto al procesador Intel i7-950. Por otro lado, la GK104 no presenta un comportamiento estable utilizando las diferentes configuraciones de ancho de banda y ρ_{max}. Utilizando el ancho de banda 64, la aceleración obtenida varía desde 33,95x con ρ_{max} = 7 hasta 36,37x con ρ_{max} = 12. Esto indica que con este ancho de banda la cantidad de datos a procesar no es lo suficientemente grande como para utilizar toda la capacidad computacional de la tarjeta. En cambio, la aceleración obtenida utilizando el ancho de banda de 128 es mucho más estable, con un promedio de 49,79x.

En resumen, se ha logrado realizar de forma exitosa una adaptación paralela del método de alineamiento de imagen FBM sobre tarjetas gráficas NVIDIA. La diferencia entre los resultados obtenidos por esta adaptación para GPU y la versión de CPU es tan pequeña que se considera despreciable. Además se logra realizar los alineamientos hasta 50 veces más rápido que en la versión secuencial.

5.2.2. Análisis detallado de los *kernels* desarrollados

A continuación se muestran los análisis que se han realizado para los dos *kernels* desarrollados: cálculo de la integral y búsqueda de la mejor correlación. Estos análisis se han llevado a cabo sobre la tarjeta gráfica GK104, que posee una arquitectura Kepler. No se ha realizado un análisis de la ejecución de los *kernels* incluidos en la librería cuFFT, pero se han hecho algunos análisis de rendimiento.

Los datos de perfilado (*profiling*) se obtienen mediante la herramienta *Nsight Visual Studio Edition 2.2* [154] proporcionada por NVIDIA. El perfilador recopila una gran variedad de datos, de los cuales se han seleccionado aquellos que se consideran más importantes para la evaluación del rendimiento de los *kernels*. Estos son:

- Tasa de transferencia de memoria: indica el porcentaje utilizado del ancho de banda disponible para transferencias en el dispositivo.
- Número de transacciones por solicitud: muestra el promedio de transacciones de memoria que se generan por cada solicitud de acceso. Este indicador se utiliza para evaluar la eficacia del patrón de acceso a memoria global.
- Serialización: indica el porcentaje de instrucciones lanzadas que necesitan ser repetidas. Se puede utilizar como medidor general de rendimiento pues incluye repeticiones de acceso a todos los tipos de memoria por un patrón no óptimo, repeticiones debidas a que todos los recursos de la tarjeta estén ocupados y repeticiones por otros motivos.
- IPC: número de instrucciones por ciclo que la tarjeta es capaz de procesar para un *kernel*. Puede verse afectado por numerosos factores como dependencias de datos, sincronizaciones entre *threads*, etc.

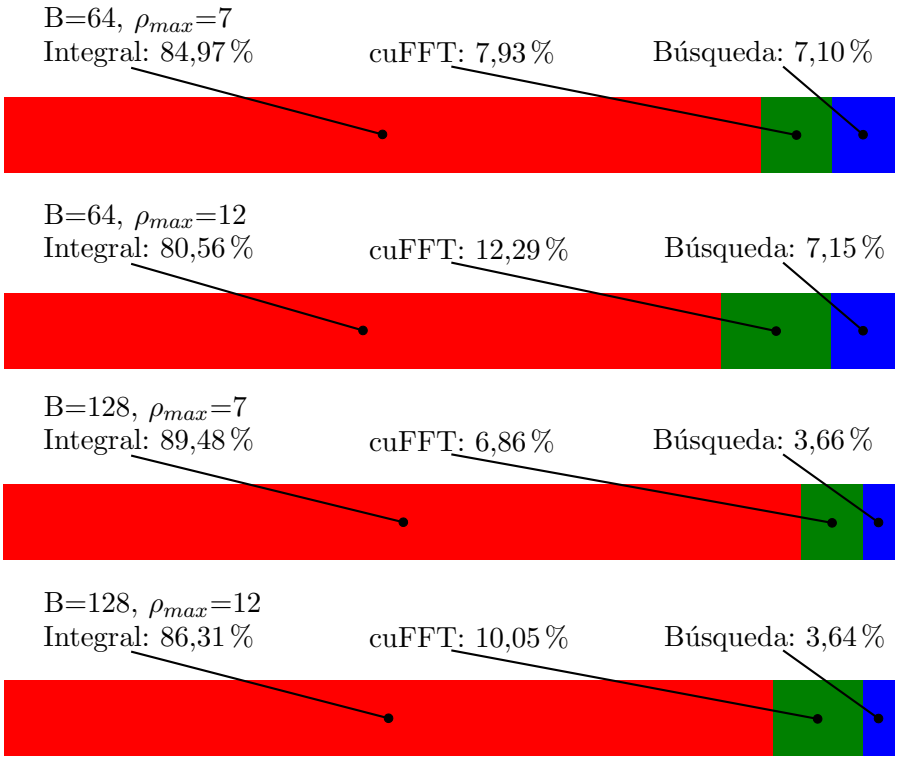


Figura 5.20: Porcentaje de tiempo de cómputo que cada kernel utiliza empleando los anchos de banda 64 y 128, con ρ_{max} = 7 y ρ_{max} = 12.

- Ocupación: indica la cantidad de *warps* activos en un multiprocesador.
- Motivos de detención de *warps*: la ejecución de un *warp* puede verse detenida por motivos como sincronizaciones entre *threads* o dependencia de datos o entre otros.
- Divergencia: muestra el porcentaje de instrucciones de salto que hacen que *threads* dentro un mismo *warp* tomen caminos diferentes.

Se ha registrado en primer lugar el porcentaje de tiempo de cómputo que cada *kernel* emplea para llevar a cabo su tarea a través de 4.000 iteraciones. Dependiendo del ancho de banda empleado y el valor de ρ_{max} las estructuras de datos tienen tamaños diferentes haciendo que varíe la carga de cada *kernel*. En la Figura 5.20 se muestran estos porcentajes empleando los anchos de banda 64 y 128, con ρ_{max} = 7 y ρ_{max} = 12. Se puede observar que el tiempo de cómputo que cada *kernel* emplea no varía mucho entre configuraciones. Las diferencias más notables se pueden apreciar entre los anchos de banda de 64 y 128 para el *kernel* que realiza la búsqueda de la mejor correlación y entre los valores ρ_{max} = 7 y ρ_{max} = 12 para la librería cuFFT. Esto indica que el muestreo rotacional es el factor que más altera la cantidad de operaciones necesarias para calcular una integral mientras que el muestreo traslacional produce un impacto mayor en el proceso de cálculo de la transformada inversa de Fourier.

Los análisis de los *kernels* se realizan empleando los anchos de banda de 64 y 128. La variable ρ_{max} se fija a 12. No se prueban más valores de ρ_{max} ya que esta variable sólo modifica el tamaño del *grid* manteniendo constante el número de *threads* por bloque. En primer lugar se detalla el

Tamaño de bloque	[64,1,1]	[128,1,1]
Warps por bloque	2	4
Ocupación	31,79/64 (49,67 %)	63,84/64 (99,75 %)
IPC	2,61	3,74
Serialización	5 %	6 %
Divergencia	11,02 %	6,56 %

Tabla 5.12: Información de perfilado para el *kernel* que calcula la integral en la tarjeta GK104 para los anchos de banda 64 y 128.

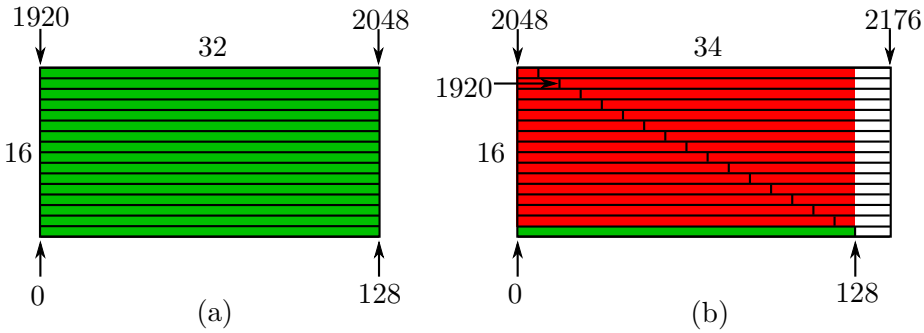


Figura 5.21: La figura ilustra un problema general de desalineamiento de memoria. Se presentan dos bloques de datos alojados en memoria global: uno de 16 x 32 bytes (a) y otro de 16 x 34 bytes (b). El espacio en blanco del bloque (b) se reserva por requerimiento del algoritmo que realiza la transformada de Fourier, y se mantiene vacío. En el bloque (a) se pueden realizar 16 lecturas consecutivas de 128 bytes. Cada lectura genera una transacción ya que las direcciones de memoria están alineadas con 128. En el bloque (b) sólo la primera lectura produce una transacción. Las 15 lecturas restantes producen dos transacciones ya que sus direcciones de memoria no están alineadas con 128.

análisis realizado para el *kernel* que calcula las integrales seguido del análisis correspondiente para el *kernel* que busca los parámetros de ajuste.

5.2.2.1. *Kernel* para el cálculo de la integral

Cada bloque de este *kernel* emplea 1.024 bytes de memoria compartida a la que tienen acceso todos sus *threads*. Los recursos consumidos por cada bloque son reducidos, logrando que por cada multiprocesador se pueda alojar su máximo de 16 bloques simultáneos. Se utiliza un total 16 KB de memoria compartida entre todos los bloques alojados. Esto permite configurar la caché L1 a un tamaño de 48 KB. La compilación de este *kernel* para la arquitectura Kepler produce un código que emplea 28 registros. En la Tabla 5.12 se muestran los datos obtenidos para la tarjeta GK104 utilizando los anchos de banda 64 y 128. El tamaño de la dimensión x del bloque es igual al ancho de banda empleado.

Comenzando por el análisis del *kernel* cuando se emplea un ancho de banda de 128, el perfilador indica una tasa de transferencia de memoria de 5,09 GB/s. Esta tasa está muy lejos del máximo teórico de 192,2 GB/s. Esto se produce por varios motivos: por una parte, los *threads*

de este *kernel* intercalan lecturas de memoria con cómputo por lo que no se realizan los suficientes accesos consecutivos a memoria como para obtener una tasa de transferencia elevada. Por otro lado, por cada solicitud de lectura se realiza un promedio de 2,69 transacciones, degradando el rendimiento de los accesos a memoria global. Esta cantidad de transacciones por solicitud se debe a que los datos de entrada correspondientes a las transformadas de Bessel tienen su parte real e imaginaria intercalada. Los diferentes *threads* de cada *warp* no leen segmentos contiguos de memoria ya que primero leen la parte real y luego la imaginaria, generando entre dos y tres transacciones por cada solicitud. Los datos escritos por este *kernel* en memoria global tienen también sus partes real e imaginaria intercalados ya que se usan a continuación como entrada por el algoritmo de la transformada de Fourier, que los lee de forma intercalada. La escritura de datos está más optimizada que la lectura, con un promedio de 1,94 transacciones por solicitud de acceso. Eso se logra a través de una escritura coalescente en memoria global mediante el empleo de memoria compartida. De este modo se reduce el número de transacciones por solicitud, pero este valor nunca puede bajar hasta 1 ya que el *padding* introducido mediante el valor complejo al final de la dimensión x de la transformada hace que las direcciones de acceso no estén adecuadamente alineadas. En la Figura 5.21 se muestra un ejemplo de cómo se produce este desalineamiento. El hecho de que para una solicitud de acceso a memoria se realicen varias transacciones hace que la instrucción de acceso a memoria tenga que repetirse una vez por cada transacción adicional. Estas repeticiones de instrucciones están indicadas por el contador de serialización que proporciona el perfilador. Este *kernel* tiene sólo un 6 % de instrucciones serializadas, el cual es un valor muy bajo. Con respecto a los accesos a la memoria compartida, por cada solicitud de lectura y escritura se realiza sólo una transacción para este *kernel*, lo que indica que no existen conflictos de bancos. Con esto se puede concluir que el *kernel* que calcula la integral no está limitado por el ancho de banda de memoria.

Las tarjetas de arquitectura Kepler poseen cuatro *schedulers* por multiprocesador que se encargan de lanzar nuevas instrucciones cada ciclo. Cada *scheduler* es capaz de lanzar en el mismo ciclo dos instrucciones siempre que estas sean independientes. Por lo tanto el máximo teórico alcanzable sería de ocho instrucciones por ciclo si el procesador nunca se detuviera. En la práctica, una cantidad de instrucciones por ciclo de cuatro o más se considera un valor alto. Este *kernel* se acerca a este valor, con un IPC de 3,74. El perfilador indica un promedio de 7,4 *warps* con instrucciones disponibles por cada ciclo activo de reloj. Esta parece una cantidad adecuada teniendo en cuenta que se necesitan al menos cuatro *warps* elegibles para los cuatro *schedulers*. Sin embargo, en el 26,38 % de las veces no hay ningún *warp* elegible y los *schedulers* no lanzan instrucciones. El análisis indica que en el 48,4 % de los casos en los que el procesador se detiene, el motivo es por dependencia de datos. En este *kernel* en concreto, esto se traduce en que el código no posee suficientes instrucciones independientes que puedan lanzarse mientras se resuelven las dependencias de las demás. El siguiente motivo de parada más frecuente, que sucede en el 15,9 % de los casos, se debe a instrucciones de sincronización. El *kernel* sólo posee una instrucción de sincronización que hace que los *threads* esperen a que toda la memoria compartida haya sido escrita puesto que unos *threads* escriben en memoria global datos que otros *threads* han generado. Por último, en el 13,1 % de los casos los *warps* se detienen debido a que la siguiente instrucción en ensamblador aún no ha llegado al procesador. Otro factor que altera el rendimiento de la

ejecución es la divergencia. Esto sucede cuando los *threads* dentro de un mismo *warp* toman caminos de ejecución diferentes. Los diferentes caminos se serializan ejecutándose primero uno y luego el otro. Una alta divergencia tiene un impacto muy negativo en el rendimiento. Este *kernel* está bastante optimizado en este aspecto. El análisis muestra tan sólo un 6,56 % de divergencia, lo cual es una cantidad bastante reducida.

Cuando se emplea el ancho de banda de 64 el rendimiento del *kernel* se reduce en gran medida. La divergencia aumenta hasta el 11 %, aunque sigue siendo un valor bajo. Este incremento se debe a que las condiciones de los saltos y bucles es muy dependiente de los tamaños que se utilicen en las estructuras de datos. Por otra parte, la serialización se reduce desde el 6 % hasta el 5 %. Con la configuración de los bucles en este ancho de banda, se realizan más solicitudes de dos transacciones y menos de tres que con el ancho de banda de 128. Esto hace que las instrucciones de acceso a memoria tengan que repetirse menos veces. En promedio se realizan 2,56 transacciones por solicitud de acceso de lectura a memoria global. La escritura se realiza exactamente del mismo modo, por lo que se mantienen las 1,94 transacciones por solicitud que con el ancho de banda 128.

La cantidad máxima de bloques alojados simultáneamente por multiprocesador es de 16. Cuando se emplea el ancho de banda de 128 se ocupaba el multiprocesador completamente. Con el ancho de banda de 64 se reduce a la mitad la cantidad de *threads* por bloque, pero se siguen alojando el máximo de 16 bloques simultáneos. Esto hace que la ocupación del multiprocesador se reduzca a la mitad, empleando 32 de los 64 *slots* disponibles para *warps*. En este caso, reducir la cantidad de *warps* tiene un impacto directo en el rendimiento. El análisis del perfilador indica que en promedio se dispone de tres *warps* elegibles por ciclo activo. El mínimo indispensable para que cada *scheduler* lance al menos una instrucción por ciclo es de cuatro *warps* disponibles. En el 49,1 % de los casos no existe ningún *warp* disponible, por lo que los *schedulers* no lanzan instrucciones. El principal motivo de parada de los *warps* es la dependencia de datos. Con este ancho de banda el procesador tiene que esperar a que los operandos estén listos en un 60,8 % de las veces en las que se detiene. El siguiente motivo de parada más frecuente, que sucede en el 13,3 % de los casos, continúa siendo la sincronización entre *threads*.

En resumen, el análisis ha demostrado que el comportamiento de este *kernel* es mejor con el ancho de banda de 128 que con el de 64 ya que con este último no se utilizan plenamente los recursos disponibles en la tarjeta. El análisis también revela que el siguiente paso para obtener un mayor rendimiento pasaría por modificar el *kernel* para conseguir que el código tenga más instrucciones independientes.

5.2.2.2. *Kernel* para la búsqueda de parámetros de ajuste

El *kernel* que realiza la búsqueda se compila utilizando 27 registros para la arquitectura Kepler. Los bloques tienen una cantidad de *threads* igual al ancho de banda utilizado. Además por cada bloque se reserva la cantidad de 2.064 *bytes* en memoria compartida repartidos entre dos vectores de 256 bytes y cuatro variables individuales. El número de bloques que se alojan simultáneamente en los multiprocesadores es el máximo de 16. Por lo tanto, la cantidad total de memoria compartida utilizada por multiprocesador es de 33.024 bytes. La única opción de

Tamaño de bloque	[64,1,1]	[128,1,1]
Warps por bloque	2	4
Ocupación	31,78/64(49,65 %)	59,29/64(92,64 %)
IPC	1,49	2,33
Serialización	0 %	0 %
Divergencia	13,85 %	6,7 %

Tabla 5.13: Información de perfilado para el *kernel* que realiza la búsqueda de la mejor correlación en la tarjeta GK104 para los anchos de banda 64 y 128. El valor de ρ_{max} utilizado es 12.

configuración de tamaños de memoria posible es de 48 KB para memoria compartida y 16 KB para caché L1. De otro modo el *kernel* no podría ejecutarse. La Tabla 5.13 muestra los resultados de los marcadores registrados para este *kernel*.

Para el ancho de banda 128 se aprecia que los accesos a memoria son óptimos, con una serialización del 0 %. Todos los accesos a memoria global y compartida, tanto de lectura como de escritura se realizan con tan sólo una transacción por solicitud. La tasa de transferencia de memoria obtenida es de 32,75 GB/s, muy lejos de la tasa máxima teórica de transferencia, lo que indica que el *kernel* no está limitado por el ancho de banda de memoria.

La ocupación del multiprocesador es prácticamente completa, utilizando un promedio de 59,29 warps activos sobre el máximo de 64. Los resultados del perfilador indican que hay una leve divergencia del 6,7 % en la ejecución del código de los *threads* del mismo *warp*. Esto no está producido realmente porque los *threads* tomen diferentes caminos sino porque cada uno de ellos realiza una cantidad diferente de iteraciones en el bucle de reducción. Esta divergencia no tiene un impacto negativo ya que los *threads* inactivos no ejecutan otro código que deba serializarse. Con respecto al rendimiento de ejecución, se obtiene una cantidad de instrucciones ejecutadas por ciclo de 2,33. Este no es un valor lo suficientemente alto como para que el *kernel* esté limitado por cantidad de instrucciones ejecutadas. El resultado del perfilador indica que hay un promedio de 2,88 *warps* disponibles por ciclo activo. Este es un valor muy bajo ya que al menos se necesitan cuatro *warps* activos por ciclo para alimentar los cuatro *schedulers*. Es más, en el 52,2 % de los casos ningún *scheduler* puede lanzar nuevas instrucciones. El principal motivo de parada que sucede en el 57 % de las veces es por dependencia de datos, seguido por sincronizaciones entre *threads* en el 33,1 % de los casos. La dependencia de datos en este *kernel* es un claro indicador de que la latencia de lectura en memoria global no se oculta. No es posible reducir la dependencia agregando más instrucciones independientes dentro del mismo *warp*. Este *kernel* realiza un proceso iterativo en serie en el que el resultado de cada iteración depende siempre de la anterior. Tampoco es posible agregar más instrucciones independientes incrementando el número de *warps* ya que actualmente se está empleando la cantidad máxima en cada multiprocesador.

Para el ancho de banda 64 se observa un comportamiento similar. Los accesos a memoria son óptimos, con una transacción por acceso tanto en lectura como en escritura, en memoria

	Integral	Búsqueda
IPC	3,74	2,33
Transferencia	5,09 GB/s	32,75 GB/s
Factor limitante	Dependencia de datos	Dependencia de datos

Tabla 5.14: Resumen de análisis de los *kernels* de FBM. Los datos mostrados corresponden al ancho de banda 128.

compartida y global. La tasa de transferencia de memoria se reduce a 20,75 GB/s ya que la cantidad de datos es menor empleando este ancho de banda. Al igual que con el ancho de banda de 128, el ancho de banda de memoria del dispositivo no es el factor limitante sobre la capacidad de cómputo.

El hecho de reducir a la mitad el tamaño de bloque, que ahora es de 64 *threads*, hace que también se reduzca la cantidad de *warps* por bloque. Este valor se reduce desde los 4 *warps*/bloque en 128 hasta los 2 *warps*/bloque en 64. Debido a la limitación del máximo de 16 bloques alojados simultáneamente por multiprocesador, la ocupación máxima con esta configuración de ancho de banda es del 49,65 %. La cantidad de instrucciones ejecutadas por ciclo se ve mermada a 1,49 por este motivo. El perfilador muestra que la cantidad promedio de *warps* elegibles por ciclo activo baja hasta 1,5, y que en el 69,92 % de los casos no hay ningún *warp* con instrucciones listas para ser ejecutadas. Al igual que en el caso anterior, el principal motivo es la espera por dependencia de datos (67 % de las veces). Al reducir la cantidad de *warps* por bloque se tienen menos instrucciones que ejecutar y el efecto de la espera por la latencia de memoria global se ve incrementado. Otro efecto producido por la reducción de la cantidad de *warps* es el incremento de la divergencia. La subida hasta el 13,85 % de este marcador no tiene ningún impacto negativo. Este valor se dobla debido a que el tamaño de bloque se reduce a la mitad, por lo que el porcentaje total de *threads* divergentes es mayor.

Por último la Tabla 5.14 muestra un resumen de los aspectos más importantes de los análisis realizados a los dos *kernels*. Los datos mostrados son los correspondientes al ancho de banda 128 ya que se ha demostrado que con este valor se utilizan completamente los recursos disponibles de la tarjeta gráfica. Los dos *kernels* desarrollados están bastante optimizados, sin embargo ninguno de ellos está completamente limitado por los recursos disponibles en el *hardware*. El motivo principal de sus limitaciones se debe a restricciones del método ejecutado.

5.2.2.3. Transformadas rápidas de Fourier

El proceso del cálculo de las transformadas de Fourier sobre los dispositivos gráficos se lleva a cabo mediante la librería cuFFT proporcionada por NVIDIA.

La cantidad de ajustes 2D que se pueden realizar simultáneamente está limitado por este paso. Esto se debe a la gran cantidad de memoria empleada por los volúmenes a transformar. La transformada puede realizarse *in-place* ó *out-of-place*. Que se realice *in-place* implica que la zona de memoria de los datos de origen es la misma que la de destino. Una transformada *out-of-place* tiene zonas de origen y destino diferentes en memoria. Realizar la transformada

B	ρ_{max}	TAMAÑO	MEMORIA
64	7	[8,8,128]	64 KB
64	12	[14,14,128]	196 KB
64	16	[18,18,128]	324 KB
128	7	[16,16,256]	512 KB
128	12	[28,28,256]	1,53 MB
128	16	[36,36,256]	2,53 MB

Tabla 5.15: Tamaños de transformadas de Fourier y espacio usado de memoria. Se muestran los valores para los anchos de banda de 64 y 128 con ρ_{max} = 7, ρ_{max} = 12 y ρ_{max} = 16. Las transformadas se hacen *out-of-place*.

B	ρ_{max}	AJUSTES SIMULTÁNEOS	TIEMPO POR AJUSTE
64	7	20.000	1,70 μs
64	12	6.480	7,83 μs
64	16	3.920	12,97 μs
128	7	2.480	13,95 μs
128	12	800	65,24 μs
128	16	480	108,78 μs

Tabla 5.16: Cantidad de transformadas que se pueden realizar simultáneamente sobre la GK104 y tiempo promedio que se emplea en realizar una única transformada.

out-of-place requiere el doble de memoria que hacerla *in-place*. Se ha tomado la decisión de hacer las transformadas *out-of-place* por motivos de eficiencia a pesar de requerir más memoria. Los espacios de memoria de entrada y salida de la transformada tienen diferentes tamaños. En concreto se aprovecha la ventaja de que los volúmenes de correlación de salida tienen dimensiones compatibles con el tamaño de *warp* y sus direcciones de memoria están alineadas con múltiplos de este mismo tamaño. De este modo, el *kernel* que realiza la búsqueda de la mejor correlación puede realizar accesos óptimos de lectura en memoria global.

En la Tabla 5.15 se muestra el tamaño de las transformadas para cada una de las combinaciones de ancho de banda y ρ_{max} empleada. Se muestra también la cantidad de memoria que emplea cada transformada individualmente, teniendo en cuenta el espacio adicional por el uso *out-of-place*.

A continuación se muestran en las Tablas 5.16 y 5.17 información relativa a las transformadas de Fourier sobre las tarjetas GK104 y GF100 respectivamente. En estas tablas se muestra para cada una de las configuraciones de ancho de banda y ρ_{max} la cantidad de transformadas que se pueden realizar simultáneamente así como el tiempo promedio que cada tarjeta toma para calcular una única transformada. La GK104 utilizada (NVIDIA GTX 680) dispone de 2 GB de RAM mientras que la GF100 (NVIDIA GTX 470) cuenta con 1,28 GB. Se puede observar que la GK104 es capaz de realizar simultáneamente algo menos del doble de transformaciones que con la GF100, siendo esto acorde con sus respectivas cantidades de memoria.

B	ρ_{max}	AJUSTES SIMULTÁNEOS	TIEMPO POR AJUSTE
64	7	11.840	2,35 μs
64	12	3.840	10,69 μs
64	16	2.320	17,82 μs
128	7	1.440	19,01 μs
128	12	480	87,73 μs
128	16	240	145,93 μs

Tabla 5.17: Cantidad de transformadas que se pueden realizar simultáneamente sobre la GF100 y tiempo promedio que se emplea en realizar una única transformada.

La librería cuFFT utiliza el algoritmo Cooley-Tukey [32] para realizar el cálculo de las transformadas de Fourier. Este método está optimizado para transformadas cuyas dimensiones sean múltiplo de 2, 3, 5 ó 7. Además, cuantos menos factores se empleen en estas dimensiones, mejor rendimiento se obtendrá. Las transformadas más rápidas se computan para ρ_{max} = 7 en ambos anchos de banda. Utilizando B=64 se generan unos volúmenes de 8 x 8 x 128 vóxeles. Sus dimensiones pueden descomponerse como $[2^3, 2^3, 2^7]$. Si el ancho de banda es 128, los volúmenes generados son de 16 x 16 x 256 y sus dimensiones se pueden descomponer como $[2^4, 2^4, 2^8]$. Las dimensiones de estas transformadas se descomponen empleando sólo el factor 2. En ambos tamaños de volumen se obtiene una proporción de 0,21 ns/vóxel al realizar la transformación. El resto de combinaciones de anchos de banda y rangos de escaneo traslacional producen volúmenes cuyas dimensiones se descomponen empleando diversas combinaciones de factores. La descomposición de las dimensiones de las otras dos transformadas que emplean el ancho de banda 64 son $[2^1 \cdot 7^1, 2^1 \cdot 7^1, 2^7]$ y $[2^1 \cdot 3^2, 2^1 \cdot 3^2, 2^7]$ para ρ_{max} = 12 y ρ_{max} = 16 respectivamente. Para ambos tamaños de volumen se registra una proporción de 0,31 ns/vóxel al realizar la transformada. Por último, las dimensiones de las otras dos transformadas que emplean el ancho de banda 128 se pueden descomponer como $[2^2 \cdot 7^1, 2^2 \cdot 7^1, 2^8]$ y $[2^2 \cdot 3^2, 2^2 \cdot 3^2, 2^8]$ para ρ_{max} = 12 y ρ_{max} = 16 respectivamente. Sus correspondientes proporciones son de 0,32 ns/vóxel y 0,33 ns/vóxel.

En resumen, se ha observado que la librería cuFFT obtiene un mayor rendimiento empleando ρ_{max} = 7. Para este caso concreto se generan estructuras de datos que coinciden con configuraciones óptimas para realizar el cómputo de las transformadas de Fourier. Emplear otros valores de ρ_{max} hacen que el rendimiento en el cómputo de las transformadas se reduzca muy ligeramente, no teniendo un gran impacto en el tiempo global de ejecución.

5.2.3. Comparativa con *Fast Rotational Matching 2D*

En esta sección se presenta la comparativa de precisión y rendimiento realizada sobre FBM con respecto a otro método llamado *Fast Rotational Matching 2D* (FRM2D). Este está implementado en el paquete de utilidades de microscopía electrónica EMAN [119] y es uno de los métodos más rápidos de ajuste 2D empleado en la actualidad. Por esta razón se ha decidido comparar FBM con este método. Además, como se verá en la siguiente sección, se ha mejorado el rendimiento de FRM2D un orden de magnitud antes de compararlo con FBM.

	B = 64	B = 128
FRM2D Original (EMAN)	208 ms	1.070 ms
FRM2D Optimizado	20 ms	103 ms

Tabla 5.18: Tiempo promedio tomado por alineamiento con la versión original y la optimizada de FRM2D.

Para realizar la comparativa entre ambos métodos se han empleado tres conjuntos de imágenes de microscopía electrónica: el conjunto de imágenes simuladas utilizado en la validación y dos conjuntos de imágenes experimentales obtenidas a través de microscopios electrónicos.

5.2.3.1. Optimización de *Fast Rotational Matching 2D*

En esta sección se muestran los resultados obtenidos a partir de la optimización llevada a cabo en FRM2D en la sección 4.1.3. La prueba que se ha realizado para registrar los tiempos consiste en alinear uno de los conjuntos de 4.000 imágenes simuladas con las 80 de referencia. Estas imágenes pertenecen al conjunto diseñado para la validación de FBM (sección 5.1.4). La Tabla 5.18 muestra el tiempo promedio tomado para el alineamiento de una imagen simulada con una de referencia tanto para la versión original como para la optimizada de FRM2D. Los tiempos se han registrado para los anchos de banda de 64 y 128, empleando un valor de ρ_{max} = 7. El procesador utilizado en esta prueba es un Intel i7-950.

A través de la mejora realizada a este método de alineamiento se ha logrado obtener una aceleración promedio de 10,39x con respecto a la versión original implementada en EMAN. Esto muestra que la mejora realizada en el método proporciona una aceleración consistente independientemente del ancho de banda utilizado. Además, la mejora aplicada tan sólo afecta al tiempo de ejecución, no viéndose modificada la precisión del alineamiento.

5.2.3.2. Comparativa con imágenes simuladas de microscopía electrónica

A continuación se realiza una comparativa de precisión y rendimiento de FBM con la versión optimizada de FRM2D utilizando el conjunto de imágenes generadas para la validación (sección 5.1.4). Este conjunto está compuesto por 76.000 imágenes simuladas y 80 de referencia con un tamaño de 128 x 128 píxeles. Se considera que una imagen está bien alineada cuando su error de ajuste no supera el 5 % del tamaño de imagen (6,4 píxeles). Las traslaciones aplicadas al conjunto de pruebas nunca exceden los 6 píxeles de distancia desde el centro por lo que se fija un valor de ρ_{max} de 7, que añade un breve margen de error. Para FRM2D el valor de ρ_{max} no influye en el muestreo traslacional de las imágenes, por lo que se mantiene fijo para todas las pruebas de este método. En cambio se prueban dos anchos de banda diferentes que corresponden con un muestreo angular de 2,8º (64) y 1,4º (128). Con respecto a FBM, sólo se muestran las líneas correspondientes a ρ_{max} = 12 ya que es el rango de exploración traslacional que proporciona los alineamientos más precisos. La Figura 5.22 muestra los resultados obtenidos en esta prueba. Ambos métodos producen alineamientos dentro del error máximo admitido hasta llegar al SNR de 0,021. En este SNR en concreto, la línea correspondiente a FBM con un ancho de banda de 64 produce un error ligeramente mayor que las demás, con un promedio de 5,57 píxeles de

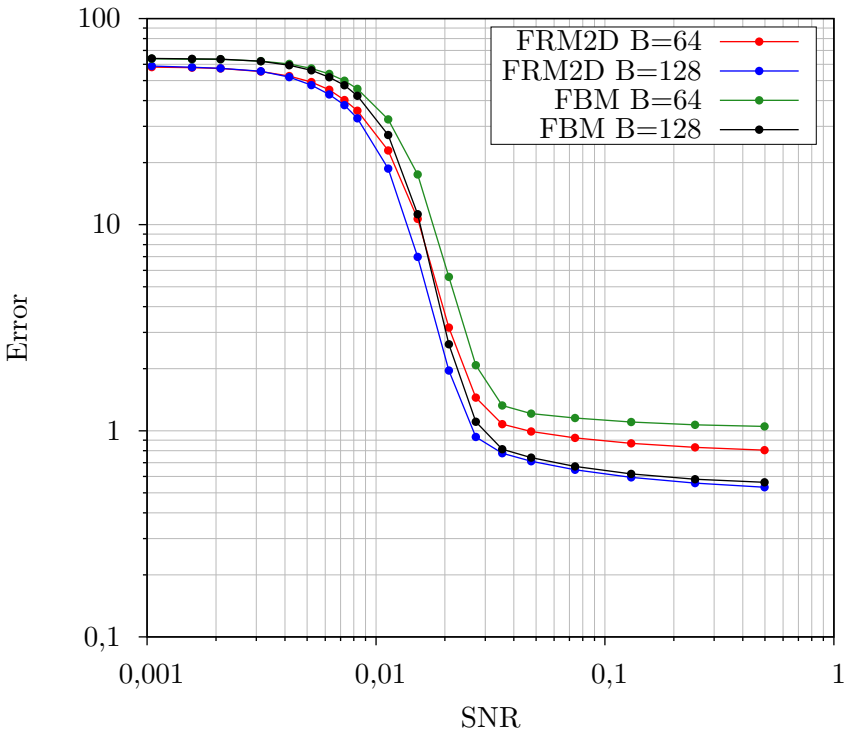


Figura 5.22: Error promedio cometido en píxeles al realizar el ajuste del conjunto de pruebas de imágenes de microscopía electrónica simuladas utilizando FRM2D. El valor de ρ_{max} para FRM2D se fija en 7. Se muestran los anchos de banda de 64 y 128. A modo de comparativa se incluyen también los resultados de FBM con ancho de banda 64 y 128. Los datos de FBM se muestran sólo para ρ_{max} = 12, que es el rango de exploración traslacional para el que se obtiene mayor precisión.

error. Las otras tres líneas muestran errores parecidos siendo estos 1,96, 2,63 y 3,16 píxeles en promedio para FRM2D con un ancho de banda de 128, FBM con ancho de banda 128 y FRM2D con ancho de banda de 64 respectivamente. A partir de este valor de SNR hacia valores inferiores el error es demasiado alto como para considerar que los alineamientos se realizan correctamente. En el otro sentido del eje, a partir de un SNR de 0,028 hacia valores más elevados, la línea de FRM2D con ancho de banda de 128 indica un error promedio inferior a 1 pixel. A partir del SNR = 0,035 la línea de FBM con ancho de banda 128 produce también un error inferior a 1 pixel. La última línea en mostrar un error inferior a 1 pixel es la de FRM2D con ancho de banda de 64 en el rango de SNR a partir de 0,05 hacia SNR superiores. La línea de FBM con ancho de banda de 64 nunca llega a producir un error promedio inferior a 1 pixel, siendo 1,2 píxeles su mínimo error promedio en el SNR de 0,5. Los resultados obtenidos en esta prueba indican que FRM2D es ligeramente menos sensible al ruido que FBM.

A continuación se muestra en la Figura 5.23 la tasa de acierto obtenida en los alineamientos de esta prueba. Ambos métodos de alineamiento en todas sus configuraciones tienen una tasa de acierto del 100 % con imágenes de SNR hasta 0,035. A partir de este SNR hacia valores inferiores, la tasa de acierto comienza a decrecer ligeramente hasta el SNR de 0,021. En este punto no hay mucha diferencia entre tasas de acierto, salvo por la correspondiente a FBM con ancho de banda

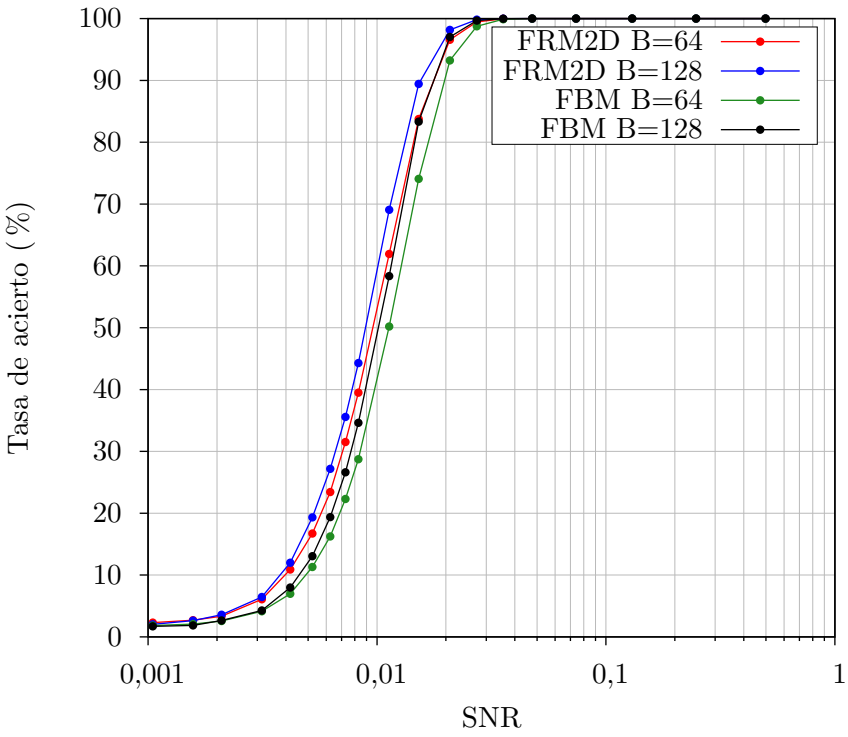


Figura 5.23: Porcentaje de aciertos para ajuste del conjunto de pruebas de imágenes de microscopía electrónica simuladas. Se muestran los resultados obtenidos por FRM2D correspondientes al ancho de banda de 64 y 128. FRM2D utiliza un ρ_{max} = 7. Se muestran las tasas de acierto para FBM correspondientes a los anchos de banda de 64 y 128 con un ρ_{max} = 12.

de 64 y ρ_{max} = 12. Esta tasa de acierto (93 %) es ligeramente inferior a las demás (alrededor del 97 %). En el siguiente valor inferior de SNR (0,015), los valores de las cuatro tasas de acierto se separan formando tres grupos. Por una parte sobresale la correspondiente a FRM2D con el ancho de banda 128 logrando un valor del 89 %. A continuación se encuentran, con valores parecidos, las tasas de acierto correspondientes a FRM2D con un ancho de banda de 64 (83,7 %) y FBM con un ancho de banda de 128 (83,3 %). Por último, FBM con el ancho de banda de 64 produce una tasa de acierto del 74 %. En cualquier caso, ninguno de los métodos con sus diferentes configuraciones proporciona resultados dentro del margen de error admitido para este nivel de ruido. Se puede concluir que desde el SNR de 0,015 hacia niveles menores no se pueden considerar útiles los alineamientos calculados por ninguno de los métodos.

Por último se muestra en la Tabla 5.19 los tiempos empleados por FBM y FRM2D en esta prueba. Los tiempos se han registrado para las versiones secuenciales de los métodos sobre la CPU Intel i7-950. El valor de tiempo indica el promedio necesario para ajustar una única pareja de imágenes de microscopía. Además de los tiempos registrados para las líneas mostradas en las gráficas, se muestran también los correspondientes a ρ_{max} = 7 con FBM. Estas líneas no se han mostrado en las gráficas de esta sección para evitar sobrecargarlas visualmente, pero se muestran en el apartado de validación (sección 5.1.4). Empleando un ρ_{max} = 7 con FBM se obtiene una precisión ligeramente inferior a FRM2D a costa de un rendimiento notablemente más elevado.

	B	ρ_{max} = 7	ρ_{max} = 12
FRM2D	64	20,418 ms	-
	128	103,28 ms	-
FBM	64	0,747 ms	2,289 ms
	128	10,150 ms	32,088 ms

Tabla 5.19: Promedio de tiempo necesario para el ajuste entre una única pareja de imágenes en FBM y FRM2D. Los tiempos se han registrado empleando la versión secuencial de los métodos sobre la CPU Intel i7-950. FRM2D no ha sido probado con ρ_{max} = 12 ya que con ρ_{max} = 7 se cubre el espacio traslacional necesario.

En comparación, FBM funciona 27,33 y 10,17 veces más rápido que FRM2D para los anchos de banda 64 y 128 respectivamente. El valor de ρ_{max} que más precisión proporciona en FBM es 12. Utilizando este valor, FBM funciona 8,92 y 3,22 veces más rápido que FRM2D para los anchos de banda de 64 y 128 respectivamente.

En resumen, a través del conjunto de pruebas con imágenes simuladas se ha podido observar que FRM2D es ligeramente más preciso que FBM. La diferencia en el error es ligeramente mayor empleando el ancho de banda de 64 que en el de 128. En cualquier caso esta diferencia no es muy elevada. El hecho de que FRM2D emplee un rango de exploración de 7 y FBM de 12 para obtener precisiones similares no es algo significativo pues en FBM esta variable modifica el patrón de muestreo traslacional y en FRM2D no. El rango de exploración de 12 en FBM es necesario en este caso para obtener una precisión similar a FRM2D. Utilizar un rango de exploración más extenso en FBM implica que los tiempos de ejecución son más elevados. La gran diferencia está en el rendimiento. FBM es al menos tres veces más rápido que la versión optimizada de FRM2D y unas 30 veces más rápido que la versión original implementada en EMAN.

5.2.3.3. Comparativa con imágenes experimentales de microscopía electrónica

FBM y FRM2D se han comparado también usando imágenes experimentales reales obtenidas por microscopios electrónicos. Se dispone de dos conjuntos de pruebas: unas imágenes de baja resolución (64 x 64 píxeles) y otras de gran resolución (320 x 320 píxeles).

Las imágenes de baja resolución empleadas son proyecciones del antígeno T grande del virus del simio 40 (SV 40) [186, 185] proporcionadas por el Dr. Sjors Scheres (MRC,UK). El conjunto de datos empleado para realizar esta prueba consiste en un grupo de 1.670 imágenes experimentales y dos de referencia. Las dos imágenes de referencia y algunas de las experimentales se muestran en la Figura 5.24. Las imágenes de este conjunto tienen un tamaño de 64 x 64 píxeles y se ha utilizado un ancho de banda de 64. Este ancho de banda produce un muestreo que cubre de forma razonable los píxeles más significativos de las imágenes de este tamaño.

Las dos imágenes de referencia han sido generadas mediante un proceso iterativo a partir de las imágenes experimentales. El programa encargado de generar las referencias se llama *MLalign2D*. Este programa está incluido en el paquete XMIPP. *MLalign2D* emplea dos imágenes

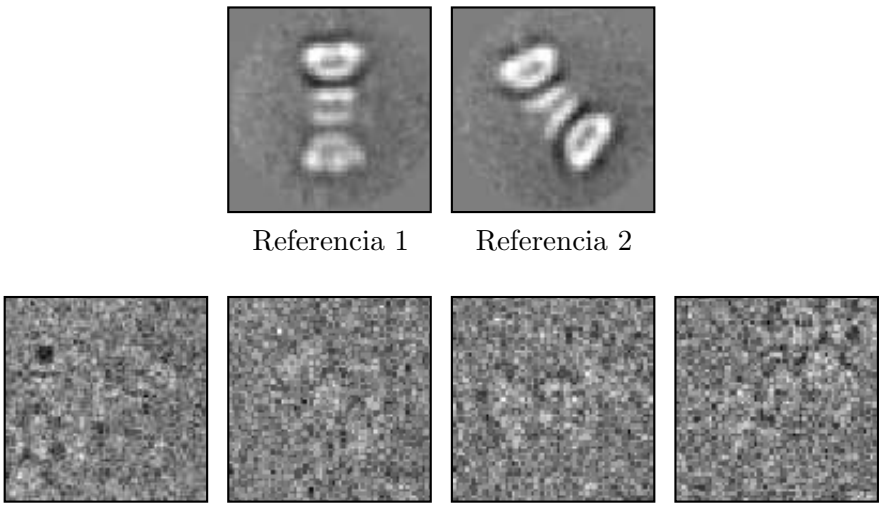


Figura 5.24: Imágenes experimentales de microscopía electrónica de 64 x 64 píxeles. En la parte superior se pueden observar las dos imágenes que se emplean como referencia. Las imágenes de la parte inferior son cuatro de las 1.670 imágenes experimentales obtenidas con un microscopio electrónico.

de referencia iniciales que han sido generadas a través del promediado de un subconjunto aleatorio de las imágenes experimentales. En cada iteración, el programa alinea cada imagen experimental con la referencia más parecida. Cuando todas las imágenes experimentales han sido alineadas se realiza un promediado de estas, obteniendo así las referencias de la siguiente iteración. La forma final de las dos imágenes de referencia empleadas en la prueba se ha conseguido tras 41 iteraciones. *MLalign2D* genera en cada iteración un fichero de texto con información relativa a los alineamientos. Este contiene para cada imagen experimental su referencia asignada así como la traslación y rotación que el programa ha encontrado óptimas para alinear dicha imagen.

La prueba realizada consiste en enfrentar todas las imágenes experimentales con cada una de las imágenes de referencia. Los resultados son a continuación contrastados con el fichero que contiene la información del alineamiento de las imágenes para la iteración 41 de *MLalign2D*. Se considera que una imagen está bien alineada cuando su error de ajuste no supera el 5 % del tamaño de imagen. En este caso se tienen imágenes de 64 x 64, siendo el máximo error admisible 3,2 píxeles. Es importante tener en cuenta que al trabajar con imágenes experimentales no existe una alineación perfecta y se asume como correcta la que nos proporcionan las clases iniciales alineadas por el Dr. Sjors. Los datos de rotación y traslación con los que se contrastan los alineamientos son los considerados óptimos por *MLalign2D*. Esto puede resultar en la inclusión de cierto grado de error en el ajuste.

La máxima traslación que se supone que estas imágenes experimentales tienen es inferior a siete píxeles según el fichero de alineamiento de *MLalign2D*. Esté será el valor de ρ_{max} empleado para FRM2D. Dado que el valor de ρ_{max} hace que varíe el patrón de puntos traslaciones producido por FBM, con este método se prueban algunos valores para ver las diferencias de error. Los ρ_{max} probados son 7, 12, 16 y 32.

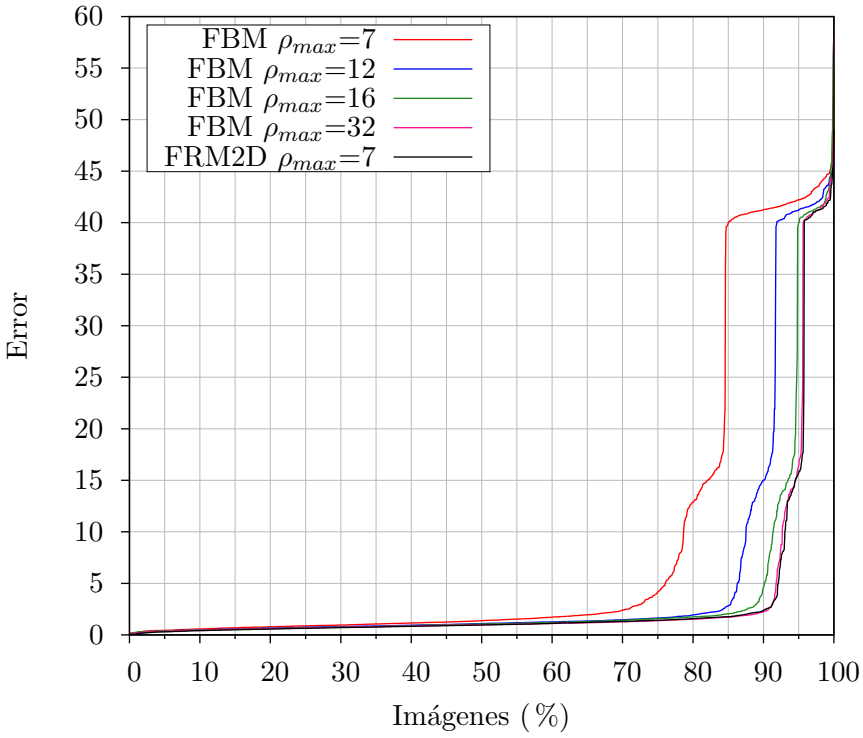


Figura 5.25: Error en píxeles cometido en el ajuste de las imágenes experimentales de microscopía electrónica de 64 x 64 píxeles. En el eje de abscisas se encuentran las diferentes imágenes del conjunto de pruebas. Se muestra el error correspondiente a FRM2D con ρ_{max} = 7 y a FBM empleando ρ_{max} = 7, ρ_{max} = 12, ρ_{max} = 16 y ρ_{max} = 32.

La Figura 5.25 muestra el error en píxeles cometido para cada una de las 1.670 imágenes del conjunto experimental. El error se muestra ordenado de menor a mayor para facilitar la lectura de la gráfica. Este se calcula a través de la Ecuación 5.1, igual que para la validación con imágenes simuladas. Como se puede observar, el error cometido es bastante elevado empleando un ρ_{max} = 7 con FBM. Tan sólo el 73,05 % de las imágenes se alinean con error inferior a 3,2 píxeles. Ampliando el rango de escaneo en cinco píxeles, hasta ρ_{max} = 12, se muestra una cantidad de error menor, alineando el 85,39 % de las imágenes con un error inferior a 3,2 píxeles. Según se aumenta el rango de escaneo del método, el error se va reduciendo. Esto se produce ya que al aumentar el rango de escaneo no sólo aumenta la cantidad de puntos traslacionales explorados, sino que además la densidad de puntos cercanos al centro de las imágenes es mayor. Empleando ρ_{max} = 16 se obtiene un 89,34 % de las imágenes alineadas con un error inferior a 3,2 píxeles. Las líneas de error correspondientes a FRM2D con ρ_{max} = 7 y FBM con ρ_{max} = 32 prácticamente se superponen. La cantidad de imágenes alineadas con un error inferior a 3,2 píxeles es del 91,38 % para ambos métodos.

A continuación se analizan para ambos métodos las tasas de imágenes para las que se identifica su referencia correcta en este conjunto de prueba. La Tabla 5.20 muestra los resultados obtenidos. Como se puede observar, según aumenta el rango de escaneo traslacional en FBM la tasa de acierto se incrementa. El motivo es el mismo que se detallaba anteriormente: al incrementar el rango de escaneo, la cantidad de puntos traslacionales explorados aumenta,

	ρ_{max}	TASA DE ACIERTO
FBM	7	81,5 %
	12	88,92 %
	16	92,27 %
	32	93,47 %
FRM2D	7	93,71 %

Tabla 5.20: Tasas de acierto para FRM2D y para los diversos valores de configuración de ρ_{max} en FBM. Los valores corresponden a la prueba de alineamiento con imágenes de microscopía de 64 x 64 píxeles.

incrementando así la precisión del método. Todas las tasas de acierto de FBM son elevadas con este conjunto de pruebas, sin embargo es necesario utilizar un ρ_{max} = 32 para alcanzar una tasa de acierto similar a la obtenida por FRM2D.

Se han registrado también los tiempos de ejecución de ambos métodos para esta prueba. El motivo de hacer esto es que el tamaño de imagen utilizado es, junto con el ancho de banda y el rango de escaneo traslacional, uno de los factores que modifica las dimensiones de las estructuras de datos generadas en FBM. Esto se traduce en unos tiempos promedio diferentes para cada tamaño de imagen, por lo que los tiempos registrados para FBM en esta prueba no coinciden con los de la anterior. Este efecto es inherente al diseño de FBM, por lo que no se da en FRM2D. Los únicos factores que modifican el tiempo por ajuste de FRM2D son el ancho de banda y el rango de escaneo traslacional. Las pruebas se han realizado sobre un procesador Intel i7-950 empleando un sólo núcleo. En la Tabla 5.21 se muestran los resultados de tiempo registrados. Se puede observar que con este tamaño de imagen (64 x 64 píxeles) FBM tarda casi seis veces menos que FRM2D con la misma configuración de ρ_{max}. Sin embargo, este reducido tiempo se obtiene a costa de utilizar un patrón de posiciones traslacionales que no dispone de suficientes puntos como para proporcionar una buena precisión. Empleando un ρ_{max} = 16 con FBM se obtienen unos tiempos de ejecución similares a los registrados para FRM2D con ρ_{max} = 7. FRM2D es ligeramente más preciso en esta configuración. Emplear ρ_{max} = 32 en FBM produce una alta cantidad de puntos traslacionales que mejoran la precisión del alineamiento en un 1,2 % con respecto a ρ_{max} = 16, sin embargo se producen unos tiempos de ajuste muy altos. Con esta configuración, FRM2D alinea 4,57 veces más rápido que FBM.

En resumen, para este tamaño de imagen (64 x 64 píxeles) se pueden obtener alineamientos con un error muy similar en tiempos muy parecidos con ambos métodos, siendo un poco más preciso FRM2D. Esto se logra configurando FBM con ρ_{max} = 16. FRM2D obtiene de este modo una tasa de acierto sólo 1,44 % más elevada que FBM tardando alrededor de 1 ms más por cada alineamiento realizado.

El siguiente conjunto de pruebas corresponde a un ejemplo más actual que el anterior [7]. Está compuesto por imágenes de mejor calidad obtenidas mediante un microscopio electrónico de última generación. El microscopio empleado es un FEI Titan Krios operando a 300 kV. Este conjunto de pruebas consiste en un grupo de 6.906 imágenes experimentales y dos imágenes de

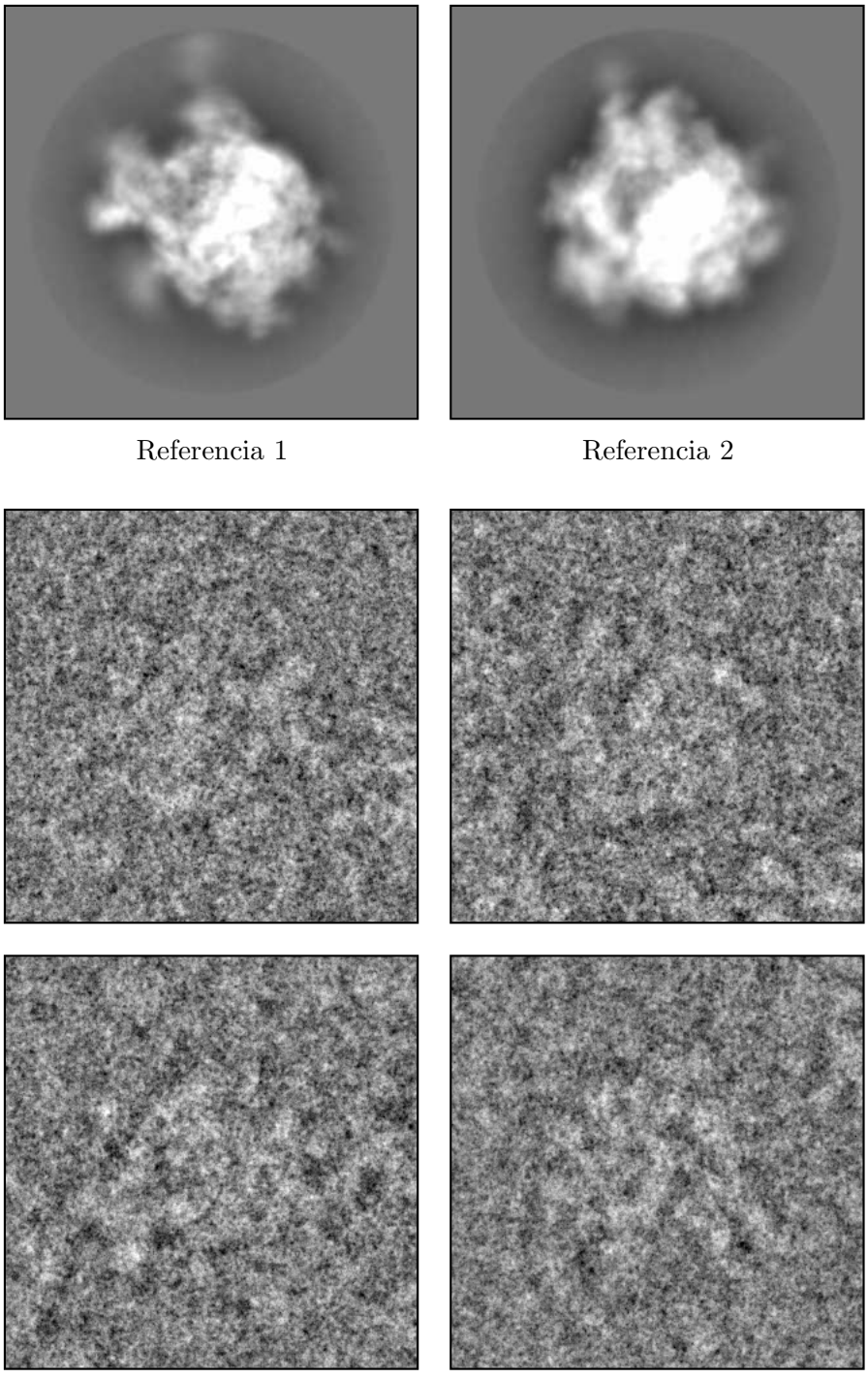


Figura 5.26: Imágenes experimentales de microscopía electrónica de 320 x 320 píxeles. En la parte superior se pueden observar las dos imágenes que se emplean como referencia y que corresponden a clases promediadas. Las imágenes de la parte inferior son 4 de las 6.906 imágenes experimentales. Las dos de la izquierda corresponden a la referencia 1 y las dos de la derecha a la referencia 2.

	ρ_{max}	TOTAL	POR AJUSTE
FBM	7	11,44 s	3,426 ms
	12	34,06 s	10,198 ms
	16	63,23 s	18,933 ms
	32	301,73 s	90,338 ms
FRM2D	7	65,97 s	19,753 ms

Tabla 5.21: Tiempos de ejecución para la comparativa mediante imágenes experimentales. Ambos métodos se ejecutan en una CPU Intel i7 950. El tamaño de las imágenes es de 64 x 64 píxeles. El ancho de banda empleado es de 64.

referencia. El tamaño de estas imágenes es de 320 x 320 píxeles, un tamaño 25 veces mayor que las imágenes experimentales de la anterior prueba. Actualmente los microscopios electrónicos generan grandes volúmenes de datos ya que se ha mejorado la resolución de sus detectores. Algunas de las imágenes de este conjunto se pueden observar en la Figura 5.26. La obtención de las imágenes de referencia se ha realizado a través de un proceso similar a la obtención de las imágenes de referencia para la prueba anterior. La diferencia principal se encuentra en que para este conjunto de pruebas se ha empleado el software de procesamiento de datos de microscopía RELION [188, 189].

El fichero de desplazamientos proporcionado por RELION indica que la cantidad de traslación máxima desde el centro de las imágenes es 28,49 píxeles. El valor seleccionado para ρ_{max} en estas pruebas es de 30, dejando disponible un ligero margen de 1,5 píxeles. No es conveniente utilizar un valor menor de ρ_{max} ya que esto produciría cierto error en los alineamientos al quedar algunas imágenes fuera del rango de exploración. La variable que se explora en esta prueba es el ancho de banda. Se comienza empleando un ancho de banda pequeño (64) para compensar el gran rango de escaneo traslacional. Los alineamientos de las imágenes se consideran correctos si el error producido es inferior a 16 píxeles. Este valor se corresponde con el 5 % del tamaño de imagen.

En la Figura 5.27 se muestran los resultados de precisión obtenidos para ambos métodos utilizando los anchos de banda 64 y 128. Las dos líneas correspondientes a FBM se encuentran prácticamente superpuestas. El error producido usando el ancho de banda 128 es ligeramente menor que utilizando 64, pero la diferencia es tan reducida que se considera despreciable. Empleando el ancho de banda de 64, FBM obtiene un 99,42 % de alineamientos con un error inferior al umbral fijado en 16 píxeles. Incrementar el ancho de banda hasta 128 tan sólo hace que esta tasa mejore en un 0,1 %. Esto indica que con imágenes de esta resolución la mejora en la precisión es escasa según se incrementa el ancho de banda. Con respecto a FRM2D, utilizando un ancho de banda de 64 se logra alinear el 83,26 % de las imágenes con un error inferior al umbral fijado. Doblando el ancho de banda hasta 128 la precisión aumenta ligeramente, logrando un 2,78 % más de imágenes con un error inferior a 16 píxeles que con el ancho de banda de 64. Se han desestimado anchos de banda superiores en FRM2D debido a su elevado coste computacional.

A continuación se analiza la tasa de imágenes experimentales correctamente emparejadas

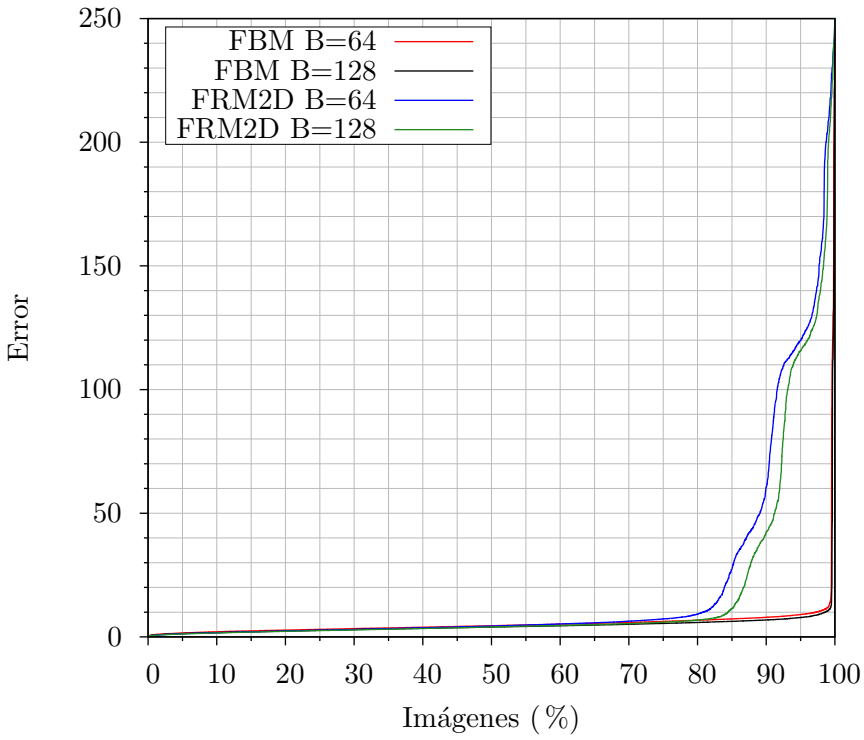


Figura 5.27: Error en píxeles cometido en el ajuste de imágenes experimentales de microscopía electrónica de 320 x 320 píxeles. Se muestra el error producido en FBM utilizando los anchos de banda 64 y 128. Se muestra también el error en FRM2D para los anchos de banda 64 y 128. El valor de ρ_{max} se fija a 30.

con sus respectivas referencias para esta prueba. La Tabla 5.22 muestra esta información. La diferencia en la precisión entre ambos métodos es considerable utilizando este conjunto de pruebas. FBM obtiene unas tasas de acierto muy elevadas y empareja correctamente un 21,14 % más de imágenes que FRM2D utilizando el ancho de banda 64. Al aumentar el ancho de banda a 128 la cantidad de aciertos logrados por FRM2D aumenta en un 3,79 %.

A continuación se examina el error cometido en los subconjuntos de imágenes cuya referencia se ha identificado correctamente. Esto permite observar mejor la precisión del ajuste, suponiendo que los datos de traslación y rotación de referencia son correctos. FBM produce alineamientos con un error inferior a 16 píxeles para el 99,96 % y 99,97 % de las imágenes correctamente identificadas empleando los anchos de banda de 64 y 128 respectivamente. La correspondiente

	B	TASA DE ACIERTO
FBM	64	99,45 %
	128	99,52 %
FRM2D	64	78,31 %
	128	82,10 %

Tabla 5.22: Tasas de acierto para FRM2D y FBM correspondientes a la prueba de alineamiento con imágenes de microscopía de 320 x 320 píxeles. En todos los casos se utiliza ρ_{max} = 30.

	B	TOTAL	POR AJUSTE
FBM	64	35 s	2,59 ms
	128	8 m 19 s	36,17 ms
FRM2D	64	1 h 18 m 10 s	339,57 ms
	128	5 h 5 m 55 s	1,33 s

Tabla 5.23: Tiempos de ejecución para la comparativa mediante imágenes experimentales. Ambos métodos se ejecutan en una CPU Intel i7 950. El tamaño de las imágenes es de 320 x 320 píxeles. El valor de ρ_{max} empleado es de 30.

cantidad en FRM2D es del 99,96 % para ambos anchos de banda. Estos resultados indican que la precisión de FRM2D es alta. El problema por el que se obtiene una cantidad elevada de error con este método para alrededor del 18 % de las imágenes experimentales de este conjunto de pruebas es porque estas se asignan a la referencia incorrecta. De este modo, los valores de traslación y rotación calculados se comparan con datos de referencia que no son los correctos.

Por último se detallan en la Tabla 5.23 los tiempos registrados para esta prueba en cada uno de los métodos con todas las configuraciones probadas. El tiempo base contra el que se compara FBM con respecto a FRM2D es el correspondiente al ancho de banda 128 ya que es para el que se obtienen los resultados más precisos. FBM configurado con el ancho de banda de 64 realiza los ajustes 512,72 veces más rápido que FRM2D con ancho de banda de 128. Seleccionando en ambos métodos un ancho de banda de 128, FBM se ejecuta 36,77 veces más rápido que FRM2D. Sin embargo, la mejora de precisión de FBM utilizando un ancho de banda de 128 frente a 64 es tan sólo del 0,1 %, por lo que no merece la pena la pérdida de rendimiento.

En resumen, con las imágenes experimentales de alta resolución (320 x 320 píxeles) que proporcionan los microscopios electrónicos actuales FBM funciona de un modo más preciso y rápido que FRM2D. Utilizando en cambio imágenes experimentales más pequeñas (64 x 64 píxeles) es posible configurar FBM para obtener una precisión similar a FRM2D empleando aproximadamente el mismo tiempo por alineamiento. Siempre que en ambos métodos se utilice el mismo rango de exploración, FBM es más eficiente que FRM2D con respecto al tiempo de ejecución independientemente del ancho de banda empleado.

5.3. Validación de FRODRUG

A lo largo de esta sección se muestran las pruebas realizadas y los resultados obtenidos para FRODRUG. En primer lugar se realiza una prueba que mide la precisión del ajuste molecular (*docking*) con ligandos individuales. A continuación se valida el método en un contexto de *virtual screening* a través de dos *benchmarks*: DUD y DUD-E. Más adelante se muestran los resultados de las pruebas de rendimiento del método, tanto en CPU como en GPU. Posteriormente se analizan los *kernels* desarrollados para la paralelización del método en GPU. Por último se realiza una comparativa de FRODRUG frente a otros métodos de ajuste molecular y *virtual screening*.

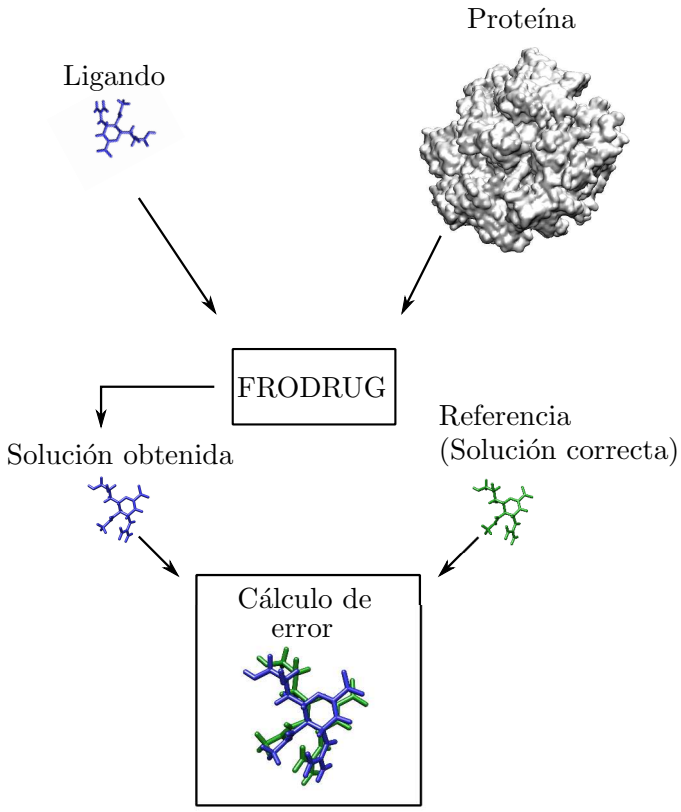


Figura 5.28: Proceso del ajuste molecular a través de complejos proteína-ligando empleando el conjunto de pruebas Astex.

5.3.1. Prueba de precisión en el *docking* con Astex

La precisión del ajuste molecular se lleva a cabo empleando el *benchmark* Astex. Este es un conjunto de pruebas de validación para métodos de *docking* [77]. Está compuesto por un conjunto de 85 complejos proteína-ligando. El *benchmark* proporciona la información para cada uno de los complejos sobre la correcta posición y orientación de cada ligando sobre su proteína. El objetivo de este *benchmark* es proporcionar a los métodos de *docking* un conjunto de datos de prueba lo más diverso posible, con complejos relevantes para el descubrimiento de nuevos fármacos. De los 85 ligandos que el *set* contiene, 23 de ellos son fármacos aprobados y 6 han estado en ensayo clínico, mientras que otros 35 se han originado a partir de proyectos de descubrimiento de fármacos. El resto de los ligandos está compuesto por sustratos naturales y elementos relacionados. El conjunto de pruebas proporciona estructuras cristalográficas experimentales de alta calidad. La resolución de todas las estructuras es de al menos 2,5 Å. Para evitar solapamiento de complejos proteína-ligando con otros conjuntos de pruebas, las estructuras seleccionadas son relativamente recientes. La estructura más antigua data del 11 de Agosto de 2000. De este modo se puede usar de forma independiente a otros conjuntos de pruebas anteriores.

El *benchmark* proporciona dos tipos de ficheros: `.pdb` y `.mol2`. Cada fichero `.pdb` contiene la estructura cristalográfica de una proteína en el formato de *Protein Data Bank* [163]. Cada fichero `.mol2` contiene el ligando extraído de dicha estructura cristalográfica. Las posiciones tridimensionales de los átomos en ambos ficheros están dispuestas según el mismo origen de

coordenadas, de este modo es posible conocer la posición y orientación correctas de cada ligando en el entorno de su correspondiente proteína. FRODRUG es capaz de leer directamente los ficheros `.pdb`, pero no los `.mol2`. Antes de realizar las pruebas es necesario convertir los nombres de los átomos de estos ficheros a otros nombres que FRODRUG sea capaz de interpretar. Para lograr esto se hace uso de la herramienta `fconv` [143]. El siguiente paso es preparar los mapas de potenciales de *Van der Waals* y electrostático. Para esto se emplea la herramienta `frodockgrid` perteneciente a FRODOCK [62]. Este procedimiento se repite para cada uno de los 85 ligandos del *benchmark*.

La prueba llevada a cabo consiste en realizar un ajuste molecular ciego de cada complejo proteína-ligando de forma individual e independiente. A continuación se mide el error obtenido entre la solución proporcionada por FRODRUG y la de referencia. El objetivo es que FRODRUG logre identificar de forma precisa el lugar de unión de la proteína y colocar su ligando en la posición y orientación correctas. La Figura 5.28 muestra un esquema explicativo de este proceso de validación.

La ejecución de FRODRUG para un complejo proteína-ligando dado produce un listado de 15 soluciones para el ligando con sus correspondientes posiciones, orientaciones y puntuación. La posición relativa de estas soluciones se indica a través de tres coordenadas de posición en el espacio y tres ángulos que indican la rotación del ligando. La calidad de las soluciones se mide a través del RMSD (*root mean square deviation*). El RMSD es la distancia promedio, medida en Angstroms, entre los átomos del ligando orientado según las soluciones obtenidas y el ligando de referencia. El RMSD se calcula como

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i^2} \tag{5.3}$$

siendo δ_i la distancia en Angstroms entre N parejas diferentes de átomos, que se calcula como:

$$\delta_i = \sqrt{(x_i^{REF} - x_i^{SOL})^2 + (y_i^{REF} - y_i^{SOL})^2 + (z_i^{REF} - z_i^{SOL})^2} \tag{5.4}$$

siendo las coordenadas REF las correspondientes a la solución de referencia y las SOL las correspondientes a la obtenida por FRODRUG. Se considera que FRODRUG ha colocado correctamente un ligando en su lugar si el RMSD obtenido es igual o inferior a 2.0 Å.

FRODRUG se puede configurar para almacenar varias soluciones por cada posición traslacional del sitio de unión. Se ha realizado la prueba con este *benchmark* almacenando una y tres soluciones por posición como se verá a continuación. El hecho de almacenar las n mejores soluciones por cada punto traslacional tiene un pequeño impacto en el resultado final ya que sobre las soluciones almacenadas se realiza un proceso de refinamiento posterior.

La Figura 5.29 muestra, para cada uno de los 85 complejos proteína-ligando, el RMSD medido en Angstroms entre la mejor solución obtenida y la de referencia. En la gráfica se muestran los datos obtenidos empleando una y tres soluciones por posición traslacional. El promedio de RMSD obtenido en la mejor solución para los 85 casos es de 0,87 Å y 0,73 Å empleando una y tres soluciones por posición traslacional respectivamente. Como se puede observar en la figura, al

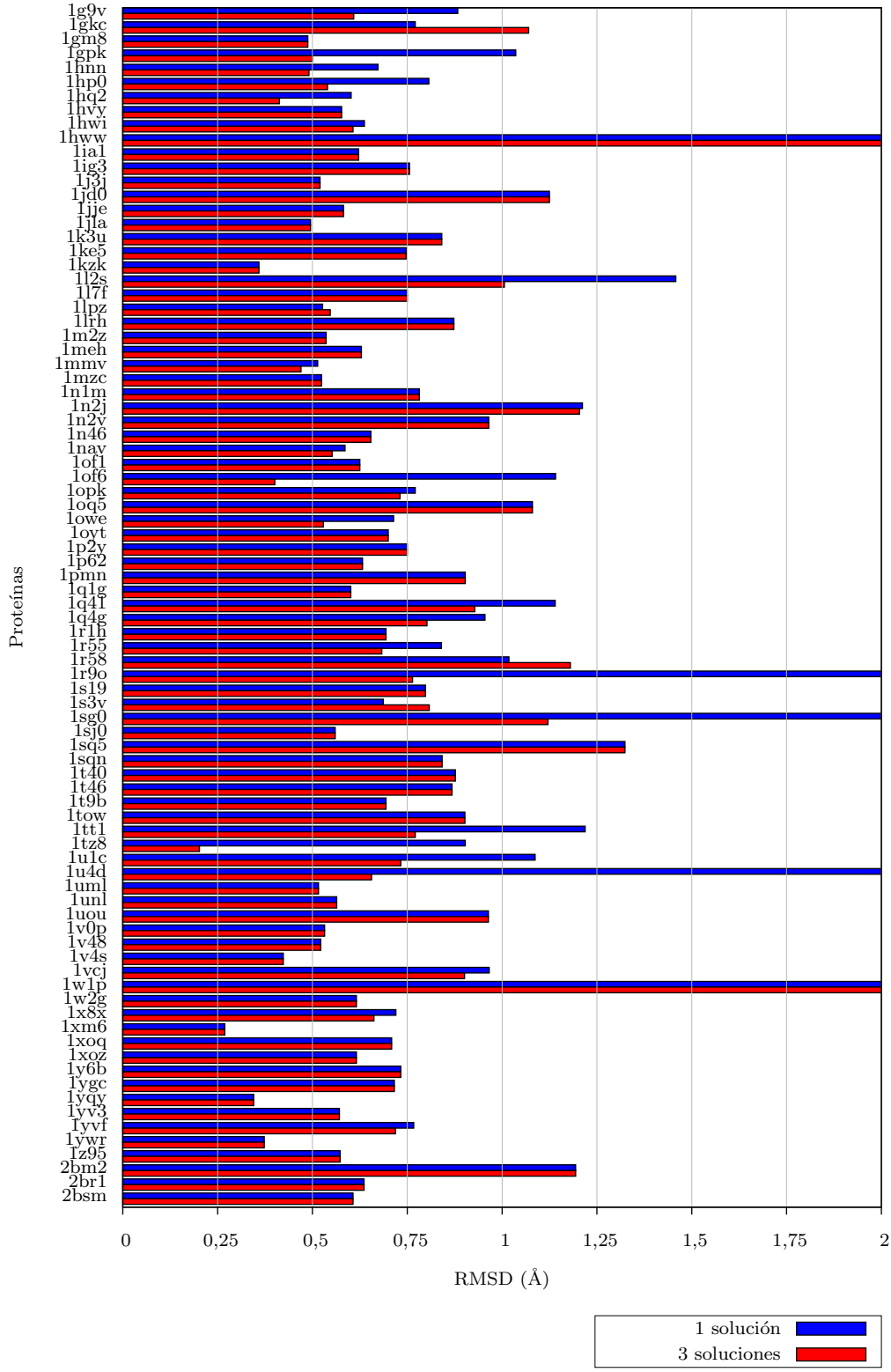


Figura 5.29: RMSD calculado para los ligandos de cada una de las proteínas del *benchmark* Astex según los resultados proporcionados por FRODRUG sobre CPU almacenando una y tres soluciones por punto.

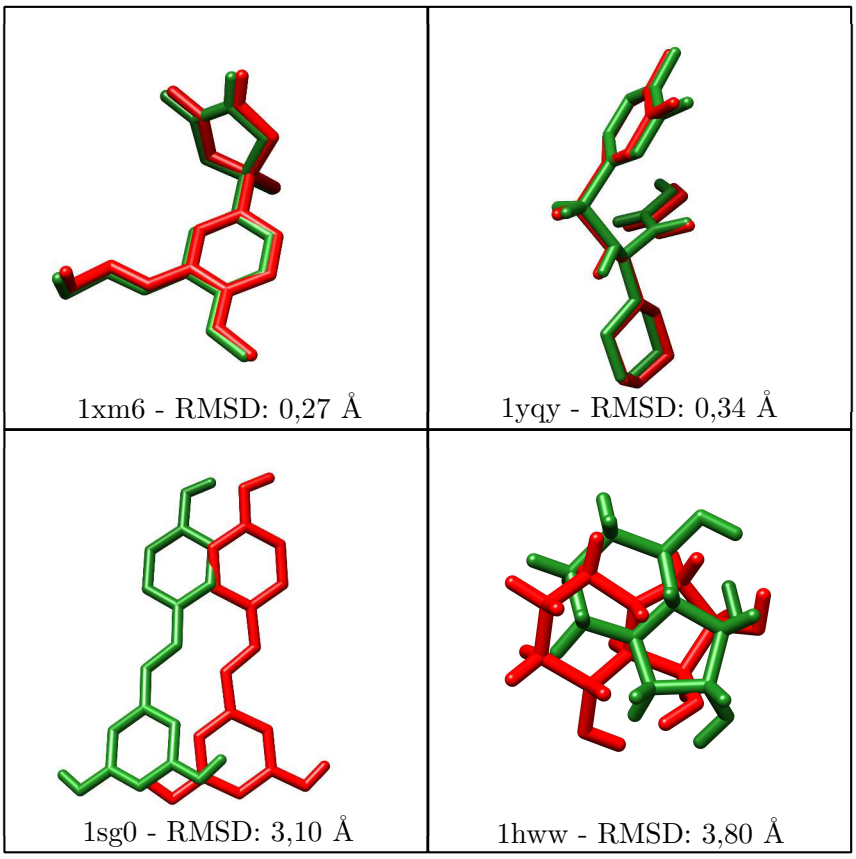


Figura 5.30: Soluciones obtenidas por FRODRUG para algunos complejos proteína-ligando de Astex. Los ligandos de referencia colocados y orientados en sus correctas posiciones se muestran en verde. Las soluciones obtenidas por FRODRUG se muestran en color rojo. En la parte superior de la figura se pueden ver dos de los complejos para los que se logra un ajuste excelente. En la parte inferior se muestran los dos complejos del *benchmark* para los que se realiza el peor ajuste.

emplear una solución por posición, para 80 de los 85 ligandos (94,12 % del conjunto) se logra realizar un ajuste molecular correcto. El error obtenido para estos 80 ligandos es inferior a 1,5 Å. Los cinco ligandos para los que no se logra realizar un ajuste correcto muestran unos errores de 2,1 Å (1w1p), 2,6 Å (19ro), 3,0 Å (1u4d), 3,1 Å (1sg0) y 3,8 Å (1hww). La Figura 5.30 muestra el resultado obtenido para dos complejos cuyo ajuste es muy preciso y otros dos para los que el ajuste se ha realizado de forma incorrecta. El ligando de referencia se muestra en verde y el obtenido por FRODRUG en rojo. En uno de los complejos para los que el ajuste se realiza mal (1sg0) se consigue que el ligando tenga la rotación correcta pero la posición traslacional incorrecta. En el otro ejemplo con ajuste erróneo (1hww) la traslación está cerca de ser correcta y la rotación es totalmente incorrecta.

Si se realiza el ajuste molecular almacenando tres soluciones por posición traslacional el resultado mejora en algunos casos. De este modo, en 83 de los 85 casos (97,65 %) se realiza el ajuste molecular correctamente con un error inferior a 1,5 Å. En los dos casos restantes (1w1p y 1hww) no se obtienen resultados correctos. El error obtenido en estos dos casos es de 2,1 Å y 2,7 Å respectivamente. Nótese que incluso en estos casos el RMSD se encuentra cerca del límite.

Por último se analiza la diferencia en el RMSD producida entre el empleo de una y tres

soluciones por posición traslacional teniendo en cuenta sólo los 80 complejos cuyo ajuste se considera correcto. De este subconjunto, el 67,5 % de los ligandos se ajustan con el mismo error independientemente de la cantidad de soluciones por posición traslacional empleadas. En el 27,5 % de los casos se produce un error menor al emplear tres soluciones por punto. La diferencia de error se desglosa del siguiente modo: alrededor de 0,75 Å para dos casos (1of6 y 1tz8), aproximadamente 0,5 Å para cuatro casos (1gpk, 1l2s, 1tt1 y 1u1c) y menor a 0,25 Å para los 16 casos restantes. De los 80 casos analizados, el 5 % (cuatro casos) obtienen un error más bajo cuando se emplea una solución por punto que cuando se emplean tres. En estos casos la diferencia es menor a 0,25 Å para tres de ellos (1lpz, 1r58 y 1s3v) y ligeramente mayor de 0,25 Å para el restante (1gkc).

Con esta prueba se demuestra que el hecho de utilizar tres soluciones por punto traslacional en lugar de una produce no sólo tres aciertos más, sino que también se consigue mejorar la precisión en el ajuste. Utilizando la versión secuencial del método sobre una CPU Intel i7-950, el tiempo total de cómputo almacenando una solución por punto es de 3 horas y 20 minutos. Almacenar tres soluciones por posición traslacional aumenta este tiempo de cómputo en tan sólo 33 segundos, lo que constituye un porcentaje muy pequeño. Esto se debe a que independientemente de la cantidad de soluciones almacenadas por cada punto del sitio de unión, se tiene un límite máximo de soluciones guardadas por ligando. De entre todas las soluciones generadas, sólo se guardan las 50 mejores y se desechan las demás. Así, el tiempo empleado para revaluar las soluciones a través de la función de puntuación basada en conocimiento es siempre constante.

En resumen, se ha logrado llevar a cabo de forma exitosa el ajuste molecular utilizando una solución por punto traslacional para el 94,12 % de los complejos proteína-ligando del *benchmark* Astex. Almacenando tres soluciones por punto traslacional se mejoran ligeramente los resultados, ajustando correctamente el 97,65 % de los complejos del *benchmark*.

Mediante este conjunto de pruebas se demuestra que el proceso básico en el cribado virtual de proteínas, el ajuste molecular, funciona correctamente en FRODRUG.

5.3.2. Validación con el conjunto de pruebas DUD

DUD es el acrónimo de *Directory of Useful Decoys* [85]. DUD es un *benchmark* cuyo propósito es determinar la eficacia de métodos de *virtual screening*. Esta clase de conjuntos de pruebas están compuestos de dos tipos de elementos: proteínas diana (*targets*) y ligandos. Los ligandos se pueden subdividir en dos grupos: principios activos e inactivos (*decoys*). Los principios activos de una proteína son los ligandos que se conoce que se unen a ella, interaccionando de algún modo en su función. Por otra parte, los inactivos son ligandos que se sabe que no se unen a dicha proteína.

El proceso de validación con este *benchmark*, ilustrado en la Figura 5.31, consiste en que para cada uno de los *targets*, FRODRUG sea capaz de diferenciar los principios activos de los inactivos. Las pruebas para llevar a cabo este propósito radican en realizar un test ciego de cribado virtual de cada *target* con sus correspondientes ligandos y *decoys*. La salida obtenida es un archivo en el que se indican las 15 mejores soluciones (traslación y rotación) para cada uno de los ligandos de entrada así como su puntuación final. De este fichero se selecciona la

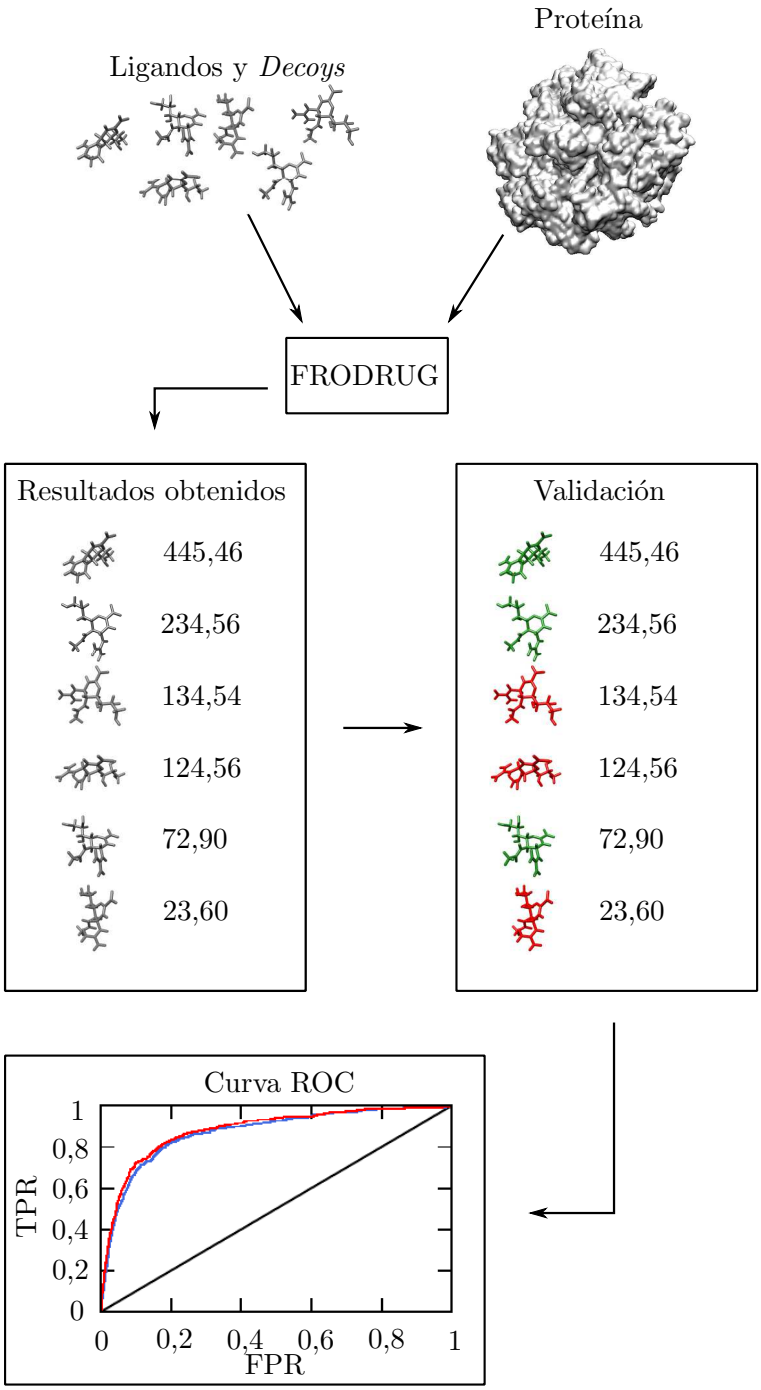


Figura 5.31: Esquema de validación de FRODRUG mediante DUD. FRODRUG lee la proteína, los ligandos y los *decoys*. Se obtiene una lista de resultados ordenados por puntuación. Por último se identifican en el listado de resultados los principios activos (verde) y los *decoys* (rojo) para generar una curva ROC.

mejor solución para cada ligando y el resultado se ordena por puntuación. A partir de estos resultados se realiza una curva ROC (*Receiver Operating Characteristic*). Una figura de curva ROC es aquella que enfrenta la tasa de verdaderos positivos (*True Positive Rate*) contra la tasa de falsos positivos (*False Positive Rate*). Estas se definen como:

$$\begin{aligned} TPR &= TP/P \\ FPR &= FP/N \end{aligned} \tag{5.5}$$

siendo TP los verdaderos positivos encontrados, P la cantidad total de positivos, FP los falsos positivos encontrados y N la cantidad total de negativos. Cada uno de los ejes de una curva ROC varía entre 0,0 (0 %) y 1,0 (100 %), indicando la cantidad de ligandos o *decoys* que se han encontrado con respecto a la cantidad total de estos. La gráfica se genera leyendo la lista de resultados ordenados por puntuación obtenida con FRODRUG. Cuando se encuentra un principio activo, la curva se incrementa en el eje de ordenadas. Al encontrar un *decoy* entre los resultados, la curva se incrementa en el eje de abscisas. La eficacia del proceso de cribado virtual se evalúa a través del cálculo del área bajo la curva (AUC). En el escenario ideal en el que las mejores puntuaciones son asignadas a los principios activos, el área bajo la curva sería de 1,0. El peor caso es en el que todos los *decoys* obtienen mejor puntuación que los compuestos activos siendo el área bajo la curva de 0,0. La gráfica incluye además una línea diagonal a modo de referencia con un AUC de 0,5 que es la que se generaría si se seleccionasen los ligandos de forma aleatoria.

Los *decoys* de este *benchmark* se seleccionan atendiendo a diversas propiedades físico-químicas tales como carga y peso moleculares, cantidad de uniones rotatorias y otras propiedades. De este modo se consigue que estos *decoys* se asemejen químicamente a los ligandos que se unen. A pesar de tener características moleculares similares, se seleccionan *decoys* cuya topología sea lo más diferente posible a la de los principios activos para minimizar la posibilidad de que puedan unirse a la proteína. Por brevedad, los principios activos se denominarán simplemente ligandos y los inactivos *decoys*.

Este conjunto de pruebas ha sido creado con el fin de aumentar y mejorar las opciones disponibles para la validación de métodos de *virtual screening*. Otros conjuntos de pruebas anteriores [166, 131] introducen algunos errores en la selección de *decoys* haciendo que el resultado del cribado se vea mejorado artificialmente. DUD trata de evitar los errores cometidos en otros *benchmarks* para proporcionar una validación más fiable. Algunos de estos errores radican en grandes diferencias en las propiedades físicas entre los ligandos y los *decoys*. Una de estas propiedades es por ejemplo su tamaño. Si se emplean *decoys* cuyo tamaño es muy diferente al de los principios activos se facilita en gran medida la tarea del método de cribado virtual. En cambio, al utilizar sólo *decoys* cuyas propiedades físicas sean similares a las de los principios activos se aumenta la dificultad en el proceso de cribado. De este modo se obtiene un *benchmark* más robusto.

La base de datos de DUD proporciona las estructuras de 40 *targets*. Casi todas ellas (38 de 40) son estructuras cristalográficas extraídas del *Protein Data Bank* [14]. Para dos de los *targets* (*pdgfrb* y *vegfr2*) no se dispone de estructura cristalográfica. La estructura de *pdgfrb*

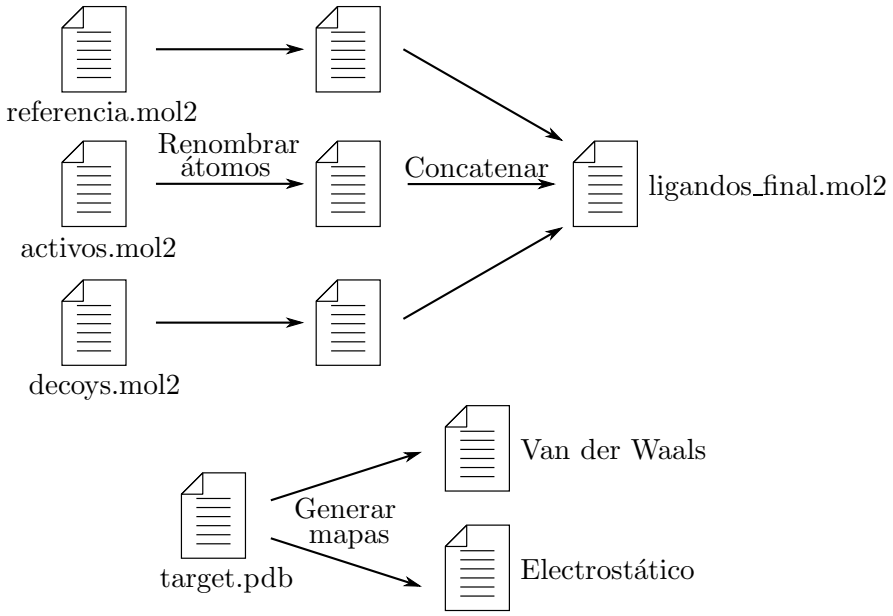


Figura 5.32: Preparación de los ficheros de DUD para poder ser empleados con FRODRUG.

que se emplea en DUD ha sido modelada por homología. En el caso de *vegfr2* se dispone de una estructura sin principio activo unido a ella. Para el propósito del conjunto de pruebas, este principio activo se obtiene también a través de homología. Los nombres de todos los *targets* están indicados en la Tabla 5.24. Para cada uno de ellos se indica además un acrónimo que se utilizará para su identificación. En total, la base de datos cuenta con 2.950 ligandos. Todos ellos han sido obtenidos de la base de datos de compuestos ZINC [90, 91]. No todos los *targets* disponen de la misma cantidad de ligandos sino que su número varía entre 12 y 416. En promedio se dispone de 74 ligandos por proteína y por cada ligando se proporcionan 36 *decoys*. El total de ligandos inactivos en la base de datos asciende a 95.316. Esta cantidad es menor a la esperada ya que algunos de estos *decoys* se utilizan para varios *targets*.

Antes de poder emplear el conjunto de pruebas, este debe ser previamente preparado. DUD proporciona para cada una de las proteínas varios ficheros empleando los formatos `.mol2` para los principios activos y *decoys*, un fichero `.pdb` conteniendo la estructura cristalográfica de la proteína y otro `.mol2` que contiene un ligando de referencia extraído dicha estructura. El diagrama general de la preparación de los ficheros se puede observar en la Figura 5.32. Antes de poder cargar los ficheros `.mol2` es necesario realizar un cambio de los nombres en los átomos para que FRODRUG los pueda interpretar adecuadamente. Esto se realiza mediante la herramienta `fconv` [143] del mismo modo que para los ligandos en Astex.

Empleando `fconv` con el fichero de principios activos, *decoys* y ligando de referencia se obtienen los correspondientes ficheros que FRODRUG puede leer. El siguiente paso para preparar los datos es concatenar estos ficheros. Los archivos `.mol2` contienen texto indicando las posiciones de los átomos y las uniones entre ellos. Tienen también unas etiquetas para indicar los puntos de comienzo de cada una de sus secciones. Por lo tanto es posible realizar una concatenación mediante la herramienta `cat` y obtener un fichero de ligandos válido para

ACRÓNIMO	NOMBRE
ace	*Angiotensin-converting enzyme*
ache	*Acetylcholine esterase*
ada	*Adenosine deaminase*
alr2	*Aldose reductase*
ampc	*AmpC beta lactamase*
ar	*Androgen receptor*
cdk2	*Cyclin dependent kinase 2*
comt	*Catechol O-methyltransferase*
cox1	*Cyclooxygenase 1*
cox2	*Cyclooxygenase 2*
dhfr	*Dihydrofolate reductase*
egfr	*Epidermal growth factor receptor kinase*
er_agonist	*Estrogen receptor agonist*
er_antagonist	*Estrogen receptor antagonist*
fgfr1	*Fibroblast growth factor receptor kinase*
fxa	*Factor Xa*
gart	*glycinamide ribonucleotide transformylase*
gpb	*Glycogen phosphorylase beta*
gr	*Glutocorticoid receptor*
hivpr	*HIV protease*
hivrt	*HIV reverse transcriptase*
hmgr	*Hydroxymethylglutaryl-CoA reductase*
hsp90	*Human heat shock protein 90 kinase*
inha	*Enoyl ACP reductase*
mr	*Mineralcorticoid receptor*
na	*Neuraminidase*
p38	*P38 mitogen activated protein kinase*
parp	*Poly(ADP-ribose) polymerase*
pde5	*Phosphodiesterase V*
pdgfrb	*Platlet derived growth factor receptor kinase*
pnp	*Purine nucleoside phosphorylase*
pparg	*Peroxisome proliferator activated receptor gamma*
pr	*Progesterone receptor*
rxra	*Retinoic X receptor alpha*
sahh	*S-adenosyl-homocysteine hydrolase*
src	*Tyrosine kinase SRC*
thrombin	*Thrombin*
tk	*Thymidine kinase*
trpsin	*Trypsin*
vegfr2	*Vascular endothelial growth factor receptor kinase*

Tabla 5.24: La tabla muestra la correspondencia de cada *target* empleado en DUD con su acrónimo. Estos acrónimos se utilizan para nombrar las curvas ROC de este *benchmark*.

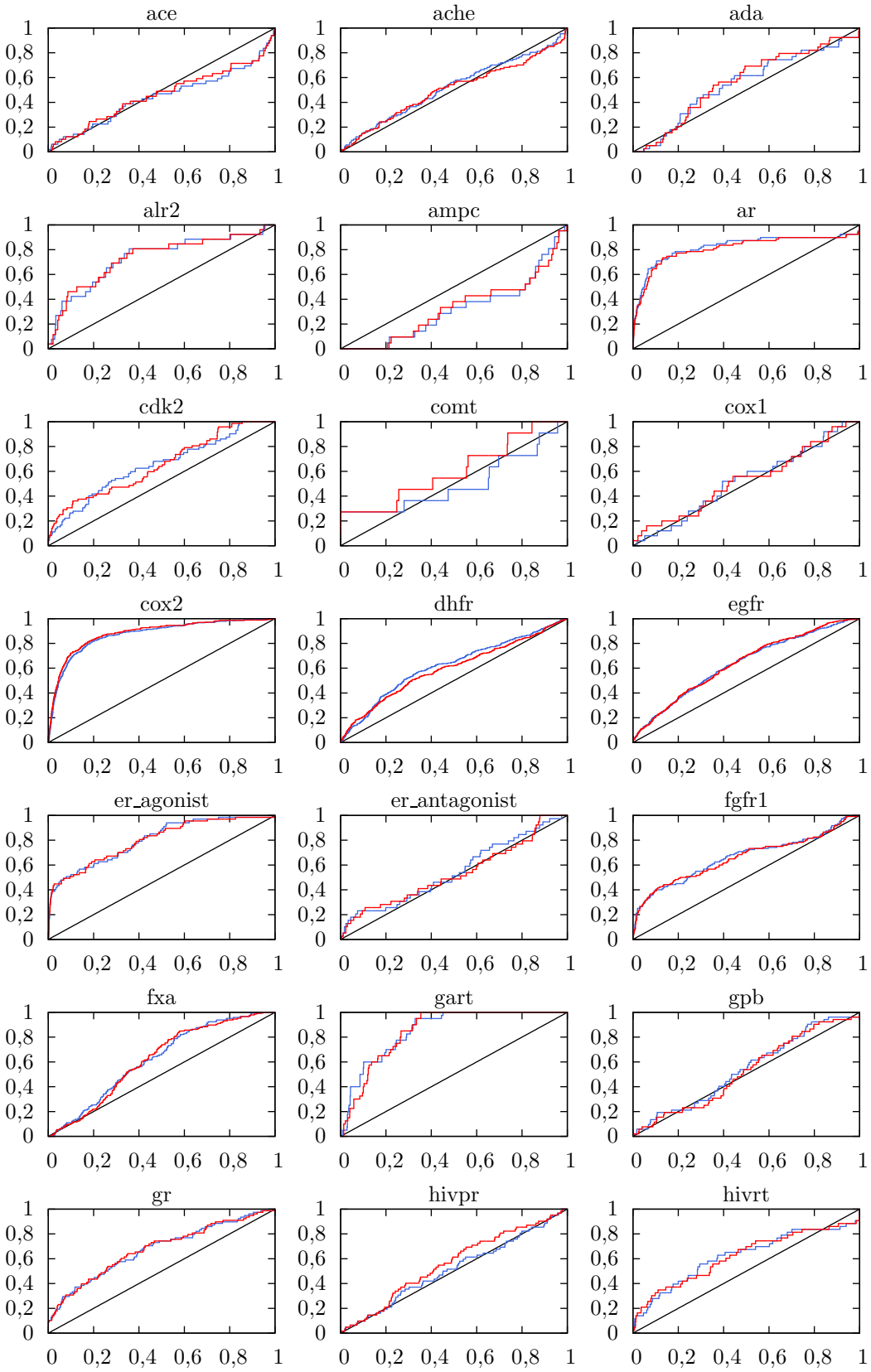
emplearlo en FRODRUG. La primera entrada de este fichero es el ligando de referencia extraído del PDB, seguido de todos los principios activos, y por último de todos los *decoys*. El orden en el que los ligandos están colocados es muy importante ya que será de utilidad al analizar los resultados posteriormente.

El siguiente paso es preparar los mapas de potenciales de *Van der Waals* y electrostático. Para esto se emplea la herramienta `frodockgrid` perteneciente a FRODOCK [62].

Una vez preparados todos los datos se procede a realizar la validación. Por cada *target* del conjunto de pruebas se genera una curva ROC. Estas se muestran en la Figura 5.33. El cribado virtual se ha llevado a cabo utilizando varias soluciones por punto para cada uno de los *targets*. Las curvas obtenidas utilizando una solución por punto se muestran en rojo mientras que las producidas utilizando tres se ilustran en azul. En principio deben obtenerse curvas por encima de la diagonal para que el proceso de cribado sea mínimamente positivo. La Tabla 5.25 muestra las áreas bajo la curva para cada uno de los *targets*.

El área bajo la curva calculada como promedio entre los 40 *targets* usando tanto una como tres soluciones por punto es de 0,63. Esto confirma que FRODRUG es capaz de seleccionar con mayor probabilidad los principios activos sobre los *decoys*, demostrando que cumple su función.

Las curvas obtenidas son buenas (área $\geq$ 0,7) en 12 de los 40 casos, lo que supone el 30 % del conjunto de pruebas. En el 60 % de los casos (24 de 40) se obtiene una curva aceptable. Estas curvas tienen un área igual o superior a 0,5 y menor a 0,7. Por último, existen cuatro casos (*ace*, *ampc*, *na* y *trpsin*) cuya área bajo la curva es pobre (área < 0,5). La peor curva se obtiene con *ampc* (AUC=0,316 con una solución por punto). Este *target* es un caso reconocido como muy problemático para los métodos actuales de virtual screening [33]. El ajuste molecular sobre esta proteína con el conjunto de principios activos y *decoys* de DUD es muy dependiente de la función de puntuación empleada así como del uso de flexibilidad en ligandos [24]. La Figura 5.34 muestra una sección del *target ace* junto con su principio activo de referencia extraído de la estructura cristalográfica. Este ligando se encuentra colocado en la posición y orientación en las que se une a dicha proteína. Como se puede observar, el lugar de unión está ubicado en una zona profunda de la proteína y la cavidad es muy estrecha. Al realizar un ajuste molecular con FRODRUG utilizando sólo el principio activo de referencia, se obtienen su posición y orientación correctas. La solución obtenida presenta un RMSD de 0,83 Å con respecto a su posición adecuada. El hecho de que se logre colocar el ligando de referencia correctamente, pero la curva ROC no sea especialmente buena, apunta a que este error se produce porque FRODRUG no soporta flexibilidad. El ligando de referencia se encuentra en la conformación adecuada para unirse a la proteína, sin embargo no es el caso del resto de principios activos para dicha proteína. Cualquier ligando que en principio pueda unirse será desechado si su estructura inicial no es capaz de ajustarse a la cavidad comprendida en el sitio de unión, incluso aunque con otra conformación sea capaz de entrar en la cavidad. Esto dificulta en gran medida la correcta detección de los principios activos. En los *targets na* y *trpsin* el sitio de unión se encuentra en su superficie. FRODRUG localiza correctamente este lugar y coloca adecuadamente los ligandos originales de referencia con un RMSD de 1,05 Å para *na* y de 0,79 Å para *trpsin*. Al igual que en el caso de *ace*, la estrechez de estos sitios de unión junto a la ausencia de flexibilidad hace que la detección


ace
ache
ada
alr2
ampc
ar
cdk2
comt
cox1
cox2
dhfr
egfr
er_agonist
er_antagonist
fgfr1
fxa
gart
gpb
gr
hivpr
hivrt
0
0,2
0,4
0,6
0,8
1

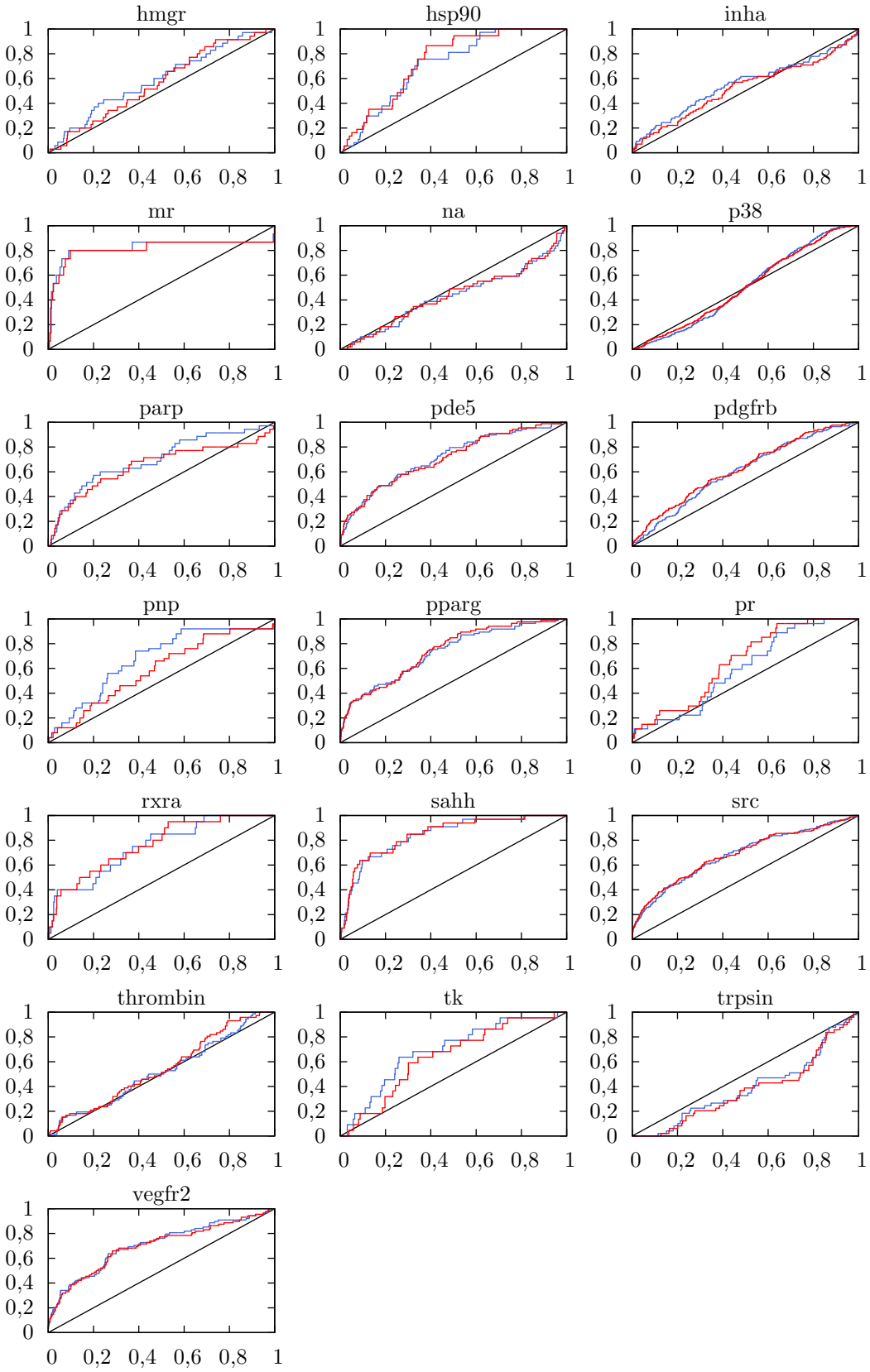


Figura 5.33: Curvas ROC para *targets* de DUD en CPU utilizando una (azul) y tres (rojo) soluciones por punto.

TARGET	AUC 1SPP	AUC 3SPP	DIFERENCIA
ace	0,444	0,461	1,7 %
ache	0,520	0,503	1,7 %
ada	0,549	0,565	1,6 %
alr2	0,734	0,735	0,1 %
ampc	0,316	0,328	1,2 %
ar	0,822	0,809	1,3 %
cdk2	0,647	0,651	0,4 %
comt	0,504	0,605	10,1 %
cox1	0,522	0,528	0,6 %
cox2	0,870	0,882	1,2 %
dhfr	0,619	0,596	2,3 %
egfr	0,632	0,637	0,5 %
er_agonist	0,802	0,799	0,3 %
er_antagonist	0,547	0,534	1,3 %
fgfr1	0,663	0,658	0,5 %
fxa	0,621	0,618	0,3 %
gart	0,854	0,848	0,6 %
gpb	0,540	0,523	1,7 %
gr	0,675	0,678	0,3 %
hivpr	0,513	0,560	4,7 %
hivrt	0,622	0,620	0,2 %
hmgr	0,594	0,568	2,6 %
hsp90	0,712	0,740	2,8 %
inha	0,552	0,521	3,1 %
mr	0,821	0,812	0,9 %
na	0,433	0,439	0,6 %
p38	0,506	0,509	0,3 %
parp	0,700	0,650	5 %
pde5	0,714	0,709	0,5 %
pdgfrb	0,604	0,625	2,1 %
pnp	0,676	0,590	8,6 %
pparg	0,728	0,741	1,3 %
pr	0,597	0,653	5,6 %
rxra	0,751	0,766	1,5 %
sahh	0,842	0,848	0,6 %
src	0,683	0,687	0,4 %
thrombin	0,522	0,542	2 %
tk	0,689	0,640	4,9 %
trpsin	0,391	0,374	1,7 %
vegfr2	0,708	0,700	0,8 %

Tabla 5.25: Área bajo la curva de las curvas ROC para los *targets* de DUD.

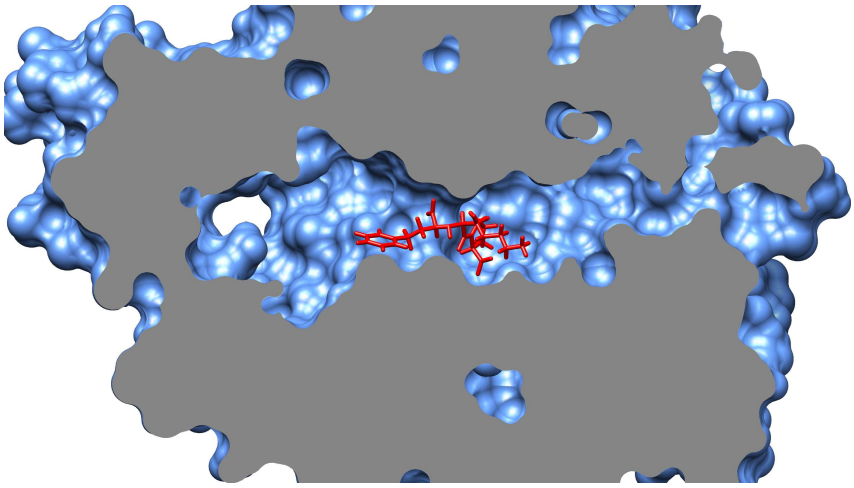


Figura 5.34: Sección del *target ace* de DUD. Su ligando de referencia está ilustrado en color rojo.

de sus principios activos resulte difícil.

La diferencia entre ejecutar FRODRUG empleando una o tres soluciones por punto es muy pequeña. En cuanto a la precisión entre estas dos configuraciones, en el 92,5 % de los *targets* (37 de 40) se presenta una diferencia en el área bajo la curva menor al 5 %. De este subconjunto de 37 *targets*, 19 de ellos obtienen un área mayor si se almacenan tres soluciones por punto que si se almacena una. En los tres casos restantes la diferencia entre áreas bajo la curva es algo más grande. Estas diferencias son de 5,6 %, 8,6 % y 10,1 % para los *targets pr*, *pnp* y *comt* respectivamente. En el caso de *comt* y *pr*, el área bajo la curva obtenida es mejor cuando se emplean tres soluciones por punto. Sin embargo, emplear esta misma cantidad de soluciones por punto empeora el área obtenida para *pnp*. Esto está producido por el efecto de la revaluación de las soluciones a través de la función de puntuación basada en conocimiento. Al aplicar esta función a las tres mejores soluciones obtenidas, en ocasiones la segunda o tercera solución se convierte en la primera. Esto no siempre produce un efecto positivo, como se puede observar en la curva de *pnp* (Figura 5.33). En este caso algunos *decoys* ascienden en puntuación debido a esto. La diferencia en área se hace más notable cuantos menos principios activos posea un *target*. Un claro ejemplo es el *target comt* con tan sólo once principios activos. De este modo se producen diferencias en el área al haber algunos intercambios en el orden de la lista de soluciones. Sin embargo, como se ha comentado anteriormente, las diferencias empleando una o tres soluciones en general no son significativas.

Por último se muestra en la Figura 5.35 un ejemplo de los resultados de ajuste molecular que FRODRUG proporciona para algunos de los principios activos identificados correctamente de tres *targets* de DUD. Estos son *cox2*, *alr2* y *vegfr2*. Para cada uno de ellos se muestra tanto en la parte izquierda como en la derecha de la figura su ligando de referencia en color verde. En color azul se muestran seis principios activos distintos; dos por cada *target*. Estos están colocados y orientados según la solución proporcionada por FRODRUG. Como se puede observar existen algunas partes de los principios activos que coinciden con la estructura de los ligandos de referencia, lo cual indica que están siendo correctamente colocados.

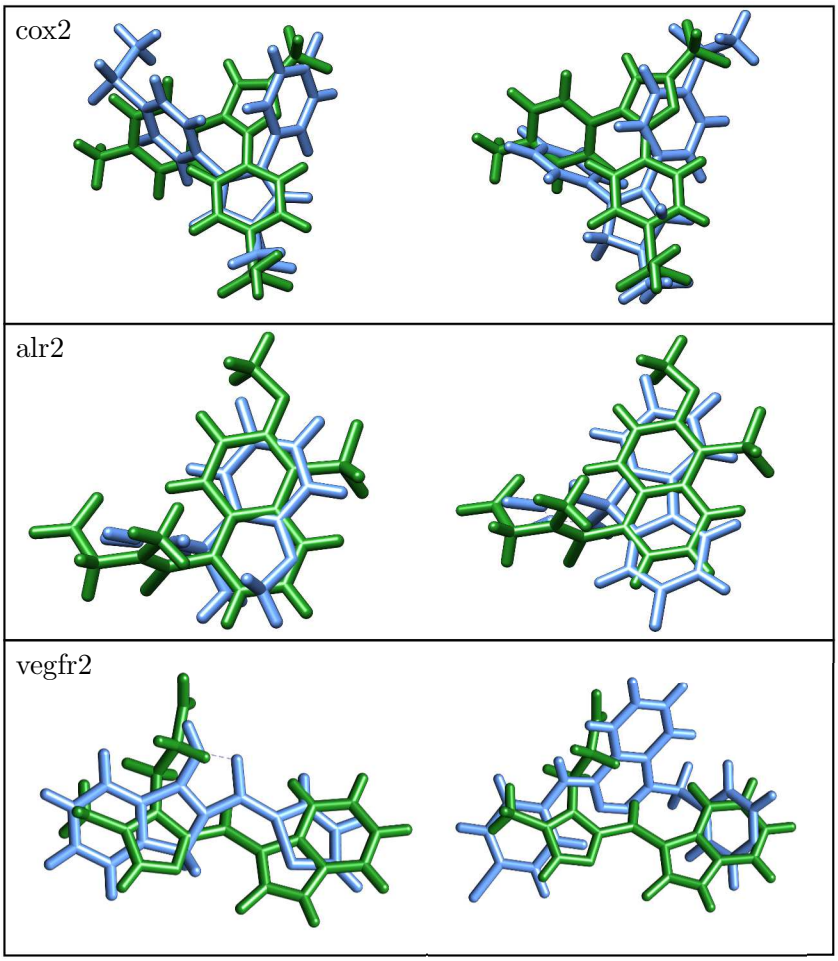


Figura 5.35: Resultados del ajuste molecular para algunos principios activos de tres targets de DUD (cox2, alr2 y vegfr2). Para cada *target* se muestra su ligando de referencia en color verde tanto en el lado izquierdo como en el derecho. Así mismo se muestra por cada *target* dos principios activos diferentes en color azul.

5.3.3. Validación con el conjunto de pruebas DUD-E

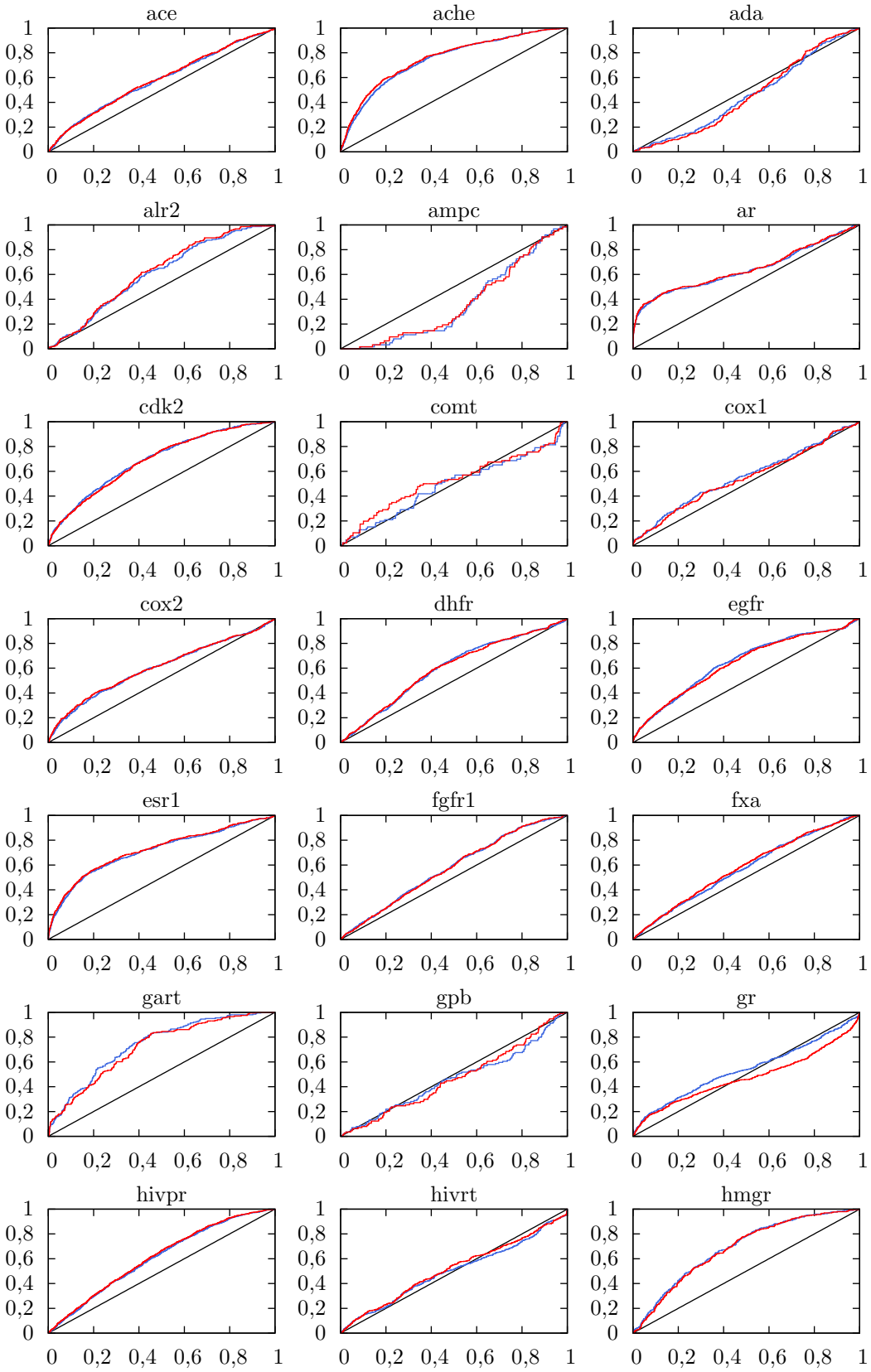
DUD-E es el acrónimo de *Directory of Useful Decoys - Enhanced*; es otro conjunto de pruebas para comprobar la robustez de métodos de *virtual screening* [138]. Esta nueva base de datos está parcialmente basada en el anterior *Directory of Useful Decoys* [85]. A través de diversas pruebas se ha encontrado que DUD tiene algunas debilidades tanto en su conjunto de principios activos como en el de *decoys* [231]. Estas deficiencias varían desde un ajuste pobre de algunas propiedades entre los principios activos y los inactivos [89, 137] hasta el hecho de que algunos *decoys* realmente se unan al *target* cuando no deberían [229]. El objetivo de DUD-E es tratar de solventar estas debilidades además de proporcionar mayor diversidad de *targets*.

DUD-E cuenta con 102 proteínas de las cuales 38 provienen de la base de datos DUD original. Para realizar esta prueba de validación de FRODRUG se emplea sólo el subconjunto de proteínas provenientes de DUD. De este modo es posible comparar los resultados de DUD-E con los obtenidos en la sección anterior.

DUD-E aumenta en gran medida la cantidad de principios activos y *decoys* proporcionados para cada uno de los *targets*. La base de datos suministra entre 100 y 600 principios activos por proteína; en promedio se dispone de 224 por cada una. Al contrario que en DUD, estos principios activos no se extraen de la base de datos ZINC sino que se obtienen de otra diferente llamada ChEMBL [63]. No obstante, los *decoys* de DUD-E siguen obteniéndose de la base de datos ZINC [90, 91]. Por cada principio activo se proporcionan de 50 *decoys*. Las proteínas de esta base de datos se han extraído del *Protein Data Bank* [14] al igual que para DUD.

Con respecto al DUD original, DUD-E realiza una pequeña alteración de la lista de *targets* disponibles. La proteína *pdgfrb* (*Platlet derived growth factor receptor kinase*) ha sido eliminada por haber sido obtenida mediante homología, mientras que las proteínas *er_agonist* y *er_antagonist* (*Estrogen receptor*) han sido unidas en un solo *target* llamado *esr1*. De este modo se reduce desde las 40 proteínas originales a las 38 que proporciona el nuevo conjunto de pruebas.

Las estructuras proporcionadas en este conjunto no se pueden emplear directamente por FRODRUG. Es necesario realizar un paso previo para preparar los datos. Este proceso es exactamente igual al que se realiza para el conjunto de pruebas DUD en el apartado anterior. De igual forma, la prueba de validación llevada a cabo es equivalente a la que se ha realizado con la versión anterior de este *benchmark*. Esta consiste en llevar a cabo un test ciego de ajuste molecular de cada *target* con su correspondiente conjunto de principios activos y *decoys*. A continuación se ordenan los resultados por puntuación y se identifican cuales son principios activos y cuales *decoys*. La Figura 5.36 muestra las curvas ROC obtenidas para DUD-E. Al igual que en la sección anterior, se emplean en la curva los colores azul y rojo para indicar la ejecución almacenando una y tres soluciones por punto traslacional respectivamente. La Tabla 5.26 muestra las correspondientes áreas bajo la curva para cada uno de los *targets* empleando ambas configuraciones así como la diferencia en área obtenida entre estas. El área bajo la curva calculada como promedio entre los 38 *targets* usando tanto una como tres soluciones por punto es de 0,62. Este área promedio es un 1 % inferior a la obtenida en DUD. Es razonable obtener un promedio de área ligeramente menor al hacer la prueba con un *benchmark* más restrictivo y robusto. Empleando tanto una solución por punto como tres, se obtienen curvas buenas (área > 0,7) para 8 de los 38 *targets* (21,05 % del conjunto). Se pueden contabilizar 25 targets con los que se produce una curva aceptable (área entre 0,5 y 0,7) si se almacena una solución por punto. Esta cifra aumenta a 26 al almacenar tres soluciones por punto. La cantidad de curvas incorrectas (área < 0,5) es de cinco si se almacena una solución por punto, y de cuatro si se almacenan tres. Existen tres *targets* cuya curva se considera aceptable utilizando una cantidad de soluciones por punto e incorrecta (por debajo de la diagonal) con la otra cantidad. Estos son *comt*, *gr* y *hivrt*. En los casos *comt* y *hivrt*, sus áreas bajo la curva calificadas como incorrectas se encuentran en el límite superior de este rango: 0,489 y 0,498 respectivamente. El tercer *target* (*gr*) obtiene un área algo más baja, 0,47, pero también cerca del umbral superior. En este caso la diferencia de área obtenida entre diferentes cantidades de soluciones por punto es más notable. Con una diferencia de 6,4 % se incrementa desde un área de 0,47 empleando tres soluciones por punto hasta una de 0,54 con una solución por punto. Con ambas cantidades de soluciones por punto, las curvas se superponen durante el primer quinto del recorrido para luego separarse. El


ace
ache
ada
alr2
ampc
ar
cdk2
comt
cox1
cox2
dhfr
egfr
esr1
fgfr1
fxa
gart
gpb
gr
hivpr
hivrt
hmgr
0
0,2
0,4
0,6
0,8
1

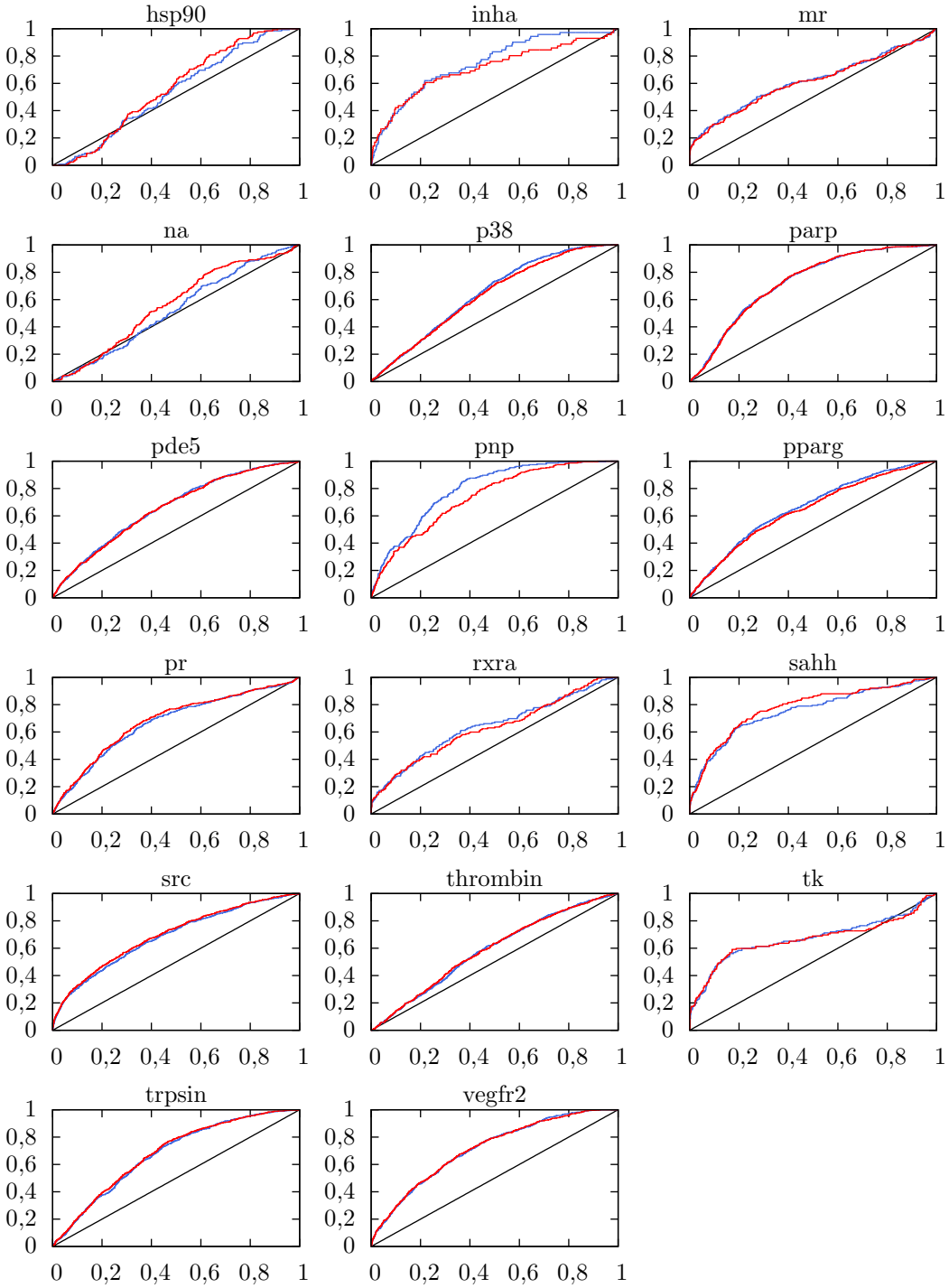


Figura 5.36: Curvas ROC para DUD-E en CPU. Las líneas azules corresponden a emplear una solución por punto; las líneas rojas corresponden a emplear tres.

TARGET	AUC 1SPP	AUC 3SPP	DIFERENCIA
ace	0,579	0,582	0,3 %
ache	0,746	0,756	1 %
ada	0,457	0,456	0,1 %
alr2	0,606	0,621	1,5 %
ampc	0,365	0,368	0,3 %
ar	0,641	0,647	0,6 %
cdk2	0,691	0,686	0,5 %
comt	0,489	0,524	3,5 %
cox1	0,557	0,544	1,3 %
cox2	0,600	0,605	0,5 %
dhfr	0,596	0,596	0 %
egfr	0,645	0,638	0,7 %
esr1	0,712	0,721	0,9 %
fgfr1	0,577	0,575	0,2 %
fxa	0,567	0,577	1 %
gart	0,737	0,717	2 %
gpb	0,463	0,468	0,5 %
gr	0,539	0,475	6,4 %
hivpr	0,605	0,613	0,8 %
hivrt	0,498	0,514	1,6 %
hmgr	0,689	0,682	0,7 %
hsp90	0,540	0,564	2,4 %
inha	0,749	0,709	4 %
mr	0,618	0,611	0,7 %
na	0,523	0,561	3,8 %
p38	0,643	0,631	1,2 %
parp	0,727	0,726	0,1 %
pde5	0,659	0,656	0,3 %
pnp	0,795	0,737	5,8 %
pparg	0,660	0,643	1,7 %
pr	0,667	0,680	1,3 %
rxra	0,639	0,627	1,2 %
sahh	0,752	0,777	2,5 %
src	0,683	0,695	1,2 %
thrombin	0,583	0,586	0,3 %
tk	0,672	0,666	0,6 %
trpsin	0,675	0,681	0,6 %
vegfr2	0,713	0,714	0,1 %

Tabla 5.26: Área bajo la curva de las curvas ROC para los *targets* de DUD-E.

TARGET	DUD	DUD-E
ache	0,52	0,76
cox2	0,88	0,60
fgfr1	0,66	0,58
gr	0,68	0,54
hivrt	0,62	0,51
hmgr	0,59	0,69
hsp90	0,74	0,56
inha	0,55	0,75
mr	0,82	0,62
trpsin	0,39	0,68

Tabla 5.27: Áreas bajo la curva obtenidas para varios *targets* en DUD y DUD-E. El área mostrada corresponde a la máxima obtenida de entre las dos configuraciones de soluciones por punto empleadas.

enriquecimiento de la gráfica es bastante bueno en este primer tramo. Cuando ha aparecido el 5 % de los *decoys* ya se ha logrado identificar el 15 % de los principios activos del *target*. Existen otros tres *targets* para los que se obtiene una curva incorrecta en ambas cantidades de soluciones por punto. Estos son *ada*, *ampc*, y *gpb*. Dos de ellos (*ada* y *gpb*) tienen un área máxima de 0,46 y 0,47 respectivamente, que son valores inferiores a los que tendría una selección al azar. Realizar un ajuste molecular sólo con el principio activo de referencia para estos dos *targets* revela que el sitio de unión se identifica correctamente. El RMSD obtenido para *ada* es de 0,82 Å mientras que el obtenido para *gpb* es de 1,47 Å. Por último, el *target ampc* repite al igual que en el test anterior con la peor área bajo la curva del conjunto de pruebas (0,37). Del mismo modo que sucedía en el *benchmark* de la sección anterior, la ausencia de la implementación de flexibilidad en ligandos hace que sea complicado identificar adecuadamente los principios activos para estas proteínas. Además, otras funciones de puntuación diferentes también podrían ayudar a la correcta identificación de estos principios activos, como en el caso de *comt*.

Al igual que para DUD, las diferencias en el área obtenidas empleando una o tres soluciones por punto son muy pequeñas. De los 38 *targets*, el 94,74 % (36 casos) presentan una diferencia entre áreas inferior al 5 %. Los dos cuyas diferencias en las áreas son superiores a este porcentaje son *gr* y *pnp* con unas diferencias de 6,4 % y 5,8 % respectivamente. Tal como se ha indicado en el párrafo anterior, para el *target gr*, seleccionar una solución por punto hace que el área obtenida pase de ser incorrecta a aceptable, aunque el enriquecimiento es bueno en ambas configuraciones. De entre todos los *targets* de este conjunto de pruebas, en 20 casos (52,63 % del conjunto) las áreas obtenidas son mejores empleando tres soluciones por punto que cuando se utiliza una. Estas tasas son muy parecidas a las obtenidas en la sección anterior, siendo esto razonable al estar DUD-E basado en el conjunto de pruebas DUD.

Prácticamente para todos los *targets* se producen curvas cuya diferencia es pequeña con respecto a las generadas para DUD. Existen no obstante ciertas proteínas para las que la diferencia es mayor. Hay cuatro casos (*ache*, *hmgr*, *inha* y *trpsin*) para los que la curva generada

en DUD-E es radicalmente mejor a la obtenida para DUD. También hay otros seis casos (*cox2*, *fgfr1*, *gr*, *hivrt*, *hsp90* y *mr*) en los que esta en DUD-E es bastante peor que en DUD. Las diferencias en el área para estos *targets* se muestran en la Tabla 5.27. El análisis de los conjuntos de principios activos y *decoys* para estos *targets* revela que prácticamente no hay intersección entre ellos. Los *targets hmgr*, *fgfr1*, *hsp90* y *mr* no comparten ningún compuesto entre los conjuntos de ambos *benchmarks*. En los casos de *hivrt*, *inha* y *ache* existe tan sólo un *decoy* que se encuentra tanto en el conjunto de DUD como en el de DUD-E. El *target gr* comparte tres *decoys* y *cox2* comparte nueve. El caso que más moléculas comparte entre las diferentes versiones del *benchmark* es *trpsin*, con 26 *decoys*. Estas cantidades son extremadamente bajas teniendo en cuenta que el conjunto de *decoys* de este último *target* contiene 26.219 moléculas para DUD-E y 1.664 en el conjunto de DUD. Examinando el resto de *targets* cuyas áreas presentan diferencias más leves se observa que tampoco hay coincidencia entre los *decoys* de ambas bases de datos. Las diferencias en las áreas bajo la curva mostradas en la Tabla 5.27 se pueden atribuir por lo tanto a la conformación en la que estén dispuestos los principios activos y los *decoys* de cada conjunto.

5.4. Análisis de rendimiento de FRODRUG

En las siguientes secciones se detallan las pruebas de rendimiento realizadas con FRODRUG. En primer lugar se examina tanto la precisión como el rendimiento en diferentes sistemas computacionales. Más adelante se realiza un estudio de eficiencia de los *kernels* desarrollados para llevar a cabo el cribado virtual en GPU. Por último se compara FRODRUG con otros métodos actuales de *docking* proteína-ligando y *virtual screening*.

5.4.1. Comparativa de CPU con diferentes sistemas

En esta sección se estudia en primer lugar el rendimiento y precisión de la implementación paralela para GPU con respecto a la versión secuencial en CPU. Para ello se realizan sobre un dispositivo gráfico las mismas pruebas llevadas a cabo en la validación de FRODRUG. Todas las pruebas realizadas en la GPU se comparan exclusivamente con los resultados obtenidos almacenando una solución por punto en CPU. Esto se debe a que la versión GPU carece de la opción de salvado de tres soluciones por posición traslacional. En segundo lugar se analiza el rendimiento y la escalabilidad de la adaptación de FRODRUG a sistemas de memoria distribuida a través de dos clústers (uno compuesto por CPUs y otro por GPUs) y dos procesadores individuales.

5.4.1.1. Comparativa con dispositivos gráficos

Todos los resultados de GPU que se muestran a continuación se han llevado a cabo sobre una tarjeta gráfica GK104. El código ha sido compilado por la versión 5.0 del compilador de NVIDIA para la arquitectura Kepler.

En primer lugar se muestra la comparativa para el conjunto de pruebas Astex de la prueba de ajuste molecular llevada a cabo en la sección 5.3.1. Este *benchmark* consiste en un conjunto de 85 complejos proteína-ligando. Cada uno de ellos proporciona la estructura cristalográfica

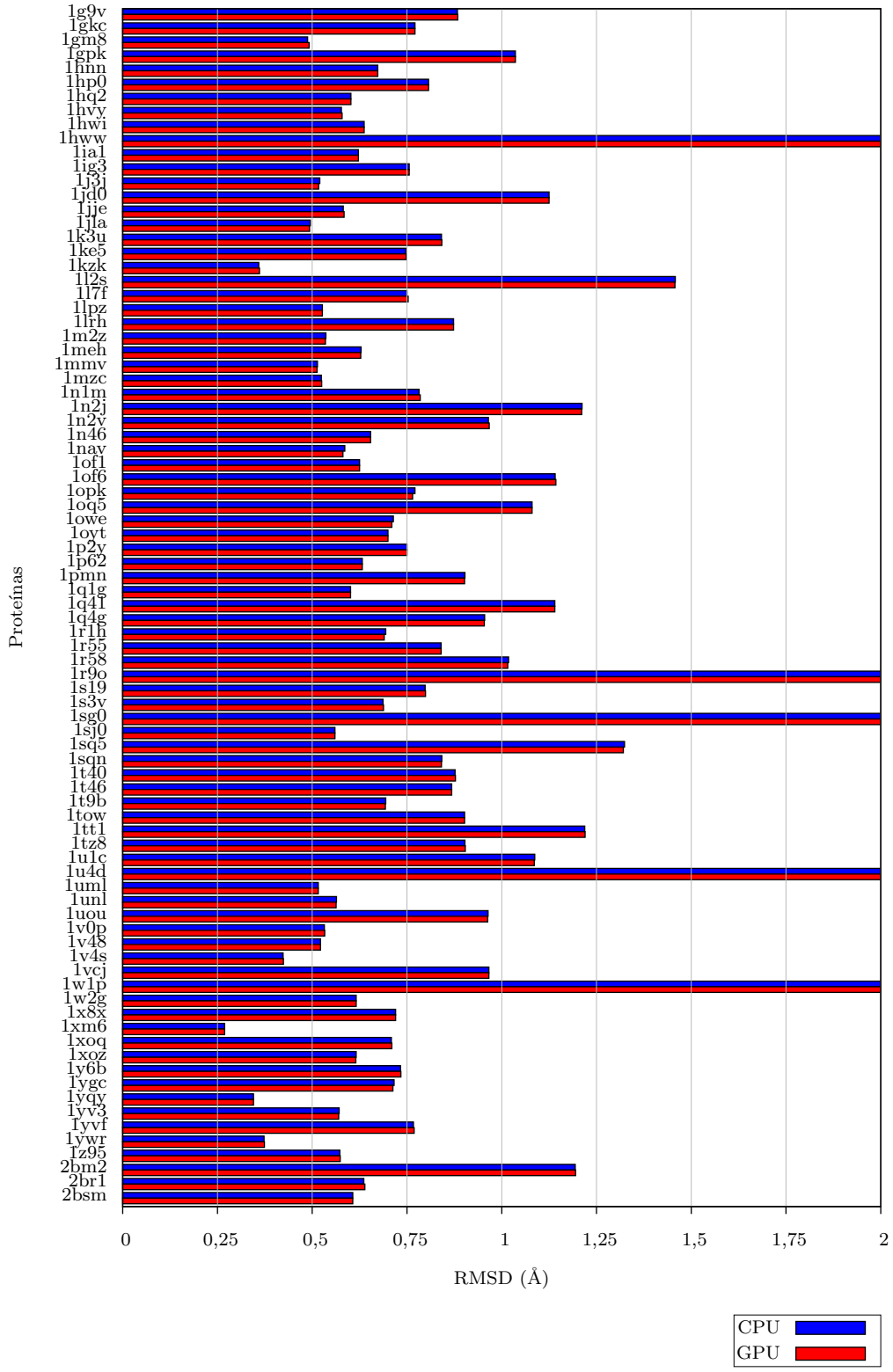


Figura 5.37: Comparativa de los RMSD obtenidos para CPU y GPU empleando una solución por punto con el *benchmark* Astex.

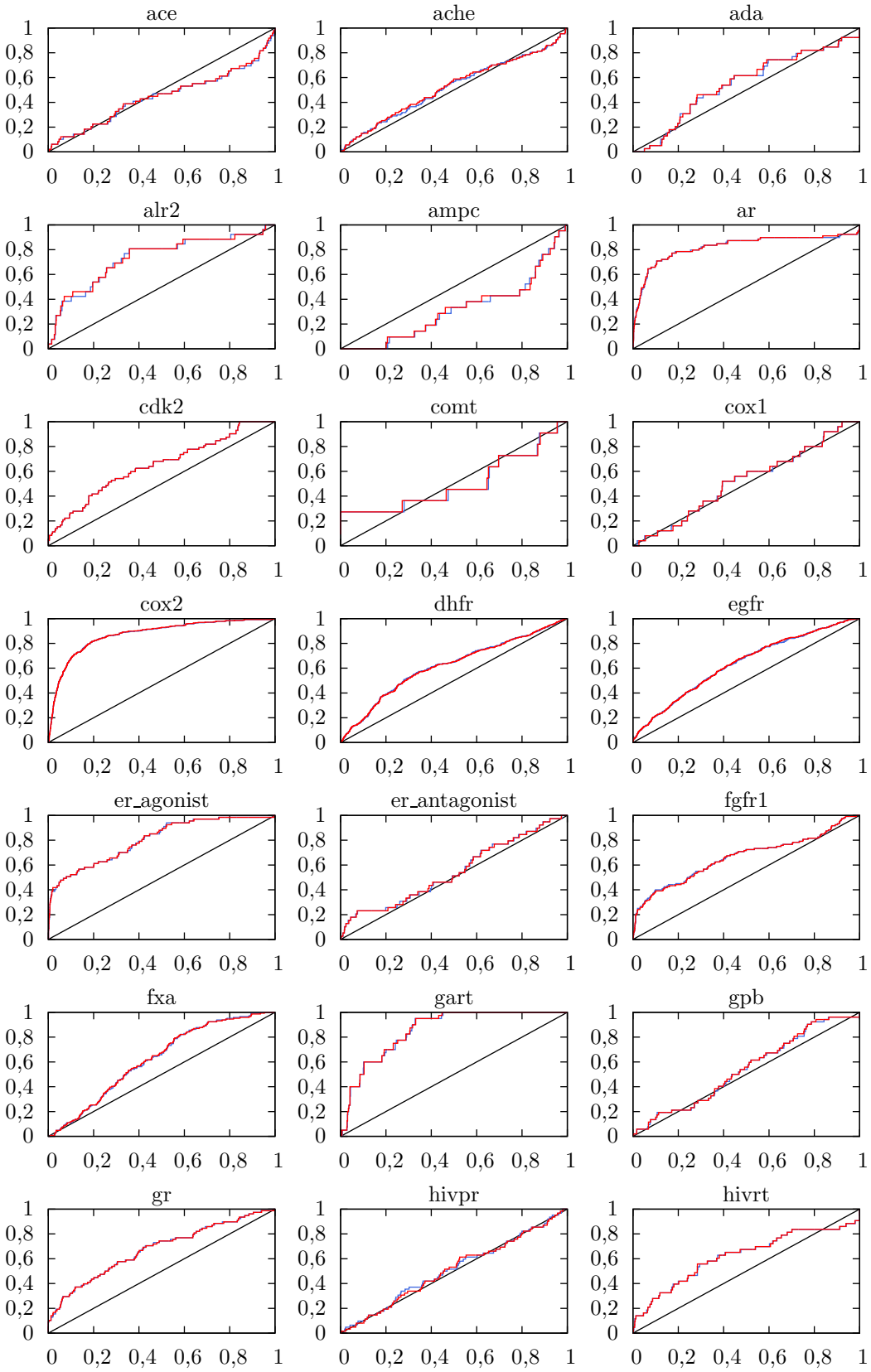
de la proteína con su respectivo ligando colocado en su conformación, posición y orientación adecuadas. De este modo es posible evaluar la calidad del ajuste molecular realizado por FRODRUG al disponer de la solución correcta con la que contrastar el resultado.

La prueba consiste en realizar el ajuste molecular de cada ligando sobre su correspondiente proteína de forma independiente a las demás. A continuación se evalúa la solución obtenida a través del RMSD con el ligando en la posición y orientación de referencia. La Figura 5.37 muestra los resultados obtenidos para esta prueba tanto en CPU como en GPU. Como se puede observar en la figura, la diferencia entre los resultados obtenidos por la CPU y la GPU es despreciable. El error máximo cometido por la GPU al realizar el ajuste molecular es de 0,006 Å con respecto a la solución proporcionada por la CPU. Entre todas las proteínas del conjunto de pruebas se calcula un error promedio de 0,001 Å. Estas diferencias tan pequeñas se consideran normales; se producen por el modo en el que se realizan los cálculos en la tarjeta gráfica. Todos los *kernels* emplean el modo más rápido pero menos preciso para trabajar en coma flotante (*fast math*).

A continuación se detallan los resultados de la comparativa entre GPU y CPU realizada a través del *benchmark* DUD. La validación de FRODRUG con DUD se ha detallado anteriormente en la sección 5.3.2. Este conjunto de pruebas ha sido diseñado para validar métodos de cribado virtual de proteínas. El *benchmark* consta de 40 proteínas. Por cada una se dispone de un conjunto de principios activos que interactúan con la proteína y un conjunto de *decoys* que no se unen a ella. Cada *target* se procesa de forma independiente a los demás, generándose una curva ROC por cada uno.

En la Figura 5.38 se muestra la comparativa de curvas ROC generadas a través de CPU (azul) y GPU (rojo) para los 40 *targets* del *benchmark* DUD. Como se puede observar, en muchas de las curvas no se puede ver la línea de CPU al estar completamente solapada por la correspondiente línea de GPU. Esto demuestra una diferencia muy reducida entre los resultados obtenidos por la versión paralela y la secuencial. Existen no obstante unos cuantos *targets* para los que se observan pequeñas discrepancias entre sus curvas ROC. Entre los casos más notables se pueden destacar *alr2*, *ampc* y *hmgr* (ver Figura 5.39). En todos los casos del conjunto de pruebas las curvas siguen un trayecto similar, indicando que en general la puntuación para cada molécula se calcula adecuadamente. Sin embargo en estos casos se puede ver que en algunos puntos las curvas de CPU y GPU se separan y a continuación vuelven a unirse. Este efecto se produce cuando dos o más ligandos con puntuaciones similares intercambian sus posiciones en el listado de resultados. En ocasiones la diferencia en la puntuación entre uno y otro ligando es una cantidad despreciable en su parte decimal. El hecho de que CPU y GPU computen los valores flotantes obteniendo diferentes resultados explica que algunos ligandos intercambien sus posiciones. En general, la diferencia entre el área bajo la curva obtenida para CPU y GPU es despreciable. La máxima diferencia de entre todos los *targets* del *benchmark* sucede en la curva del *target ache*. Se ha registrado una diferencia de 0,46 % para este caso. En promedio se calcula una diferencia de 0,17 % entre la curva de CPU y GPU para todos los casos de este conjunto de pruebas.

En último lugar se realiza la comparativa correspondiente al conjunto de pruebas DUD-E (sección 5.3.3). La Figura 5.40 muestra las curvas ROC correspondientes para este *benchmark*


ace
ache
ada
alr2
ampc
ar
cdk2
comt
cox1
cox2
dhfr
egfr
er_agonist
er_antagonist
fgfr1
fxa
gart
gpb
gr
hivpr
hivrt
0
0,2
0,4
0,6
0,8
1

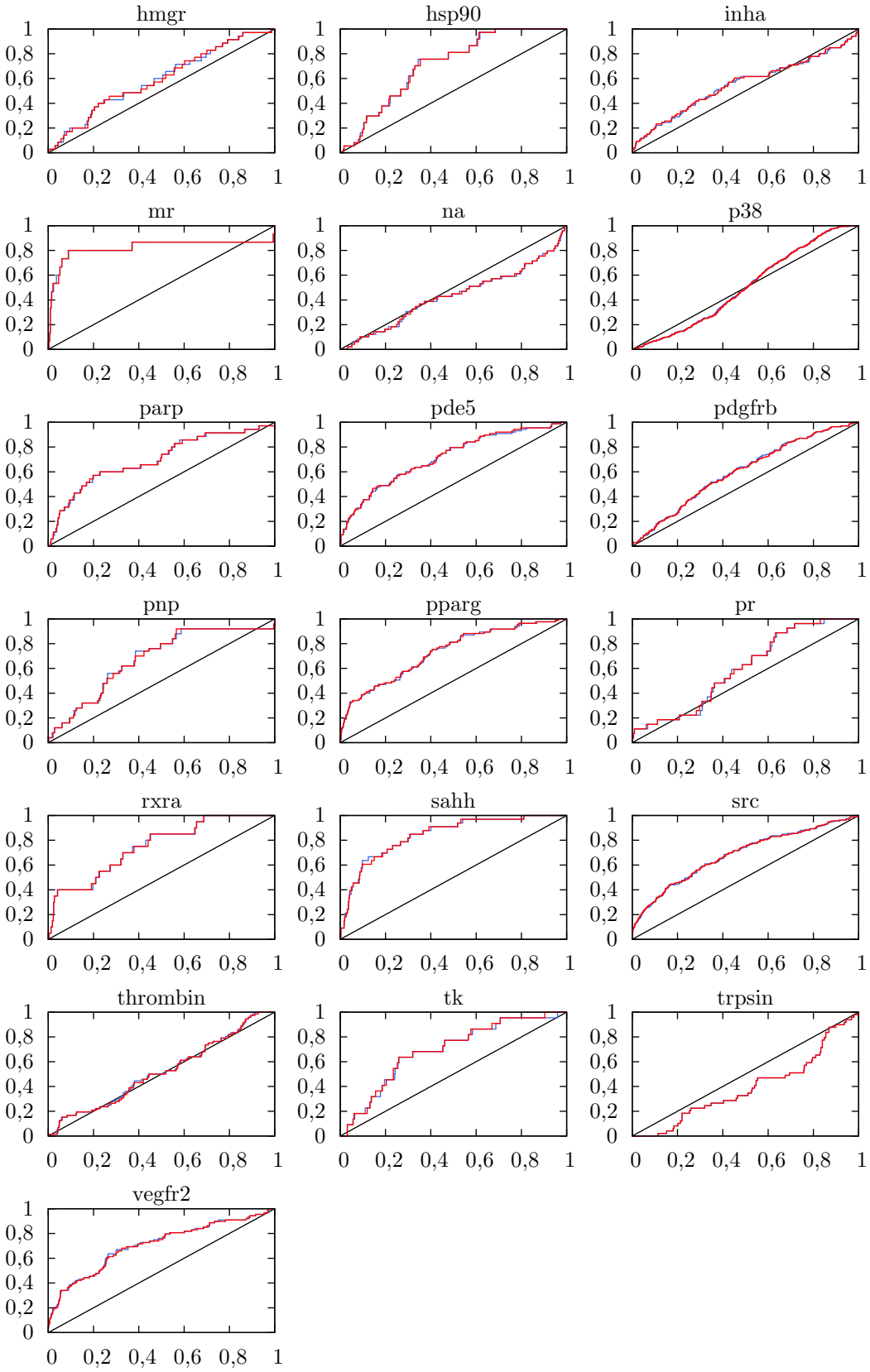


Figura 5.38: Comparativa de curvas ROC para DUD en CPU (azul) y GPU (rojo). Se almacena una solución por punto.

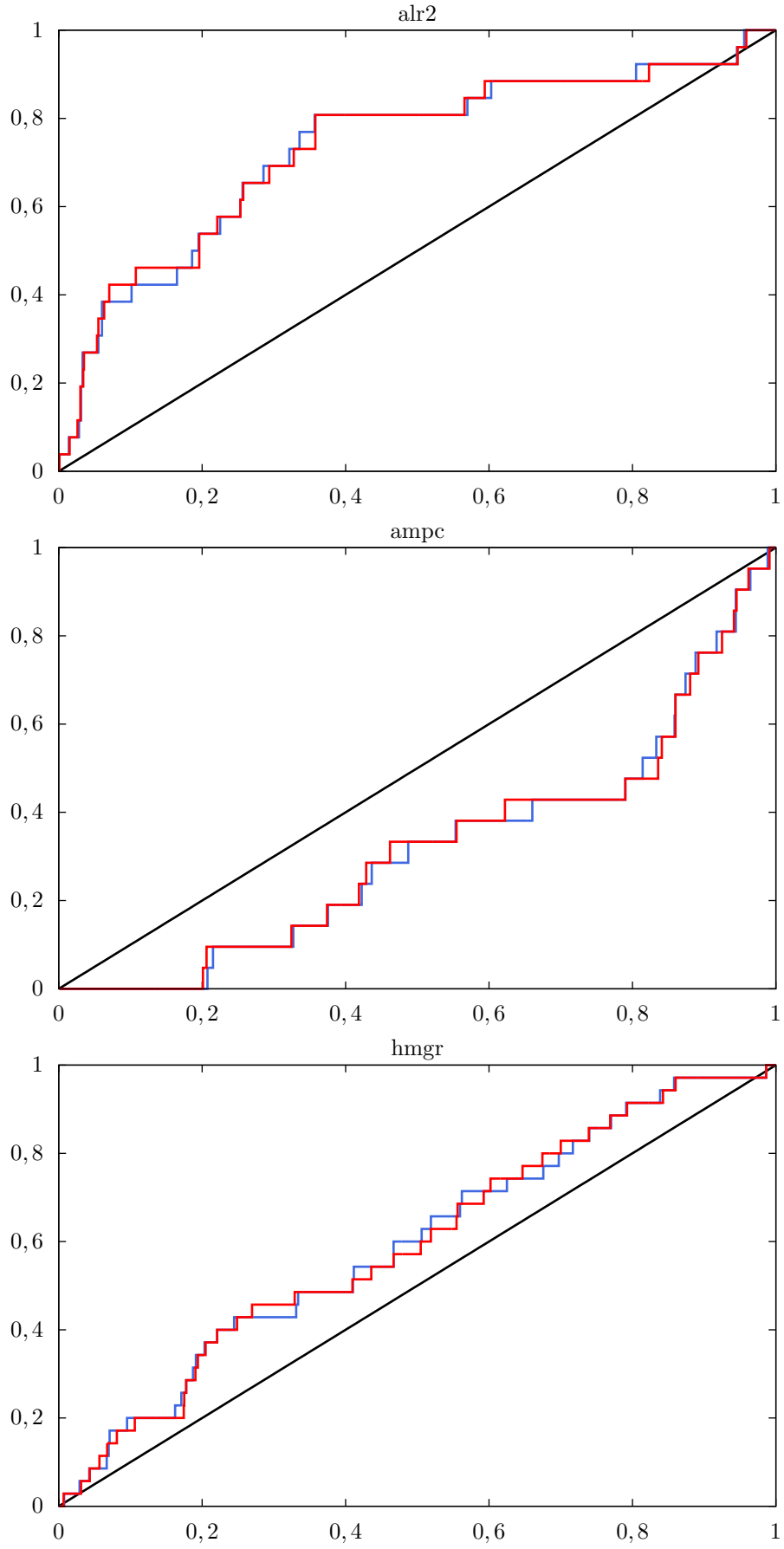
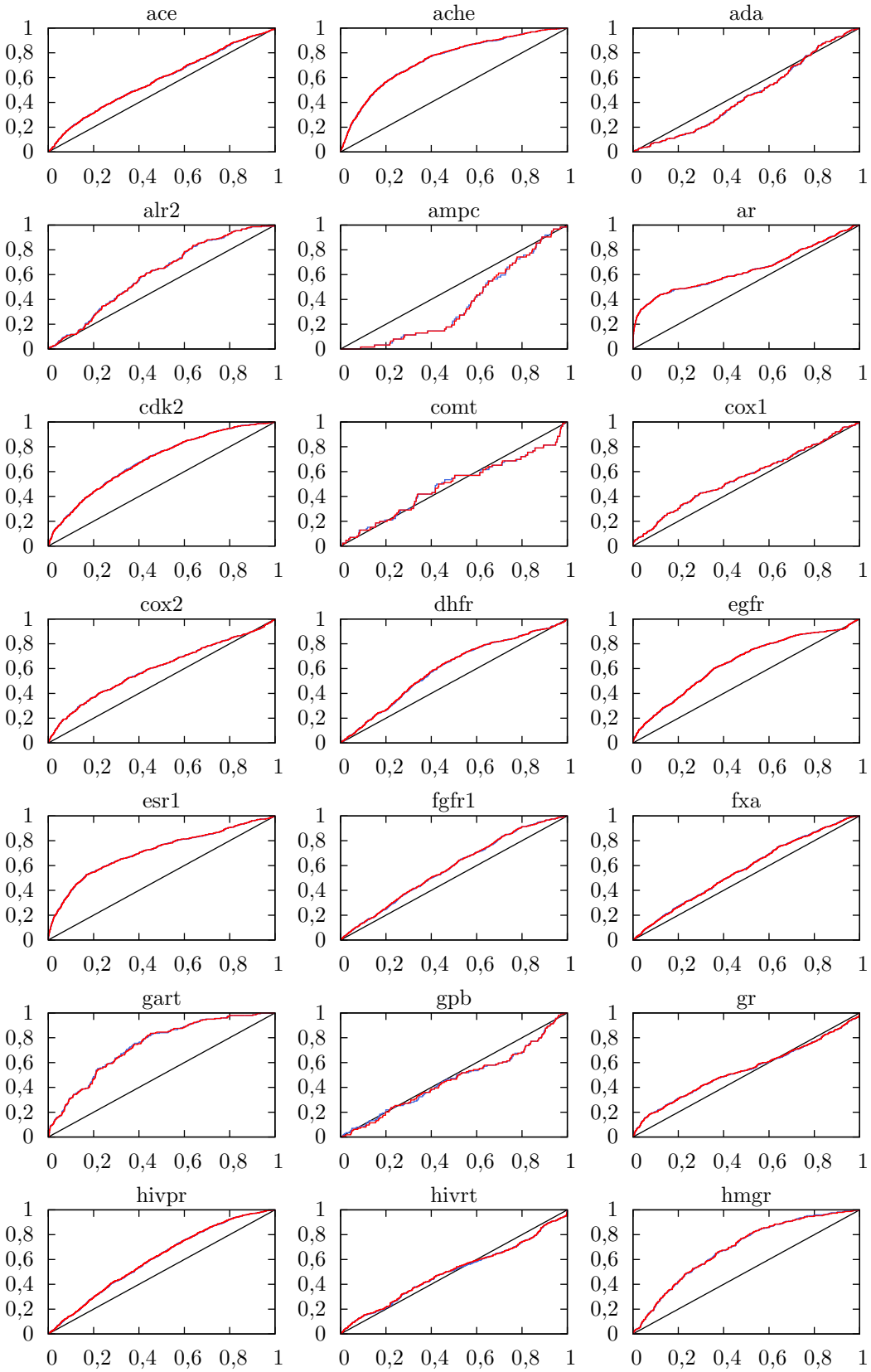


Figura 5.39: Ampliación de curvas ROC para los *targets alr2*, *ampc* y *hmgr* de DUD.


ace
ache
ada
alr2
ampc
ar
cdk2
comt
cox1
cox2
dhfr
egfr
esr1
fgfr1
fxa
gart
gpb
gr
hivpr
hivrt
hmgr
0
0,2
0,4
0,6
0,8
1

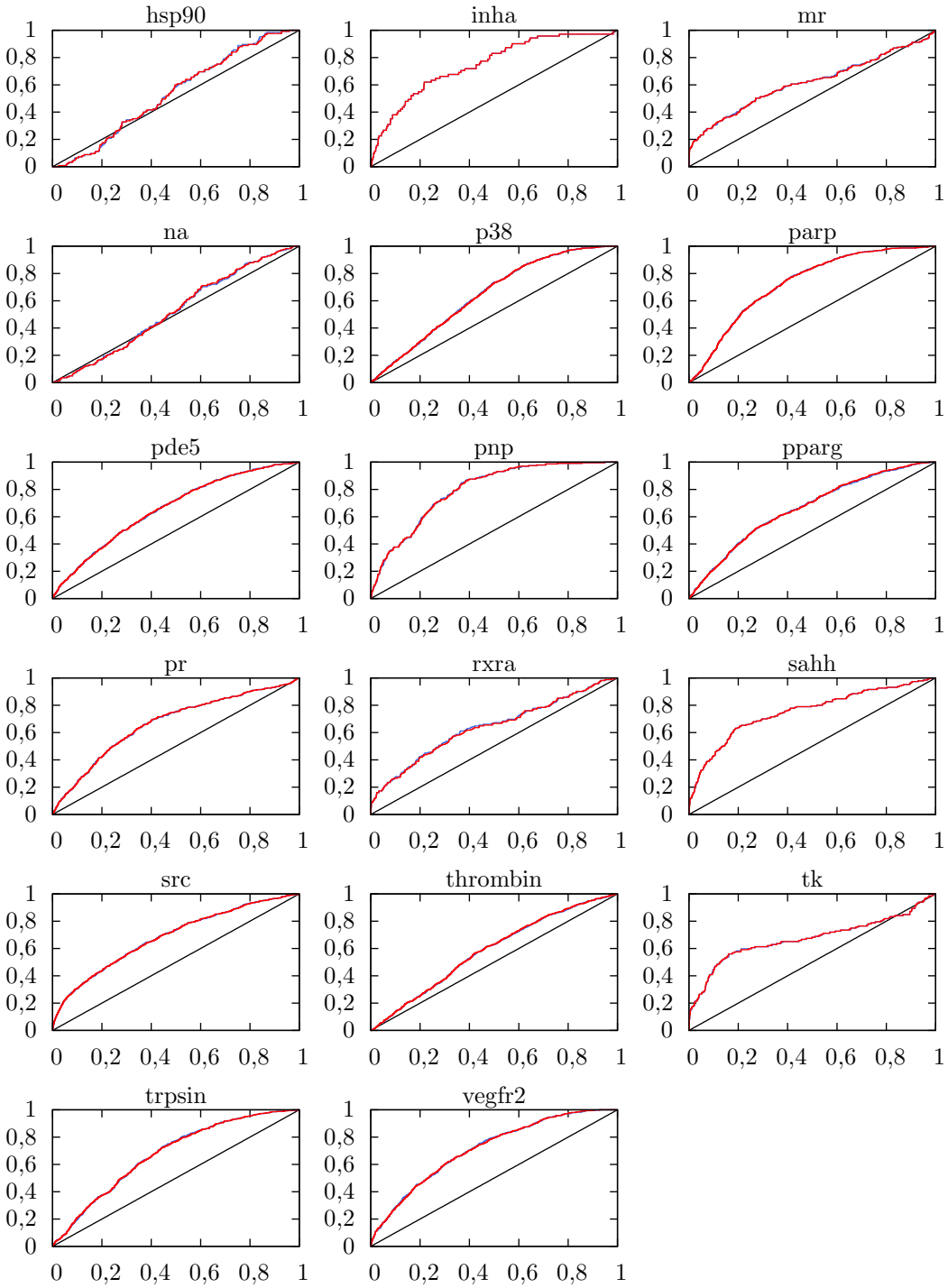


Figura 5.40: Comparativa de curvas ROC para DUD-E en CPU y GPU. Las líneas azules corresponden a usar la CPU empleando una solución por punto. Las líneas rojas corresponden a usar la GPU.

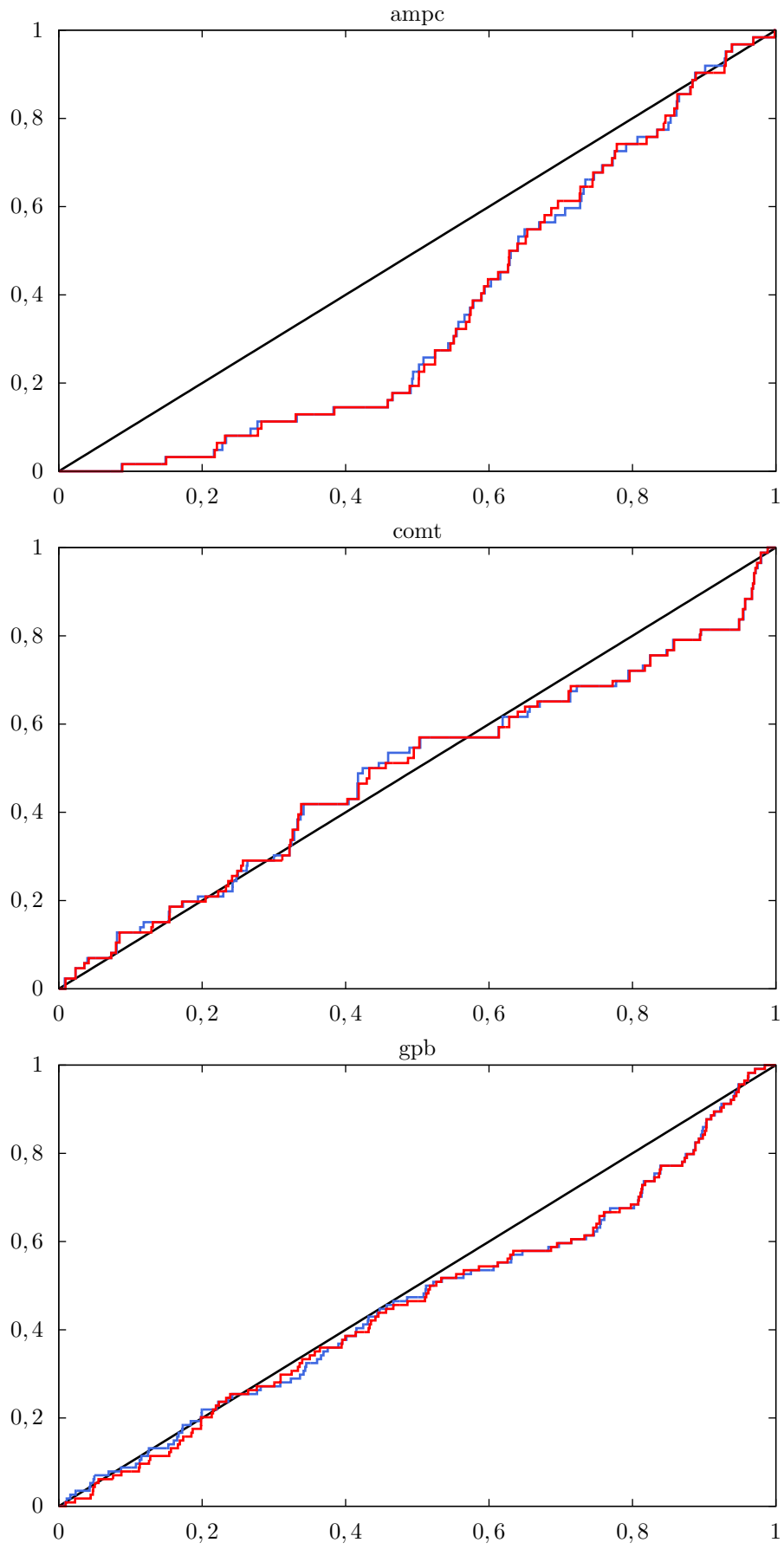


Figura 5.41: Ampliación de curvas ROC para los *targets ampc*, *comt* y *gpb* de DUD-E.

HARDWARE	TIEMPO POR PUNTO
Intel i7-950	636,40 μs
GK104	21,29 μs

Tabla 5.28: Tiempo promedio registrado para un ajuste molecular de un ligando en un punto del sitio de unión para CPU y GPU. El tiempo registrado para CPU corresponde con utilizar un sólo proceso.

para CPU (azul) y GPU (rojo). Al igual que con DUD, las curvas de CPU están solapadas por las de GPU prácticamente en su totalidad. El hecho de que los conjuntos de ligandos y *decoys* de este *benchmark* contengan una cantidad significativamente mayor de moléculas hace que las diferencias entre CPU y GPU se vuelvan más imperceptibles. Existen algunos casos (*ampc*, *comt* y *gpb*) en los que se pueden percibir pequeñas diferencias producidas por el mismo fenómeno que sucedía con DUD: el intercambio de posiciones entre dos o más ligandos en la lista de soluciones. La Figura 5.41 muestra las curvas ROC ampliadas para estos tres *targets*. La mayor discrepancia registrada se encuentra en el *target gpb*. Su diferencia entre áreas es del 0,34 %. En promedio para todos los casos del *benchmark* se calcula una diferencia del 0,09 % entre la versión paralela y la secuencial.

En conclusión, las diferencias registradas entre los resultados obtenidos por las versiones paralela y secuencial de FRODRUG para los tres conjuntos de pruebas empleados son tan pequeñas que se consideran despreciables.

El rendimiento en el cribado virtual se ha analizado en base a los tiempos de ejecución registrados para cada uno de los *targets* de DUD y DUD-E. Los tiempos de Astex se han omitido ya que al ser un *benchmark* destinado a *docking* cada proteína posee tan sólo un ligando que ajustar. Esto hace que la carga computacional no sea lo suficientemente elevada para obtener una estadística significativa. Por otro lado, DUD y DUD-E poseen miles de ligandos por *target*. Cada uno de los *targets* de ambos conjuntos de pruebas tiene cantidades diferentes tanto en su conjunto de ligandos como en los puntos en el espacio seleccionados para el sitio de unión. Con el fin de proporcionar una medida consistente, el tiempo mostrado a continuación es el que se necesita en promedio para calcular el ajuste de un punto traslacional del sitio de unión. Esto se debe a que el tiempo empleado por punto es constante independientemente de su número en el sitio de unión o del tamaño del ligando. El tiempo correspondiente a la versión secuencial del método se ha registrado sobre un procesador Intel i7-950 a 3,3 GHz. El tiempo correspondiente a la adaptación paralela sobre GPU se ha registrado empleando una tarjeta GK104 y la versión 5.0 de CUDA con la opción de matemática rápida habilitada. La Tabla 5.28 muestra el tiempo registrado en promedio por cada punto del sitio de unión. Según los tiempos registrados, la GPU logra realizar el proceso de cribado virtual 29,89 veces más rápido que la CPU empleando un único proceso. A modo de referencia, el tiempo completo empleado por el dispositivo gráfico para cribar en su totalidad el *benchmark* más grande disponible (DUD-E) es de algo más de 13 horas. La cantidad total de ligandos cribados en este tiempo es de 631.286. El mismo proceso en CPU tarda en completarse más de dos semanas.

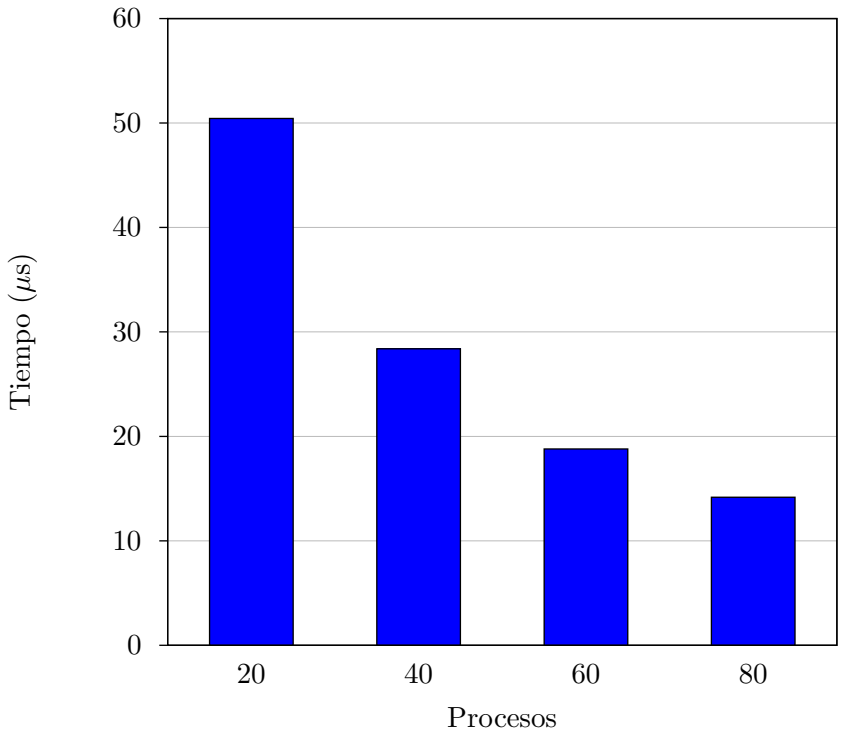


Figura 5.42: Tiempos promedio por punto registrados en el proceso de cribado virtual para el *target egfr* sobre el clúster Ladón empleando diferentes cantidades de procesos. Los tiempos corresponden a almacenar una solución por punto.

5.4.1.2. Comparativa con sistemas de memoria distribuida

En la siguiente sección se analiza el comportamiento de escalado de FRODRUG en sistemas de memoria distribuida. Para ello se emplea la versión del método basada en MPI. La librería utilizada es OpenMPI 1.4.2. En esta sección no se analiza la precisión de esta adaptación ya que los resultados son iguales a los recogidos durante la validación. Los tiempos de ejecución se han registrado en cuatro sistemas diferentes. Dos de ellos son clústers heterogéneos de los cuales uno está compuesto por CPUs y el otro por GPUs. Los otros dos sistemas son ordenadores independientes con CPUs multinúcleo. Uno contiene un procesador Intel i7-950 y el otro un Xeon E5-2650. El motivo de realizar también la prueba de escalado en procesadores individuales es analizar el comportamiento de la adaptación con MPI en sistemas controlados que no se ven afectados por factores externos. En primer lugar se muestran los datos registrados por los sistemas de CPU: el clúster y los procesadores individuales. Más adelante se detallan los resultados recogidos para el clúster de GPUs.

Las pruebas de escalado consisten en llevar a cabo el cribado virtual sobre un *target* ilustrativo de DUD-E (*egfr*) con su correspondiente conjunto de principios activos y *decoys*. El motivo de seleccionar este *target* es que su grupo de ligandos es uno de los más amplios del *benchmark*, contando con 36.275 moléculas. La prueba se repite varias veces empleando diferente número de procesos, registrando el tiempo promedio por cada punto del sitio de unión en cada prueba. El tamaño del volumen seleccionado para elegir los puntos a probar del sitio de unión es de

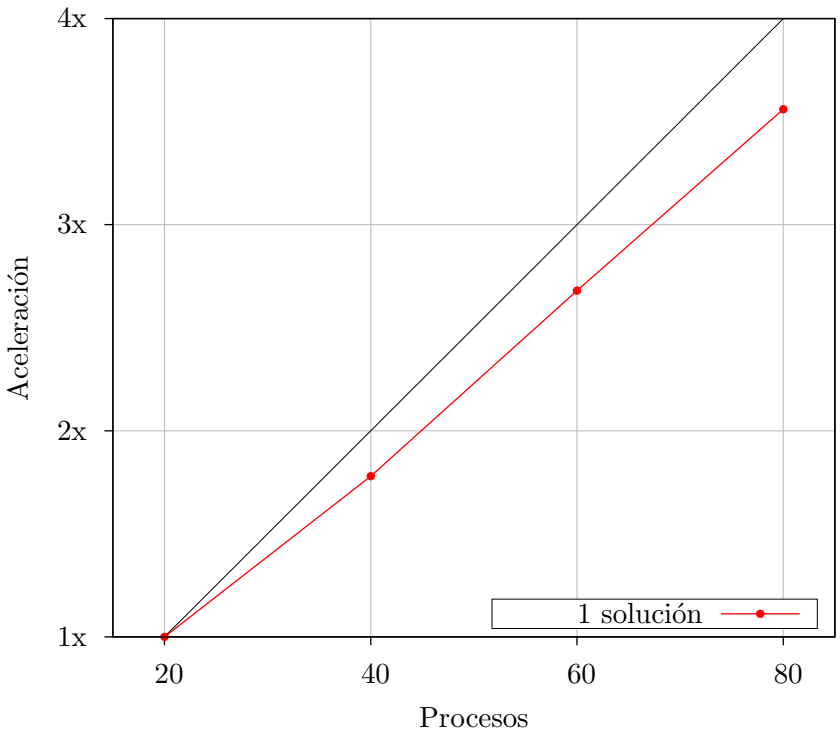


Figura 5.43: Aceleración obtenida en FRODRUG con MPI sobre el clúster Ladón utilizando diferente número de procesos. Se muestra la aceleración obtenida almacenando una solución por punto y una aceleración lineal de referencia.

PROCESADOR	NÚCLEOS	FRECUENCIA
Intel Xeon X5650	48	3,06 GHz
Intel Xeon E5-2650	256	2,8 GHz

Tabla 5.29: Procesadores pertenecientes al clúster Ladón. La cantidad indicada es el total de núcleos disponibles para cada tipo de procesador.

20x20x20 $Å^3$. Esto genera un conjunto de 1.396 puntos traslacionales.

La prueba de escalado con la adaptación sobre MPI requiere como mínimo que se generen dos procesos. Uno de ellos es el proceso maestro que actúa como servidor de ligandos. Este no realiza cómputo alguno. El otro es un proceso esclavo que recibe identificadores de ligando por parte del proceso maestro y realiza los ajustes moleculares.

Para comenzar se analiza la escalabilidad en el clúster Ladón de CPUs del Instituto de Química-Física Rocasolano perteneciente al Consejo Superior de Investigaciones Científicas. En la Tabla 5.29 se muestra el conjunto de procesadores que lo compone. Las ejecuciones en este sistema se realizan almacenando sólo una solución por punto. Esto es así porque en este clúster heterogéneo, el sistema de gestión de trabajos *Sun Grid Engine* selecciona los procesadores a emplear dependiendo de la carga del sistema. Esto deriva en una reducida probabilidad de que dos ejecuciones consecutivas, una almacenando tres soluciones por punto y otra una, utilicen los mismos procesadores. La prueba en este sistema se realiza para 20, 40, 60 y 80 procesos.

PROCESADOR	FRECUENCIA	NÚCLEOS	CACHÉ L3
Intel i7 950	3,3 GHz	4	8 MB
Intel Xeon E5-2650	2,8 GHz	8	20 MB

Tabla 5.30: CPUs individuales empleadas en las pruebas de rendimiento.

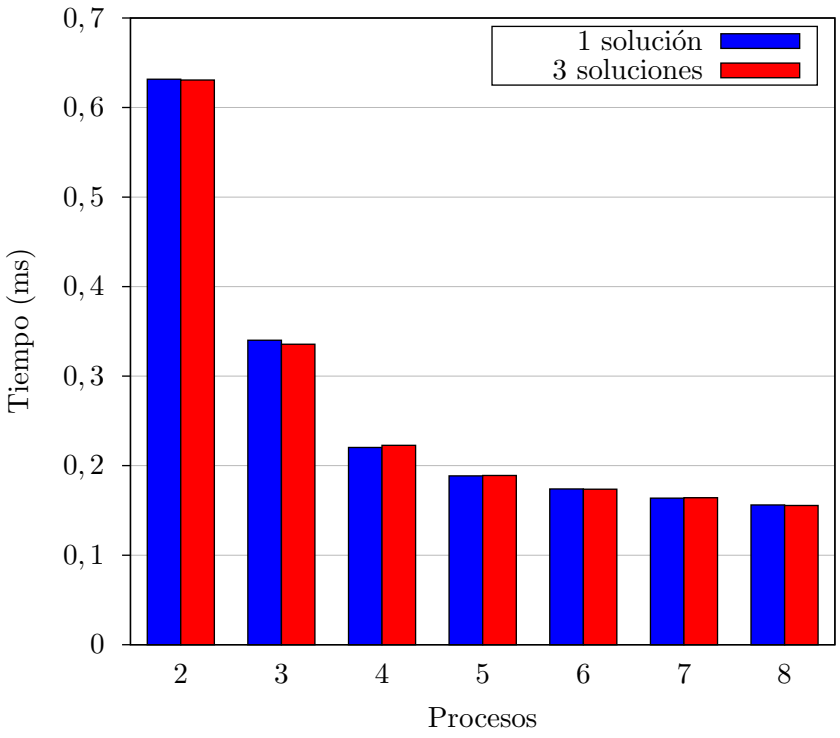


Figura 5.44: Tiempos promedio por punto registrados en el proceso de cribado virtual para el *target egfr* sobre el procesador Intel i7-950 empleando diferente número de procesos. Se muestra el tiempo registrado almacenando una y tres soluciones por punto.

A pesar de que el clúster cuenta con más procesadores, existe un límite máximo de uso de 80 núcleos por usuario de forma simultánea. La Figura 5.42 muestra los resultados relativos al tiempo de ejecución obtenido para FRODRUG sobre Ladón mientras que la Figura 5.43 muestra la correspondiente curva de aceleración. Tomando como base el tiempo promedio por punto para 20 procesos (50,44 μs) se registran unas aceleraciones de 1,78x, 2,68x y 3,56x para 40, 60 y 80 procesos respectivamente. Estos valores, próximos a los 2x, 3x y 4x esperados, son adecuados teniendo en cuenta que cada ejecución se realiza con un conjunto diferentes de procesadores debido a la heterogeneidad del sistema. En general el escalado de FRODRUG en esta clase de sistemas es muy bueno siempre que se disponga de la suficiente cantidad de datos a procesar como para ser distribuidos entre las CPUs disponibles.

El análisis continúa con los dos procesadores individuales. Se dispone de las CPUs Intel i7-950 y Xeon E5-2650. Sus características se muestran en la Tabla 5.30. En primer lugar se comienza con el análisis sobre el procesador Intel i7-950. Este dispone de 8 MB de caché y soporta el conjunto de instrucciones SSE4.2. El número de procesos elegidos para realizar la prueba va desde el mínimo necesario de dos (uno maestro y uno esclavo) hasta el máximo disponible de ocho. El

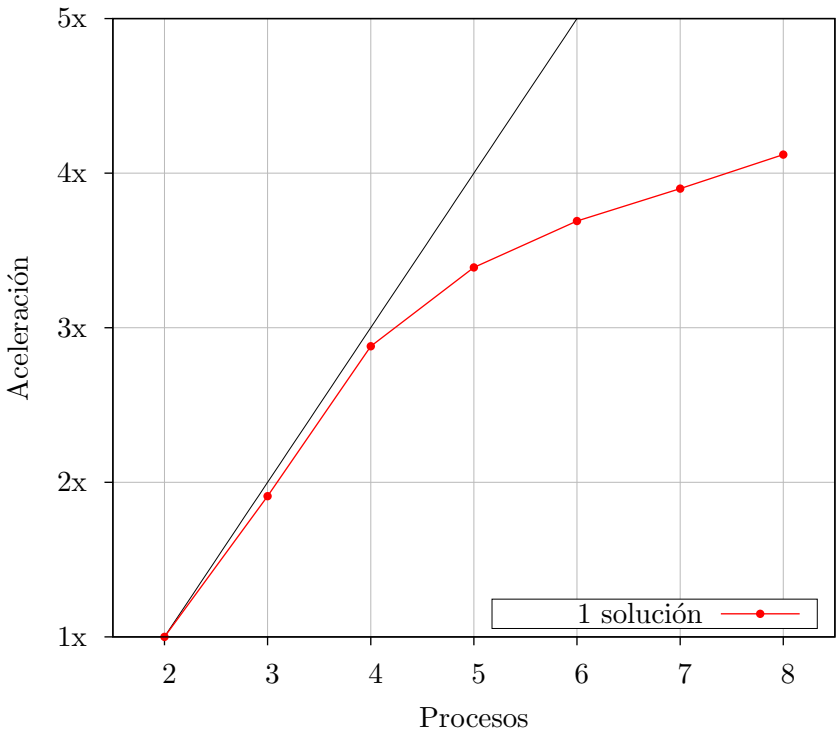


Figura 5.45: Aceleración obtenida en FRODRUG con MPI sobre el procesador Intel i7-950 utilizando diferente número de procesos. Se muestra la aceleración obtenida almacenando una solución por punto y una aceleración lineal de referencia.

procesador consta de cuatro núcleos físicos pero gracias a la tecnología *Hyper Threading* cuenta con ocho virtuales. En la Figura 5.44 se muestran los tiempos promedio registrados para calcular el ajuste molecular por cada punto del sitio de unión en el procesador Intel i7-950 utilizando tanto una como tres soluciones por punto. Las diferencias en los tiempos registrados por punto entre almacenar una o tres soluciones es tan pequeña que puede considerarse despreciable. La máxima diferencia registrada se ha producido al emplear tres procesos. En este caso al almacenar una solución por punto se ha utilizado 4,57 μs más que al almacenar tres. Se puede observar cuando se emplean dos o cuatro procesos que el tiempo registrado para tres soluciones por punto es ligeramente mayor que para una solución. Estas ligeras discrepancias se deben por lo tanto a factores externos al sistema. En la Figura 5.45 se muestra la aceleración obtenida para todas las cantidades de procesos utilizados almacenando una solución por posición traslacional. Los datos correspondientes a almacenar tres soluciones por punto no se muestran ya que prácticamente se superponen con los de una solución por punto. Contando como tiempo base el registrado para dos procesos puede verse que al utilizar tres, el tiempo registrado es prácticamente la mitad. La aceleración obtenida en este caso es de 1,91x. Esto es razonable ya que, teniendo en cuenta que el proceso maestro no realiza cómputo alguno, el número de procesos que realizan ajustes moleculares es el doble. Al emplear cuatro procesos la aceleración aumenta a 2,92x, empleando aproximadamente un tercio del tiempo base. A partir de este punto la aceleración se incrementa muy lentamente hasta un máximo de 4,12x. El descenso en el rendimiento sucede justo en el punto en que los núcleos físicos se utilizan por completo y comienzan a usarse las unidades

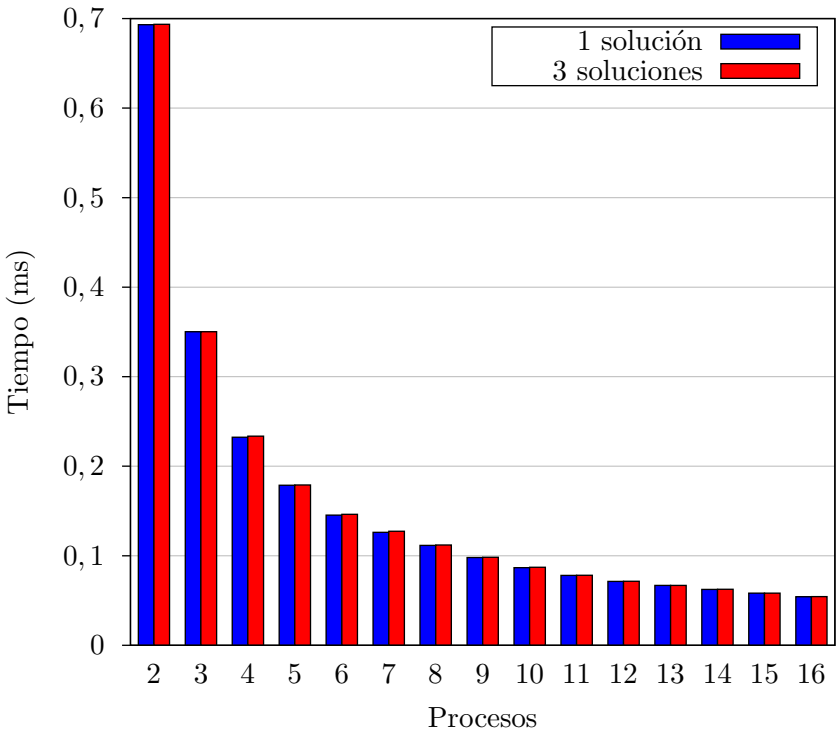


Figura 5.46: Tiempos promedio por punto registrados en el proceso de cribado virtual para el *target egfr* sobre el procesador Intel Xeon empleando diferente número de procesos. Se muestra el tiempo registrado almacenando una y tres soluciones por punto.

duplicadas por *Hyper Threading*.

El siguiente procesador evaluado es el Intel Xeon E5-2650. Este cuenta con 20 MB de memoria caché. Este procesador contiene ocho núcleos físicos; la cantidad total disponible de estos asciende a 16 debido al *Hyper Threading*. Dispone también de la extensión de conjunto de instrucciones AVX. La Figura 5.46 muestra el tiempo promedio empleado por FRODRUG para realizar el ajuste molecular sobre un punto del sitio de unión almacenando una y tres soluciones por punto. Del mismo modo que para el procesador i7, las diferencias registradas con respecto al tiempo de ajuste entre las dos configuraciones son muy reducidas. La más elevada se presenta para siete procesos, con una discrepancia de tan sólo 1,3 μs, siendo una diferencia despreciable. Como se puede observar, el promedio obtenido por punto del sitio de unión empleando 16 procesos con esta CPU (54,21 μs) está cerca del tiempo empleado por 20 procesos en el clúster Ladón (50,44 μs). Esto indica un comportamiento consistente entre sistemas teniendo en cuenta los factores externos que pueden afectar al tiempo de cómputo (comunicación entre nodos, heterogeneidad, ocupación del clúster, etc...). La aceleración obtenida para FRODRUG con el procesador Intel Xeon E5-2650 se muestra en la Figura 5.47. La curva corresponde a la ejecución almacenando una solución por punto traslacional. Como se puede ver, el escalado en este procesador es ligeramente mejor que en el Intel i7. El tiempo base tomado es de 693 μs, registrado empleando dos procesos. La aceleración obtenida para tres, cuatro, cinco y seis procesos es de 1,97x, 2,98x, 3,88x y 4,77x respectivamente. El escalado utilizando estas distintas configuraciones de procesos resulta prácticamente lineal. A partir de emplear siete procesos el rendimiento comienza a descender

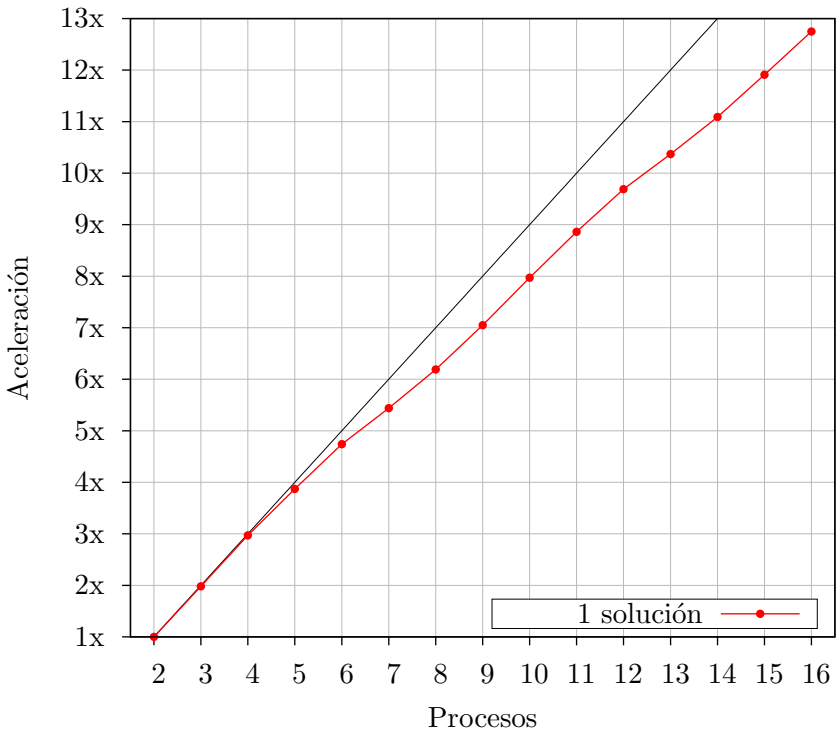


Figura 5.47: Aceleración obtenida en FRODRUG con MPI sobre el procesador Xeon E5-2650 utilizando diferente número de procesos. Se muestra la aceleración obtenida almacenando una solución por punto y una aceleración lineal de referencia. La curva correspondiente a almacenar tres soluciones por posición es similar a la obtenida almacenando sólo una.

GPU (CODE NAME)	FRECUENCIA	NÚCLEOS	ARQUITECTURA
NVIDIA GTX 680 (GK104)	1,06 GHz	1536	Kepler
NVIDIA GTX 470 (GF100)	1,22 GHz	448	Fermi
NVIDIA GTX 570 (GF110)	1,46 GHz	480	Fermi

Tabla 5.31: Modelos de dispositivos gráficos empleados en las pruebas de rendimiento con el clúster de GPUs.

ligeramente. Se obtiene una aceleración de 5,49x para siete y de 6,21x para ocho procesos. En este punto el rendimiento comienza a aumentar más lentamente debido a que se están utilizando la totalidad de los núcleos físicos disponibles. La aceleración llega a un máximo de 12,79x empleando 16 procesos, con un tiempo promedio por punto de 54 μs. Este valor se encuentra cerca de la máxima aceleración posible para 16 procesos (15x) sin embargo no es posible alcanzar ese nivel de rendimiento debido a que la tecnología de *Hyper Threading* no duplica los núcleos completos sino sólo algunas de sus unidades internas. Esto desemboca en que dos procesos ejecutándose en el mismo núcleo compiten por los recursos disponibles, degradando el rendimiento general.

Por último se ha realizado la prueba en un pequeño clúster de GPU consistente en tres equipos denominados A, B y C. Entre estos tres equipos han sido distribuidas cuatro tarjetas gráficas. Las características de los modelos de GPU empleados se encuentran en la Tabla 5.31. El primer equipo (A) cuenta con una CPU Intel i7-950 y dispone de dos dispositivos gráficos:

TARJETA	% DE LIGANDOS	TIEMPO POR PUNTO
$GK104^{(A)}$	28,34 %	21,60 μs
$GF100^{(A)}$	21,63 %	30,18 μs
$GF110^{(B)}$	24,82 %	24,33 μs
$GK104^{(C)}$	25,21 %	21,74 μs

Tabla 5.32: Porcentaje de ligandos ajustados y el tiempo promedio por punto que emplea cada una de las tarjetas. Para cada tarjeta se indica si está instalada en el sistema A, B o C, según se detalla en el texto.

un GK104 y un GF100. El segundo equipo (B), dispone de un procesador Intel Xeon E5620 a 2,4 GHz y de una GPU GF110. El último equipo (C), cuyo procesador es un Intel Core 2 Quad Q6600 a 2,40 GHz dispone de otra tarjeta GK104. Los equipos están interconectados a través de una red Ethernet que transmite 1 Gbps. Debe destacarse que el único modo de realizar esta prueba es con un sistema operativo Linux en todos los equipos. La combinación de CUDA + MPI no está disponible en Windows. Esto se debe a la restricción en el acceso a dispositivos gráficos al ejecutar tareas remotas a través de MPI en este sistema operativo. La Tabla 5.32 muestra los resultados de la prueba. Sólo se almacena una solución por punto al ser la única configuración disponible para GPU. En la prueba se ha registrado para cada tarjeta tanto el tiempo promedio por punto así como el porcentaje del conjunto de ligandos que ha cribado cada una de ellas. El proceso maestro que hace de servidor en esta prueba se ejecuta en el sistema A. Según puede observarse en la tabla, la cantidad de ligandos procesados por cada tarjeta, así como el tiempo promedio empleado por cada punto ajustado, se corresponden con su pertinente capacidad de cómputo. Existe una pequeña discrepancia entre la cantidad de ligandos computada entre ambas tarjetas GK104. El tiempo promedio por punto en ambas GPUs es similar, pero la instalada en el sistema A ajusta un 3,13 % más ligandos que la del sistema C. La explicación a esta diferencia reside en las características de sus CPUs. El proceso maestro, residente en el ordenador A, comienza a servir ligandos aproximadamente al mismo tiempo que el proceso de la GK104 del mismo equipo empieza a solicitarlos. En el caso del sistema C, que cuenta con una CPU menos potente, el proceso esclavo requiere más tiempo para realizar sus tareas de inicio. Debido a esto comienza a solicitar ligandos más tarde que el proceso esclavo del sistema A.

Teniendo en cuenta el tiempo global de ejecución, el clúster de GPU es capaz de realizar en promedio el ajuste de un ligando en un punto del sitio de unión en tan sólo 6,53 μs. Esta cantidad equivale a algo menos de la mitad del tiempo que se utiliza en realizar la misma operación utilizando 80 procesos en el clúster de CPU.

En general, las aceleraciones obtenidas por las adaptaciones de FRODRUG para GPU y sistemas de memoria distribuida son elevadas. La Tabla 5.33 muestra las mayores aceleraciones logradas por cada uno de los cuatro sistemas evaluados en esta sección tomando como referencia el tiempo registrado para la versión secuencial de FRODRUG utilizando un sólo proceso sobre el procesador Intel i7-950.

En resumen, se ha demostrado que el escalado de FRODRUG en sistemas de memoria

SISTEMA	ACELERACIÓN	TIEMPO POR PUNTO
Clúster Ladón	44,9x	14,17 μs
Intel i7-950	4,1x	155,82 μs
Intel Xeon E5-2650	11,7x	54,21 μs
Clúster de GPUs	97,5x	6,53 μs

Tabla 5.33: Aceleraciones máximas y tiempo promedio obtenido para FRODRUG en los cuatro sistemas evaluados en esta sección con respecto al tiempo registrado para la versión secuencial utilizando un sólo proceso sobre el procesador Intel i7-950.

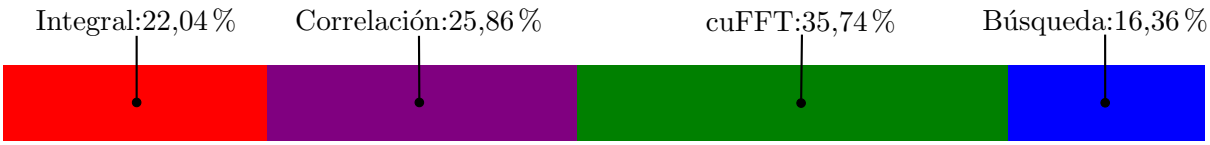


Figura 5.48: Porcentajes de tiempo de ejecución de los *kernels* de FRODRUG sobre una tarjeta GK104.

distribuida es muy bueno. Las aceleraciones obtenidas a través de las diferentes pruebas son casi lineales. Mientras exista una cantidad suficiente de ligandos que distribuir entre los diferentes procesos, el rendimiento sólo se ve reducido por las limitaciones del *hardware*.

5.4.2. Análisis detallado de los *kernels* desarrollados

Para continuar con la sección de análisis de resultados se procede a detallar el estudio realizado sobre los tres *kernels* que conforman la adaptación de FRODRUG a GPU: cálculo de integral, cálculo de correlación y búsqueda de las mejores correlaciones. Las técnicas y herramientas utilizadas para el análisis son las mismas que las empleadas en el apartado de análisis de los *kernels* correspondientes a FBM (sección 5.2.2). La Figura 5.48 muestra el porcentaje de tiempo de ejecución que emplea cada uno de los *kernels* desarrollados así como el de la librería cuFFT de NVIDIA sobre una GPU GK104.

5.4.2.1. *Kernel* para el cálculo de la integral

En primer lugar se realiza el análisis del *kernel* que calcula la parte de la integral del método. La Tabla 5.34 muestra los datos más relevantes registrados en el análisis. Este *kernel* utiliza bloques que contienen 512 *threads*. La cantidad de memoria compartida por bloque es de 11.392 *bytes*. Esta memoria se distribuye entre dos bloques de 32 x 33 valores y tres vectores de 32 datos flotantes para uso temporal, junto con cinco vectores de 128 valores para almacenamiento de datos procedentes de la memoria constante. El número máximo de bloques que pueden alojarse simultáneamente en el multiprocesador para este *kernel* es de cuatro. El factor limitante es la cantidad de *warps* por multiprocesador. Este sólo puede alojar 64 *warps* a la vez y cada bloque posee 16. Teniendo en cuenta la cantidad de bloques, en total se emplean 45.568 *bytes* de memoria compartida en cada multiprocesador. Esto sólo deja la opción de configurar una caché L1 de 16 KB y una memoria compartida de 48 KB para que el *kernel* pueda ser ejecutado.

Tamaño de bloque	[32,16,1]
Warps por bloque	16
Ocupación	63,18/64 (98,72 %)
IPC	2,82
Serialización	7 %
Divergencia	0 %

Tabla 5.34: Información de perfilado para el *kernel* que calcula la integral de FRODRUG en la tarjeta GK104.

Algunos de los datos que este *kernel* lee de memoria global tienen una disposición tal que no es posible realizar lecturas coalescentes de forma directa. Se ha tratado de optimizar no obstante estas lecturas lo máximo posible. Se ha registrado una cantidad promedio de 1,05 transacciones de memoria por cada solicitud de lectura, el cual es un valor casi óptimo. Con respecto a las escrituras de memoria global se logra realizar siempre una transacción por solicitud, que es lo mínimo posible. Tanto las lecturas como escrituras de los bancos de memoria compartida se realizan con una transacción por solicitud, lo que indica que no existen conflictos de bancos. La tasa de transferencia de datos global para este *kernel* es de 9,91 GB/s, valor que está muy lejos del máximo teórico de 192,2 GB/s. Por lo tanto el rendimiento de este *kernel* no está limitado por el ancho de banda de memoria disponible.

El análisis revela que la cantidad de instrucciones por ciclo ejecutadas es de 2,82. Este valor está algo por debajo de las cuatro instrucciones por ciclo deseables, indicando que el *kernel* tampoco está limitado por cantidad de instrucciones ejecutadas. El análisis muestra que en promedio por cada ciclo existen 6,76 *warps* disponibles para ser ejecutados. Esta cantidad en principio parece ser aceptable puesto que tan sólo se necesitan cuatro *warps* por ciclo en la arquitectura Kepler. Sin embargo parece ser una cantidad demasiado ajustada como para que el procesador gráfico trabaje con fluidez. En el 47,8 % de los ciclos no hay ningún *warp* disponible para ser ejecutado en los multiprocesadores. El perfilador no es capaz de indicar el motivo de la parada de los *warps* en el 35,1 % de los casos. Existe un conjunto de motivos de parada no documentados que la herramienta de perfilado engloba en una categoría llamada "otros". En esta se incluyen motivos como que todos los recursos computaciones estén ocupados o que haya sucedido un fallo de caché, entre otros. Por lo tanto no es posible identificar el motivo concreto de estas paradas. Sin embargo lo más probable en este caso es que las paradas se deban a los accesos que este *kernel* realiza a memoria constante. Cuando un conjunto de *threads* accede simultáneamente a diferentes direcciones en este tipo de memoria los accesos se serializan, resultando en una transacción por cada dirección leída. El siguiente motivo de parada, que sucede en el 24,2 % de los casos, es el de dependencia de datos. Esto sucede cuando una instrucción no puede ser ejecutada al no estar sus operandos disponibles. El tercer motivo de parada, en el 17,9 % de los casos, es la sincronización entre *threads* del bloque. Estas instrucciones de sincronización han sido incluidas por decisiones de diseño y las paradas por este motivo no pueden ser reducidas.

El flujo de ejecución del *kernel* es óptimo. Se ha logrado obtener una divergencia del 0 %,

Tamaño de bloque	[32,16,1]
Warps por bloque	16
Ocupación	62,64/64 (97,87 %)
IPC	2,97
Serialización	2,05 %
Divergencia	0 %

Tabla 5.35: Tabla de perfilado para el *kernel* que genera los volúmenes de correlación de FRODRUG en la tarjeta GK104.

lo que quiere decir que todos los *threads* dentro del mismo *warp* toman el mismo camino. La serialización en este *kernel* es baja, de tan sólo el 7 %. Los accesos a memoria global y compartida son casi óptimos, por lo que esta serialización puede achacarse al acceso a la sección de datos ubicada en memoria constante.

Las decisiones de optimización que han llevado a la versión final de este *kernel* priorizaban la ausencia de divergencia y los patrones de acceso a memoria adecuados, ya que son los factores que más impacto tienen en el rendimiento. En este aspecto se ha logrado cumplir con los objetivos de optimización.

5.4.2.2. *Kernel* para la generación del volumen de correlación

El siguiente *kernel* analizado es el encargado de generar los volúmenes de correlación. La Tabla 5.35 muestra la información más relevante del análisis. Las decisiones de diseño aplicadas al *kernel* que genera los volúmenes de correlación son las mismas que las aplicadas al que calcula las integrales. Este *kernel* tiene el mismo tamaño de bloque que el anterior, y por lo tanto la misma cantidad de *warps*. Su ocupación en el multiprocesador es completa (64 *warps*) lo que significa que se alojan simultáneamente cuatro bloques. La cantidad de memoria utilizada por bloque es de 4.224 bytes. En total se emplean 16.896 bytes de memoria compartida entre los cuatro bloques. Esto fuerza una configuración de memoria equilibrada con 32 KB de caché L1 y 32 KB de memoria compartida, aunque se desperdicien casi 16 KB de memoria. Si se seleccionase una memoria compartida de 16.384 *bytes* y una caché de 49.152 *bytes* el *kernel* no podría lanzarse.

La tasa de transferencia de memoria de este *kernel* es de 64,38 GB/s, lo que supone un 35,11 % de la capacidad total del bus. El *kernel* no está por lo tanto limitado por el ancho de banda de memoria. La serialización registrada para este *kernel*, de tan sólo 2,05 %, es muy baja. En gran parte se debe a los accesos que el multiprocesador realiza en memoria local y en parte al patrón de escritura de memoria global. Las lecturas realizadas en memoria global por este *kernel* son óptimas, con una sola transacción por solicitud. No lo son por el contrario las escrituras, ya que producen en promedio 1,97 transacciones por solicitud. Esto se debe a que la memoria de los volúmenes que el *kernel* escribe no se encuentra alineada correctamente. Sólo una de cada 16 solicitudes produce una transacción; el resto producen dos. La memoria se dispone de este modo debido a un requisito del procedimiento que realiza a continuación la transformada inversa de

Fourier. Este método necesita que sean agregados dos valores flotantes al final de cada fila del volumen, siendo esto lo que produce el desalineamiento. Un ejemplo de este comportamiento se encuentra detallado en la Figura 5.21 dispuesta en la sección de análisis de los *kernels* de FBM. Las lecturas y escrituras de memoria compartida son óptimas, realizándose sólo una transacción por solicitud.

El *kernel* tampoco está limitado por cantidad de instrucciones ejecutadas por ciclo, con tan sólo 2,97. Este número es algo más bajo al deseable de cuatro, pero se acerca de forma razonable. El perfilador indica una cantidad promedio de 6,85 *warps* listos para ejecutar por cada ciclo de reloj. Esta cantidad parece no ser suficiente ya que en el 44,7 % de los casos los *schedulers* no pueden lanzar instrucciones al no haber *warps* preparados. En este *kernel*, el principal motivo de parada de los *warps* es el de la dependencia de datos en un 37,6 % de los casos. En un diseño preliminar del *kernel*, el porcentaje era más elevado. Con el fin de reducir la cantidad de paradas por este motivo se hizo un nuevo diseño para que se realizase el cálculo de dos vóxeles del volumen por cada *thread* en lugar de uno. De este modo se añade más paralelismo a nivel de instrucción para cada *thread* y se logra reducir la cantidad de paradas hasta el porcentaje actual. En el 28,3 % la razón es la sincronización entre *threads*. Estas instrucciones son necesarias al realizar este *kernel* su cómputo en dos pasos siendo indispensable el intercambio de datos entre *threads*. En tercer lugar, con un 20,1 %, se encuentra el conjunto denominado como "otros", que no proporciona ninguna información concreta ya que engloba varios motivos no documentados.

Al igual que para el *kernel* anterior, en este se obtiene una divergencia del 0 %. Esto se traduce en un óptimo flujo de ejecución en el que todos los *threads* dentro de un mismo *warp* siempre toman el mismo camino de ejecución.

En general el *kernel* se comporta bien con respecto al trabajo que tiene que desempeñar. Sus accesos de escritura en memoria son los mejores posibles teniendo en cuenta el desalineamiento existente y su flujo de ejecución es óptimo.

5.4.2.3. *Kernel* para la búsqueda de parámetros de ajuste

Por último se realiza el análisis del *kernel* que lleva a cabo la búsqueda de los mejores valores en los volúmenes de correlación. La Tabla 5.36 muestra los datos más relevantes del análisis. La principal diferencia de este *kernel* con respecto a los otros es la cantidad de *threads* por bloque que utiliza. En este caso se dispone de bloques que contienen 1.024 *threads* cada uno. Cada bloque emplea un total de 16.384 bytes de memoria compartida. El máximo número de bloques que pueden alojarse simultáneamente en un multiprocesador es de dos, pues cada uno emplea 32 *warps*. Por lo tanto es necesario reservar 32.768 bytes de memoria compartida. Los 32 KB restantes se asignan a memoria caché L1. Este *kernel* reutiliza gran cantidad de los datos que lee y todos ellos se alojan en memoria compartida, por lo que la memoria caché tiene poco uso. La serialización de 1,94 % se debe principalmente a los accesos que los multiprocesadores realizan a la caché L1 con respecto a la memoria local empleada.

La tasa de transferencia de memoria que este *kernel* utiliza no es especialmente alta, con tan solo 47,18 GB/s. Esto supone un 25,73 % de la capacidad total del bus, por lo que el *kernel* no está limitado por ancho de banda. Los accesos de memoria realizados son óptimos tanto para

Tamaño de bloque	[32,1,32]
Warps por bloque	32
Ocupación	59,72/64 (93,31 %)
IPC	1,82
Serialización	1,94 %
Divergencia	5,42 %

Tabla 5.36: Tabla de perfilado para el *kernel* que busca el valor más alto en el volumen de correlación de FRODRUG en la tarjeta GK104.

memoria global como compartida. La lectura y escritura se llevan a cabo a través de una sola transacción por solicitud de acceso.

La cantidad de instrucciones ejecutadas por ciclo es baja, siendo tan sólo 1,82. Esto se debe al objetivo del algoritmo llevado a cabo por este *kernel*. La búsqueda en paralelo del mejor valor entre una colección de datos es un proceso que presenta una intensidad aritmética baja. En promedio existen 4,78 *warps* disponibles para ser ejecutados por ciclo. Esta cifra es claramente insuficiente ya que en el 66,4 % de los ciclos no existe ningún *warp* disponible para ser ejecutado. Los principales motivos de parada son indicativos del escaso rendimiento obtenido en esta etapa. En el 43,5 % de los casos existe dependencia de datos y en el 28,3 % se debe a sincronizaciones entre *threads*. Estos dos motivos superan el 70 % de las causas de parada. El algoritmo hace una búsqueda comparando la mitad de los valores disponibles con la otra mitad, reduciendo en cada paso la cantidad de valores disponibles en un 50 %. Los motivos de parada indicados se justifican ya que cada iteración requiere de los datos de la anterior, y los *threads* deben ser sincronizados en cada una de ellas. Por esta parte no ha sido posible optimizar más el proceso. Por otro lado se ha logrado que la divergencia entre *threads* del mismo *warp* sea muy baja: 5,42 %.

En general este *kernel* se comporta excepcionalmente bien para el trabajo para el que ha sido diseñado. Se cumplen los objetivos prioritarios de optimización consistentes en una baja divergencia y en unos accesos óptimos a memoria.

La Tabla 5.37 muestra un resumen de los aspectos más importantes recogidos en los análisis realizados sobre los tres *kernels* utilizados por FRODRUG. El *kernel* que genera la integral muestra un rendimiento algo menor al esperado para un código cuya tarea principal es realizar cómputo. Esto es principalmente debido a los tiempos de espera generados por sus accesos a memoria constante. Los otros dos *kernels* muestran unos datos en concordancia con lo esperado. El *kernel* que genera los volúmenes de correlación tiene una carga equilibrada entre cómputo y transferencia de datos mientras que el *kernel* de búsqueda no realiza apenas cálculos y se dedica en su mayor parte a leer la memoria en busca de los valores más altos.

5.4.2.4. Uso de registros en los *kernels*

Los tres *kernels* desarrollados para FRODRUG hacen un uso elevado de registros por *thread*. Esto hace que la ocupación de los multiprocesadores se vea reducida debido a la limitada cantidad de este recurso. El compilador dispone de una opción mediante la cual se fuerza un límite en

	Integral	Correlación	Búsqueda
IPC	2,82	2,97	1,82
Transferencia	9,91 GB/s	64,38 GB/s	47,18 GB/s
Factor limitante	Memoria constante	Dependencia de datos	Dependencia de datos

Tabla 5.37: Resumen de análisis de los *kernels* de FRODRUG.

la cantidad de registros por *thread* empleados. Al reducir el número de registros por *thread* es posible aumentar la ocupación de los multiprocesadores. Los *threads* siguen utilizando la misma cantidad de datos, por lo que los datos que no puedan ser almacenados en los registros tendrán que ser guardados en memoria local. Almacenar datos en memoria local es algo poco deseado en dispositivos antiguos con arquitectura Tesla (capacidad computacional 1.0) pues la memoria local reside en el mismo lugar que la memoria global, y su acceso tiene la misma latencia. En los dispositivos actuales la memoria local reside en las cachés de los multiprocesadores. Siendo su latencia la misma que para memoria compartida, merece la pena explorar la posibilidad de restringir la cantidad de registros para aumentar la ocupación. El *kernel* que realiza el cálculo de la integral emplea 40 registros, el que genera el volumen de correlación utiliza 37 y la búsqueda de las mejores correlaciones requiere 38. Sus ocupaciones respectivas son 75 %, 75 % y 50 %. La opción de limitación en el compilador se fija a 32 registros por *thread*. Esto produce ocupaciones del 100 % en los tres *kernels*. La mejora global en tiempo de ejecución registrada activando esta opción es de 1,09x. No es una mejora sustancial, pero en este caso utilizar memoria local aumenta el rendimiento.

5.4.3. Comparativa con otros métodos

Por último se realiza una comparativa de FRODRUG con los más recientes métodos utilizados en el campo. Los datos contrastados en esta comparativa son los correspondientes a los *benchmarks* Astex para ajuste molecular y DUD para cribado virtual de proteínas. No se posee el código ejecutable de la mayoría de los métodos comparados. Por este motivo una comparativa de tiempos de ejecución entre diferentes métodos no sería posible. Sin embargo se presentan los tiempos proporcionados por los investigadores para sus métodos correspondientes siempre que estén disponibles, aunque sea en distintas plataformas.

Esta comparativa pretende situar nuestros desarrollos preliminares en el contexto actual del ajuste molecular y cribado virtual. La comparativa no es del todo justa si consideramos que FRODRUG no está completamente desarrollado y carece de flexibilidad tanto en ligandos como en proteínas. Los métodos con los que se compara son desarrollos completos que en su mayor parte cuentan con flexibilidad en proteína y ligando. En primer lugar se presenta la comparativa con respecto al ajuste molecular seguida de la correspondiente al cribado virtual.

5.4.3.1. Ajuste molecular

En relación a la comparativa de ajuste molecular, se compara en diferentes métodos el número de complejos proteína-ligando del *benchmark* Astex para los que se realiza un ajuste molecular con un error igual o inferior a 2 Å. La cantidad de complejos proteína-ligando que el *benchmark*

MÉTODO	RMSD ≤ 2Å(%)
FROFRUG	97,65 %
LigDockCSA	84,7 %
Autodock	81,7 %
GOLD	81 %
Quantum.Ligand.Dock	78 %
Lead Finder	74,1 %
Molegro Virtual Docker	74 %
CRDOCK	62 %

Tabla 5.38: Porcentaje de ligandos cuyo ajuste se realiza con un error igual o inferior a 2 Åpara diferentes métodos de ajuste molecular. Estos están ordenados de mayor a menor según el porcentaje que obtienen.

proporciona es de 85. En la Tabla 5.38 se muestran los datos recopilados para esta comparativa.

Como puede observarse, FRODRUG realiza un ajuste molecular correcto para casi la totalidad de los complejos proteína-ligando proporcionados en el *benchmark*, superando al resto de los métodos de ajuste molecular. Es necesario indicar que todos los métodos de *docking* salvo FRODRUG cuentan con implementación de flexibilidad, lo que hace que su tasa de acierto se reduzca. De entre estos métodos, sólo tres de ellos (LigDockCSA [196], Quantum.Ligand.Dock [101] y Molegro Virtual Docker [216, 177]) se utilizan exclusivamente para *docking*. El resto se emplean también para *virtual screening* y se hablará de ellos a continuación, en la comparativa de *virtual screening*. Quantum.Ligand.Dock es un método de *docking* que cuenta con implementación paralela. Al contrario que FRODRUG, la mayor parte de su implementación GPU se realiza con OpenCL en lugar de con CUDA. LigDockCSA realiza el ajuste molecular a través de un proceso iterativo con un comportamiento similar a los algoritmos genéticos. Su función de puntuación es la misma que usa AutoDock, pero añadiéndole un potencial energético de torsión. Por último, Molegro Virtual Docker es otro método que cuenta con implementación paralela sobre dispositivos gráficos [200]. En este caso su adaptación GPU se ha desarrollado con CUDA al igual que la de FRODRUG. Este es el único método para el que se reportan tiempos empleados en el ajuste molecular, aunque de forma incompleta. Los tiempos recopilados se corresponden con la ejecución sobre una tarjeta NVIDIA GTX 570 y un procesador Intel Core2 Quad Q8200 empleando cuatro núcleos y realizando el ajuste molecular sobre tres complejos diferentes (1hvr, 1stp y 3ptb). Los correspondientes tiempos son 0,34 s, 0,23 s y 0,18 s. Estos tiempos son mejorados por FRODRUG, con un promedio de 0,08 s por ligando empleando el mismo modelo de tarjeta.

5.4.3.2. Cribado virtual

A continuación se realiza la comparativa de FRODRUG con otros métodos de *virtual screening*. Los datos que se contrastan en esta comparativa son las áreas promedio obtenidas por cada uno de los métodos para los *targets* del conjunto de pruebas DUD así como los tiempos de ejecución cuando son proporcionados por los investigadores de cada método. El promedio de

MÉTODO	AUC PROMEDIO
Lead Finder	0,73
Glide 4.5	0,72
Autodock Vina	0,70
GOLD 3.2	0,70
rDock	0,69
CRDOCK	0,66
Surflex 2.1	0,66
FRODRUG	0,63
ICM 3.5-1	0,63
FlexX 2.0.3	0,61
PhDOCK	0,59
DOCK 6.1	0,55

Tabla 5.39: Áreas bajo la curva obtenidas por diferentes métodos de cribado virtual para el conjunto de pruebas DUD. Los métodos están ordenados de mayor a menor según el área que obtienen.

tiempo calculado para el ajuste de un único ligando es de 70,38 ms empleando FRODRUG con la mejor GPU disponible (GK104); el promedio por punto del sitio de unión es de 21,29 μs.

La Tabla 5.39 muestra el área promedio para cada método. Los métodos se encuentran ordenados de mayor a menor área obtenida. Se puede observar que el método que obtiene el mejor área es Lead Finder [209, 145]. El proceso de ajuste empleado por este método está basado en algoritmos genéticos. El método implementa flexibilidad de ligandos codificando las distintas conformaciones de estos en diferentes cromosomas. No se dispone de información sobre el *hardware* en el que se han registrado los tiempos. Se indica sin embargo que en promedio se emplea un minuto por ligando utilizando un sitio de unión que contiene 216 puntos.

GLIDE soporta también flexibilidad de ligandos. En el caso de este método, se realiza una búsqueda exhaustiva de las diferentes conformaciones de los ligandos proporcionados. El sitio de unión utilizado por este método se genera de un modo similar al que se hace en FRODRUG en el aspecto de que el volumen de búsqueda se centra en el ligando extraído de la estructura cristalográfica. Se desconocen los tamaños del sitio de unión empleados para los resultados mostrados. Se ha registrado para GLIDE un tiempo promedio de 3,7 segundos por *target* sobre un Intel Xeon X5660 a 2,8 GHz.

Autodock Vina [42, 220] emplea, al igual que FRODRUG, el ligando de referencia para centrar el volumen en la que se sitúan los puntos del sitio de unión. El tamaño de este varía en dependencia del *target* desde los 56.365 Å^3 hasta los 88.845 Å^3. FRODRUG ha empleado en las pruebas volúmenes de búsqueda de 20 x 20 x 20 Å^3 (13.824 Å^3) pero la cantidad final de puntos traslacionales también depende del *target*. En promedio para los 40 *targets*, Vina produce un área bajo la curva de 0,70, un valor moderadamente superior al 0,63 obtenido por FRODRUG. Las diferencias más notables en áreas se producen para los *targets ada*, *er_antagonist*, *hivpr* y

tk. En dos de estos casos (*er_antagonist* y *hivpr*) el área bajo la curva obtenida por Vina es muy superior a la obtenida por FRODRUG: 0,90 en ambos casos contra 0,55 y 0,56 respectivamente. En los otros dos casos (*ada* y *tk*) FRODRUG produce un área bajo la curva bastante mejor: 0,56 y 0,69 frente a 0,29 y 0,46 respectivamente. El tiempo promedio registrado para Vina es de 147,5 segundos por cada *target* de DUD sobre un Intel Xeon X5660 a 2.8 GHz.

GOLD [97, 98] se comporta de un modo similar a GLIDE, utilizando algoritmos genéticos para realizar el cribado virtual. Se utilizan del mismo modo los cromosomas para codificar las diferentes conformaciones de los ligandos. Se dispone de cuatro diferentes funciones de puntuación para realizar el cribado virtual [114]. El dato mostrado en la Tabla 5.39 para este método es el correspondiente al mayor área obtenida de entre las diferentes funciones de puntuación. Este área se obtiene con las funciones Chemscore y ChemPLP. Los nombres de las dos funciones restantes son ASP y Goldscore. GOLD define el sitio de unión como los puntos del espacio sobre la superficie del *target* que se encuentran situados a 10 Å o menos de cada uno de los átomos del ligando, cuando este se encuentra colocado en su lugar adecuado. No se ha encontrado información relativa a tiempos de ejecución para este método.

En rDock [182] se utiliza el mismo método empleado en GOLD para generar el sitio de unión: explorar los puntos contenidos en el espacio cercano al ligando extraído de la estructura cristalográfica. Sin embargo, el valor por defecto para este método es una distancia de 6 Å o menos de los átomos del ligando. En rDock se registran unos tiempos promedios de 29,8 segundos por cada *target* del conjunto de pruebas DUD sobre un Intel Xeon X5660 a 2.8 GHz.

CRDOCK [24] es un método modular que dispone de varias funciones de búsqueda y puntuación disponibles que pueden ser seleccionadas dependiendo de las características del problema. Este método obtiene en promedio un área bajo la curva un 3 % superior a la obtenida por FRODRUG y reporta tiempos promedios de aproximadamente 13 segundos para el ajuste molecular de un ligando contando con flexibilidad. Este tiempo se ha registrado sobre un Intel i5 a 3,3 GHz.

Surflex [94] es un método que utiliza la función de puntuación empírica *Hammerhead* [93]. El área promedio obtenida es similar a la de CRDOCK. Con respecto a sus tiempos de ejecución se dispone de poca información, salvo que los investigadores de este método indican que el ajuste molecular para un ligando es de "unos pocos segundos" si no se utiliza flexibilidad y se añaden alrededor de 10 segundos por cada unión rotable si se utiliza flexibilidad. La plataforma empleada en este caso para registrar los tiempos es un Pentium 3 a 933 MHz.

FlexX [172] es un método que utiliza un sistema de muestreo basado en un árbol de búsqueda para colocar los ligandos en los puntos del sitio de unión. Además, para colocarlos inicialmente emplea algoritmos adaptados de visión artificial. El promedio de área bajo la curva obtenida por este método es inferior a la generada por FRODRUG en tan sólo un 2 %.

DOCK [110] es el método comparado que menor área bajo la curva obtiene, siendo un 8 % inferior a la obtenida por FRODRUG. Sobre este método no se posee información relativa a tiempos de ejecución, sin embargo se dispone de información correspondiente a una versión anterior de DOCK (3.5.54). En [85] se indica que con esta versión el cribado virtual de cada

target individual se lleva a cabo en un tiempo que varía desde varias horas hasta varios días sobre un Pentium 4 a 2,8 GHz.

PhDOCK [99] es un método basado en farmacóforo implementado dentro de DOCK. Este emplea la misma técnica que FRODRUG con respecto a la revaluación de sus soluciones con el fin de afinar un poco el resultado. PhDOCK refina las 10 mejores soluciones encontradas. La diferencia en el área obtenida por este método con respecto a FRODRUG no es muy elevada, siendo tan sólo un 4 % inferior.

BINDSURF [202] sólo informa de curvas ROC relativas a 3 de los 40 *targets* de DUD. Estos son *tk*, *mr* y *gpb*. Las áreas bajo la curva obtenidas son 0,7, 0,69 y 0,67 respectivamente. FRODRUG obtiene 0,69, 0,82 y 0,54 para estos *targets* respectivamente. BINDSURF es el único de los métodos de cribado virtual comparados que cuenta con implementación para GPU utilizando CUDA. El tiempo promedio registrado por este método para realizar el ajuste molecular de un único ligando en una conformación sobre una proteína depende de la cantidad de puntos del espacio comprobados. Este tiempo varía desde los 6,8 segundos comprobando 64 puntos hasta los 54,5 segundos comprobando 356 puntos. Este es el método más competitivo en tiempo, con un promedio de 106,25 ms por punto del sitio de unión. Sin embargo, FRODRUG obtiene un rendimiento más elevado con 70,38 ms por punto.

Se puede observar que FRODRUG comparte el octavo puesto de la Tabla 5.39 junto con ICM [1]. Mientras que la función de puntuación de FRODRUG cuenta con tan solo dos términos energéticos, la de ICM cuenta con cinco; estos se calculan mediante el uso del campo de fuerza molecular de Merck [76] y el campo de fuerza ECEPP/3 [141].

La diferencia en el área obtenida por FRODRUG con respecto al método que mejor realiza el cribado es de 0,1. Esta no es una cantidad en absoluto despreciable. Sin embargo debe tenerse en cuenta que el resto de métodos de cribado virtual cuentan con flexibilidad de ligandos, al contrario que FRODRUG.

En resumen, se ha desarrollado una herramienta de *virtual screening* y *docking* que produce resultados prometedores tanto en rendimiento como en precisión. En cuestión de ajuste molecular supera a los métodos actuales mientras que se obtienen unos resultados razonables con respecto al cribado virtual. Además, FRODRUG es capaz de realizar los ajustes en tiempos que son órdenes de magnitud menores a los registrados en otros métodos. Sin embargo, para poder ser competitivos es necesario continuar el desarrollo con la inclusión y optimización de nuevos potenciales y la consideración de la flexibilidad en ligandos.

Capítulo 6

Conclusiones

El desarrollo de esta tesis ha llevado a obtener las siguientes conclusiones:

1. Se ha adaptado y optimizado el nuevo método *Fast Bessel Matching* para alineamiento de imágenes 2D sobre CPU y GPU y se ha comprobado que:

   1.1. FBM es capaz de realizar alineamientos correctos tanto con imágenes simuladas como experimentales con un error inferior a 1 pixel.

   1.2. El muestreo óptimo de los píxeles de una imagen de microscopía se obtiene fijando el ancho de banda al tamaño del lado de esta en píxeles.

   1.3. La versión para sistemas de memoria distribuida con MPI escala de un modo prácticamente lineal.

   1.4. La implementación sobre tarjetas gráficas NVIDIA logra realizar los alineamientos, en una tarjeta GK104, hasta 50 veces más rápido que la versión secuencial prácticamente con la misma precisión.

   1.5. Para procesar los alineamientos sobre dispositivos gráficos se han desarrollado y optimizado dos *kernels*:

      1.5.1. El *kernel* que calcula la integral, cuyas principales características son:

         - El patrón de acceso a memoria global es aceptable, con promedios de 2,69 transacciones por solicitud en lectura y 1,97 en escritura.
         - La cantidad de instrucciones por ciclo ejecutadas es buena, con un valor de 3,74.
         - La divergencia de ejecución entre *threads* del mismo *warp* es baja, con tan sólo un 6,56 %.

      1.5.2. El *kernel* de búsqueda de los mejores ángulos en el alineamiento:

         - Los accesos a memoria global son óptimos, con una transacción por solicitud de acceso tanto en lectura como en escritura.
         - La cantidad de instrucciones por ciclo de reloj ejecutadas es aceptable para un proceso poco eficiente en paralelo: 2,33.

- La divergencia de ejecución entre *threads* del mismo *warp* es baja, con tan sólo un 6,7 %.

1.6. Se ha comparado FBM con *Fast Rotational Matching 2D*:

1.6.1. Primero se ha optimizado FRM2D, método incluido en el paquete de herramientas de microscopía electrónica EMAN, y se ha conseguido obtener un rendimiento 10,39 veces más alto conservando la misma precisión.

1.6.2. Usando imágenes de microscopía simuladas con un tamaño de 128 x 128 píxeles se ha podido observar que la versión optimizada de FRM2D es ligeramente más precisa que FBM pero más de tres veces más lenta.

1.6.3. Utilizando imágenes experimentales reales a baja resolución (64 x 64 píxeles) los resultados obtenidos con ambos métodos son comparables tanto en tiempo como en precisión.

1.6.4. Empleando imágenes experimentales de 320 x 320 píxeles, que proporcionan los microscopios electrónicos actuales, FBM proporciona un 17,35 % más de ajustes correctos que la versión optimizada de FRM2D funcionando 500 veces más rápido. Sobre un Intel i7-950, el tiempo promedio por ajuste de FBM es de 2,59 ms mientras que el de FRM2D optimizado es de 1,33 s.

2. Se ha adaptado y optimizado sobre CPU y GPU el nuevo método FRODRUG para cribado virtual de proteínas y se ha comprobado que:

2.1. Dada una proteína, FRODRUG es capaz de identificar con éxito muchos de los ligandos que se unen a ella desechando gran cantidad de los que no lo hacen.

2.2. La versión paralela de FRODRUG logra realizar el cribado virtual 30 veces más rápido que la versión secuencial y prácticamente con la misma precisión usando una GPU GK104. El tiempo requerido para el ajuste de un único punto traslacional en esta GPU es de 21,29 μs y en promedio, un ligando se puede cribar en 70,38 ms.

2.3. La capacidad de escalado de la versión para sistemas de memoria distribuida que emplea MPI es prácticamente lineal.

2.4. Empleando un clúster de GPU compuesto por dos GPUs GK104, una GF110 y una GF100 el tiempo de ajuste por punto traslacional es de 6,53 μs. Se estima que FRODRUG sería capaz de realizar el cribado de un millón de ligandos en un promedio de seis horas, tiempos que son inalcanzables por los actuales métodos de cribado virtual basados en estructura.

2.5. Se han desarrollado y optimizado tres *kernels* para realizar el cribado virtual sobre dispositivos gráficos. Estos son:

2.5.1. El *kernel* que calcula la integral del método:

- Su patrón de acceso de lectura a memoria global es casi óptimo, produciéndose en promedio una cantidad de 1,05 transacciones por solicitud. Las escrituras son óptimas con una transacción por solicitud.
- La cantidad de instrucciones por ciclo ejecutadas por este *kernel* es relativamente baja: 2,82.

- La ejecución de los diferentes *threads* dentro del mismo *warp* se realiza de forma óptima con un 0 % de divergencia.

2.5.2. El *kernel* que genera los volúmenes de correlación:

- Los accesos de lectura a memoria global son óptimos, generándose una transacción por solicitud. Sin embargo, las escrituras generan en promedio 1,97 transacciones por solicitud.
- La cantidad de instrucciones por ciclo ejecutadas por este *kernel* es aceptable: 2,97.
- La ejecución de los diferentes *threads* dentro del mismo *warp* se realiza con un 0 % de divergencia.

2.5.3. El *kernel* que realiza la búsqueda de los ángulos de ajuste:

- Sus accesos a memoria global son óptimos, generándose una transacción por solicitud de acceso tanto en lectura como en escritura.
- La cantidad de instrucciones por ciclo de reloj ejecutadas es buena considerando que es un proceso poco eficiente en paralelo: 1,82.
- La divergencia de ejecución entre *threads* del mismo *warp* es muy baja, con tan sólo un 5,42 %.

2.6. Comparativamente FRODRUG proporciona unos resultados aceptables:

2.6.1. Se ha logrado llevar a cabo de forma exitosa el ajuste molecular para el 97,65 % de los complejos proteína-ligando del *benchmark* Astex con un error inferior a 2 Å.

2.6.2. Se ha logrado realizar el cribado virtual obteniendo unos resultados que superan a una selección aleatoria de ligandos en el 90 % de los casos que componen el conjunto de pruebas DUD y en el 87 % de los de DUD-E. El área promedio obtenida en los casos de DUD es de 0,63 y de 0,62 en los de DUD-E.

2.6.3. Con respecto a otros métodos de cribado virtual, FRODRUG se sitúa en un punto intermedio en lo relativo a la precisión, sin embargo es capaz de realizar los ajustes en tiempos que son órdenes de magnitud menores a los registrados en otros programas. Esta versión preliminar de FRODRUG puede ser mejorada considerando la flexibilidad de ligandos e incluyendo nuevos potenciales energéticos.

6.1. Aportaciones principales

A lo largo de este trabajo de tesis se han obtenido resultados parciales cuya difusión ha generado las siguientes publicaciones en congresos internacionales indexadas en Scopus:

- S. García, J. Kovacs, P Chacón. *Ultra-fast registration of 2D electron microscopy images.* 11th International Conference on Modeling and Applied Simulation, MAS 2012, Held at the International Multidisciplinary Modeling and Simulation Multiconference, I3M 2012, pp. 212 - 217. URL: `http://www.scopus.com/inward/record.url?eid=2-s2.0-84898848084&partnerID=40&md5=70c575e1d46e20659c12781cf425020f`

- S. García, E. Ramirez-Aportela, J.I. Garzon, P. Chacón, A. Sanz Montemayor, R. Cabido. *FRODRUG: A Virtual Screening GPU Accelerated Approach for Drug Discovery*. Parallel, Distributed and Network-Based Processing (PDP), 2014. 22nd Euromicro International Conference. pp. 594 - 600. URL: `http://www.scopus.com/inward/record.url?eid=2-s2.0-84899434160&partnerID=40&md5=e739b1226f6e98f722fb571806da8e95`

Además, se enumeran otros trabajos en congresos internacionales indexados en Web Of Science relacionados de forma menos directa con la investigación de esta tesis:

- J.R. López-Blanco, E. Ramirez, S. García, P. Chacon. *Imods: Fast Exploration of Macromolecular Collective Motions*. Biophysical Society 58th Annual Meeting, San Francisco. Publicación: Biophysical Journal, Volume 106, Issue 2, Supplement 1, 28 January 2014, Page 653a. Número de acceso Web Of Science: WOS:000337000403645.

- J.R. López-Blanco, E. Ramirez, S. García, P. Chacon. *Exploring Macromolecular Machine Motions*. Biophysical Society 57th Annual Meeting, Philadelphia. Publicación: Biophysical Journal, Volume 104, Issue 2, Supplement 1, 29 January 2013, Page 69a. Número de acceso Web Of Science: WOS:000316074300354.

- J.R. López-Blanco, E. Ramirez, S. García, P. Chys, P. Chacon. *Modeling Macromolecular Flexibility with Normal Mode Analysis in Internal Coordinates*. Biophysical Society 56th Annual Meeting, San Diego. Publicación: Biophysical Journal, Volume 102, Issue 3, Supplement 1, 31 January 2012, Page 394a. Número de acceso Web Of Science: WOS:000321561202583.

- J.R. Lopéz-Blanco, E. Ramírez, P. Chys, S. García and P. Chacón. *Addressing Macromolecular Flexibility*. 22nd IUBMB & 37th FEBS Congress, Sevilla. Publicación: FEBS Journal, Volume 279, Supplement 1, September 2012, Page 529. Número de acceso Web Of Science: WOS:000308128602727.

- J.L. Herraiz, S. España, S. García, R. Cabido, A.S. Montemayor, M. Desco, J.J. Vaquero, J.M. Udias. *GPU acceleration of a fully 3D Iterative Reconstruction Software for PET using CUDA*. 2009 IEEE Nuclear Science Symposium and Medical Imaging Conference. Publicación: Nuclear Science Symposium Conference Record (NSS/MIC), October 2009, pages 4064-4067. Número de acceso Web Of Science: WOS:000280505102148.

Apéndices

Apéndice A

Funciones MPI

Las funciones síncronas empleadas por el proceso maestro son las siguientes:

```
MPI_Send(void *buffer,
         int count,
         MPI_Datatype datatype,
         int dest,
         int tag,
         MPI_Comm comm)
```

MPI_Send es la función síncrona que se utiliza para enviar datos. Los datos que se desean enviar se almacenan en el vector apuntado por *buffer*. La variable *count* indica la cantidad de datos de tipo *datatype* que hay en el vector. El proceso al que se envía el mensaje se indica mediante la variable *dest*. La variable *tag* se puede utilizar para identificar el tipo de mensaje que se está enviando. Usualmente, cada tipo de mensaje definido en el protocolo de comunicación está asociado a un *tag* único. Por último, la variable *comm* se utiliza para especificar el comunicador de MPI a través del que se envía el mensaje. Un comunicador de MPI agrupa varios procesos definiendo un identificador único para cada uno de ellos. En la adaptación para sistemas de memoria distribuida se utiliza únicamente el comunicador por defecto (MPI_COMM_WORLD), que engloba todos los procesos creados.

```
MPI_Recv(void *buffer,
         int count,
         MPI_Datatype datatype,
         int source,
         int tag,
         MPI_Comm comm,
         MPI_Status *status)
```

MPI_Recv es la función síncrona que se utiliza para recibir datos. El vector apuntado por el puntero *buffer* almacenará como máximo una cantidad *count* de datos de tipo *datatype*. La variable *source* sirve para indicar de qué proceso en concreto se desea recibir datos. El proceso maestro utilizará la constante MPI_ANY_SOURCE para recibir mensajes de cualquier otro proceso en el comunicador empleado. La variable *tag* se utiliza para indicar que únicamente se desea recoger un mensaje que esté etiquetado con el número especificado. El comunicador

(*comm*) tiene la misma función que en la función de envío: especificar el conjunto de procesos con el que se trabaja. La variable *status* contiene información sobre el proceso emisor del mensaje, la cantidad de datos enviados, el *tag* empleado, etc.

Las funciones asíncronas empleadas por los procesos esclavos son las siguientes:

```
MPI_Isend(void *buffer,
          int count,
          MPI_Datatype datatype,
          int dest,
          int tag,
          MPI_Comm comm,
          MPI_Request *request)
```

La función MPI_Isend es muy similar a su versión síncrona. Las variables son las mismas y cumplen la misma función, salvo por la nueva variable de tipo MPI_Request. En esta variable se guarda la información relacionada con la solicitud de comunicación. Al ser MPI_Isend una función asíncrona, esta retorna inmediatamente sin haber completado el envío. Mediante la variable *request* se puede comprobar en cualquier momento el estado de la solicitud.

Para comprobar el estado de una solicitud de comunicación asíncrona, se emplea la siguiente función:

```
MPI_Test(MPI_Request *request,
         int *flag,
         MPI_Status *status)
```

La variable *request* se corresponderá con la de la petición que se desee comprobar. La variable *flag* devolverá el valor 1 si la solicitud de comunicación se ha completado. La variable *status* contendrá la información del envío del mismo modo que en la versión síncrona.

La versión asíncrona de la función de recepción de mensajes está definida del siguiente modo:

```
MPI_Irecv(void *buffer,
          int count,
          MPI_Datatype datatype,
          int source,
          int tag,
          MPI_Comm comm,
          MPI_Request *request)
```

Las variables de esta función se comportan del mismo modo que las ya explicadas anteriormente.

Apéndice B

Scripts para la generación de imágenes de microscopía con Xmipp

Listado B.1: Fichero de parámetros ref.par

```
# XMIPP_STAR_1 *
data_block1
# X and Y projection dimensions [Xdim Ydim]
_dimensions2D   '128 128'
# Rotation range and number of samples [Start Finish NSamples]
_projRotRange    '0 342 10'
# Rotation angle added noise  [noise (bias)]
_projRotNoise   '0'
# Tilt range and number of samples for Tilt
_projTiltRange    '20 160 8'
# Tilt angle added noise
_projTiltNoise   '0'
# Psi range and number of samples
_projPsiRange    '0 0 0'
# Psi added noise
_projPsiNoise   '0'
# Noise applied to pixels [noise (bias)]
_noisePixelLevel   '0'
# Noise applied to particle center coordenates [noise (bias)]
_noiseCoord   '0'
```

Listado B.2: Fichero de configuración de CTF

```
# XMIPP_STAR_1 *
#
data_noname
 _CTF_Sampling_rate 3
 _CTF_Voltage 300
 _CTF_Defocus_U 20000
```

```
 _CTF_Defocus_V 19000
 _CTF_Defocus_angle 23.4347
 _CTF_Spherical_aberration 2.
 _CTF_Chromatic_aberration 1.91072e-05
 _CTF_Energy_loss 0.00844884
 _CTF_Lens_stability 0
 _CTF_Convergence_cone 0.000105177
 _CTF_Longitudinal_displacement 462.366
 _CTF_Transversal_displacement 0.000508047
 _CTF_Q0 0.07
 _CTF_K 1.04159
 _CTFBG_Gaussian_K 11.1188
 _CTFBG_Gaussian_SigmaU 14419.4
 _CTFBG_Gaussian_SigmaV 14347.8
 _CTFBG_Gaussian_CU 0.0227578
 _CTFBG_Gaussian_CV 0.0233244
 _CTFBG_Gaussian_Angle 3.54688e-07
 _CTFBG_Sqrt_K 22.5324
 _CTFBG_Sqrt_U 6.88956
 _CTFBG_Sqrt_V 7.14675
 _CTFBG_Sqrt_Angle 86.7604
 _CTFBG_Baseline 0.162732
 _CTFBG_Gaussian2_K 1.5983
 _CTFBG_Gaussian2_SigmaU 502.434
 _CTFBG_Gaussian2_SigmaV 499.628
 _CTFBG_Gaussian2_CU 0.0228452
 _CTFBG_Gaussian2_CV 0.0267223
 _CTFBG_Gaussian2_Angle 3.70661
 _CTFCrit_Fitting 0.331973
 _CTFCrit_Corr13 0.469147
 _CTFCrit_PSDStdQ 0.954441
 _CTFCrit_PSDPCA1 69.3456
 _CTFCrit_PSDPCARuns 4.50895
```

Listado B.3: Fichero de parámetros rand.par

```
# XMIPP_STAR_1 *
data_block1
# X and Y projection dimensions [Xdim Ydim]
_dimensions2D   '128 128'
# Rotation range and number of samples [Start Finish NSamples]
_projRotRange    '0 342 10'
# Rotation angle added noise  [noise (bias)]
_projRotNoise   '0'
# Tilt range and number of samples for Tilt
_projTiltRange    '20 160 8'
# Tilt angle added noise
_projTiltNoise   '0'
```

```
# Psi range and number of samples
_projPsiRange    '0 180 50'
# Psi added noise
_projPsiNoise   '0'
# Noise applied to pixels [noise (bias)]
_noisePixelLevel   '0'
# Noise applied to particle center coordenates [noise (bias)]
_noiseCoord   '1.5'
```

Listado B.4: *Script* para la generación del conjunto de pruebas de microscopía electrónica

```
#!/bin/bash
rm *.xmd
rm *.stk

#Reference images
xmipp_phantom_transform -i 3M3Y.pdb
                        -o 3M3YC.pdb
                        --operation scale 0.7 0.7 0.7
                        --center_pdb

xmipp_phantom_project -i 3M3YC.pdb
                      -o r
                      --params ref.par
                      --sampling_rate 3

xmipp_phantom_simulate_microscope -i r.stk
                                  -o ref.spi
                                  --ctf ctfdesc.txt
                                  --noNoise

#Experimental images
xmipp_phantom_project -i 3M3YC.pdb
                      -o rand
                      --params rand.par
                      --sampling_rate 3

#apply ctf simulation
xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp1.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 1.0
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp2.spi
```

```
                                    --ctf ctfdesc.txt
                                    --targetSNR 0.33
                                    --after_ctf_noise
                                    --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                    -o exp3.spi
                                    --ctf ctfdesc.txt
                                    --targetSNR 0.15
                                    --after_ctf_noise
                                    --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                    -o exp4.spi
                                    --ctf ctfdesc.txt
                                    --targetSNR 0.08
                                    --after_ctf_noise
                                    --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                    -o exp5.spi
                                    --ctf ctfdesc.txt
                                    --targetSNR 0.05
                                    --after_ctf_noise
                                    --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                    -o exp6.spi
                                    --ctf ctfdesc.txt
                                    --targetSNR 0.037
                                    --after_ctf_noise
                                    --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                    -o exp7.spi
                                    --ctf ctfdesc.txt
                                    --targetSNR 0.028
                                    --after_ctf_noise
                                    --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                    -o exp8.spi
                                    --ctf ctfdesc.txt
                                    --targetSNR 0.021
                                    --after_ctf_noise
                                    --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
```

```
                                  -o exp9.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.015
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp10.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.011
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp11.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.008
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp12.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.007
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp13.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.006
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp14.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.005
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp15.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.004
                                  --after_ctf_noise
                                  --defocus_change 50
```

```
xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp16.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.003
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp17.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.002
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp18.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.0015
                                  --after_ctf_noise
                                  --defocus_change 50

xmipp_phantom_simulate_microscope -i rand.xmd
                                  -o exp19.spi
                                  --ctf ctfdesc.txt
                                  --targetSNR 0.001
                                  --after_ctf_noise
                                  --defocus_change 50
```

Bibliografía

[1] R. Abagyan, M. Totrov, and D. Kuznetsov. ICM–a new method for protein modeling and design: applications to docking and structure prediction from the distorted native conformation. *Journal of computational chemistry*, 15(5):488–506, 1994.

[2] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. United States Departament of Commerce, 1972.

[3] J.C. Adams and P.N. Swarztrauber. Spherepack 2.0: A model development facility. *NCAR Technical Note NCAR/TN-436+STR*, 1997. doi: DOI:10.5065/D6Z899CF.

[4] J.J. Agresti, E. Antipov, A.R. Abate, K. Ahn, A.C. Rowat, J.C. Baret, M. Marquez, A.M. Klibanov, A.D. Griffiths, and D.A. Weitz. Ultrahigh-throughput screening in drop-based microfluidics for directed evolution. *Proceedings of the National Academy of Sciences*, 107 (9):4004–4009, 2010.

[5] D. Akhmed-Zaki, N. Danaev, B. Matkerim, and A. Bektemessov. Design of Distributed Parallel Computing Using by MapReduce/MPI Technology. In Victor Malyshkin, editor, *Parallel Computing Technologies*, volume 7979 of *Lecture Notes in Computer Science*, pages 139–148. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39957-2. doi: 10.1007/978-3-642-39958-9_12. URL `http://dx.doi.org/10.1007/978-3-642-39958-9_12`.

[6] R. Alasdair, A. Bruce, J.G. Mills, and A.G. Smith. *CHIMP/MPI user guide*, 1994.

[7] A. Amunts, A. Brown, X.C. Bai, J.L. Llácer, T. Hussain, P. Emsley, F. Long, G. Murshudov, S.H.W. Scheres, and V. Ramakrishnan. Structure of the yeast mitochondrial large ribosomal subunit. *Science*, 343(6178):1485–1489, 2014.

[8] R. Apweiler, A. Bairoch, C.H. Wu, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M.J. Martin, D.A. Natale, C. O'Donovan, N. Redaschi, and L.S.L. Yeh. Uniprot: the universal protein knowledgebase. *Nucleic Acids Research*, 32(suppl 1):D115–D119, 2004. doi: 10.1093/nar/gkh131. URL `http://nar.oxfordjournals.org/content/32/suppl_1/D115.abstract`.

[9] X. Bai, I.S. Fernandez, G. McMullan, and S.H.W. Scheres. Ribosome structures to near-atomic resolution from thirty thousand cryo-EM particles. *eLife*, 2, 2013.

[10] J. Bajorath. Integration of virtual and high-throughput screening. *Nat Rev Drug Discov*, 1:882–894, 2002. ISSN 1474-1776. URL `http://dx.doi.org/10.1038/nrd941`.

[11] T.S. Baker and R.H. Cheng. A model-based approach for determining orientations of biological macromolecules imaged by cryoelectron microscopy. *Journal of Structural Biology*, 116(1):120 – 130, 1996. ISSN 1047-8477. doi: http://dx.doi.org/10.1006/jsbi.1996.0020. URL `http://www.sciencedirect.com/science/article/pii/S1047847796900209`.

[12] L.A. Barroso. The price of performance. *Queue*, 3(7):48–53, September 2005. ISSN 1542-7730.

[13] W.T. Baxter, R.A. Grassucci, H. Gao, and J. Frank. Determination of signal-to-noise ratios and spectral SNRs in cryo-EM low-dose imaging of molecules. *Journal of Structural Biology*, 166(2):126 – 132, 2009.

[14] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[15] Bessel Functions from GNU C Library, 2014. URL `http://www.gnu.org/software/gsl/manual/html_node/Bessel-Functions.html`.

[16] D. Blythe. The direct3d 10 system. *ACM Trans. Graph.*, 25(3):724–734, July 2006. ISSN 0730-0301. doi: 10.1145/1141911.1141947. URL `http://doi.acm.org/10.1145/1141911.1141947`.

[17] A. Bondi. Van der waals volumes and radii. *The Journal of Physical Chemistry*, 68(3):441–451, 1964.

[18] G. Bosilca, A. Bouteiller, F. Cappello, S. Djilali, G. Fedak, C. Germain, T. Herault, P. Lemarinier, O. Lodygensky, F. Magniette, V. Neri, and A. Selikhov. MPICH-V: Toward a Scalable Fault Tolerant MPI for Volatile Nodes. In *Supercomputing, ACM/IEEE 2002 Conference*, pages 29–29, Nov 2002. doi: 10.1109/SC.2002.10048.

[19] D. M. Brink and G. R. Satchler. *Angular Momentum*. Oxford: Clarendon Press, 1993.

[20] A. Brunton, J. Lang, and E. Dubois. Spherical Harmonic Transforms and Convolutions on the GPU. *Journal of Graphics, GPU, and Game Tools*, 15(1):13–27, 2010.

[21] A. Buades, B. Coll, and J.M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490?530, 2005. doi: 10.1137/040616024.

[22] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: Stream Computing on Graphics Hardware. *ACM Trans. Graph.*, 23(3):777–786, August 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015800. URL `http://doi.acm.org/10.1145/1015706.1015800`.

[23] Business Wire. PCI-SIG Announces PCI Express 4.0 Evolution to 16 GT/s, Twice the Throughput of PCI Express 3.0 Technology, Nov 29 2011.

[24] A.C. Cabrera, J. Klett, H.G. Dos Santos, A. Perona, R. Gil-Redondo, S.M. Francis, E.M. Priego., F. Gago, and A. Morreale. CRDOCK: An Ultrafast Multipurpose Protein-Ligand Docking Tool. *Journal of Chemical Information and Modeling*, 52(8):2300–2309, 2012.

[25] D. Castaño-Díez, D. Moser, A. Schoenegger, S. Pruggnaller, and A.S. Frangakis. Performance evaluation of image processing algorithms on the GPU. *Journal of Structural Biology*, 164(1):153 – 160, 2008. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2008.07.006. URL `http://www.sciencedirect.com/science/article/pii/S1047847708001792`.

[26] D. Castaño-Díez, M. Scheffer, A. Al-Amoudi, and A. S. Frangakis. Alignator: A GPU powered software package for robust fiducial-less alignment of cryo tilt-series. *Journal of Structural Biology*, 170(1):117 – 126, 2010. ISSN 1047-8477.

[27] M. Cecchi, F. Capannini, A. Dorigo, A. Ghiselli, F. Giacomini, A. Maraschini, M. Marzolla, S. Monforte, F. Pacini, L. Petronzio, and F. Prelz. The gLite Workload Management System. In Nabil Abdennadher and Dana Petcu, editors, *Advances in Grid and Pervasive Computing*, volume 5529 of *Lecture Notes in Computer Science*, pages 256–268. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01670-7. doi: 10.1007/978-3-642-01671-4_24. URL `http://dx.doi.org/10.1007/978-3-642-01671-4_24`.

[28] G. Chrysos. Intel Xeon Phi Coprocessor (codename knights corner). In *Proceedings of the 24th Hot Chips Symposium, HC*, 2012.

[29] Y. Cong, J. A. Kovacs, and W. Wriggers. 2D fast rotational matching for image processing of biophysical data. *Journal of Structural Biology*, 144(1-2):51 – 60, 2003. ISSN 1047-8477.

[30] Y. Cong, W. Jiang, S. Birmanns, Z. H. Zhou, W. Chiu, and W. Wriggers. Fast rotational matching of single-particle images. *Journal of Structural Biology*, 152(2):104 – 112, 2005. ISSN 1047-8477.

[31] Y. Cong, M. L. Baker, J. Jakana, D. Woolford, E. J. Miller, S. Reissmann, R. N. Kumar, A. M. Redding-Johanson, T. S. Batth, A. Mukhopadhyay, S. J. Ludtke, J. Frydman, and W. Chiu. 4.0-Å resolution cryo-EM structure of the mammalian chaperonin TRiC/CCT reveals its unique subunit arrangement. *Proceedings of the National Academy of Sciences*, 107(11):4967–4972, 2010.

[32] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297 – 301, 1965.

[33] J. B. Cross, D. C. Thompson, B. K. Rai, J. C. Baber, K. Y. Fan, Y. Hu, and C. Humblet. Comparison of several molecular docking programs: Pose prediction and virtual screening accuracy. *Journal of Chemical Information and Modeling*, 49(6):1455–1474, 2009.

[34] R.A. Crowther. Procedures for three-dimensional reconstruction of spherical viruses by fourier synthesis from electron micrographs. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 261(837):221–230, 1971. ISSN 0080-4622. doi: 10.1098/rstb.1971.0054.

[35] R.A. Crowther, D.J. DeRosier, and A. Klug. The reconstruction of a three-dimensional structure from projections and its application to electron microscopy. *Proceedings of the*

Royal Society of London A: Mathematical, Physical and Engineering Sciences, 317(1530): 319–340, 1970. ISSN 0080-4630. doi: 10.1098/rspa.1970.0119.

[36] M.D. Cummings, R.L. DesJarlais, A.C. Gibbs, V. Mohan, and E.P. Jaeger. Comparison of automated docking programs as virtual screening tools. *Journal of Medicinal Chemistry*, 48(4):962–976, 2005.

[37] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113, January 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327492. URL `http://doi.acm.org/10.1145/1327452.1327492`.

[38] D.J. DeRosier and A. Klug. Reconstruction of three dimensional structures from electron micrographs. *Nature*, 217:130–134, 1968.

[39] J. Diaz, C. Muñoz-Caro, and A. Niño. A survey of parallel programming models and tools in the multi and many-core era. *Parallel and Distributed Systems, IEEE Transactions on*, 23(8):1369–1386, Aug 2012. ISSN 1045-9219. doi: 10.1109/TPDS.2011.308.

[40] L. Domanski, P. Vallotton, and D. Wang. Two and three-dimensional image deconvolution on graphics hardware. *18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation*, pages 1010 – 1016, 2009.

[41] A. Duran and M. Klemm. The intel many integrated core architecture. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 365–366. IEEE, 2012.

[42] J.D. Durrant, A.J. Friedman, K.E. Rogers, and J.A. McCammon. Comparing neural-network scoring functions and the state of the art: Applications to common library screening. *Journal of Chemical Information and Modeling*, 53(7):1726–1735, 2013.

[43] I.E. Dzyaloshinskii, E.M. Lifshitz, and L.P. Pitaevskii. General Theory of Van Der Waals' Forces. *Soviet Physics Uspekhi*, 4(2):153, 1961.

[44] E. Elsen, V. Vishal, M. Houston, V.S. Pande, P. Hanrahan, and E. Darve. N-body simulations on gpus. *CoRR*, abs/0706.3060, 2007. URL `http://arxiv.org/abs/0706.3060`.

[45] L. F. Estrozi and J. Navaza. Ab initio high-resolution single-particle 3D reconstructions: The symmetry adapted functions way. *Journal of Structural Biology*, 172(3):253 – 260, 2010. ISSN 1047-8477.

[46] L.F. Estrozi and J. Navaza. Fast projection matching for cryo-electron microscopy image reconstruction. *Journal of Structural Biology*, 162(2):324 – 334, 2008.

[47] Fastest Fourier Transform in the West 3.3.3, 2012. URL `http://www.fftw.org/`.

[48] FEI Company. An introduction to electron microscopy, 2010. URL `http://www.fei.com/documents/introduction-to-microscopy-document/`.

[49] R. Fernando and M.J. Kilgard. *The Cg Tutorial: The definitive guide to programmable real-time graphics*. Addison-Wesley, 2003.

[50] J.J. Fernández and S. Li. An improved algorithm for anisotropic nonlinear diffusion for denoising cryo-tomograms. *Journal of Structural Biology*, 144(1?2):152 – 161, 2003. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2003.09.010. URL `http://www.sciencedirect.com/science/article/pii/S104784770300176X`.

[51] J. Fernández-Recio, M. Totrov, and R. Abagyan. Soft protein-protein docking in internal coordinates. *Protein Sci*, 11:280–291, 2002.

[52] M. Flynn. Some computer organizations and their effectiveness. *Computers, IEEE Transactions on*, C-21(9):948–960, Sept 1972. ISSN 0018-9340. doi: 10.1109/TC.1972.5009071.

[53] I. Foster. Globus online: Accelerating and democratizing science through cloud-based services. *Internet Computing, IEEE*, 15(3):70–73, May 2011. ISSN 1089-7801. doi: 10.1109/MIC.2011.64.

[54] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kauffman, 1998.

[55] A.S. Frangakis and R. Hegerl. Noise reduction in electron tomographic reconstructions using nonlinear anisotropic diffusion. *Journal of Structural Biology*, 135(3):239 – 250, 2001. ISSN 1047-8477. doi: http://dx.doi.org/10.1006/jsbi.2001.4406. URL `http://www.sciencedirect.com/science/article/pii/S1047847701944065`.

[56] J. Frank. *Three-Dimensional Electron Microscopy of Macromolecular Assemblies*. Oxford University Press, New York, 2006.

[57] J. Frank, M. Radermacher, P. Penczek, J. Zhu, Y. Li, M. Ladjadj, and A. Leith. SPIDER and WEB: Processing and Visualization of Images in 3D Electron Microscopy and Related Fields. *Journal of Structural Biology*, 116(1):190 – 199, 1996. ISSN 1047-8477. doi: http://dx.doi.org/10.1006/jsbi.1996.0030. URL `http://www.sciencedirect.com/science/article/pii/S1047847796900301`.

[58] R. A. Friesner, J. L. Banks, R. B. Murphy, T. A. Halgren, J. J. Klicic, D. T. Mainz, M. P. Repasky, E. H. Knoll, M. Shelley, J. K. Perry, D. E. Shaw, P. Francis, and P. S. Shenkin. Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy. *Journal of Medicinal Chemistry*, 47(7):1739–1749, 2004.

[59] H.A. Gabb, R.M. Jackson, and M.J.E. Sternberg. Modelling protein docking using shape complementarity, electrostatics and biochemical information. *Journal of Molecular Biology*, 272(1):106 – 120, 1997.

[60] E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, and

T.S. Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Proceedings, 11th European PVM/MPI Users Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[61] J. I. Garzón, J. Kovacs, R. Abagyan, and P. Chacón. ADP_EM: fast exhaustive multi-resolution docking for high-throughput coverage. *Bioinformatics*, 23(4):427–433, 2007.

[62] J. I. Garzón, J. R. Lopéz-Blanco, C. Pons, J. Kovacs, R. Abagyan, J. Fernandez-Recio, and P. Chacon. FRODOCK: a new approach for fast rotational protein-protein docking. *Bioinformatics*, 25(19):2544–2551, 2009.

[63] A. Gaulton, L.J. Bellis, A.P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, and J.P. Overington. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 2011.

[64] R. Gil-Redondo, J. Estrada, A. Morreale, F. Herranz, J. Sancho, and A. R. Ortiz. VSDMIP: virtual screening data management on an integrated platform. *Journal of computer-aided molecular design*, 23(3):171–184, 2009.

[65] M.K. Gilson and B. Honig. Calculation of the total electrostatic energy of a macromolecular system: Solvation energies, binding energies, and conformational analysis. *Proteins: Structure, Function, and Bioinformatics*, 4(1):7–18, 1988. ISSN 1097-0134. doi: 10.1002/prot.340040104. URL `http://dx.doi.org/10.1002/prot.340040104`.

[66] P. J. Goodford. A computational procedure for determining energetically favorable binding sites on biologically important macromolecules. *Journal of Medicinal Chemistry*, 28(7): 849–857, 1985.

[67] N. Grigorieff. FREALIGN: High-resolution refinement of single particle structures. *Journal of Structural Biology*, 157(1):117 – 125, 2007. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2006.05.004. URL `http://www.sciencedirect.com/science/article/pii/S1047847706001699`. Software tools for macromolecular microscopy.

[68] N. Grigorieff and S.C. Harrison. Near-atomic resolution reconstructions of icosahedral viruses from electron cryo-microscopy. *Current Opinion in Structural Biology*, 21(2):265 – 273, 2011. ISSN 0959-440X. doi: http://dx.doi.org/10.1016/j.sbi.2011.01.008. URL `http://www.sciencedirect.com/science/article/pii/S0959440X11000236`.

[69] M.M. Gromiha, J. An, H. Kono, M. Oobatake, H. Uedaira, and A. Sarai. Protherm: Thermodynamic database for proteins and mutants. *Nucleic Acids Research*, 27(1):286–288, 1999. doi: 10.1093/nar/27.1.286. URL `http://nar.oxfordjournals.org/content/27/1/286.abstract`.

[70] M.M. Gromiha, Y. Yabuki, M.X. Suresh, A.M. Thangakani, M. Suwa, and K. Fukui. TMFunction: database for functional residues in membrane proteins. *Nucleic Acids Research*, 37(suppl 1):D201–D204, 2009. doi: 10.1093/nar/gkn672. URL `http://nar.oxfordjournals.org/content/37/suppl_1/D201.abstract`.

[71] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 1999.

[72] L. Gu, X. Li, and J. Siegel. An Empirically Tuned 2D and 3D FFT Library on CUDA GPU. In *Proceedings of the 24th ACM International Conference on Supercomputing*, ICS '10, pages 305 – 314. ACM, 2010.

[73] M. Guizar-Sicairos and J.C. Gutiérrez-Vega. Computation of quasi-discrete Hankel transforms of integer order for propagating optical wave fields. *J. Opt. Soc. Am. A*, 21(1): 53–58, Jan 2004.

[74] M. Hachman. Powerful Voodoo From 3Dfx. *Electronic Engineering Times*, page 16, Nov 25 1996.

[75] T. A. Halgren, R. B. Murphy, R. A. Friesner, H. S. Beard, L. L. Frye, W. T. Pollard, and J. L. Banks. Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening. *Journal of Medicinal Chemistry*, 47(7): 1750–1759, 2004.

[76] T.A. Halgren. Merck molecular force field. v. extension of mmff94 using experimental data, additional computational data, and empirical rules. *Journal of Computational Chemistry*, 17(5-6):616–641, 1996. ISSN 1096-987X. doi: 10.1002/(SICI)1096-987X(199604)17:5/6⟨616::AID-JCC5⟩3.0.CO;2-X. URL `http://dx.doi.org/10.1002/(SICI)1096-987X(199604)17:5/6<616::AID-JCC5>3.0.CO;2-X`.

[77] M. J. Hartshorn, M. L. Verdonk, G. Chessari, S. C. Brewerton, W. T. M. Mooij, P. N. Mortenson, and C. W. Murray. Diverse, high-quality test set for the validation of protein-ligand docking performance. *J. Med. Chem*, 50:726–741, 2007.

[78] Jr. Healy, D.M., D.N. Rockmore, P.J. Kostelec, and S. Moore. Ffts for the 2-sphere-improvements and variations. *Journal of Fourier Analysis and Applications*, 9(4):341–385, 1998. ISSN 1069-5869. doi: 10.1007/s00041-003-0018-9. URL `http://dx.doi.org/10.1007/s00041-003-0018-9`.

[79] Jim Held. Single-chip cloud computer. In *Intel Labs Single-chip Cloud Computer Symposium*, page 85, 2010.

[80] M. Helmer-Citterich and A. Tramontano. PUZZLE: A New Method for Automated Protein Docking Based on Surface Shape Complementarity. *Journal of Molecular Biology*, 235(3): 1021 – 1031, 1994. ISSN 0022-2836. doi: http://dx.doi.org/10.1006/jmbi.1994.1054. URL `http://www.sciencedirect.com/science/article/pii/S0022283684710540`.

[81] J.L. Herraiz, S. España, S. Garcia, R. Cabido, A.S. Montemayor, M. Desco, J.J. Vaquero, and J.M. Udias. GPU acceleration of a fully 3D Iterative Reconstruction Software for PET using CUDA. In *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*, pages 4064–4067, Oct 2009. doi: 10.1109/NSSMIC.2009.5402402.

[82] E. W. Hobson. *The theory of spherical and ellipsoidal harmonics*. Cambridge University Press, 1931.

[83] A. J. Hopfinger. *Conformational Properties of Macromolecules*. Academic Press: New York, 1973.

[84] M. Huang, M. Mehalel, R. Arvapalli, and S. He. An Energy Efficient 32-nm 20-MB Shared On-Die L3 Cache for Intel Xeon Processor E5 Family. *IEEE Journal of Solid-State Circuits*, 48(8):1954–1962, Aug 2013. ISSN 0018-9200. doi: 10.1109/JSSC.2013.2258815.

[85] N. Huang, B. K. Shoichet, and J. J. Irwin. Benchmarking sets for molecular docking. *Journal of Medicinal Chemistry*, 49(23):6789–6801, 2006.

[86] W. Humphrey, A. Dalke, and K. Schulten. VMD: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33 – 38, 1996. ISSN 0263-7855. doi: http://dx.doi.org/10.1016/0263-7855(96)00018-5.

[87] I.O. Hupca, J. Falcou, L. Grigori, and R. Stompor. Spherical Harmonic Transform with GPUs. In *Euro-Par 2011: Parallel Processing Workshops*, volume 7155, pages 355–366. Springer Berlin Heidelberg, 2012.

[88] J. Hursey, J.M. Squyres, T.I. Mattox, and A. Lumsdaine. The Design and Implementation of Checkpoint/Restart Process Fault Tolerance for Open MPI. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8, March 2007. doi: 10.1109/IPDPS.2007.370605.

[89] J.J. Irwin. Community benchmarks for virtual screening. *Journal of Computer-Aided Molecular Design*, 22(3-4):193–199, 2008.

[90] J.J. Irwin and B.K. Shoichet. ZINC - A Free Database of Commercially Available Compounds for Virtual Screening. *Journal of Chemical Information and Modeling*, 45 (1):177–182, 2005.

[91] J.J. Irwin, T. Sterling, M.M. Mysinger, E.S. Bolstad, and R.G. Coleman. ZINC: A Free Tool to Discover Chemistry for Biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, 2012.

[92] A. Jahn, G. Hinselmann, N. Fechner, and A. Zell. Optimal assignment methods for ligand-based virtual screening. *Journal of Cheminformatics*, 1(1):14, 2009.

[93] A.N. Jain. Scoring noncovalent protein-ligand interactions: A continuous differentiable function tuned to compute binding affinities. *Journal of Computer-Aided Molecular Design*, 10(5):427–440, 1996. ISSN 0920-654X. doi: 10.1007/BF00124474.

[94] A.N. Jain. Surflex: Fully Automatic Flexible Molecular Docking Using a Molecular Similarity-Based Search Engine. *Journal of Medicinal Chemistry*, 46(4):499–511, 2003.

[95] F. Jiang and S.H. Kim. Soft docking: Matching of molecular surface cubes. *Journal of Molecular Biology*, 219(1):79 – 102, 1991. ISSN 0022-2836. doi: http://dx.doi.org/10.

1016/0022-2836(91)90859-5. URL `http://www.sciencedirect.com/science/article/pii/0022283691908595`.

[96] W. Jiang, M.L. Baker, Q. Wu, C. Bajaj, and W. Chiu. Applications of a bilateral denoising filter in biological electron microscopy. *Journal of Structural Biology*, 144(1?2):114 – 122, 2003. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2003.09.028. URL `http://www.sciencedirect.com/science/article/pii/S1047847703002053`. Analytical Methods and Software Tools for Macromolecular Microscopy.

[97] G. Jones, P. Willett, and R.C. Glen. Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *Journal of Molecular Biology*, 245(1):43 – 53, 1995.

[98] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking. *Journal of molecular biology*, 267(3):727–748, 1997.

[99] D. Joseph-McCarthy, B. E. Thomas, M. Belmarsh, D. Moustakas, and J. C. Alvarez. Pharmacophore-based molecular docking to account for ligand flexibility. *Proteins: Structure, Function, and Bioinformatics*, 51(2):172–188, 2003.

[100] L. Joyeux and P. A. Penczek. Efficiency of 2D alignment methods. *Ultramicroscopy*, 92 (2):33 – 46, 2002. ISSN 0304-3991.

[101] A.A. Kantardjiev. Quantum.Ligand.Dock: protein-ligand docking with quantum entanglement refinement on a GPU system. *Nucleic Acids Research*, 40(W1):W415–W422, 2012.

[102] E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A.A. Friesem, C. Aflalo, and I.A. Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences*, 89(6): 2195–2199, 1992. URL `http://www.pnas.org/content/89/6/2195.abstract`.

[103] J. Kessenich. *OpenGL Shading Language 4.50 Specification*, 2015. URL `https://www.opengl.org/registry/doc/GLSLangSpec.4.50.pdf.pdf`.

[104] Khronos OpenCL Working Group. *The OpenCL Specification 2.0*, 2014. URL `https://www.khronos.org/registry/cl/specs/opencl-2.0.pdf`.

[105] E. Kilgariff and R. Fernando. The GeForce 6 series GPU architecture. In *GPU Gems 2*, chapter 30, pages 471–491. Addison Wesley, 2005.

[106] O. Korb, T. Stützle, and T.E. Exner. Empirical Scoring Functions for Advanced Protein-Ligand Docking with PLANTS. *Journal of Chemical Information and Modeling*, 49(1): 84–96, 2009. doi: 10.1021/ci800298z. URL `http://dx.doi.org/10.1021/ci800298z`.

[107] J. A. Kovacs and W. Wriggers. Fast rotational matching. *Acta Crystallographica Section D*, 58(8):1282–1286, Aug 2002.

[108] J. A. Kovacs, P. Chacón, Y. Cong, E. Metwally, and W. Wriggers. Fast rotational matching of rigid bodies by fast fourier transform acceleration of five degrees of freedom. *Acta Crystallographica Section D*, 59(8):1371–1376, Aug 2003.

[109] J. A. Kovacs, R. Abagyan, and M. Yeager. Fast Bessel Matching. *Journal of Computational and Theoretical Nanoscience*, 4(1):84–95, 2007.

[110] I.D. Kuntz, J.M. Blaney, S.J. Oatley, R. Langridge, and T.E. Ferrin. A geometric approach to macromolecule-ligand interactions. *Journal of Molecular Biology*, 161(2):269 – 288, 1982. ISSN 0022-2836.

[111] J. E. Lennard-Jones. On the Determination of Molecular Fields. *Proc. R. Soc. Lond. A*, 106(738):463–477, 1924.

[112] X. Li, N. Grigorieff, and Y. Cheng. GPU-enabled FREALIGN: Accelerating single particle 3D reconstruction and refinement in fourier space on graphics processors. *Journal of Structural Biology*, 172(3):407 – 412, 2010. ISSN 1047-8477.

[113] Y. Liang, E.Y. Ke, and Z.H. Zhou. IMIRS: a high-resolution 3D reconstruction package integrated with a relational image database. *Journal of Structural Biology*, 137(3):292 – 304, 2002. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/S1047-8477(02)00014-X. URL `http://www.sciencedirect.com/science/article/pii/S104784770200014X`.

[114] J. W. Liebeschuetz, J. C. Cole, and O. Korb. Pose prediction and virtual screening performance of gold scoring functions in a standardized test. *Journal of Computer-Aided Molecular Design*, 26(6):737–748, 2012. ISSN 0920-654X. doi: 10.1007/s10822-012-9551-4.

[115] E. Lindahl, B. Hess, and D. van der Spoel. GROMACS 3.0: a package for molecular simulation and trajectory analysis. *Molecular modeling annual*, 7(8):306–317, 2001. ISSN 0949-183X. doi: 10.1007/s008940100045. URL `http://dx.doi.org/10.1007/s008940100045`.

[116] E. Lindholm, M.J. Kilgard, and H. Moreton. A user-programmable vertex engine. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 149–158, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X. doi: 10.1145/383259.383274. URL `http://doi.acm.org/10.1145/383259.383274`.

[117] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. NVIDIA Tesla: A Unified Graphics and Computing Architecture. *Micro, IEEE*, 28(2):39–55, March 2008. ISSN 0272-1732. doi: 10.1109/MM.2008.31.

[118] H. Liu, L. Cheng, S. Zeng, C. Cai, Z.H. Zhou, and Q. Yang. Symmetry-adapted spherical harmonics method for high-resolution 3d single-particle reconstructions. *Journal of Structural Biology*, 161(1):64 – 73, 2008. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2007.09.016. URL `http://www.sciencedirect.com/science/article/pii/S1047847707002353`.

[119] S.J. Ludtke, P.R. Baldwin, and W. Chiu. EMAN: Semiautomated Software for High-Resolution Single-Particle Reconstructions. *Journal of Structural Biology*, 128(1):82 – 97, 1999.

[120] T.M. MacRobert. *Spherical harmonics: An elementary treatise on harmonic functions, with applications*. Pergamon Press, 1967.

[121] R. Marabini, I. M. Masegosa, M. C. San Martín, S. Marco, J. J. Fernández, L. G. de la Fraga, C. Vaquerizo, and J. M. Carazo. Xmipp: An image processing package for electron microscopy. *Journal of Structural Biology*, 116(1):237 – 240, 1996. ISSN 1047-8477.

[122] A.I. Margaris. Local Area Multicomputer (LAM-MPI). *Computer and Information Science*, 6(2):1–8, 2013.

[123] W.R. Mark, R.S. Glanville, K. Akeley, and M.J. Kilgard. Cg: A system for programming graphics hardware in a c-like language. *ACM Trans. Graph.*, 22(3):896–907, July 2003. ISSN 0730-0301. doi: 10.1145/882262.882362. URL `http://doi.acm.org/10.1145/882262.882362`.

[124] R. Marroquim and A. Maximo. Introduction to gpu programming with glsl. In *Computer Graphics and Image Processing (SIBGRAPI TUTORIALS), 2009 Tutorials of the XXII Brazilian Symposium on*, pages 3–16, Oct 2009. doi: 10.1109/SIBGRAPI-Tutorials.2009.9.

[125] R.L. Martino, C.A. Johnson, E.B. Suh, B.L. Trus, and T.K. Yap. Parallel computing in biomedical research. *Science*, 265(5174):902–908, 1994.

[126] T.G. Mattson, B.A. Sanders, and B.L. Massingill. *Patterns of parallel programming*. Addison-Wesley, 2004.

[127] M. McCool. Data-parallel programming on the Cell BE and the GPU using the RapidMind development platform. In *GSPx Multicore Applications Conference*, volume 9, 2006.

[128] M. McCool, Z. Qin, and T.S. Popa. Shader metaprogramming. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '02, pages 57–68, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. ISBN 1-58113-580-7. URL `http://dl.acm.org/citation.cfm?id=569046.569055`.

[129] M. McCool, S. Du Toit, T. Popa, B. Chan, and K. Moule. Shader algebra. *ACM Trans. Graph.*, 23(3):787–795, August 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015801. URL `http://doi.acm.org/10.1145/1015706.1015801`.

[130] M.R. McGann, H.R. Almond, A. Nicholls, J.A. Grant, and F.K. Brown. Gaussian docking functions. *Biopolymers*, 68(1):76–90, 2003. ISSN 1097-0282. doi: 10.1002/bip.10207. URL `http://dx.doi.org/10.1002/bip.10207`.

[131] S.L. McGovern and B.K. Shoichet. Information Decay in Molecular Docking Screens against Holo, Apo, and Modeled Conformations of Enzymes. *Journal of Medicinal Chemistry*, 46(14):2895–2907, 2003.

[132] C. McInnes. Virtual screening strategies in drug discovery. *Current Opinion in Chemical Biology*, 11(5):494 – 502, 2007.

[133] G. McMullan, S. Chen, R. Henderson, and A.R. Faruqi. Detective quantum efficiency of electron area detectors in electron microscopy. *Ultramicroscopy*, 109(9):1126 – 1143, 2009. ISSN 0304-3991. doi: http://dx.doi.org/10.1016/j.ultramic.2009.04.002. URL `http://www.sciencedirect.com/science/article/pii/S0304399109001120`.

[134] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 3.0*, 2012. URL `http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf`.

[135] C. Müller. *Spherical Harmonics*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1966. ISBN 9783540036005.

[136] J. Montrym and H. Moreton. The GeForce 6800. *Micro, IEEE*, 25(2):41–51, March 2005. ISSN 0272-1732. doi: 10.1109/MM.2005.37.

[137] M.M. Mysinger and B.K. Shoichet. Rapid context-dependent ligand desolvation in molecular docking. *Journal of Chemical Information and Modeling*, 50(9):1561–1573, 2010.

[138] M.M. Mysinger, M. Carchia, J.J. Irwin, and B.K. Shoichet. Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking. *Journal of Medicinal Chemistry*, 55(14):6582–6594, 2012.

[139] C.A. Navarro, N. Hitschfeld-Kahler, and L. Mateu. A Survey on Parallel Computing and its Applications in Data-Parallel Problems Using GPU Architectures. *Communications in computational physics*, 15(2):285–329, 2014.

[140] J. Navaza. On the three-dimensional reconstruction of icosahedral particles. *Journal of Structural Biology*, 144(1?2):13 – 23, 2003. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2003.09.007. URL `http://www.sciencedirect.com/science/article/pii/S1047847703001722`. Analytical Methods and Software Tools for Macromolecular Microscopy.

[141] G. Nemethy, K.D. Gibson, K.A. Palmer, C.N. Yoon, G. Paterlini, A. Zagari, S. Rumsey, and H.A. Scheraga. Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides. *The Journal of Physical Chemistry*, 96(15): 6472–6484, 1992.

[142] G. Neudert and G. Klebe. DSX: A Knowledge-Based Scoring Function for the Assessment of Protein-Ligand Complexes. *Journal of Chemical Information and Modeling*, 51(10): 2731–2745, 2011.

[143] G. Neudert and G. Klebe. fconv: format conversion, manipulation and feature computation of molecular data. *Bioinformatics*, 27(7):1021–1022, 2011.

[144] R. Norel, S.L. Lin, H.J. Wolfson, and R. Nussinov. Molecular surface complementarity at protein-protein interfaces: The critical role played by surface normals at well placed, sparse, points in docking. *Journal of Molecular Biology*, 252(2):263 – 273, 1995. ISSN 0022-2836. doi: http://dx.doi.org/10.1006/jmbi.1995.0493. URL `http://www.sciencedirect.com/science/article/pii/S0022283685704937`.

[145] F.N. Novikov, V.S. Stroylov, A.A. Zeifman, O.V. Stroganov, V. Kulkov, and G.G. Chilov. Lead Finder docking and virtual screening evaluation with Astex and DUD test sets. *Journal of Computer-Aided Molecular Design*, 26(6):725–735, 2012.

[146] A. Nukada and S. Matsuoka. Auto-tuning 3-D FFT Library for CUDA GPUs. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 30:1–30:10, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-744-8. doi: 10.1145/1654059.1654090. URL `http://doi.acm.org/10.1145/1654059.1654090`.

[147] A. Nukada, Y. Ogata, T. Endo, and S. Matsuoka. Bandwidth intensive 3-D FFT kernel for GPUs using CUDA. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–11, Nov 2008. doi: 10.1109/SC.2008.5213210.

[148] NVIDIA
Corporation. Arquitectura Fermi, 2009. URL `http://www.nvidia.com/content/pdf/fermi_white_papers/nvidia_fermi_compute_architecture_whitepaper.pdf`.

[149] NVIDIA Corporation. Arquitectura Kepler, 2012. URL `http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf`.

[150] NVIDIA Corporation. NVIDIA cuFFT 5.0, 2013. URL `https://developer.nvidia.com/cufft`.

[151] NVIDIA Corporation. CUDA 5.0 C Programming guide, 2013. URL `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`.

[152] NVIDIA Corporation. Arquitectura Maxwell en GeForce GTX 980, 2014. URL `http://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_980_Whitepaper_FINAL.PDF`.

[153] NVIDIA Corporation. Cuda compute capability, 2014. URL `https://developer.nvidia.com/cuda-gpus`.

[154] NVIDIA Corporation. Nvidia nsight visual studio edition, 2014. URL `https://developer.nvidia.com/nvidia-nsight-visual-studio-edition`.

[155] M. Oneppo. Hlsl shader model 4.0. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, pages 112–152, New York, NY, USA, 2007. ACM. ISBN 978-1-4503-1823-5. doi: 10.1145/1281500.1281576. URL `http://doi.acm.org/10.1145/1281500.1281576`.

[156] N. Opalka, M. Chlenov, P. Chacón, W. J. Rice, W. Wriggers, and S. A. Darst. Structure and function of the transcription elongation factor GreB bound to bacterial RNA polymerase. *Cell*, 114(3):335–345, 2003.

[157] Open MPI 1.6.2, Septiembre 2012. URL `http://www.open-mpi.org/software/ompi/v1.6/`.

[158] OpenACC.org. *OpenACC standard version 2.0*, 2014. URL `http://www.openacc.org/sites/default/files/OpenACC%202%200.pdf`.

[159] OpenMP Architecture Review Board. *OpenMP 4.0 Complete Specifications*, 2013. URL `http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf`.

[160] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A.E. Lefohn, and T.J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007. ISSN 1467-8659.

[161] J.D. Owens, M. Houston, D. Luebke, S. Green, J.E. Stone, and J.C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008.

[162] L. Pan, L. Gu, and J. Xu. Implementation of medical image segmentation in cuda. In *Information Technology and Applications in Biomedicine, 2008. ITAB 2008. International Conference on*, pages 82–85, May 2008. doi: 10.1109/ITAB.2008.4570542.

[163] PDB File format, 2011. URL `http://www.wwpdb.org/documentation/format33/v3.3.html`.

[164] P. Penczek, M. Radermacher, and J. Frank. Three-dimensional reconstruction of single particles embedded in ice. *Ultramicroscopy*, 40(1):33 – 53, 1992.

[165] H. Perez-Sanchez and W. Wenzel. Optimization methods for virtual screening on novel computational architectures. *Current Computer - Aided Drug Design*, 7(1):44–52, 2011.

[166] T.A. Pham and A.N. Jain. Parameter Estimation for Scoring Protein-Ligand Interactions Using Negative Training Data. *Journal of Medicinal Chemistry*, 49(20):5856–5868, 2006.

[167] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kalé, and K. Schulten. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26(16):1781–1802, 2005. ISSN 1096-987X. doi: 10.1002/jcc.20289. URL `http://dx.doi.org/10.1002/jcc.20289`.

[168] T.C. Pilkington, B. Loftis, T. Palmer, and T.F. Budinger. *High-Performance Computing in Biomedical Research*. CRC Press, 1992.

[169] N. D Prakhov, A. L Chernorudskiy, and M. R. Gainullin. VSDocker: a tool for parallel high-throughput virtual screening using AutoDock on Windows-based computer clusters. *Bioinformatics*, 26(10):1374–1375, 2010.

[170] S. Pruggnaller, M. Mayr, and A.S. Frangakis. A visualization and segmentation toolbox for electron microscopy. *Journal of Structural Biology*, 164(1):161 – 165, 2008.

[171] M. Radermacher. Three-dimensional reconstruction of single particles from random and nonrandom tilt series. *Journal of Electron Microscopy Technique*, 9(4):359–394, 1988. ISSN 1553-0817. doi: 10.1002/jemt.1060090405. URL `http://dx.doi.org/10.1002/jemt.1060090405`.

[172] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. A fast flexible docking method using an incremental construction algorithm. *Journal of molecular biology*, 261(3):470–489, 1996.

[173] M. Reinecke. Libpsht - algorithms for efficient spherical harmonic transforms. *A&A*, 526:A108, 2011. doi: 10.1051/0004-6361/201015906. URL `http://dx.doi.org/10.1051/0004-6361/201015906`.

[174] T. Risbo. Fourier transform summation of Legendre series and D-functions. *Journal of Geodesy*, 70(7):383–396, 1996.

[175] D.W. Ritchie and V. Venkatraman. Ultra-fast FFT protein docking on graphics processors. *Bioinformatics*, 26(19):2398–2405, 2010.

[176] D.W. Ritchie, D. Kozakov, and S. Vajda. Accelerating and focusing protein-protein docking correlations using multi-dimensional rotational FFT generating functions. *Bioinformatics*, 24(17):1865–1873, 2008.

[177] O. Roche, R. Kiyama, and C.L. Brooks. Ligand-Protein DataBase: Linking Protein-Ligand Complex Structures to Binding Data. *Journal of Medicinal Chemistry*, 44(22):3592–3598, 2001.

[178] P.B. Rosenthal and R. Henderson. Optimal determination of particle orientation, absolute hand, and contrast loss in single-particle electron cryomicroscopy. *Journal of Molecular Biology*, 333(4):721 – 745, 2003.

[179] P.E. Ross. Why cpu frequency stalled. *IEEE Spectr.*, 45(4):72–72, April 2008. ISSN 0018-9235. doi: 10.1109/MSPEC.2008.4476447. URL `http://dx.doi.org/10.1109/MSPEC.2008.4476447`.

[180] H.G. Rudenberg and P.G. Rudenberg. Chapter 6 - origin and background of the invention of the electron microscope: Commentary and expanded notes on memoir of reinhold rüdenberg. In Peter W. Hawkes, editor, *Advances in Imaging and Electron Physics*, volume 160 of *Advances in Imaging and Electron Physics*, pages 207 – 286. Elsevier, 2010. doi: http://dx.doi.org/10.1016/S1076-5670(10)60006-7. URL `http://www.sciencedirect.com/science/article/pii/S1076567010600067`.

[181] A. Ruiz, J. Kong, M. Ujaldon, K. Boyer, J. Saltz, and M. Gurcan. Pathological image segmentation for neuroblastoma using the gpu. In *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pages 296–299, May 2008. doi: 10.1109/ISBI.2008.4540991.

[182] S. Ruiz-Carmona, D. Alvarez-Garcia, N. Foloppe, A.B. Garmendia-Doval, S. Juhos, P. Schmidtke, X. Barril, R.E. Hubbard, and S.D. Morley. rDock: A Fast, Versatile and

Open Source Program for Docking Ligands to Proteins and Nucleic Acids. *PLoS Comput Biol*, 10(4):e1003571, 04 2014.

[183] R. Di Salvo and C. Pino. Image and Video Processing on CUDA: State of the Art and Future Directions. In *Proceedings of the 13th WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering*, MACMESE'11, pages 60–66, Stevens Point, Wisconsin, USA, 2011. World Scientific and Engineering Academy and Society (WSEAS). ISBN 978-1-61804-046-6. URL `http://dl.acm.org/citation.cfm?id=2074857.2074871`.

[184] N. Sauton, D. Lagorce, B. O. Villoutreix, and M. A. Miteva. MS-DOCK: Accurate multiple conformation generator and rigid docking protocol for multi-step virtual ligand screening. *BMC bioinformatics*, 9(1):184, 2008.

[185] S. H. Scheres, M. Valle, and J. M. Carazo. Fast maximum-likelihood refinement of electron microscopy images. *Bioinformatics*, 21(suppl 2):ii243–ii244, 2005.

[186] S. H. Scheres, M. Valle, R Nuñez, C. Sorzano, R. Marabini, G. Herman, and J. M. Carazo. Maximum-likelihood multi-reference refinement for electron microscopy images. *Journal of Molecular Biology*, 348(1):139 – 149, 2005. ISSN 0022-2836.

[187] S. H. Scheres, R. Núñez-Ramirez, C. O. Sorzano, J. M. Carazo, and R. Marabini. Image processing for electron microscopy single-particle analysis using XMIPP. *Nat. Protocols*, 3(6):977 – 990, 2008. ISSN 1754-2189.

[188] S.H. Scheres. RELION: Implementation of a Bayesian approach to cryo-EM structure determination. *Journal of Structural Biology*, 180(3):519 – 530, 2012.

[189] S.H. Scheres. A Bayesian View on Cryo-EM Structure Determination. *Journal of Molecular Biology*, 415(2):406 – 418, 2012.

[190] S.H.W. Scheres. Chapter Eleven - Classification of Structural Heterogeneity by Maximum-Likelihood Methods. In Grant J. Jensen, editor, *Cryo-EM, Part B: 3-D Reconstruction*, volume 482 of *Methods in Enzymology*, pages 295 – 320. Academic Press, 2010.

[191] S.H.W. Scheres, M. Valle, P. Grob, E. Nogales, and J.M. Carazo. Maximum likelihood refinement of electron microscopy data with normalization errors. *Journal of Structural Biology*, 166(2):234 – 240, 2009.

[192] M. Schmeisser, B. C. Heisen, M. Luettich, B. Busche, F. Hauer, T. Koske, K. H. Knauber, and H. Stark. Parallel, distributed and GPU computing technologies in single-particle electron microscopy. *Acta Crystallographica Section D*, 65(7):659–671, Jul 2009.

[193] J. Schmid, J.A. Iglesias Guitián, E. Gobbetti, and N. Magnenat-Thalmann. A GPU framework for parallel segmentation of volumetric images using discrete deformable models. *The Visual Computer*, 27(2):85–95, 2011. ISSN 0178-2789. doi: 10.1007/s00371-010-0532-0. URL `http://dx.doi.org/10.1007/s00371-010-0532-0`.

[194] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: A many-core x86 architecture for visual computing. *ACM Trans. Graph.*, 27(3):18:1–18:15, August 2008. ISSN 0730-0301. doi: 10.1145/1360612.1360617. URL `http://doi.acm.org/10.1145/1360612.1360617`.

[195] L. Shi, W. Liu, H. Zhang, Y. Xie, and D. Wang. A survey of GPU-based medical image computing techniques. *Quantitative Imaging in Medicine and Surgery*, 2(3):188–206, 2012.

[196] W.H. Shin, L. Heo, J. Lee, J. Ko, C. Seok, and J. Lee. LigDockCSA: Protein-ligand docking using conformational space annealing. *Journal of Computational Chemistry*, 32 (15):3226–3232, 2011.

[197] B.K. Shoichet and I.D. Kuntz. Protein docking and complementarity. *Journal of Molecular Biology*, 221(1):327 – 346, 1991. ISSN 0022-2836. doi: http://dx.doi.org/10.1016/0022-2836(91)80222-G. URL `http://www.sciencedirect.com/science/article/pii/002228369180222G`.

[198] B.K. Shoichet and I.D. Kuntz. Predicting the structure of protein complexes: a step in the right direction. *Chemistry & Biology*, 3(3):151 – 156, 1996. ISSN 1074-5521. doi: http://dx.doi.org/10.1016/S1074-5521(96)90256-2. URL `http://www.sciencedirect.com/science/article/pii/S1074552196902562`.

[199] F.J. Sigworth, P.C. Doerschuk, J.M. Carazo, and S.H.W. Scheres. Chapter Ten - An Introduction to Maximum-Likelihood Methods in Cryo-EM. In Grant J. Jensen, editor, *Cryo-EM, Part B: 3-D Reconstruction*, volume 482 of *Methods in Enzymology*, pages 263 – 294. Academic Press, 2010.

[200] M. Simonsen, C.N.S. Pedersen, M.H. Christensen, and R. Thomsen. GPU-accelerated High-accuracy Molecular Docking Using Guided Differential Evolution: Real World Applications. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 1803–1810. ACM, 2011.

[201] M. J. Sippl. Knowledge-based potentials for proteins. *Current Opinion in Structural Biology*, 5(2):229 – 235, 1995.

[202] I. Sánchez-Linares, H. Pérez-Sánchez, J. M. Cecilia, and J. M. García. High-Throughput parallel blind Virtual Screening using BINDSURF. *BMC bioinformatics*, 13(Suppl 14): S13, 2012.

[203] V. Soman. Accelerating spherical harmonic transforms on the nvidia gpu. *Department of Electrical Engineering, University of Wisconsin*, 2009.

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

[205] S.M. Stagg, G.C. Lander, J. Pulokas, D. Fellmann, A. Cheng, J.D. Quispe, S.P. Mallick, R.M. Avila, B. Carragher, and C.S. Potter. Automated cryoEM data acquisition and analysis of 284.742 particles of GroEL. *Journal of Structural Biology*, 155(3):470 – 481, 2006. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2006.04.005. URL `http://www.sciencedirect.com/science/article/pii/S1047847706001535`.

[206] Radek Stompor. S^{2}HAT, 2012. URL `http://www.apc.univ-paris7.fr/APC_CS/Recherche/Adamis/MIDAS09/software/s2hat/s2hat.html`.

[207] J.E. Stone, J.C. Phillips, P.L. Freddolino, D.J. Hardy, L.G. Trabuco, and K. Schulten. Accelerating molecular modeling applications with graphics processors. *Journal of Computational Chemistry*, 28(16):2618–2640, 2007. ISSN 1096-987X. doi: 10.1002/jcc.20829. URL `http://dx.doi.org/10.1002/jcc.20829`.

[208] J.A. Stratton, S.S. Stone, and W.W. Hwu. MCUDA: An Efficient Implementation of CUDA Kernels for Multi-core CPUs. In JoséNelson Amaral, editor, *Languages and Compilers for Parallel Computing*, volume 5335 of *Lecture Notes in Computer Science*, pages 16–30. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-89739-2. doi: 10.1007/978-3-540-89740-8_2. URL `http://dx.doi.org/10.1007/978-3-540-89740-8_2`.

[209] O.V. Stroganov, F.N. Novikov, V.S. Stroylov, V. Kulkov, and G.G. Chilov. Lead finder: An approach to improve accuracy of protein-ligand docking, binding energy estimation, and virtual screening. *Journal of Chemical Information and Modeling*, 48(12):2371–2385, 2008.

[210] V.S. Sunderam. PVM: A Framework for Parallel Distributed Computing. *Concurrency: Pract. Exper.*, 2(4):315–339, November 1990. ISSN 1040-3108. doi: 10.1002/cpe.4330020404. URL `http://dx.doi.org/10.1002/cpe.4330020404`.

[211] M. Szydlarski, P. Esterie, J. Falcou, L. Grigori, and R. Stompor. Parallel Spherical Harmonic Transforms on heterogeneous architectures (GPUs/multi-core CPUs). Technical Report RR-7635, INRIA, May 2012.

[212] TACC Stampede Supercomputer, 2013. URL `http://www.tacc.utexas.edu/stampede`.

[213] H. D. Tagare, A. Barthel, and F. J. Sigworth. An adaptive expectation-maximization algorithm with GPU implementation for electron cryomicroscopy. *Journal of Structural Biology*, 171(3):256 – 265, 2010. ISSN 1047-8477.

[214] G. Tang, L. Peng, P.R. Baldwin, D.S. Mann, W. Jiang, I. Rees, and S.J. Ludtke. EMAN2: An extensible image processing suite for electron microscopy. *Journal of Structural Biology*, 157(1):38 – 46, 2007. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2006.05.009. URL `http://www.sciencedirect.com/science/article/pii/S1047847706001894`.

[215] D. Tarditi, S. Puri, and J. Oglesby. Accelerator: Using Data Parallelism to Program GPUs for General-purpose Uses. *SIGOPS Oper. Syst. Rev.*, 40(5):325–335, October 2006. ISSN 0163-5980. doi: 10.1145/1168917.1168898. URL `http://doi.acm.org/10.1145/1168917.1168898`.

[216] R. Thomsen and M.H. Christensen. MolDock: A New Technique for High-Accuracy Molecular Docking. *Journal of Medicinal Chemistry*, 49(11):3315–3321, 2006.

[217] Top 500 List, 2014. URL `http://www.top500.org/lists/2014/11/`.

[218] M. Totrov and R. Abagyan. Flexible protein-ligand docking by global energy optimization in internal coordinates. *PROTEINS*, Suppl 1:215 – 220, 1997.

[219] M. Totrov and R. Abagyan. Protein-ligand docking as an energy optimization problem. In *Drug-Receptor Thermodynamics*, pages 603 – 624. John Wiley and Sons, 2001.

[220] O. Trott and A. J. Olson. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

[221] P. van der Heide, X.P. Xu, B.J. Marsh, D. Hanein, and N. Volkmann. Efficient automatic noise reduction of electron tomographic reconstructions based on iterative median filtering. *Journal of Structural Biology*, 158(2):196 – 204, 2007. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2006.10.030. URL `http://www.sciencedirect.com/science/article/pii/S1047847706003327`.

[222] M. van Heel, M. Schatz, and E. Orlova. Correlation functions revisited. *Ultramicroscopy*, 46(14):307–316, 1992.

[223] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar. An 80-tile 1.28tflops network-on-chip in 65nm cmos. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 98–589, Feb 2007.

[224] H. F. G. Velec, H. Gohlke, and G. Klebe. DrugScore(CSD)-Knowledge-Based Scoring Function Derived from Small Molecule Crystal Data with Superior Recognition Rate of Near-Native Ligand Poses and Better Affinity Prediction. *Journal of Medicinal Chemistry*, 48(20):6296–6303, 2005.

[225] C.M. Venkatachalam, X. Jiang, T. Oldfield, and M. Waldman. Ligandfit: a novel method for the shape-directed rapid docking of ligands to protein active sites. *Journal of Molecular Graphics and Modelling*, 21(4):289 – 307, 2003. ISSN 1093-3263. doi: http://dx.doi.org/10.1016/S1093-3263(02)00164-X. URL `http://www.sciencedirect.com/science/article/pii/S109332630200164X`.

[226] V. Venkatraman, V.I. Pérez-Nueno, L. Mavridis, and D.W. Ritchie. Comprehensive Comparison of Ligand-Based Virtual Screening Tools Against the DUD Data set Reveals Limitations of Current 3D Methods. *Journal of Chemical Information and Modeling*, 50 (12):2079–2093, 2010.

[227] P. Vicat-Blanc, B. Goglin, R. Guillier, and S. Soudan. *Computing networks: from cluster to cloud computing*. ISTE/Wiley, 2011.

[228] N. J. Vilenkin. *Special functions and the theory of group representations*. American Mathematical Society, 1968.

[229] S.M. Vogel, M.R. Bauer, and F.M. Boeckler. DEKOIS: Demanding Evaluation Kits for Objective in Silico Screening - A Versatile Tool for Benchmarking Docking Programs and Scoring Functions. *Journal of Chemical Information and Modeling*, 51(10):2650–2665, 2011.

[230] N. Volkmann. Chapter two - methods for segmentation and interpretation of electron tomographic reconstructions. In Grant J. Jensen, editor, *Cryo-EM, Part C: Analyses, Interpretation, and Case studies*, volume 483 of *Methods in Enzymology*, pages 31 – 46. Academic Press, 2010.

[231] I. Wallach and R. Lilien. Virtual decoy sets for molecular docking benchmarks. *Journal of Chemical Information and Modeling*, 51(2):196–202, 2011.

[232] P.H. Walls and M.J.E. Sternberg. New algorithm to model protein-protein recognition based on surface complementarity: Applications to antibody-antigen docking. *Journal of Molecular Biology*, 228(1):277 – 297, 1992. ISSN 0022-2836. doi: http://dx.doi.org/10.1016/0022-2836(92)90506-F. URL `http://www.sciencedirect.com/science/article/pii/002228369290506F`.

[233] W.P. Walters, M.T. Stahl, and M.A. Murcko. Virtual screening - an overview. *Drug Discovery Today*, 3(4):160 – 178, 1998.

[234] C. Wang, F. Mueller, C. Engelmann, and S.L. Scott. Hybrid Checkpointing for MPI Jobs in HPC Environments. In *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, pages 524–533, Dec 2010. doi: 10.1109/ICPADS.2010.48.

[235] C. Wang, F. Mueller, C. Engelmann, and S.L. Scott. Proactive process-level live migration and back migration in HPC environments. *Journal of Parallel and Distributed Computing*, 72(2):254 – 267, 2012. ISSN 0743-7315. doi: http://dx.doi.org/10.1016/j.jpdc.2011.10.009. URL `http://www.sciencedirect.com/science/article/pii/S0743731511002085`.

[236] G.L. Warren, C.W. Andrews, A.M. Capelli, B. Clarke, J. LaLonde, M.H. Lambert, M. Lindvall, N. Nevins, S.F. Semus, S. Senger, G. Tedesco, I.D. Wall, J.M. Woolven, C.E. Peishoff, and M.S. Head. A critical assessment of docking programs and scoring functions. *Journal of Medicinal Chemistry*, 49(20):5912–5931, 2006. doi: 10.1021/jm050362n. URL `http://dx.doi.org/10.1021/jm050362n`.

[237] D.Y. Wei and C.C. Yin. An optimized locally adaptive non-local means denoising filter for cryo-electron microscopy data. *Journal of Structural Biology*, 172(3):211 – 218, 2010.

[238] Z. Weng, S. Vajda, and C. Delisi. Prediction of protein complexes using empirical free energy functions. *Protein Science*, 5(4):614–626, 1996. ISSN 1469-896X. doi: 10.1002/pro.5560050406. URL `http://dx.doi.org/10.1002/pro.5560050406`.

[239] B. Wilkinson. *Grid Computing*. Chapman & Hall/CRC, 2010.

[240] H.G. Willett. Chip set brings home the arcade – startup makes high-end graphics affordable. *Electronic Buyers' News*, page 8, Nov 06 1995.

[241] X. Yan, R.S. Sinkovits, and T.S. Baker. AUTO3DEM-an automated and high throughput program for image reconstruction of icosahedral particles. *Journal of Structural Biology*, 157(1):73 – 82, 2007. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2006.08.007. URL `http://www.sciencedirect.com/science/article/pii/S1047847706002425`.

[242] C. Yang, P.A. Penczek, A. Leith, F.J. Asturias, E.G. Ng, R.M. Glaeser, and J. Frank. The parallelization of SPIDER on distributed-memory computers using MPI. *Journal of Structural Biology*, 157(1):240 – 249, 2007. ISSN 1047-8477. doi: http://dx.doi.org/10.1016/j.jsb.2006.05.011. URL `http://www.sciencedirect.com/science/article/pii/S1047847706001924`.

[243] M.I. Zavodszky and L.A. Kuhn. Side-chain flexibility in protein-ligand binding: The minimal rotation hypothesis. *Protein Science*, 14(4):1104–1114, 2005. ISSN 1469-896X. doi: 10.1110/ps.041153605. URL `http://dx.doi.org/10.1110/ps.041153605`.

[244] X. Zhang, X. Zhang, and Z.H. Zhou. Low cost, high performance GPU computing solution for atomic resolution cryoEM single-particle reconstruction. *Journal of Structural Biology*, 172(3):400 – 406, 2010.