UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de una aplicación para visualizar, manipular y fusionar volúmenes médicos

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERA EN COMPUTACIÓN

PRESENTA:

YARELY MONSERRAT ALVAREZ CONTRERAS

DIRECTOR DE TESIS:

DR. BORIS ESCALANTE RAMÍREZ

México D.F., 2013.

Abstract

The present thesis show a detailed process that visualize, handle and fusion both two volume of medical data (created from a half of medical images in RAW in VTK format) that operate in a Windows 7 system with a NVIDIA Quadro GPU and display this visualization in stereo This project arise for the LAPI -Laboratorio Avanzado de Procesamiento de Imágenes scheme requeriments from last project implement by Virtual Reality Deparment of DGTIC than visualize, handle and segment a set of medical image using the Hermite Discrete Transform. The plus of last project is visualization at same time of two images volume, implementation of Hermite Transform rotate, implementation of Hermite Inverse Transform, a scheme of fusion but without segmentation and finally display the volumes in stereo with Crystal Eyes.

The antecedents of the work is the fundamental of medical images, what is a medical image, how to obtain a medical image, differents types of medical images and purpose; the scientific visualization, techniques of volume visualization and stereo visualize.

Finally this work show how implementing the scheme of Fusion with Discrete Hermite Transform and display the volumes using the Visualization Toolkit VTK using a pipeline visualization functional model and display the medical volumes in the IXTLI with Crystal Eyes stereo.

Keywords: Visualization, 3D interaction, VTK, Fusion, Hermite transform

Agradecimientos

Al Dr.Boris Escalante Ramírez y Sonia Cruz Techica quienes me involucraron en este proyecto y apoyaron en todo lo que fuera posible.

Una especial mención al Departamento de Realidad Virtual(Mat.María del Carmen Ramos, José Larios y Rodrigo Díaz) de la Dirección General de Cómputo y Tecnologías de la Información y Comunicación por las asesorías y el apoyo otorgado para la conclusión de éste trabajo.

Muchas gracias a Julio Cesár Saynez por su constante motivación para que este trabajo diera resultado.

Gracias por el apoyo económico del Proyecto PAPIIT IN113611 y esta tesis forma parte de los productos del Programa de Fortalecimiento a la Docencia e Investigación del Observatorio IXTLI clave IX100610 *Fusión de datos 3D en imagenología médica.*

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Planteamiento del problema

El *procesamiento dígital de imágenes* es una disciplina que busca métodos y algoritmos para mejorar la apariencia de una escena capturada digitalmente. Por otro lado una imagen médica es una representación puntual en un plano o volumen de propiedades físicas o químicas del cuerpo humano. Digitalmente una imagen médica se representa como un arreglo espacial de intensidades. Así el procesamiento de imágenes médicas se encuentra evolucionando ya que se siguen investigando nuevos procedimientos para ayudar a la medicina a efectuar diagnósticos de ciertas patologías *in vivo*.

Se denomina modalidad a toda técnica de exploración con un equipo de adquisición y el elemento que define a cada modalidad de la imagen es el tipo de energía utilizada (entre ellas rayos X, rayos γ, ultrasonido, ondas de radio y luz visible) y la técnica de adquisición de la imagen(ejemplos: tomografía computarizada, radiología, resonancia magnética entre otros).

A últimas fechas los fabricantes de equipos de adquisión de imágenes médicas se están centrando en modalidades híbridas de imágenes ya sea para usos específicos de sistemas y con aplicaciones que ya efectúan el procesamiento y también aprovechando la naturaleza tridimensional de las imágenes se visualizan tridimensionalmente.

Así que el procesamiento que atañe a este trabajo es la fusión de imágenes médicas la cual detecta características sobresalientes y combina esos detalles en una imagen

sintética. Las técnicas de fusión se pueden clasificar en técnicas del dominio espacial y de transformada. La fusión en el dominio espacial el proceso se efectúa con los niveles de gris y las relaciones posicionales de los píxeles. En el dominio de la transformación se representan caracteríscas explícitas que por lo general definen cambios espaciales que ocurren con los niveles de gris.

Se han reportado en la literatura varias técnicas de fusión a nivel pixel empleando una transformada, algunas de ellas son el análisis multiresolución mediante la transformada wavelet, el análisis geométrico empleando la transformada contourlet, la transformada curvelet y la transformada Hermite.

La motivación de este trabajo es realizar la fusión usando la *transformada Hermite* y trabajarla en un contexto tridimensional que describan de forma didáctica los elementos que componen al volumen. Se usa la transformada Hermite por tratarse de un modelo de representación del sistema de visión humano y apegarse al órgano base del procesamiento de imágenes -la visión. Además la transformada Hermite permite un análisis multiresolución para describir las estructuras sobresalientes de una imagen y reduce el ruido presente sin ayuda de otros artefactos[26].

Se desarrollaron varias aplicaciones que realizan la DHT y operaciones de segmentación usando la librería de Volumizer, cuyo productos fueron una tesis de licenciatura, una investigación de la fusión de datos con la DHT y una tesis de maestría.

Se tomó como base el programa para visualización de volúmenes y realización de la DHT con resolución fija realizado por Héctor Barrón y Ramón Tapia[24](del departamento de visualización de la DGTIC antes DGSCA). Este programa permite cargar una secuencia de imágenes y también a partir de formatos establecidos(NRRD, NIFTI), visualizar un volumen(con la técnica de texture mapping). Ya visualizado, el programa permite filtrar niveles de gris y asignar mapas de colores preestablecidos. Además de efectuar cortes al volumen para visualizar zonas de interés y mostrar los coeficientes de primer orden de la DHT. A pesar de diseñar el programa modularmente y usar librerías de licencia libre, no se pudo continuar el desarrollo por falta de documentación y diseñado para visualizar sólo un volumen y no tres volúmenes.

No existe un programa que efectúe el esquema de fusión propuesto en el Laboratorio

Avanzado de Procesamiento de Imágenes *LAPI*. El problema es desarrollar una aplicación que realice el esquema de fusión propuesto y permita manipular los volúmenes formados, que el algoritmo implementado sea rápido, que maneje una interfaz intuitiva para el usuario, que permita una visualización estereoscópica, que sea una aplicación con posibilidad de crecimiento y que cumpla como ejemplo didáctico de lo que trata el Algoritmo de Fusión.

1.2. Objetivos general y específico

Desarrollar una aplicación para cargar imágenes médicas de diversos formatos y también de formatos convencionales, para generar dos volúmenes; manipularlos y realizar la fusión de sus características más importantes en un nuevo volumen.

1.2.1. Objetivos particulares

Desarrollar una interfaz gráfica con Qt. Cargar datos de imágenes médicas. Visualizar datos de volúmenes de imágenes mediante librerías que existan para este propósito. Implementar los algoritmos para realizar la fusión: transformada discreta Hermite(DHT), transformada inversa discreta Hermite(IDHT) y el esquema de fusión. Hacer una presentación en el Observatorio IXTLI y la fusión resultante se visualice estereoscópicamente.

Capítulo 2

Antecedentes

2.1. Visualización científica

La visualización de datos genera representaciones gráficas de conjuntos de datos o procesos de la naturaleza científica, de ingeniería o de medicina es una aplicación de los gráficos por computadora. Generalmente, esto se conoce como **visualización científica**. Y el término **visualización de negocios** se usa para conjuntos de datos relacionados con el comercio, la industria y otras áreas no científicas.[13]

Investigadores, analistas y demás, frecuentemente necesitan tratar con grandes cantidades de información o estudiar el comportamiento de procesos de elevada complejidad. Las simulaciones numéricas por computadora, normalmente, producen grandes cantidades de archivos de datos que contienen miles o incluso millones de valores. De modo similar, las cámaras de un satélite u otras fuentes de grabación acumulan archivos de datos más rápido de los que pueden ser interpretados. Rastrear estos grandes conjuntos de números para determinar tendencias y relaciones es un proceso tedioso y totalmente inefectivo. Sin embargo, si los datos se convierten a un formato gráfico, las tendencias y relaciones aparecen inmediatamente.

Existen muchas clases distintas de conjuntos de datos, por lo que los esquemas de visualización efectivos dependen de las características de los datos. Una colección de datos puede contener valores escalares, vectores, tensores de orden superior o cualquier combinación de estos tipos de datos. Los conjuntos de datos pueden estar distribuidos sobre

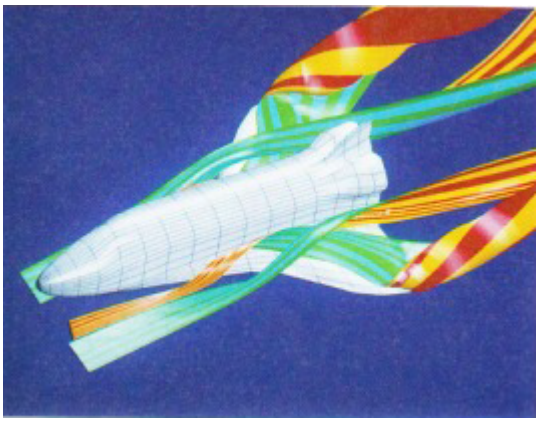


FIGURA 2.1: Visualización de superficies de corriente que fluyen a través de una lanzadera espacial.Proporcionado por Jeff Hultquist y Eric Raible, NASA Ames.*(Cortesia de Sam Uselton.Nasa Ames Research Center.)[13]*

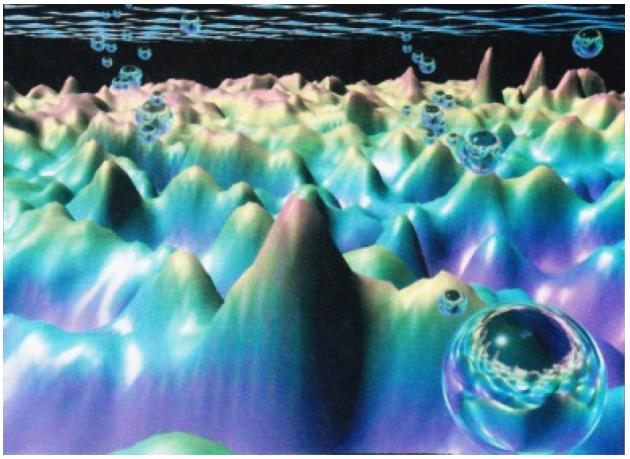


FIGURA 2.2: Visualización del suelo oceánico

una región bidimensional en el espacio, una región tridimensional o en un espacio de dimensión superior(volumen mas tiempo).

2.2. Imágenes médicas

Los médicos consideran que la imagen médica ha sido el avance técnico que mayor impacto ha tenido en su práctica clínica[19].La imagen médica es la interpretación de una o más propiedades físicas o químicas del cuerpo humano distribuidas espacialmente en un plano o volumen. Para efectos de procesamiento de imágenes esas representaciones se almacena de forma digital.

Una imagen digital es un arreglo o matriz de valores de intensidad, donde cada elemento corresponde a un valor discreto. Una caracteristica importante en una imagen es la calidad que depende del tamaño de los pixeles respecto al tamaño de la imagen y del número de valores de tonos gris disponibles para la descripción del rango de intensidades.

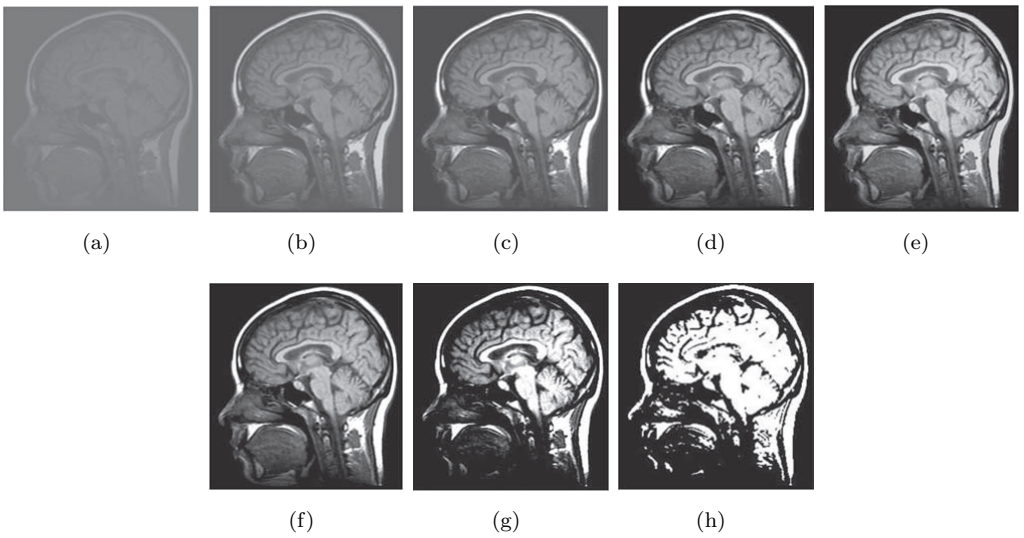


FIGURA 2.3: Una serie de imágenes que muestran distintos contrastes. La imagen de más bajo contraste es la (a) y la más alta es la (h)[15]

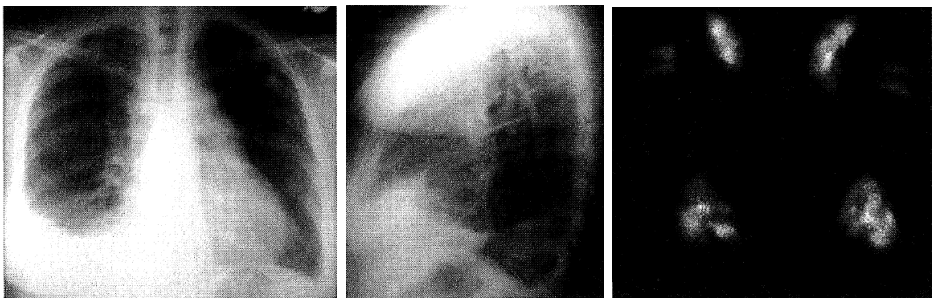
Hay otros parámetros que se consideran para el análisis de las imágenes, éstos son el contraste, la resolución y la cuantización de niveles de gris.

El *contraste* es el cambio en la luminancia que existe entre dos puntos de la imagen, para los fines de procesamiento de imágenes médicas nos interesa saber qué parámetro físico o químico se representa en forma de intensidad luminosa. Así podemos decir que el contraste es lo que se ve en la imagen.

La *resolución* puede ser espacial o temporal. La *resolución espacial* es la distancia mínima que la imagen es capaz de resolver o separar detalles. La *resolución temporal* determina la capacidad del sistema de imagen para congelar situaciones en el tiempo, relacionado con la velocidad de adquisición de la imagen.

La *cuantización de niveles de gris* es el número de valores de gris disponibles en la imagen y depende del número de bits usado durante la cuantización, n bits por pixel corresponden a 2^n intensidades de gris. Las imágenes no sólo contienen información de intéres, también datos ilegítimos o erróneos que los distorsionan. Esta información superpuesta se denomina *ruido* si es de carácter aleatorio y si depende de problemas en el sistema de obtención de la imagen se conoce como *artefacto*.

En lo que respecta a un volumen de imágenes médicas, esta se obtiene al apilar un conjunto de arreglos bidimensionales que representan los diferentes cortes anatómicos



(a) (b) (c)

FIGURA 2.4: Imagen morfológica y funcional.(a) y (b)Radiografía de tórax.(c)Gammagrafía pulmonar, en la que se aprecia una clara embolia pulmonar, invisible en las radiografías de tórax[19]

.

que conforman alguna región de intéres. La unidad de minima de volumen es conocida como voxel. Así también en los sistemas de volumenes se generan errores de cuantización.

Los sistemas de imágenes usan distintas señales físicas del paciente para producir imágenes. Para clasificar a las imágenes médicas una forma es de acuerdo a la modalidad de ésta. La modalidad es la técnica de exploración realizada con un equipo de adquisición de datos susceptibles de ser utilizados para realizar un diagnóstico. El elemento básico que define a la modalidad es el tipo de energía utilizada. Las modalidades fundamentales de imagen médica, en función del tipo de energía que utilizan, son: radiología (radiación electromagnética:rayos X), ecografía(energía mecánica, ultrasonidos), medicina nuclear(radiación electromagnética:radiación gamma) y resonancia magnética (radiación electromagnética: ondas de radio). Todas las modalidades de imagen médica, experimenta continuos avances técnicos[19].

Un criterio de clasificación de las modalidades es de acuerdo a la naturaleza del contraste de las imágenes, en base a ello se dividen en morfológicas(o estructurales) y funcionales. Las primeras producen imágenes con buena resolución y detallan la anatomía del paciente. Las segundas aportan información sobre el funcionamiento de los diferentes órganos o sistemas: algún rasgo de su metabolismo, su capacidad para acumular ciertas sustancias, entre otras[19].

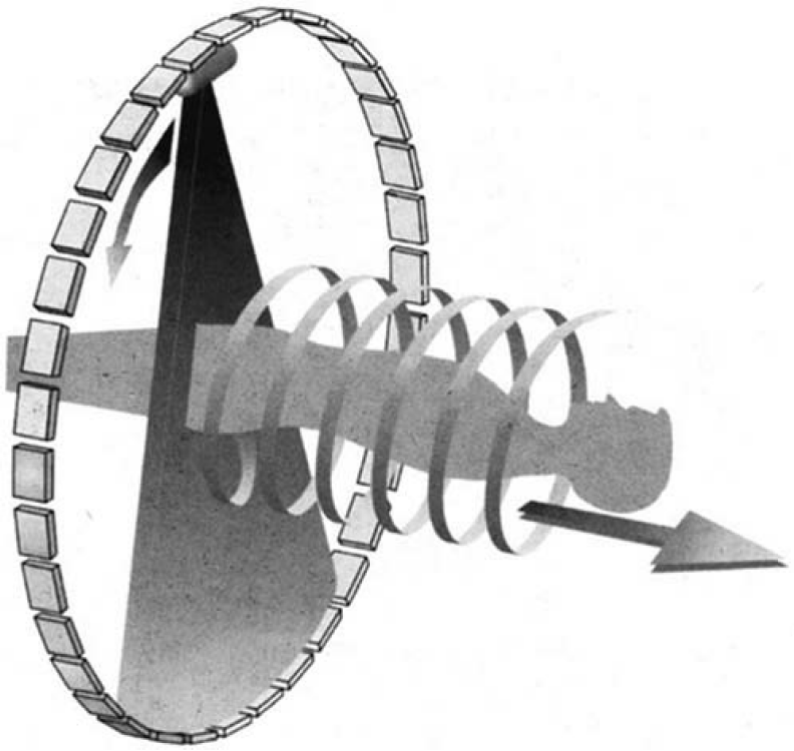


FIGURA 2.5: Esquema del CT helicoidal[15]

2.2.1. Tomografía computalizada (CT)

La tomografía computalizada(mejor conocida por los acrónimos *CT* ***C****omputer* ***T****omography*) es la adquisición de imágenes de rayos X mediante un tubo emisor de rayos X y un detector que giran describiendo un circulo alrededor del paciente. A partir de la información recogida, la computadora reconstruye los cortes tomográficos de la zona deseada. Similar a la radiografía convencional existe la falta de contraste entre tejidos blandos.

Un avance reciente de ésta modalidad es la CT helicoidal. A diferencia de la CT convencional, el tubo de rayos X y el detector se sincroniza con el de la cama del paciente obteniendo un volumen completo, mucho más rápido y sin fijar un determinado número de cortes ofreciendo la CT helicoidal una mayor velocidad de adquisición y mejora en la calidad tridimensional de los datos.

2.2.2. Imagen ultrasónica

Esta modalidad de imagen utiliza ondas ultrasónicas(energía mecánica), denominada ecografía, que se propagan por el tejido, reflejándose en las interfaces entre materiales de

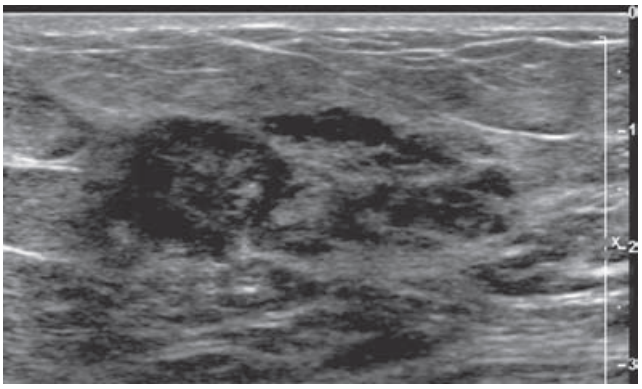


FIGURA 2.6: Imagen de ultrasonido de un carcinoma lobular

distinta densidad (similar a un SONAR marino). La radiación utilizada es no ionizante, y por lo tanto inofensiva; la imagen no es proyectiva, ya que sólo se observa un plano de corte; y las imágenes son dinámicas, en tiempo real. Se requiere de un entrenamiento específico para la interpretación de las imágenes ya que existen transiciones entre tejidos de distintas densidades. La ecografía proporciona imágenes útiles que separa estructuras sólidas y líquidas.Entre las ventajas se encuentran la inocuidad, sencillez de realización y reducido precio. Sus desventajas son la reducida resolución espacial, escaso contraste, cierta dificultad de interpretación y difucultad de penetración en el tejido (especialmente en el hueso mostrándose opaco).

2.2.3. Imagen nuclear

El fundamento de la medicina nuclear se sustenta en marcar con algún átomo radioactivo *trazadores* que se incorporan a las rutas metabólicas del organismo. El contraste de las imágenes de medicina nuclear viene determinado por la concentración alcanzada por el trazador en los diferentes órganos y sistemas. Las imágenes que se obtienen son un mapa de distribución del trazador en el organismo.

En medicina nuclear se utilizan principalmente dos tipos de radioisotópos(y dan lugar a dos tecnologías de imagen): *emisión de radiación gamma o fotón único* y *emisores de positrones*. Estos últimos tienen muchas ventajas como producir imágenes de mejor resolución y permiten marcar un mayor número de componentes biológicos. Sin embargo tienen pocos minutos de vida lo que hace dificil su manejo y exige disponer de un ciclotrón en las inmediaciones para producirlo.

Existen dos tipos de imágenes con esta modalidad, la primera se denomina *SPECT (Single Photon Emision Computed Tomography)*, tomografía computalizada de emisión de fotón único generado por la emisión de radiación gamma y se obtiene con un aparato denominado gammacámara que hace rotar alrededor del paciente el emisor de rayos gamma y va capturando en la computadora resconstrucciones tomográficas. El segundo son las imágenes *PET Positron Emission Tomography*, tomografía de emisión de positrones que utiliza una cámara PET que produce directamente corte tomográficos, detectando las parejas de fotones que produce la aniquilación del positrón.

2.2.4. Imagen por resonancia magnética (MRI)

La imagen por resonancia magnética(MRI) es una técnica no ionizante que usa radiofrecuencias(200MHz a 2GHz), radiación electromagnética y altos campos magnéticos(alrededor de 1-2 Tesla, comparado con el campo magnético terrestre que es de $0{,}5x10^{-4}$ Tesla).

Las imágenes por resonancia magnética proveen detalles anatómicos y fisiológicos, es decir, estructura y función con capacidad de representación tridimensional, excelente visualización de tejidos (componentes químicos) y alta resolución espacial. Similar a las demás modalidades de imágenes sobre todo la tomografía computacional, la reconstrución tomográfica se obtiene de una recolección de frecuencias que se capturan en un escaner MRI. La gran desventaja de esta modalidad es el alto costo del escaner.

2.3. Visualización tridimensional

La visualización de imágenes médicas surge como una necesidad para aprovechar y "digerir"la enorme cantidad de datos producidos por las distintas modalidades de imágenes.

Los datos tridimensionales pueden ser considerados como una pila de imágenes, ya sea de una sola modalidad o de una combinación de modalidades, donde los cortes(slices) bidimensionales han sido previamente registrados.

Un volumen típico comprime 50 cortes, cada uno con una resolución de 512x512x12 bits. Cada corte es una reconstrucción de un plano con un grosor de corte de aproximadamente 1 mm, el cual evita la superposición de objetos en la dirección z que ocurre

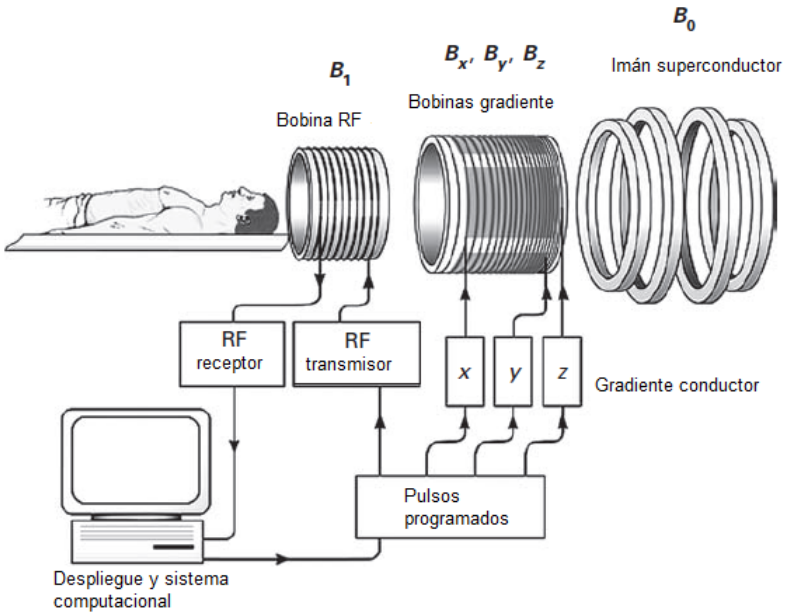
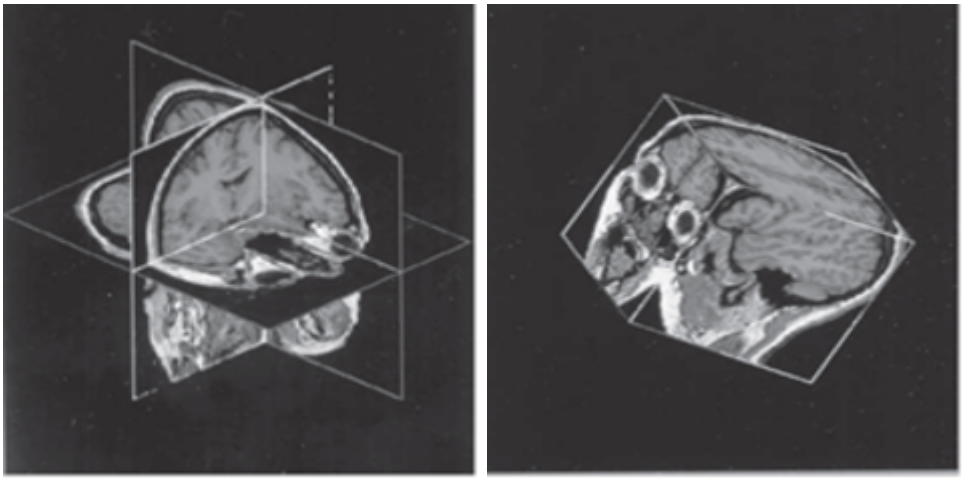


FIGURA 2.7: Componentes básicos de un escaner MRI[15]



(a) (b)

FIGURA 2.8: Secciones ortogonales de un volumen médico como (a)una intersección de planos ortogonales y (b) un volumen cúbico

en una imagen plana; estos cortes están generalmente continuos o separados por unos cuantos milímetros. Una ventaja inmediata de esto es que los datos pueden visualizarse desde cualquier punto de vista, pues el reformateo de los datos para mostrar las orientaciones ortogonales es particularmente simple y con una interpolación apropiada las secciones oblicuas también pueden ser seleccionadas. Esto se conoce como *visualización multiplanar* 2.8.

La visualización de imágenes médicas tridimensionales usualmente se lleva a cabo por

técnicas de renderización de superficies(surface rendering) o de volúmenes(volume rendering). La elección de la técnica adecuada depende de la naturaleza de las imágenes y del resultado deseado en la aplicación clínica.[15]

2.3.1. Renderización de superficies

En las técnicas de renderización de superficies primero se extraen las superficies de las estructuras que se desean visualizar. Esto requiere una etapa de segmentación y clasificación, en la cual se hace una búsqueda de los voxeles que pertenecen a bordes y además están conectados. Una vez que las estructuras han sido clasificadas y sus bordes han sido identificados, dicho bordes pueden ser representados mediante una malla triangular, los cuales se conectan entre sí para conformar la superficie.[15]

Las superficies con el mismo valor para una propiedad(brillo, gradiente o textura) se denominan isosuperficies, éstas pueden ser generadas directamente de los voxeles mediante el algoritmos 'marching cubes'. La idea básica de este algoritmo es la definición de un cubo(voxel) con los valores de los pixeles de sus vértices. Posteriormente se comparan los ocho vértices del cubo con el valor de umbral introducido, que será el de la superficie buscada, si al menos hay un vértice que esté por debajo del umbral y otro que esté por encima, el cubo formará parte de la isosuperficie final; cuando todos los vértices del cubo están por debajo o por encima del umbral no se procesan ya que no formarán parte de la isosuperficie final. Al determinar los lugares del cubo por donde pasa la superficie, se pueden generar triángulos que unan los puntos de intersección, los cuales finalmente se unen para obtener la superficie buscada.

Una vez que la superficie es obtenida, se elige una dirección de vista y una proyección. Una opción es visualizar en perspectiva donde los objetos más cercanos se ven más grandes que aquellos que están más distantes; no obstante, en ocasiones es más práctico que una región mantenga su tamaño constante en diferentes vistas. Así, una vista sin perspectiva usualmente se emplea para ver puntos que están fuera de una estructura, mientras que la vista en perspectiva se elige cuando se desea ver dentro de una estructura.

En la renderización de superficie el valor del sombreado de un voxel está definido por la orientación original de la superficie y la localización del voxel. Como consecuencia, la imagen 3D vista con la reconstrucción de superficie muestra sólo la parte externa del

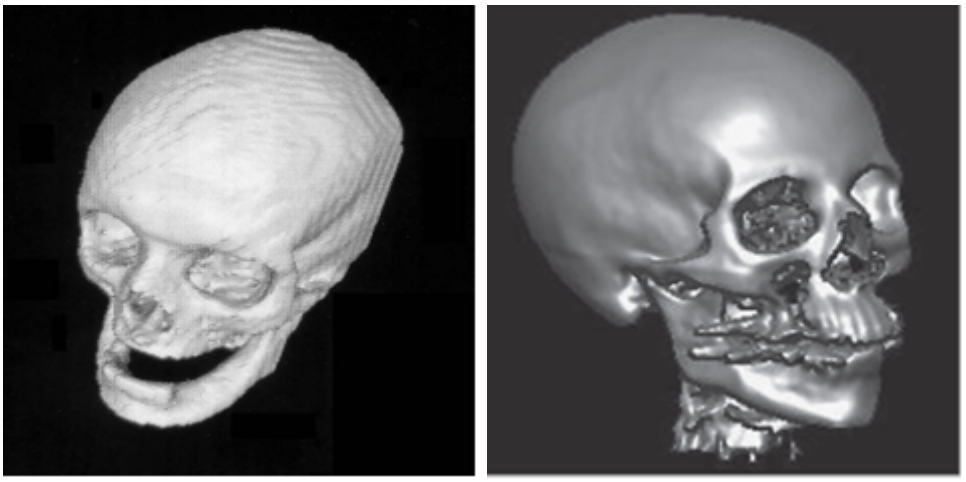
objeto, no pudiéndose analizar las estructuras internas del objeto estudiado, desperdiciando una gran cantidad de datos del volumen, pues los polígonos que quedan fuera del plano de vista son removidos del objeto.Figura 2.9(a)

La ventaja de esta técnica es la rapidez con la que se lleva a cabo su ejecución en comparación con otras formas de representación 3D, lo cual refleja en la velocidad que presenta en el manejo del volumen(rotaciones, traslaciones, deformaciones, etc.).En contraposición, las desventajas que se tienen son varias, el brillo de las superficies no depende del tejido que representa y difícilmente se obtienen bordes bien definidos de tejidos suaves; por otro lado, como la decisión sobre la superficie será visualizada se hace durante la extracción de contornos, la interactividad es limitada. Otra desventaja es que la técnica propicia los errores de submuestreo y aliasing originados por la naturaleza discreta de la malla triangular.

2.3.2. Renderización de volúmenes

La renderización de volúmenes presenta un despliegue de la imagen tridimensional completa después de proyectarla sobre un plano bidimensional. La aproximación más común se besa en técnica de *ray casting*, en las cuales un arreglo bidimensional de rayos es proyectado a través de una imagen tridimensional. Cada rayo interseca a la imagen 3D a lo largo de una serie de voxeles, los cuales son ponderados para alcanzar la representación deseada. Si las estructuras en la imagen 3D han sido segmentadas y clasificadas, los voxeles pueden ser ponderados de tal forma que se obtenga una representación translucida. Una aproximación alterna es la técnica denominada proyección de máxima intensidad(Maximun Intensity Projection: MIP), la cual es más rápida porque despliega solo los voxeles con la intensidad máxima a lo largo de cada rayo, evitando así el proceso de segmentación; sin embargo, no provee una buena sensación de profundidad por lo que resulta una pobre elección para la mayoría de las imágenes 3D.

Para esta representación de datos hay dos valores asociados con cada voxel, un valor de luminancia y uno de opacidad $\alpha(i)$. El primero se refiere al valor del voxel, o bien puede calcularse de un modelo de reflexión usando el gradiente local; mientras que la opacidad se deriva del tipo de tejido, por ejemplo, si $\alpha(i) = 0$, el rayo pasa a través del i-esimo voxel como si este fuera transparente, y si $\alpha(i) = l$, el voxel es opaco. Ambos valores pueden ser modificados durante el proceso de proyección para producir ciertos efectos



(a) (b)

FIGURA 2.9: Técnicas de visualización de volúmenes biomédicos.(a)Renderización tridimensional de superficie de un cráneo.(b)Renderización volumétrica de un cráneo[15]

sin tener que aplicar toda la técnica desde cero, estos efectos pueden ser el cambio del grado de transparencia/opacidad o la selección de diferentes superficies segmentadas. El sombreado profundo también puede incorporarse mediante la adición de una función de distancia del observador en relación al valor desplegado. Figura 2.9(b).

A pesar de que la renderización de volúmenes requiere un gran número de rayos para generar resultados satisfactorios, consumiendo muchos recursos de cómputo, los equipos actuales son lo suficientemente rápidos para permitir la exploración y edición de datos tridimensionales en tiempo real.[15]

2.3.3. Realidad virtual

La medicina ha empezado a utilizar sistemas de realidad virtual(Virtual Reality: VR), que comúnmente están especializados en alguna de sus áreas, e incorporan estereoscopia y visualización tridimensional con un diseño de interfaz muy intuitivo. Sus principales aplicaciones son el entrenamiento en el ámbito educativo mediante el uso de simuladores, la planeación de terapias y la asistencia quirúrgica. Estos sistemas van desde equipos de cómputo de escritorio con amplios recursos para el despligue de gráficos; sistemas inmersivos, donde el usuario típicamente utiliza un casco o unos lentes para la visualización y se procura que se involucre en un ambiente tridimensional que incluye estímulos visuales, auditivos y en ocasiones sensaciones de fuerza tangibles; hasta realidad aumentada, en la cual el usuario ve el mundo real con objetos virtuales superpuestos.[15]

Los sistemas de visión estereoscópica emplean un par estéreo(visión del ojo izquierdo y del ojo derecho) de imágenes 3D renderizadas en un plano, ambas imágenes presentan pequeñas diferencias entre sí debidas a la separación entre los ojos y pueden observarse con ayuda de lentes stereo[1], produciendo así el despliegue de un volumen virtual 3D. La ventaja más obvia de este tipo de visualización es la incorporación de la información de profundidad ya que el paralaje entre el par estéreo es uno de los elementos utilizados por el cerebro para percibir la profundidad, sin embargo, existe otra ventaja que se refiere a inherente habilidad del ojo humano para jugar el papel de un filtro 3D que suprime el efecto del ruido en la imagen. El ruido en imágenes, como las MRI, usualmente aparece alrededor de las estructuras de intéres; cuando la imagen renderizada es presentada en 2D, el ruido aparece proyectado sobre las estructuras haciéndolas ver más oscuras, pero cuando los datos se presentan en estéreo 3D, el ruido es atenuado gracias a la percepción de profundidad.[23]

En la endoscopia virtual, el ambiente tridimensional se construye de los datos obtenidos de una CT helicoidal. Un ejemplo de su aplicación, que ha tenido un impacto clínico importante en cuanto a detección de uno de los tipos de cáncer más comunes, es la colonoscopia virtual, pues es un procedimiento más rápido que la colonoscopia convencional además de que es mínimamente invasivo y no requiere anestesia.[15]

En relación al uso de la VR en la etapas de planeación de una intervención quirúgica, se utilizan simulaciones y realidad aumentada mediante el uso de imágenes pre-calculadas de estudios previos sobre el paciente, ya que esto ayuda a que el procedimiento sea más preciso y en consecuencia menos invasivo. Por ejemplo, en la planeación de resección de tumores cerebrales, se requiere de la reconstrucción del volumen a partir de las imágenes adquiridas mediante las modalidades que permitan una mejor localización de la lesión; en el caso de estudio que se presenta para ilustrar esta aplicación se combinaron dos modalidades en tiempo real para la planeación de la extracción de un tumor pituitario, un angiograma adquirido por MRI(para visualizar la piel y los tejidos blandos del cerebro) y una CT(para el hueso); la técnica de renderización fue *ray casting*.[18]

[1]Existen varios tipos de lentes dependiendo del sistema de visión estereóscopica empleado, pero en generla evitan que cada ojo perciba lateralmente la imagen correspondiente al otro ojo.

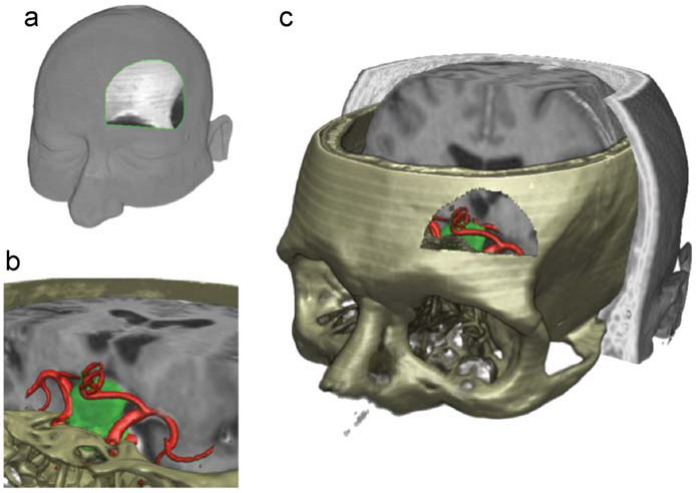


FIGURA 2.10: Planeación de una resección de tumor pituitario de la región subfrontal derecha del cerebro con ayuda de la visualización 3D.(a)Incisión en la piel.(b)Vista microscópica de la lesión.(c)Aproximación del orificio durante la planeación de la cirugía [18]

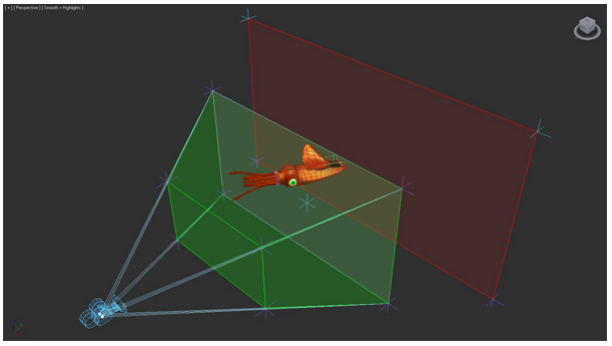


FIGURA 2.11: Una proyección de visualización estereoscópica

2.3.4. Visualización estereoscópica

Mostrar una convincente imagen en tres dimensiones a partir de una imagen en dos dimensiones ha formado un reto para productores hace más de 160 años, partiendo de la tecnología del anaglifo desarrollada en la década de 1850. Esta primera técnica llamada Anaglifo consiste en sobreponer la imagen dos veces con una pequeña diferencia en posición, además de imprimirla en distintos colores y con la ayuda de unos anteojos uno con un lente filtro color rojo y el otro de color azul dan la impresión de que la imagen tiene profundidad. Esta técnica se adoptó en los cines Hollywoodenses en la década de 1950 y en la actualidad los cines nos muestran filmes en 3D con ayuda de unos lentes polarizados y ya no con filtros como sucede con la técnica anaglifa. Aunque la tecnología ha madurado significativamente, los conceptos fundamentales se han mantenido. Pero en qué consiste la visualización estereoscópica. La visualización estereoscópica permite

(a) (b)

FIGURA 2.12: Ejemplo de dos sistemas de visión estereoscópica.(a)Crystal Eyes de la empresa Real3D[3].(b)Nvidia 3D Vision de Nvidia

la presentación de pares de imágenes, que representan una visión del mundo de ojo izquierdo/ojo derecho. El cerebro interpreta estas dos imágenes como una única sobre la cual es capaz de estimar distancias y profundidades. Para que esta especie de engaño al cerebro, mostrándole la escena virtual desde dos puntos de vista, tenga efecto es necesario poder aislar en cada uno de los ojos la visualización de sólo una de las dos imágenes de la escena correspondiente. Existen dos técnicas habituales en la generación de visión 3D estéreo en tiempo real denominadas estéreo pasivo o estéreo activo.

Estereorización activa El estéreo activo (o estereoreal) consistente en presentar en secuencia y alternativamente, las imágenes izquierda y derecha, sincronizadamente con unas gafas dotadas con obturadores de cristal líquido (denominadas LCS, Liquid Crystal Shutter glasses o LCD, Liquid Crystal Display glasses[3]), de forma que cada ojo ve solamente su imagen correspondiente. A una frecuencia elevada, el parpadeo es imperceptible. A este sistema se le denomina de gafas estéreo, porque la polarización se efectúa en la misma gafa.

Estereorización Pasiva El estéreo pasivo (o anáglifos) se consigue mediante la generación de imágenes independientes para cada ojo polarizadas mediante lentes ópticas en el sistema de proyección, y desfasadas 90º. Sobre la pantalla se superponen las dos perspectivas, mientras que el observador utiliza gafas con lentes polarizadas que filtrarán las imágenes, de forma que a cada ojo llegue sólo la adecuada. La polarización empleada

suele ser normalmente de tipo lineal. Visionarc ha desarrollado sistemas de visualización para alguno de sus proyectos que emplean indistintamente polarización vertical y también circular, que es un poco más difícil de conseguir[28].

Capítulo 3

La transformada Hermite

La transformada Hermite, introducida al área de procesamiento digital de imágenes por Martens[16], es un modelo de representación basado en el sistema de visión humano. La transformada Hermite modela los campos receptivos presentes en el ojo humano para las primeras etapas de visión (similares a las derivadas de Gaussiana excepto por un factor de escala); en su etapa de análisis, separa la información visual contenida en la imagen en información de baja frecuencia (promedio) y alta frecuencia qué hace a la transformada Hermite un buen descriptor de patrones característicos como bordes y texturas. La transformada Hermite es un caso particular de una transformada más general denominada transformada polinomial.

3.1. Transformada polinomial

La tranformada polinomial es una técnica de descomposición local de señales por medios de polinomios.[16] El transformada polinomial consta de dos pasos. El primero, la señal original $L(x)$ es localizada al multiplicarla por una función ventana $V(x)$ en varias posiciones equidistantes. A partir de la función ventana $V(x)$ es posible definir una función de ponderación periódica T:

$$W(x) = \sum_{k} V(x - kT) \tag{3.1}$$

El periodo de la función de ponderación es T. Haciendo $W(x)$ diferente de cero para cualquier posición x, se obtiene

$$L(x) = \frac{1}{W(x)} \sum_k L(x)V(x - kT) \tag{3.2}$$

El segundo paso consiste en aproximar la información local dentro de cada ventana de análisis en términos de una familia de polinomios ortogonales. Para llevar a cabo la expansión, se toman como funciones base los polinomios $Q_n(x)$, donde n es el grado del polinomio. Estos polinomios están determinados totalmente por la función ventana de tal forma que deben ser ortonormales respecto a $V^2(x)$ es decir

$$\int_{-\infty}^{+\infty} V^2(x)Q_m(x)Q_n dx = \delta_{mn} \tag{3.3}$$

En una transformada polinomial, los coeficientes de una señal $L(x)$(proyecciones) pueden ser obtenidos como

$$L_n(kT) = \int_{-\infty}^{+\infty} L(x)Q_n(x - kT)V^2(x - kT)dx \tag{3.4}$$

seguido de un submuestreo en múltiplos de T. Es posible notal que la señal original $L(x)$ se convoluciona con las funciones de análisis (funciones filtro) denotadas por:

$$D_n(x) = Q_n(-x)V^2(-x) \tag{3.5}$$

es decir:

$$L_n(kT) = \int_{-\infty}^{+\infty} L(x)D_n(x - kT)dx \tag{3.6}$$

El mapeo de la señal original $L(x)$ a los coeficientes $L_n(kT)$ es conocido como transformada polinomial directa. El proceso de reconstrucción de la señal a partir de los coeficientes es conocido como transformada polinomial inversa 3.7 y básicamente consiste en interpolar los coeficientes $L_n(kT); kentero$ con las funciones de síntesis (funciones patrón) $P_n(x)$ 3.8 y sumar sobre todos los órdenes n.

$$L(x) = \sum_{n=0}^{\infty} \sum_k L_n(kT)P_n(x - kT) \tag{3.7}$$

$$P_n(x) = \frac{Q_n(x)V(x)}{W(x)} \tag{3.8}$$

Las transformadas polinomiales directa e indirecta se muestran en la Figura 3.1

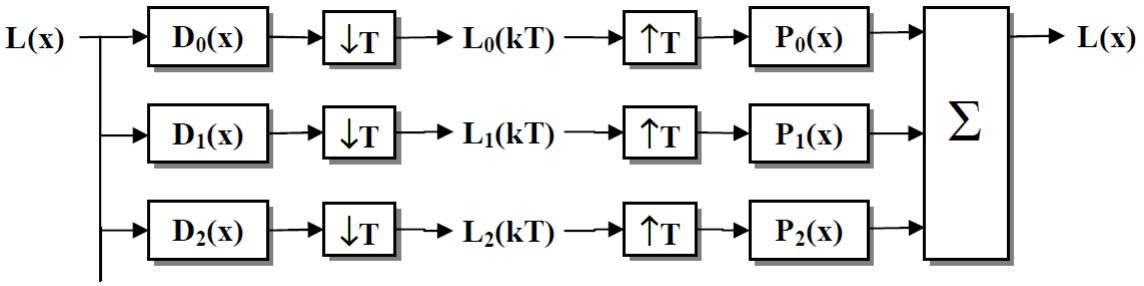


FIGURA 3.1: Diagrama de bloques de la transformada polinomial directa e inversa

3.2. La transformada Hermite

Cuando en la transformada polinomial las ventanas de análisis empleadas son funciones gaussianas se habla de la transformada Hermite.Dicha ventana Gaussiana tiene la propiedad de ser isotrópica (invariable con la rotación), separable en coordenadas cartesianas y sus derivadas simulan algunos procesos a nivel de corteza visual del sistema de visión humano; se define de la siguiente manera:

$$V(x) = \frac{1}{\sqrt{\sqrt{\pi}\sigma}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.9}$$

donde el factor de normalización $\frac{1}{\sqrt{\sqrt{\pi}\sigma}}$ es tal que $V^2(x)$ tiene energía unitaria. Los polinomios ortogonales que están asociados a $V^2(x)$ son conocidos como polinomios de Hermite son las soluciones a la ecuación diferencial[25]:

$$y'' - 2xy' + 2ny = 0 \tag{3.10}$$

donde x es una variable espacial y y es una función de x. La ecuación 3.10 se conoce como la ecuación de onda del oscilador harmónico(resultado de la ecuación de Schrödinger). La forma matemática de los polinomios de Hermite dada por la fórmula de Rodrigues[20] es:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n e^{-x^2}}{dx^n}, n = 0, 1, 2, ..., \tag{3.11}$$

donde el operador $\frac{d^n}{dx^n}$ denota la n-ésima derivada de la función. La definición de la transformada Hermite involucra tres funciones cuyas propiedades se describen a continuación.

Función de ponderación. Como la función de ponderación $W(x)$ es periódica con periodo T, puede ser expandida a través de una serie de Fourier

$$W(x) = \frac{\sqrt{2\sqrt{\pi}\sigma}}{T} w(x) \tag{3.12}$$

donde

$$w(x) = 1 + 2\sum_{k=1}^{\infty} e[-\frac{1}{2}(k\frac{2\pi\sigma}{T})^2] \cos k\frac{2\pi x}{T} \tag{3.13}$$

El contraste de esta función de ponderación se determina por el parámetro de muestreo $\tau = T/\sigma$. Como usualmente se quiere limitar el número de descomposiciones locales, es mejor hacer que τ sea tan grande como sea posible. Por otro lado, considerando la ecuación 3.2, se observa que $W(X)$ debe ser aproximamente constante ya que la división entre $W(x)$ podría introducir una sensibilidad a la variación de la escala.

Función de análisis. Las funciones de análisis (funciones filtro) determima que información se hace explícita en los coeficientes de la transformada Hermite, por lo que las principales propiedades de esta transformada están determinadas por estas funciones. A partir de la ecuación 3.5 se pueden derivar las funciones filtro específicas

$$D_n(x) = \frac{(-1)^n}{\sqrt{2^n n!}} \cdot \frac{1}{\sigma\sqrt{\pi}} H_n(\frac{x}{\sigma}) e^{-\frac{x^2}{\sigma^2}} \tag{3.14}$$

que podemos reescribirla en términos de la definición de derivadas de Gaussiana evaluada en $x = (x/\sigma)$ como

$$D_n(x) = \frac{1}{\sqrt{2^n n!}} \cdot \frac{1}{\sigma\sqrt{\pi}} \frac{d^n}{dx^n} e^{-\frac{x^2}{\sigma^2}} \tag{3.15}$$

Es aquí donde se produce la conexión entre los perfiles de los campos receptivos del Sistema de Visión Humano (HVS Human Vision System) modelados por las derivadas de Gaussiana, y los polinomios de Hermite ortonormales respecto a la ventana de análisis.

$$G_n(x) = (-1)^n H_n(x) e^{-x^2} \tag{3.16}$$

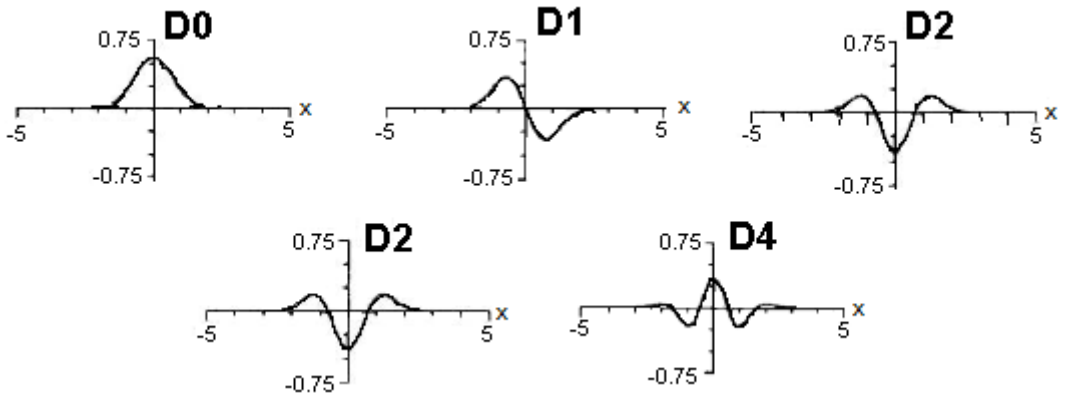


FIGURA 3.2: Funciones filtro para $\sigma = 1$

La transformada de Fourier de $D_n(x)$ es

$$d_n(\omega) = \frac{1}{\sqrt{2^n n!}}(j\omega\sigma)^n e^{-\frac{(\omega\sigma)^2}{4}} \tag{3.17}$$

teniendo un valor extremo cuando $(\omega\sigma)^2 = 2n$, indica que los filtros de órdenes mayores analizan frecuencias sucesivamente más altas de la señal. Sin embargo, para filtros de órdenes muy grandes, los picos en frecuencia se aproximan demasiado, por lo que tales filtros proporcionan poca información adicional. Debido a esto, en la práctica, la transformada de Hermite se limita a pocos términos.Las funciones filtro para $n = 0, ..., 4$ se muestran en la Figura ??.

Funciones de síntesis. Las funciones de síntesis (funciones patrón) se usan durante la reconstrucción de la señal original que se hace a partir de los coeficientes de la transformada Hermite. Estas funciones están dadas por la ecuación

$$P_n(x) = H_n(\frac{x}{\sigma})\frac{V(x)}{W(x)} = \frac{T}{\sqrt{2^n n!}}\frac{1}{\sigma\sqrt{2\pi}}H_n(\frac{x}{\sigma})\frac{e^{\frac{x^2}{2\sigma^2}}}{w(x)} \tag{3.18}$$

donde $w(x)$ es la función de ponderación de la ecuación 3.13. Si $w(x) = 1$ (para valores de muestreo del parámetro $\tau < 2$), la función patrón es igual a la función Hermite de grado n. La función Hermite tiene la propiedad de ser isomorfa a su transformada de Fourier:

$$P_n(\omega) = \frac{T}{\sqrt{2^n n!}}(-j)^n H_n(\omega\sigma)e^{\frac{(\omega\sigma)^2}{2}} \tag{3.19}$$

3.3. La transformada Hermite discreta

La transformada Hermite cuenta con una aproximación discreta (DHT: Discrete Hermite Transform) basada en los polinomios de Krawtchouk[**?**]. Dado que los filtros de análisis de la tranformada Hermite son similares a las derivadas Gaussianas excepto por un factor de escala, es posible utilizar la forma discreta de las derivadas Gaussianas dadas por los coeficientes binomiales:

$$C_N^x = \frac{N!}{x!(N-x)!} \tag{3.20}$$

donde N es la longitud de la ventana binomial. Los polinomios ortonormales de Krawtchouk son obtenidos a partir del producto de los polinomios de Krawtchouk por una ventana binomial definida por:

$$v^2(x) = C_N^x/2^N \tag{3.21}$$

de tal manera que los polinomios ortonormales de Krawtchouk están definidos como:

$$K_n(x) = \frac{1}{\sqrt{C_N^n}} \sum_{\tau=0}^{n} (-1)^{n-\tau} C_{N-x}^{n-\tau} C_x^\tau \tag{3.22}$$

para $x = 0, .., N$ y $n = 0, ..., D_{max}$ donde D_{max} es el máximo orden de la derivada y $D_{max} \leq N$. La transformada discreta de Hermite de longitud N aproxima a la transformada Hermite continua por la siguiente relación $\sigma = \sqrt{M/2}$. Ejemplos de los polinomios de Krawtchouk se presentan a continuación para $D = 1, D = 2$ y $D = 4$ con $D = N$:

$$K_1 = \frac{1}{\sqrt{2^1}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$K_2 = \frac{1}{\sqrt{2^2}} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & -2 \\ 1 & -1 & 1 \end{bmatrix}$$

$$K_4 = \frac{1}{\sqrt{2^4}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 4 & 2 & 0 & -2 & -4 \\ 6 & 0 & -2 & 0 & 6 \\ 4 & -2 & 0 & 2 & -4 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix}$$

3.3.1. La transformada Hermite 2D

Un caso especial se presenta en las transformadas polinomiales bidimensional resulta cuando la función ventana es separable:

$$V(x, y) = V(x)V(y) \tag{3.23}$$

y la rejilla de muestreo es cuadrada. Las funciones de análisis y síntesis son por ende separables y pueden ser implementadas de manera muy eficiente. En el caso de la transformada Hermite, se trabaja con ventanas Gaussianas, las cuales al ser isotrópicas podemos escribir $\sigma = \sigma_x = \sigma_y$

$$V(x, y) = \frac{1}{2\pi\sigma^2} e(-\frac{(x^2 + y^2)}{2\sigma^2}) \tag{3.24}$$

Los polinomios de Hermite ortonormales con respecto a $V^2(x, y)$ pueden escribirse como:

$$Q_{n-m,m}(x, y) = \frac{1}{\sqrt{2^n(n-m)!m!}} H_{n-m}(\frac{x}{\sigma}) H_m(\frac{y}{\sigma}) \tag{3.25}$$

Los coeficientes de la señal localizada $f_{n-m,m}(x, y)$ se obtienen al filtrar la imagen con las funciones filtro separables dada por:

$$D_{n-m,m}(x, y) = Q_{n-m,m}(-x, -y)V^2(-x, -y) \tag{3.26}$$

y haciendo enseguida un submuestro a la señal de salida en las direcciones horizontal y vertical en múltiplos de T; es decir, que para todo $p, q \in S$,

$$L_{n-m,m}(p, q) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} L(x, y) D_{n-m,m}(x - p, y - q) dx dy \tag{3.27}$$

La imagen sintetizada se obtiene por un proceso de interpolar los coeficientes polinomiales con funciones de síntesis

$$P_{n-m,m}(x,y) = \frac{Q_{n-m,m}(x,y)V(x,y)}{W(x,y)}$$

y sumar para todo orden n y m de la siguiente manera:

$$L(x,y) = \sum_{n=0}^{\infty}\sum_{m=0}^{n}\sum_{(p,q)\in S} L_{n-m,m}(p,q)P_{n-m,m}(x-p,y-q) \tag{3.28}$$

En la siguiente imagen se muestra la descomposición de Hermite sobre una imagen

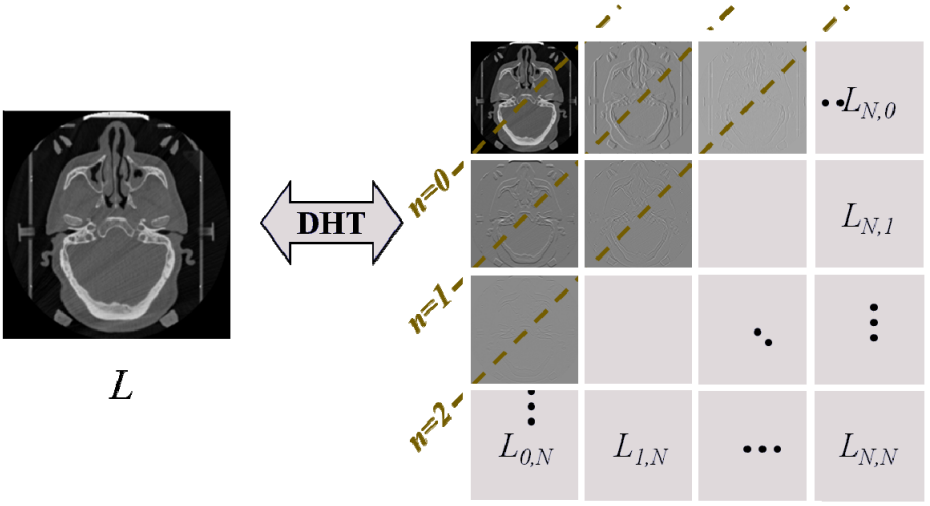


FIGURA 3.3: Descomposición de una imagen con la transformada Hermite

3.3.2. Transformada de Hermite 2D rotada

La transformada Hermite tiene ventajas de compactar su energía en algunos coeficientes mediante la transformada rotada(adaptively steering transform)[5]. El termino de filtro steerable describe un conjunto de filtros los cuales son copias rotadas de cada uno, una copia del filtro de cualquier orientación es construida como una combinación lineal de un conjunto de filtros base[29]. La propiedad de steering en los filtros de Hermite puede ser considerada por que los filtros son productos de las polinomiales con una función ventana radialmente simétrica. Los $N+1$ filtros de Hermite 2D de *enésimo orden* forman una base de los filtros de orden n sensibles a la orientación θ en particular, es decir, los filtros rotados en orden n pueden ser construidos a través de combinaciones lineales de

los filtros de orden n La trasformada Fourier de las funciones de análisis bidimensional $D_{n-m,m}(x,y)$ expresada en coordenadas polares es:

$$d_{n-m,m}(\omega_x,\omega_y) = \alpha_{n-m,m}(\theta)d_n(\omega) \tag{3.29}$$

donde $\omega_x = \omega\cos\theta, \omega_y = \omega\,\mathrm{sen}\,\theta$ y $d_n(\omega)$ es la transformada de Fourier de la función filtro de Hermite en una dimensión $D_n(r)$ donde r es la coordenada radial. De esta manera, se observa que la transformada de Fourier de cada función de análisis puede escribirse como un producto que independientemente expresa una preferencia de orientación y una preferencia de frecuencia radial. La frecuencia radial de las funciones de análisis está dada por la ecuación 3.17, mientras que la selectividad direccional del filtro se expresa por la siguiente función angular:

$$\alpha_{n-m,m}(\theta) = \sqrt{C_n^m}\cos^{n-m}\theta\sin^m\theta \tag{3.30}$$

A partir de la ecuación 3.30 se puede ver que los filtros cuyo orden es el mismo n pero con distintos indice direccional m distinguen entre diferentes orientaciones en la imagen. De acuerdo con lo anterior, la transformada Hermite rotada se puede definir de la siguiente forma:

$$L^{\theta}_{n-m,m}(x,y) = \sum_{m=0}^{n} L_{n-m,m}\alpha_{n-m,m}(\theta) \tag{3.31}$$

donde $L^{\theta}_{n-m,m}(x,y)$ denota una versión rotada o proyectada de los coeficientes $L_{n-m,m}(x,y)$ con respecto a un ángulo de preferencia θ.

Para su obtención, primero se aplica la transfomada Hermite y los coeficientes de esta transformación son rotados en un orientación local estimada, de acuerdo al criterio de máxima energía de orientación para cada posición de la ventana, lo que implica que estos filtros pueden dirigir la orientación de un patrón 1D independiente de su estructura interna.

Los coeficientes polinomiales bidimensionales se proyectan en coeficientes de una dimensión sobre un eje que hace un ángulo θ con el eje x. Los coeficientes polinomiales de primer orden llegan a semejar un detector de bordes óptimo, $\theta = \tanh L_{01}/L_{10}$. La Figura ?? muestra los pasos para una descomposición direccional de Hermite sobre una imagen, dónde se observa que la energía de los coeficientes se concentra en la dirección horizontal.

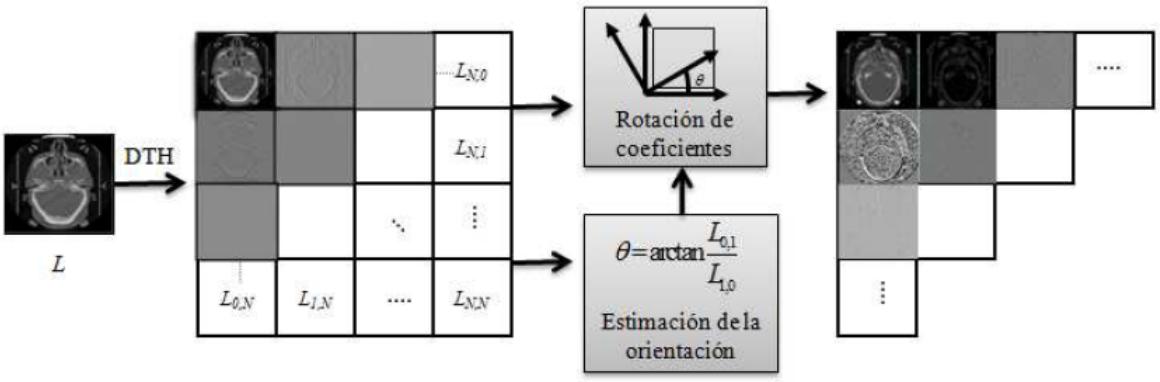


FIGURA 3.4: Descomposición de una imagen con la transformada Hermite rotada

3.3.3. Transformada de Hermite 2D multiresolución

En una aplicación de análisis de visión computacional por lo general se desea saber que parte de la información contenida en una imagen debe agregarse con mayor intensidad (bordes, esquinas y líneas) y que parte de información deber ser atenuada (texturas finas o ruido). Para ello, resulta apropiado efectuar un análisis derivativo para discernir entre las diferentes estructuras de interés[8][9].

En representaciones espacio-escala bidimensional la señal $L(x, y)$ es convolucionada con versiones escaladas de una Gaussiana normalizada para eliminar detalles, de manera que

$$L(x, y, s) = L(x, y) * G(x, y, s)$$

es la representación de la señal a la escala s, donde $G(x, y, s)$ es la Gaussiana con el parámetro de escala s (un medio de la varianza de la Gaussiana). La estructura local de una señal se infiere a partir de las derivadas de las representaciones escaladas, las cuales se pueden obtener convolucionando la función $L(x, y)$ con los operadores derivadas de Gaussianas escalados

$$G_{n-m,m}(x, y, s) = G_{n-m}(x, s)G_m(y, s)$$

para $m = 0, 1, ..., n$ y $n = 0, 1, ...\infty$ donde n es el orden de la derivación total.

Así, para las señales bidimensionales la representación multirresolución se lleva a cabo filtrando la señal en cascada a lo largo de cada coordenada bajo un esquema piramidal, en la cual la imagen se descompone en un número de subimágenes pasobaja o pasobanda,

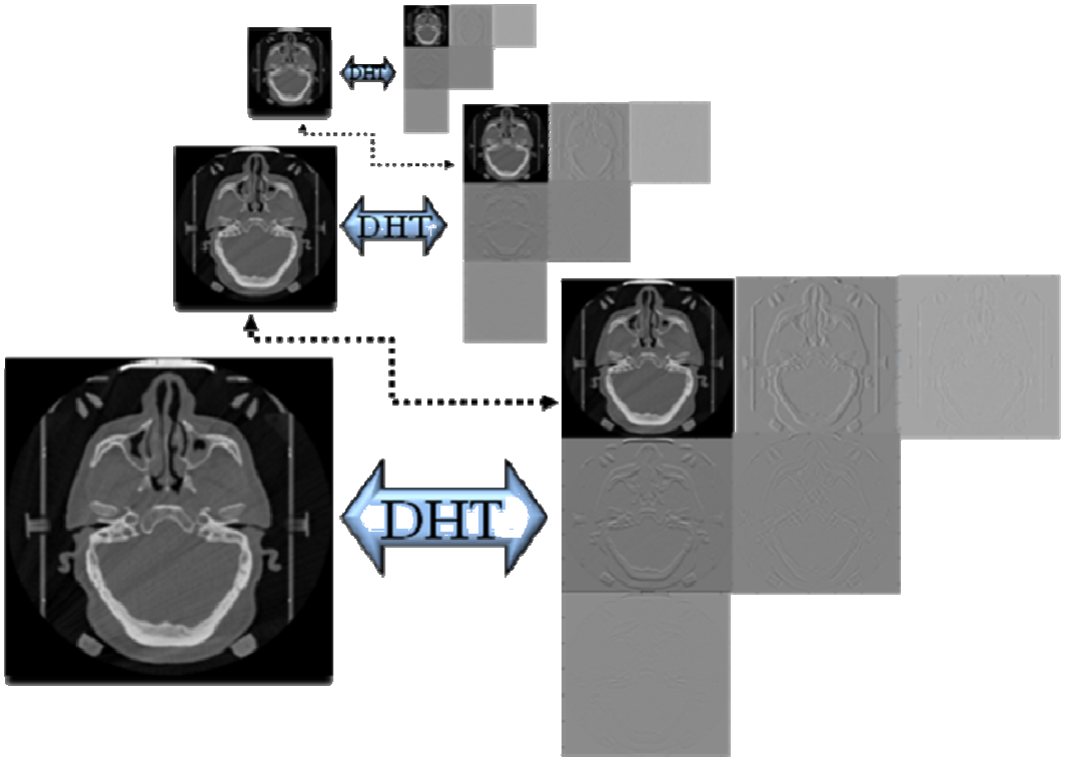


FIGURA 3.5: Expansión multiresolución de la transformada Hermite

las cuales son submuestreadas en proporción a su resolución obteniéndose la información de las estructuras de la imagen a diferentes escalas.

Otro factor físico que influye en la representación de los patrones visuales es la inclinación relativa del sistema de visión con respecto a un eje de referencia en la escena. Dicha inclinación hará que las estructuras de la imagen se perciban con una orientación determinada, la cual sólo puede ser vista por operadores sensibles a la orientación. De esta manera, para propósitos de análisis de orientación es conveniente trabajar con las versiones rotadas de estos operadores, es decir

$$G_{n-m,m}(x, y, s, \theta) = G_{n-m,m}(x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta, s) \tag{3.32}$$

donde θ es el ángulo de rotación[17]. En particular, las derivadas direccionales de Gaussianas denotadas por $G_{n,0}(x, y, s, \theta)$ se expresa como

$$G_{n,0}(x, y, s, \theta) = \sum_{m=0}^{n} \alpha_{n-m,m}(\theta) G_{n-m,m}(x, y, s) \tag{3.33}$$

donde $m = 0, 1, ..., n$ y $n = 0, 1, ...\infty$ con la función de ángulo de la ecuación 3.30. En

este caso, la expansión de las diferencias de Gaussianas (DoG: Diference of Gaussians) se escribe de la siguiente manera:

$$DoG(x, y, s_{k-1}, s_k) = \sum_{n=1}^{\infty} \sum_{j=0}^{n} \frac{c_n(-\tau s_k)^n}{n!} G_{2n,0}(x, y, s_k, \theta_j) \tag{3.34}$$

donde $\theta_j = \theta_0 + j_\pi/(n+1)$ para $j = 0, 1, ...n$ y

$$c_n = \frac{1}{\sum_{j=1}^{n} \sin^{2n}(\frac{j\pi}{n+1})} = \frac{1}{n+1} \sum_{m=0}^{n} \frac{(C_n^m)^2}{C_{2n}^{2m}} \tag{3.35}$$

para $n = 0, 1, ..., \infty$ Nótese que el orden de derivación n determina la resolución angular y que el ángulo θ_0 es un parámetro libre que puede elegirse según convenga; mientras que el parámetro de espaciamiento entre escalas,τ, generalmente se requiere que sea constante para todo entero k en $\tau = (s_k - s_{k-1})$, lo cual está de acuerdo con los datos psicofísicos que sostiene la existencia en el Sistema de visión humano de un número de canales cuyas frecuencias centrales mantienen una relación de aproximadamente una octava, es decir,$\tau \approx 0{,}75$

Partiendo de la expresión 3.34 es posible construir una transformada que resulte más apropiada para la descripción de imágenes, ya que las funciones de análisis que se obtienen incluyen las tres transformaciones geométricas fundamentales: traslación, rotación y escalamiento. El proceso de análisis se formaliza interpretando las operaciones de convolución como producto interno en la siguiente expresión

$$L_n^{k,j}(\xi, n) = \langle L(x, y), G_n^{k,j}(x, y, \xi, \eta) \rangle_{x,y} \tag{3.36}$$

mediante las funciones de análisis

$$G_n^{k,j}(x, y, \xi, \eta) = \frac{1}{2s_k} G_{n,0}^*(\frac{x\cos\theta_j + y\sin\theta_j}{\sqrt{2s_k}} - \xi, \frac{-x\sin\theta_j + y\cos\theta_j}{\sqrt{2s_k}} - \eta) \tag{3.37}$$

donde $\theta_j = \theta_0 + j_\pi/(n+1)$ para $j = 0, ..., n$; $n = 1, 2, ...$ y $s_k = (1-\tau)^{-k}$ para todo k entero. Este análisis es referido como **transformada Hermite multirresolución multidireccional**(MMHT: multidirectional multiresolution Hermite transform) debido a que emplea las derivadas direccionales de la Gaussiana a múltiples escalas y orientaciones y en consecuencia los coeficientes generados $L_n^{(k,j)}$ contienen no sólo información sobre la localización sino también sobre orientación de las estructuras de la imagen a

distintas escalas. De manera análoga, al proceso de reconstrucción de la señal a partir de esta representación se le denomina MMHT inversa y se expresa como

$$L(x,y) = \sum_{k=-\infty}^{\infty} \sum_{n=1}^{\infty} \sum_{j=0}^{n} c_n \tau^n D_n^{(k,j)}(x,y) \tag{3.38}$$

donde las funciones $D_n^{(k,j)}(x,y)$ para $n = 1, 2, ...$aportan los detalles direccionales de la señal a todas las escalas.

$$D_n^{(k,j)}(x,y) = \langle L_n^{(k,j)}(\xi,\eta), G_n^{k,j}(x,y,\xi,\eta) \rangle_{(\xi,\eta)} \tag{3.39}$$

Para el caso discreto, el filtro DoG pueder ser aproximado por una Diferencia de Binomiales (DoB)y dado que las funciones Binomiales son de soporte compacto, su representación mediante las diferencias binomiales se expresa como una suma finita y a partir de esta aproximación se construye la versión discreta de la MMHT[17].

3.4. Transformada de Hermite 3D

En el caso tridimensional[16] los coeficientes de Hermite $L_{l,m-l,n-m}$ se obtienen mediante la convolución de la señal original $L(x,y,z)$ con los filtros de análisis $D_l(x), D_{m-l}(y)$ y $D_{n-m}(z)$ seguido de un submuestreo sobre una malla tridimensional S. Por otro lado, la transformada Hermite discreta inversa se define como:

$$L(x,y,z) = \sum_{n=0}^{\infty} \sum_{m=0}^{n} \sum_{l=0}^{m} \sum_{(p,q,r)\in\ S} L_{l,m-l,n-m}(p,q,r) P_l(x-p) P_{m-l}(y-q) P_{n-m}(z-r) \tag{3.40}$$

donde $P_l(x), P_{m-l}(y)$ y $P_{n-m}(z)$ son los filtros de síntesis para las direcciones x, y y z respectivamente. La figura muestra la distribución tridimensional de los coeficientes de la transformada Hermite hasta orden 2 en cada dirección, de manera que en lugar de tener 9 coeficientes por cada pixel en una imagen, en este caso se obtienen 27 coeficientes por cada voxel del volumen.

La transformada de Fourier de los filtros de análisis se puede expresar en coordenadas esféricas de la siguiente manera:

$$d_1(\omega_x) d_{m-1}(\omega_y) d_{n-m}(\omega_z) = \alpha_{l,m-l}(\theta) \alpha_{m,n-m}(\phi) d_n(\omega) \tag{3.41}$$

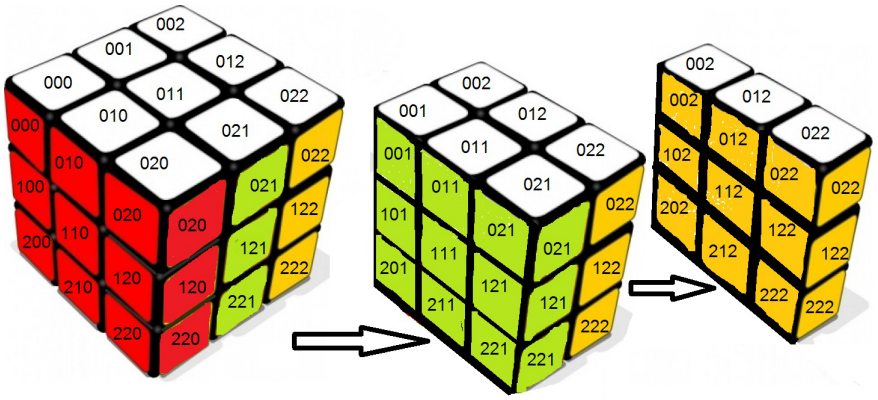


FIGURA 3.6: Distribución de los coeficientes de segundo orden de un voxel

donde $\omega_x = \omega \cos\theta \cos\phi$, $\omega_y = \omega \sin\theta \cos\phi$ $\omega_z = \omega \sin\phi$ y $d_n(\omega)$ es la transformada de Fourier del filtro de Hermite unidimensional $D_n(r)$, donde r es la coordenada polar y la función α que expresa la selectividad direccional del filtro es la misma que la introducida en la ecuación (3.30).

Por otro lado, el mejor ajuste de la señal original $L(x, y, z)$ mediante un patrón K unidimensional

$$K\left((x\cos\theta + y\sin\theta)\cos\phi + z\sin\phi\right) \tag{3.42}$$

se encuentra al maximizar la energía direccional

$$\sum_{n=0}^{\infty} K^2_{n,\theta,\phi} = \sum_{n=0}^{\infty}\left[\sum_{m=0}^{n}\sum_{l=0}^{m}\alpha_{l,m-l}(\theta)\alpha_{m,n-m}(\phi)L_{l,m-l,n-m}\right] \tag{3.43}$$

para todo (θ, ϕ). En consecuencia, la mejor aproximación 1D de los coeficientes de Hermite 3D está dada por

$$\widehat{L}_{l,m-l,n-m} = K_{n,\theta,\phi}\alpha_{l,m-l}(\theta)\alpha_{m,n-m}(\phi) \tag{3.44}$$

para $n = 0, ...\infty, m = 0...n$ y $l = 0...m$, con (θ, ϕ) como los ángulos óptimos, donde $\widehat{L}_{l,m-l,n-m}$ son los coeficientes Hermite rotados. muestra la HT y la HT rotada sobre un volumen mostrando el cambio de sistema de referencia y la aproximación de los ángulos θ y ϕ de acuerdo a la dirección del gradiente.

Una descomposición multirresolución usando la HT puede ser obtenida a través de un esquema piramidal[9][8]. En una descomposición piramidal el volumen es descompuesto en un número de subvolúmenes paso banda o paso baja, los cuales son posteriormente

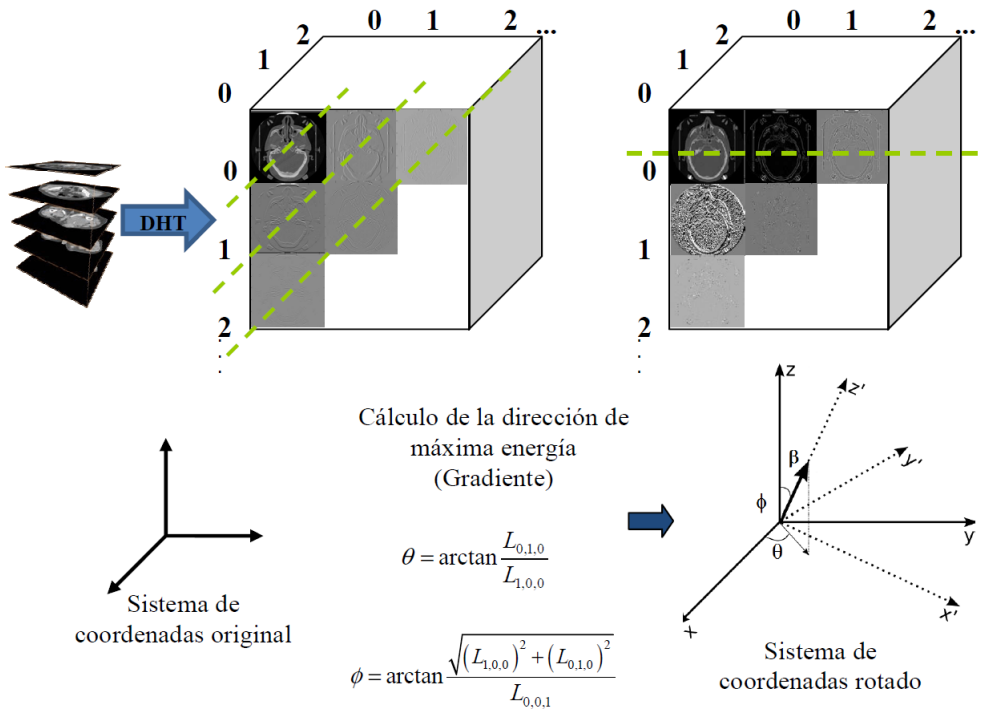


FIGURA 3.7: La transformada Hermite 3D en un sistema coordenadas cartesiano y su rotación sobre un volumen.

submuestreados en proporción a su resolución espacial. En cada subvolumen se transforman los coeficientes de orden cero para obtener una versión escalada de la anterior.

Una vez obtenida la descomposición en coeficientes Hermite de cada nivel, los coeficientes pueden ser proyectados a una dimensión por su orientación local de máxima energía, de esta forma se obtiene la **transformada Hermite multirresolución localmente orientada 3D**. Así este esquema multirresolución de la HT permite obtener información sobre la localización y orientación de la estructura de la imagen a distintas escalas.

3.4.1. Algoritmo rápido para la obtención de la DHT 3D

Para lograr una implementación discreta de la transformada Hermite se sigue el análisis de Hashimoto y Sklansky para la obtención de aproximaciones discretas de derivadas de Gaussianas[21]. Para explicar el método se partirá del caso unidimensional y posteriormente se extenderá al caso tridimensional. El procedimiento toma como base un vector

definido como se muestra a continuación:

$$d_{N,k} = 2^{N-k} \begin{bmatrix} d_{N,k}(N) \\ d_{N,k}(N-1) \\ \vdots \\ d_{N,k}(0) \end{bmatrix} \quad para\ k = 0, 1, ..., N \tag{3.45}$$

En el caso de la transformada en una dimensión, las estimaciones de las derivadas pueden expresarse como

$$\hat{f}^{(k)} = \frac{1}{2^{N-k}} d^T_{(N,k)} f_0 \ para\ k = 0, 1, ..., N \tag{3.46}$$

donde f_0 es un vector de datos en una dimensión de tamaño $(N+1)$ y las derivadas son estimadas en el centro de la ventana. El factor de escala, 2^{N-k}, se elige de tal forma que los vectores tengan las mínimas expresiones enteras. Cada elemento del vector resultante representa un coeficiente de orden k de la transformada Hermite en un punto dado.

Para determinar el vector $d_{N,k}$, se define una matriz $[P_n]$ de dimensiones $N+1$ por $N+1$ cuyo $k+1$ ésimo renglón es $d^T_{N,k}$. Es así entonces, que el conjunto de ecuaciones 3.46 se puede expresar en forma matricial:

$$\mathbf{F} = (F_0, F_1, ...F_N)^T \mathbf{F} = [P_N] f_0 \tag{3.47}$$

donde $F_k = 2^{N-k} \hat{f}^k$. Las matrices $[P_N]$ para $N = 1, 2, 4$ son las siguientes:

$$N = 1 \begin{vmatrix} 1 & 1 \\ -1 & 1 \end{vmatrix} \quad N = 2 \begin{vmatrix} 1 & 2 & 1 \\ -1 & 0 & 1 \\ 1 & -2 & 1 \end{vmatrix} \quad N = 3 \begin{vmatrix} 1 & 3 & 3 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & 3 & -3 & 1 \end{vmatrix} \quad N = 4 \begin{vmatrix} 1 & 4 & 6 & 4 & 1 \\ -1 & -2 & 0 & 2 & 1 \\ 1 & 0 & -2 & 0 & 1 \\ 1 & -4 & 6 & -4 & 1 \end{vmatrix} \tag{3.48}$$

En tres dimensiones, las derivadas parciales se estiman mediante el uso de operadores de una dimensión para cada dirección. Podemos escribir una matriz de derivadas parciales [F] como se muestra en la Figura 3.8. Los elementos $F^{(k)}$ representan la derivada parcial estimada con factor de escala $2^{2N-k-l-m}$, donde los elementos de las derivadas parciales que tiene el mismo orden, cuentan también con el mismo factor de escala.

En general, la estimación de filtros de derivadas de orden múltiple puede implementarse mediante la aplicación de diversos algoritmos:

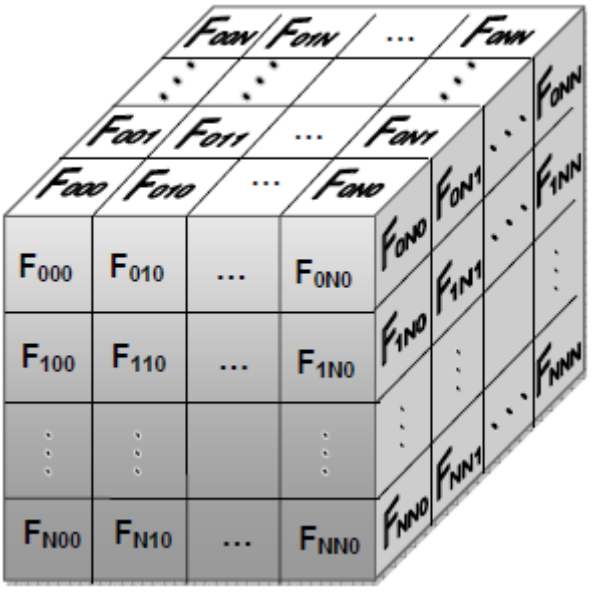


FIGURA 3.8: Matriz de derivadas parciales en tres dimensiones

1. Efectuando la convolución de la imagen tridimensional con un conjunto de filtros de tres dimensiones,

2. realizando multiplicaciones de matrices en cada ventana de datos,

3. convolucionando la imagen con un conjunto de máscaras de una dimensión en cada una de las direcciones de ésta, y

4. mediante convoluciones repetidas por dos secuencias 1, 1 y 1, -1 en las tres direcciones.

De los algoritmos mencionados, la implementación de un algoritmo rápido para estimar las derivadas de orden multiple[1], la repetición de convoluciones puede ser implementada con el menor número de operaciones, requiriendo únicamente adiciones y diferencias, ésto colocando en paralelo varias configuraciones de filtros de primer orden en cascada.

Dado que las funciones de transferencia de los filtros de estimación de derivadas contienen términos en común, las operaciones redundantes pueden ser eliminadas al compartir los resultados intermedios. Para valores muy grandes de N, es posible construir el algoritmo colocando filtros de primer orden sucesivamente en cascada, tal y como lo muestra la figura 3.9, en la que se observa que los filtros son implementados con una suma y un

[1]Para este trabajo se realizaron hasta nivel dos, ya que en el trabajo efectuado en [26] la fusión de datos tiene buenos resultados con una transformada a dos niveles de resolución

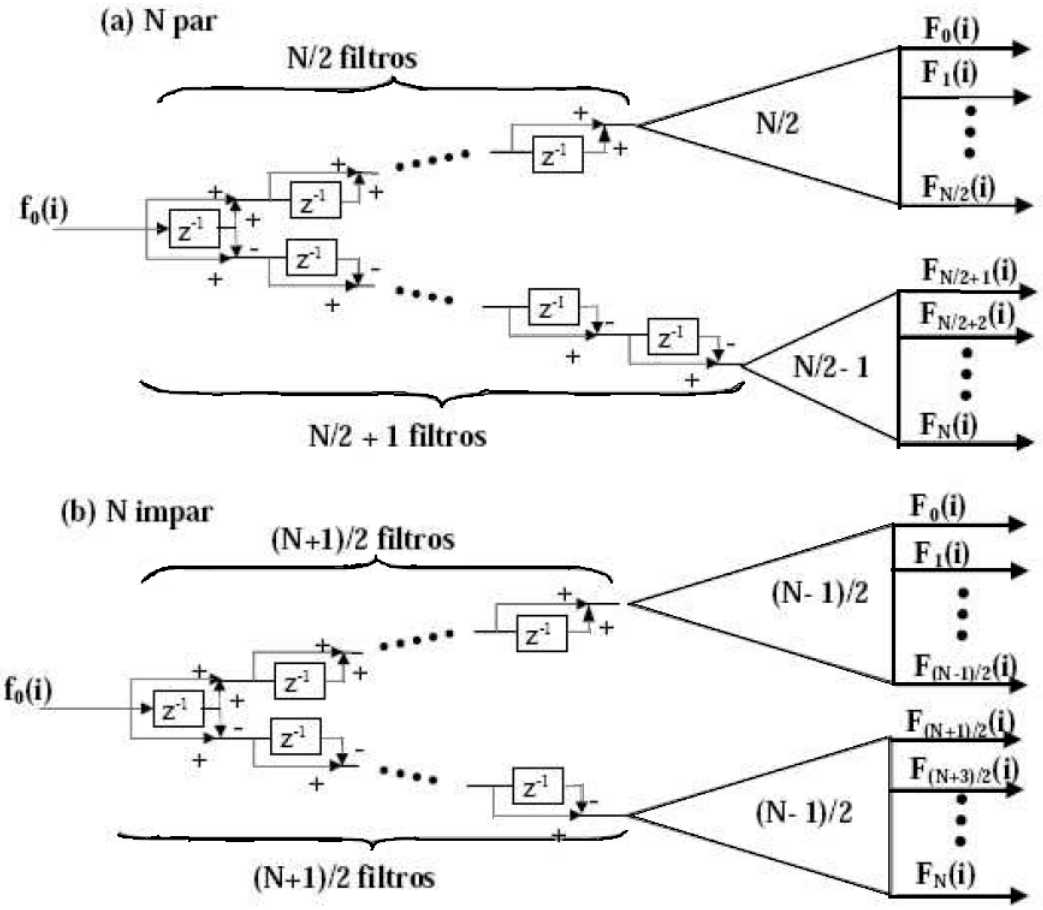


FIGURA 3.9: Algoritmo rápido para valores grandes de N

retraso, y cada triángulo representa un algoritmo rápido para un tamaño en específico dependiendo si N es número par o impar.

Cuando $N+1$ es un potencia de dos, el número de sumas por muestra a analizar está dado por $(N+1)\log_2(N+1)$ debido a que, si observamos bien, la estructura completa se separa en dos subestructuras y cada una de estas a su vez se separa en otras dos y asi sucesivamente hasta un número de veces regido por el factor $\log_2(N+1)$. En caso de que $N+1$ no sea un potencia de dos, el entero más cercano a $(N+1)\log_2(N+1)$ es aquél que nos dará el número exacto de sumas a realizar.

Una propiedad importante de los filtros de estimación de derivadas es su separabilidad, esto es, que las derivadas parciales puede ser aproximadas al aplicar el algoritmo rápido para una sola dimensión de datos de entrada y después a las demás del resultado. La estructura del algoritmo es la misma que se muestra en la Figura 3.9, sólo que en lugar de obtener las derivacioens hasta segundo orden en cada dirección se aplican en total seis filtros en cascada. Los elementos que componen el algoritmo son:

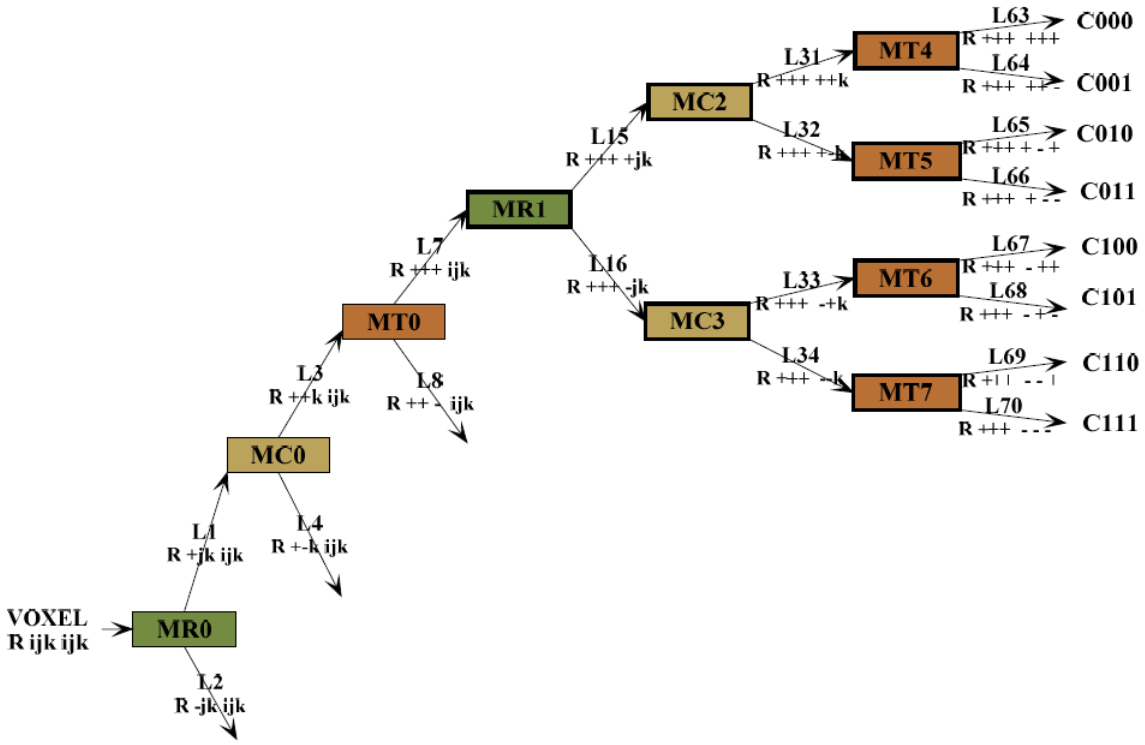


FIGURA 3.10: Estructura de registros para la implementación de la DHT 3D con N=2

- Variable de entrada 'voxel' que contiene el valor del voxel en turno a procesar.
- Registros para almacenar los retrasos en cada una de las direcciones. Se requierem 18 registros para almacenar los retrasos de columnas (MC), 9 registros por cada columna para los retrasos de renglones (MR) y 36 registros por cada corte para los retrasos entre cortes(MT).
- Las variables auxiliares L1,...,L62 que contienen los valores de salida de los filtros y que posteriormente son almacenados en los arreglos MR, MC y MT correspondientes.
- Los coeficientes polinomiales de salida, C000,C001,C002,C010,...,C222

El volumen se procesa a partir del voxel(en la posición x=1,y=1,z=1), ubicado en la esquina superior izquierda del volumen, siguiendo con los voxeles de ese mismo renglón tomando columna a columna el voxel correspondiente; posteriormente este valor se almacena en la primera posición del arreglo MR y se tendrá ahora en la entrada al voxel que ocupa la posición (0,0,1), y así sucesivamente.La Figura muestra sólo una parte de dicha estructura.

El árbol empleado en el algoritmo rápido de la transformada Hermite se basa en la aplicación sucesiva de sumas y diferencias para obtener las distintas combinaciones de derivadas en las tres dimensiones x, y y z.

3.4.2. Algoritmo para la rotación de coeficientes de la DHT

La rotación de coeficientes consiste en la proyección de los coeficientes de la transformada Hermite a una dirección, con lo cual se obtienen los ángulos que indican la dirección de máxima energía y dos coeficientes unidireccionales, uno que contiene la información de los coeficientes de primer orden C001,C010,C100, y otro con la información de coeficientes de segundo orden C011,C110,C101,C002,C020,C200.

3.4.2.1. Proyección de los coeficientes en 3D a 1D

El primer coeficiente unidireccional se obtiene de la siguiente forma

$$h_1(\theta,\phi) = \cos(\phi)L_{0,0,1} + \sin(\theta)\sin(\phi) + \cos(\theta)\sin(\phi)L_{1,0,0} \tag{3.49}$$

donde los ángulos θ y ϕ se aproximan de la siguiente forma:

$$\phi = \arctan\frac{\sqrt{(L_{1,0,0})^2 + (L_{0,1,0})^2}}{L_{0,0,1}} \quad y \quad \theta = \arctan\frac{L_{0,1,0}}{L_{1,0,0}} \tag{3.50}$$

A continuación se muestra el pseudocódigo de la función que proyecta los coeficientes en 3D a 1D para obtener el primer coeficiente unidireccional a partir de la información de coeficientes de primer orden:

Algorithm 1 Proyeccion 3d a 1d

for $t = 1$ to cortes **do**
  for $i = 1$ to renglones **do**
    for $j = 1$ to columnas **do**
      $\theta(i,j,t) = \arctan[C010(i,j,t)/C100(i,j,t)]$
      $\phi(i,j,t) = \arctan(\sqrt{C100(i,j,t)^2 + C010(i,j,t)^2}/C001(i,j,t))$
      $Mh1(i,j,t) = \cos\phi * C001(i,j,t) + \sin\theta * \sin\phi * C010(i,j,t) + \cos\theta * \sin\phi * C100$
    end for
  end for
end for

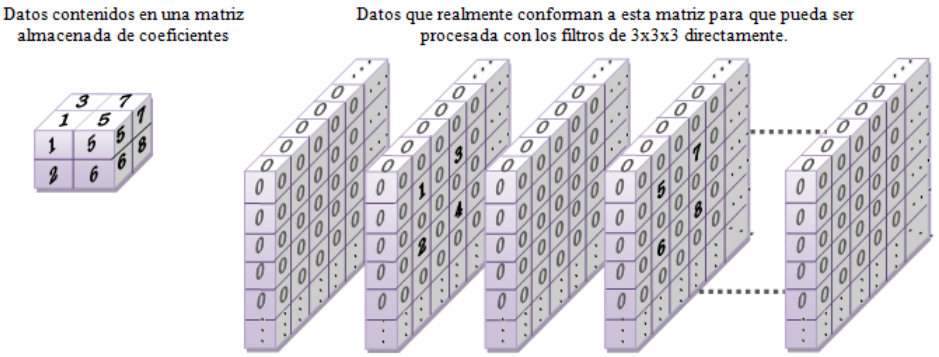


FIGURA 3.11: Disposición de los datos obtenidos en el programa contra si significado real

3.5. Algoritmo rápido para la obtención de la IDHT 3D

Para la reconstrucción se deben considerar ceros de la interpolación. Para ello se debe suponer que entre cada voxel de los coeficientes de la transformada existen ceros que rellenan la matriz, los cuales no están realmente almacenados en ésta por ser información redundante. En la Figura 3.11 se muestra un ejemplo de la disposición real de los datos con lo que en realidad representan. Debido a esta compactación de datos, la solución consistió en identificar los casos posibles que se presentan al aplicar el filtro tridimensional en los datos de la matriz no submuestreada considerando para ello que hay ceros entre cada localidad. Haciendo estas suposiciones, se consideran ocho casos en los cuales las posiciones de los ceros imaginarios entre los datos tomaban una disposición similar. En la Figura 3.12 se muestran dichos casos, los primeros cuatro corresponden al barrido de columnas y renglones de los cortes pares mientras que los casos restantes se refieren al barrido de los cortes impares(partiendo de la idea de que el primer corte es par ya que comienza a contar desde cero). Como se puede observar, todos los casos envuelven muchos ceros dentro de la región de filtrado para un voxel en particular, por lo que si se considera un filtro genérico de $3x3x3$ de la forma mostrada en la Figura 3.13 y que el barrido se hace sobre la columna j y el renglón i y el corte k, tomando el centro del cubo como el lugar al que apuntan los valores (i,j,k), se plantean las siguientes funciones para generar el filtrado para cada matriz de coeficientes, combinando al final estos valores para obtener la matriz recuperada.

Caso 1: $F_{222} * V_{(i,j,k)}$

Caso 2: $F_{212} * V_{(i,j-1,k)} + F_{232} * V_{(i,j+1,k)}$

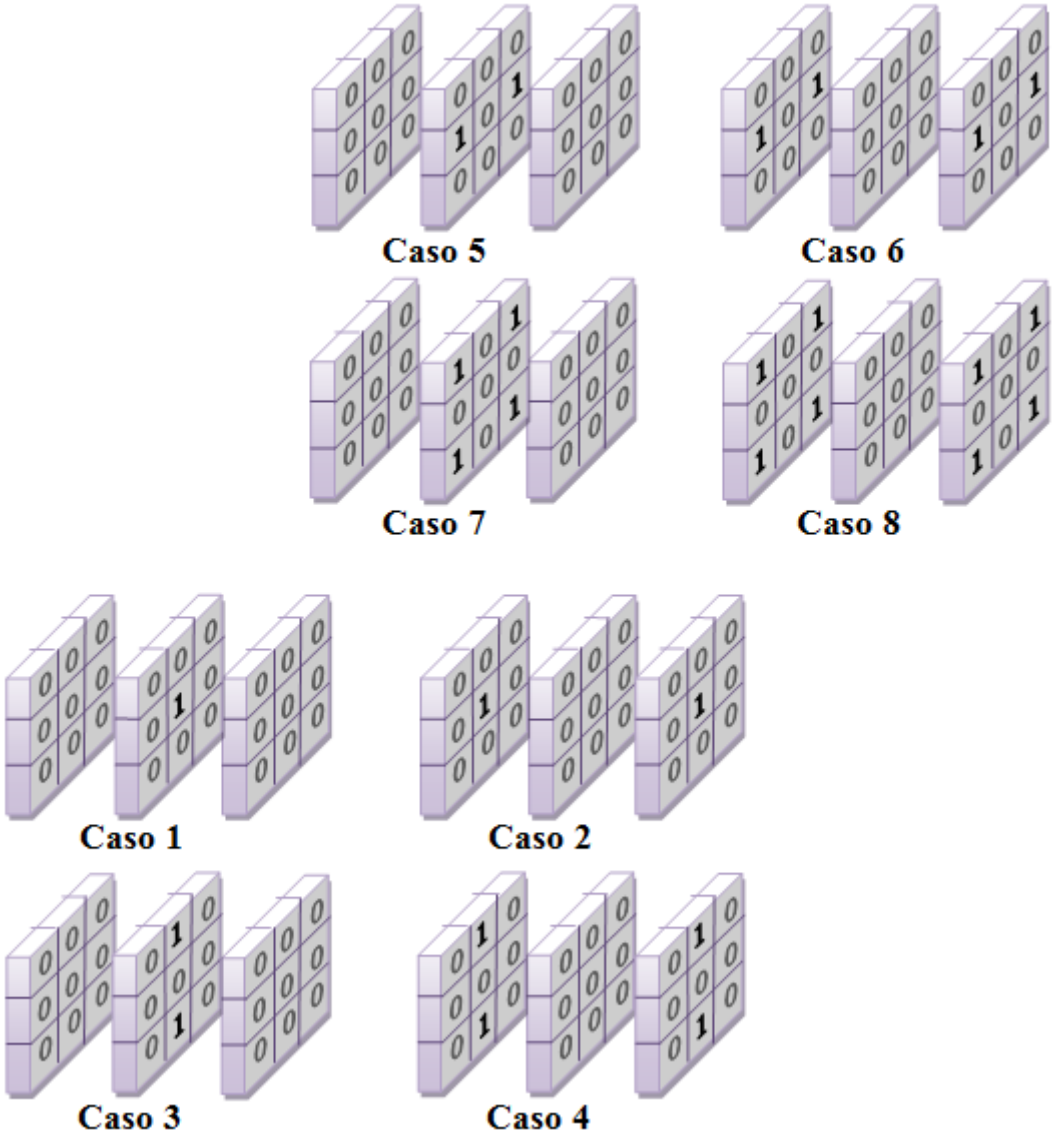


FIGURA 3.12: Cuatro disposiciones posibles de los ceros que existen entre los datos de las matrices de coeficientes

Caso 3: $F_{122} * V_{(i-1,j,k)} + F_{322} * V_{(i+1,j,k)}$

Caso 4: $F_{112} * V_{(i-1,j-1,k)} + F_{312} * V_{(i+1,j-1,k)} + F_{132} * V_{(i-1,j+1,k)} + F_{332} * V_{(i+1,j+1,k)}$

Caso 5: $F_{221} * V_{(i,j,k-1)} + F_{223} * V_{(i,j,k+1)}$

Caso 6: $F_{211} * V_{(i,j-1,k-1)} + F_{213} * V_{(i,j-1,k+1)} + F_{231} * V_{(i,j+1,k-1)} + F_{233} * V_{(i,j+1,k+1)}$

Caso 7: $F_{121} * V_{i-1,j,k-1} + F_{123} * V_{(i-1,j,k+1)} + F_{321} * V_{(i+1,j,k-1)} + F_{323} * V_{(i+1,j,k+1)}$

Caso 8: $F_{111} * V_{(i-1,j-1,k-1)} + F_{113} * V_{(i-1,j-1k+1)} + F_{311} * V_{(i+1,j-1,k-1)} + F_{313} * V_{(i+1,j-1,k+1)}$
$+ F_{131} * V_{(i-1,j+1,k-1)} + F_{133} * V_{(i-1,j+1,k+1)} + F_{331} * V_{(i+1,j+1,k-1)} + F_{333} * V_{(i+1,j+1,k+1)}$

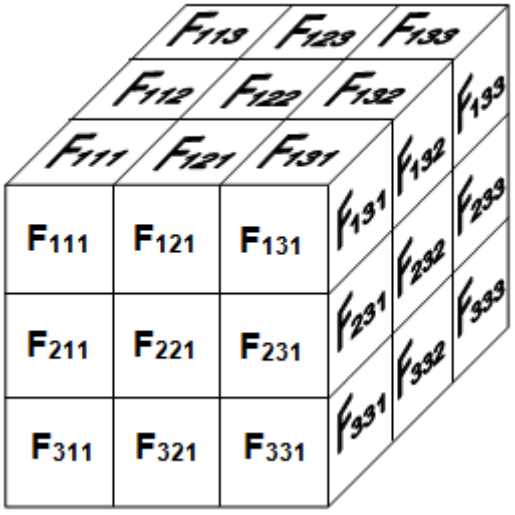


FIGURA 3.13: Representación de un filtro genérico de 3x3x3

De esta manera, los valores de F_{MNO} tomarán los valores adecuados dependiendo del filtro de síntesis que se requiera utilizar, por ejemplo, se está procesando la matriz de coeficientes de orden 000, los valores del filtro F_{MNO} deberán coincidir con los valores para este filtro de antitransformación. Cabe mencionar que en este proceso aún se pueden eliminar algunas sumas, puesto que en ciertos casos F_{MNO} para ciertos valores de 'M','N' y 'O' y para determinado orden, se tienen valores nulos que simplifican aún más las sumas de los casos anteriores. Adicionalmente falta considerar un factor de normalización dependiendo del orden del filtro de síntesis, este factor está relacionado con una potencia de 2 y puede ser considerado al momento de operar con las matrices de coeficientes multiplicándolas por su factor antes de analizar los casos.

Siguiendo este razonamiento, se observa que nuevamente se requieren localidades de memoria que almacenen los valores contenidos en las columnas, renglones y cortes anteriores al voxel procesado. Estos registros son los siguientes:

- MA(18,núm. de columnas)- Almacenan los valores del renglón anterior de las matrices de coeficientes, sólo son 18 porque no todos los 27 filtros de síntesis tienen valores diferentes de cero en la columna central. Para considerar el factor de normalización los valores se multiplican por una constante, para los filtros F000,F002,F020,F200, F220,F202,F022 y F222 el factor es 2, para F001, F021,F201,F221,F100,F102,F120 y F122 el factor es 4 y para F101 y F121 el factor es 8.

- MS(27,núm. de columnas)- Almacenan las sumas y/o diferencias de valores adyacentes del renglón anterior de las matrices de coeficientes. Se considera la suma cuando los signos de los coeficientes de los filtros de síntesis entre las columnas $j-1$ y $j+1$ son iguales, en caso contrario, se guarda la diferencia de los valores contiguos de las matrices de coeficientes. Por ejemplo, para la matriz de coeficientes de orden 001 se guardan las sumas de los valores contiguos ya que su filtro de síntesis tiene el mismo signo entre las posiciones 111 y 131, 113 y 311, 313 y 333. En cambio, en la matriz de coeficientes de orden 010 se guarda las diferencias porque en las posiciones mencionadas los signos son contrarios. Si los signos son iguales pero negativos, o bien, la diferencia está invertida, por ejemplo -111 y +131, entonces al aplicar alguno de los 8 casos planteados se cambia el signo de ese término en la sumatoria. Para considerar el factor de normalización, algunas de estas sumas y/o diferencias se afecta por una potencia de 2, para los filtros F001, F010, F012, F021,F201,F221,F100,F102,F120,F122,F210 y F212 el factor es 2, para F011,F101,F110,F112,F121 y F211 el factor es 4 y para F111 el factor es 8.

- MZA(18, núm.renglones,núm.columnas)- Almacena los valores del renglón anterior de las matrices de coeficientes que tiene las mismas características descritas para MA.

- MZS(27,núm.renglones,núm.columnas)- Almacena todas las sumas y/o diferencias del corte anterior con las mismas consideraciones de MS.

- Mtmp(Arreglo temporal que almacena ya sea los valores de MA o MS de un renglón anterior para procesar los casos 3,4,7 y 8).

Capítulo 4

Algoritmo de fusión

4.1. Esquema de fusión basado en la transformada Hermite

El algoritmo propuesto plantea la fusión de datos tridimensionales mediante el uso de la transformada Hermite rotada multirresolución (hasta 2 niveles) y la aplicación de distintas reglas de selección de coeficientes para bajas y altas frecuencias. A grandes razgos, incluye las siguientes etapas:

1. Obtención de la HT multirresolución de los volúmenes de entrada, a fin de obtener descriptores para texturas y bordes que faciliten su identificación para su preservación durante el proceso de fusión.

2. Detección de la orientación de máxima energía para rotar los coeficientes de la HT, obteniéndose así proyecciones en una dimensión para cada nivel de descomposición. Para ello se calculan los ángulos θ y ϕ de acuerdo a la dirección del gradiente y se aplican las funciones de ángulos a los coeficientes a proyectar.

3. Selección de coeficientes de la descomposición multirresolución obtenida en el paso 1) con la información de los coeficientes rotados en el paso 2) para conformar la HT multirresoluciób del resultado de la fusión. La regla de fusión se conforma de dos métodos de selección dependiendo del tipo de frecuencia

    - **Selección de coeficientes de baja frecuencia.** Los coeficientes de aproximación en el caso de la HT equivalen a los coeficientes de orden cero (L_{000}),

los cuales contienen las características de las zonas homogéneas. Para formar el coeficiente de orden cero del volumen fusionado simplemente se promedian los valores de los coeficientes L_{000} de ambos volúmenes.

- **Selección de coeficientes de alta frecuencia.** Se toma el coeficiente de primer orden rotado de cada nivel de resolución de los volúmenes de entrada para aplicar una regla de selección de coeficientes paso altas. Se considera sólo ese coeficiente puesto que la información de los bordes del volumen en una determinada localidad espacial. El resultado de esta regla de selección genera un mapa de decisión que se aplicará a los coeficientes de la HT multirresolución del paso 1) y con ello se genera el conjunto de coeficientes de detalle del volumen fusionado.

4. Obtención del volumen fusionado al aplicar la HT inversa multirresolución donde los coeficientes fusionados en cada nivel pueden reconstruir el conjunto de coeficientes de orden cero del nivel superior.

La elección de las reglas de selección de coeficientes depende del tipo de datos de entrada y las caracterísitcas deseadas en los datos fusionados, por ejemplo, en la elección del método de selección de coeficientes de baja frecuencia, el promedio no siempre resulta la mejor opción puesto que puede reducir notablemente el contraste del resultado, y en estos casos suele ser mejor quedarse con el coeficiente L_{000} de la imagen con mejor contraste. En el caso de los coeficientes de alta frecuencia, existen diversas reglas de selección: dependencia lineal, selección del máximo, Wronskiano.

4.1.1. Selección de coeficientes de baja frecuencia

Dado que el coeficiente L_{000}contiene aproximadamente las mismas características que los datos fuente, se aplica alguna de las siguientes reglas de selección:

1. Selección del coeficiente del volumen A

$$L_{000_F}(i,j,k) = L_{000_A}(i,j,k) \tag{4.1}$$

2. Selección del coeficiente del volumen B

$$L_{000_B}(i,j,k) = L_{000_B}(i,j,k) \tag{4.2}$$

3. Selección del promedio de los volúmenes A y B

$$L_{000_F}(i,j,k) = \frac{1}{2}\langle L_{000_A}(i,j,k) + L_{000_B}(i,j,k)\rangle \tag{4.3}$$

Si se selecciona el coeficiente del volumen A o B, el volumen fusionado tendrá mayor cantidad de características de A o B respectivamente. Ahora si se selecciona el promedio de ambos, el resultado contendrá la información promediada de los coeficientes de ambas imágenes.

4.1.2. Selección de coeficientes de alta frecuencia

Los coeficientes de frecuencias altas contienen valores que reflejan las diferencias entre una vecindad de voxeles y en consecuencia pueden ser positivos o negativos. En el caso de las imágenes en escala de grises, los valores absolutos de estos coeficientes representan la intensidad de las fluctuaciones de brillo de la escena a un escala dada, por lo que los valores más grandes implican más distinción entre los cambios de brillo que típicamente corresponden a las características sobresalientes de los objetos.

Para este caso la selección de los coeficientes se emplea los siguiente métodos:

1. Selección del máximo
2. Selección del máximo absoluto
3. Selección del máximo por verificación de consistencia
4. Dependencial lineal mediante el Wroskiano

4.1.2.1. Selección del máximo y máximo absoluto

Para esta técnica, de los pares de valores de los coeficientes que se tienen de los volumenes, se selecciona el valor que tenga mayor magnitud, así el volumen resultante (volumen

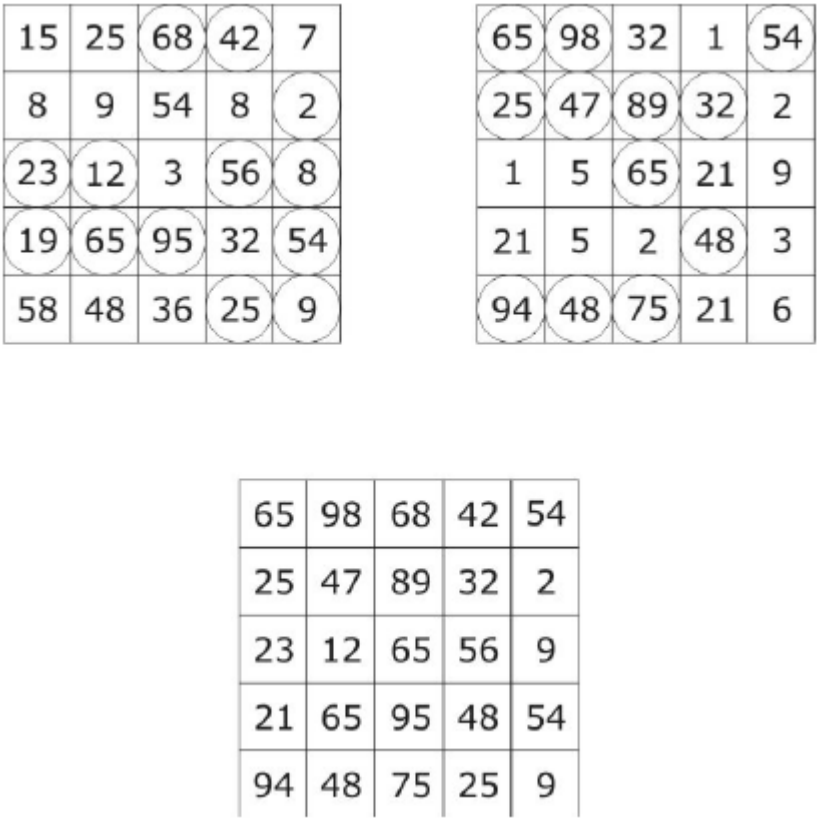


FIGURA 4.1: Selección del máximo.La figura superior izquierda muestra los coeficientes del volumen A, la figura del lado superior derecho muestra los coeficientes del volumen B. La figura inferior central muestra los coeficientes seleccionados de A y B para el volumen fusionado

fusionado) contendrá la información de los dos volúmenes[10]. Un ejemplo de cómo funcionaría esta técnica, se muestra en la Figura 4.1 con ejemplo en dos dimensiones. Los valores de los coeficientes llegan a ser número negativos, asi que el método de selección máximo absoluto consiste en tomar los valores absolutos de las matrices de proyección de los volumenes de manera a al método de selección del máximo.

4.1.2.2. Máximo y verificación de consistencia

Este es igual que el método de selección del máximo; solo que después de los máximos coeficienes, en esta, para cada posición y tomando una vecindad, por ejemplo de conectividad ocho, se determina a qué coeficiente corresponde cada vecino y sumando después los pesos (según el coeficiente al que correspondan), dicha suma que de mayor, va a indicar el coeficiente al cual pertenece esa posición; si el valor inicial corresponde a la suma de mayor peso, no se modifica y se continúa evaluando el valor de la siguiente

posición, de no ser así, se cambia por el valor en esa posición del coeficiente de mayor peso. Para entender el proceso, suponemos que la matriz A y la matriz B son las matrices Mh (coeficientes rotados)de los volumes A y B a fusionar:

$$A = \begin{matrix} 0{,}7577 & 0{,}7060 & 0{,}8235 & 0{,}4387 & 0{,}4898 \\ 0{,}7431 & 0{,}0318 & 0{,}6948 & 0{,}3816 & 0{,}4456 \\ 0{,}3922 & 0{,}2769 & 0{,}3171 & 0{,}7655 & 0{,}6463 \\ 0{,}6555 & 0{,}0462 & 0{,}9502 & 0{,}7952 & 0{,}7094 \\ 0{,}1712 & 0{,}0971 & 0{,}0344 & 0{,}1869 & 0{,}7547 \end{matrix}$$

$$B = \begin{matrix} 0{,}2760 & 0{,}4984 & 0{,}7513 & 0{,}9593 & 0{,}8407 \\ 0{,}6797 & 0{,}9597 & 0{,}2551 & 0{,}5472 & 0{,}2543 \\ 0{,}6551 & 0{,}3404 & 0{,}5060 & 0{,}1386 & 0{,}8143 \\ 0{,}1626 & 0{,}5853 & 0{,}6991 & 0{,}1493 & 0{,}2435 \\ 0{,}1190 & 0{,}2238 & 0{,}8909 & 0{,}2575 & 0{,}9293 \end{matrix}$$

A partir de las matrices A y B obtendremos dos nuevas matrices mediante la operación *Filtrado estadístico de orden 2* donde se reemplaza cada elemento en la matriz por el elemento ORDER-ésimo del conjunto de vecinos ordenados de menor a mayor y devuelve el valor mayor. Ese nuevo valor se almacena en un nueva matriz A1 y B1 las cuales quedan de la siguente manera:

$$A1 = \begin{matrix} 0{,}7577 & 0{,}7577 & 0{,}8235 & 0{,}8235 & 0{,}8235 & 0{,}4898 & 0{,}4898 \\ 0{,}7577 & 0{,}7577 & 0{,}8235 & 0{,}8235 & 0{,}8235 & 0{,}4898 & 0{,}4898 \\ 0{,}7577 & 0{,}7577 & 0{,}8235 & 0{,}8235 & 0{,}8235 & 0{,}7655 & 0{,}7655 \\ 0{,}7431 & 0{,}7431 & 0{,}9502 & 0{,}9502 & 0{,}9502 & 0{,}7952 & 0{,}7952 \\ 0{,}6555 & 0{,}6555 & 0{,}9502 & 0{,}9502 & 0{,}9502 & 0{,}7952 & 0{,}7952 \\ 0{,}6555 & 0{,}6555 & 0{,}9502 & 0{,}9502 & 0{,}9502 & 0{,}7952 & 0{,}7952 \\ 0{,}6555 & 0{,}6555 & 0{,}9502 & 0{,}9502 & 0{,}9502 & 0{,}7952 & 0{,}7547 \end{matrix}$$

$$B1 = \begin{matrix} 0{,}6797 & 0{,}9597 & 0{,}9597 & 0{,}9597 & 0{,}9593 & 0{,}9593 & 0{,}9593 \\ 0{,}9597 & 0{,}9597 & 0{,}9597 & 0{,}9597 & 0{,}9593 & 0{,}9593 & 0{,}9593 \\ 0{,}9597 & 0{,}9597 & 0{,}9597 & 0{,}9597 & 0{,}9593 & 0{,}9593 & 0{,}9593 \\ 0{,}9597 & 0{,}9597 & 0{,}9597 & 0{,}9597 & 0{,}8143 & 0{,}8143 & 0{,}8143 \\ 0{,}6551 & 0{,}6551 & 0{,}8909 & 0{,}8909 & 0{,}9293 & 0{,}9293 & 0{,}9293 \\ 0{,}5853 & 0{,}5853 & 0{,}8909 & 0{,}8909 & 0{,}9293 & 0{,}9293 & 0{,}9293 \\ 0{,}2238 & 0{,}5853 & 0{,}8909 & 0{,}8909 & 0{,}9293 & 0{,}9293 & 0{,}9293 \end{matrix}$$

Ahora a partir de las matrices A1 y B1 contruiremos una matriz binaria de $A1 > B1$ en la que se coloca 1 si el valor de A1 es mayor que B1 y cero en caso contrario. Posteriormente esa matriz binaria se le aplica una operación de convolución(en la que una ventana de $3x3$ pasa por los valores y suma el contenido)

$$A1 > B1 \; es \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{matrix}$$

$$convolucion(A1 > B1) \; es \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 3 & 3 & 4 & 3 & 2 \\ 6 & 6 & 7 & 5 & 3 \\ 9 & 9 & 9 & 6 & 3 \end{matrix}$$

Y finalmente para obtener nuestro mapa binario que se encarga de selecciona los coeficientes de los volúmenes A y B, los valores de la matriz almacenada con la convolución se comparan con el valor 13, es decir convolucion$(A1 > B1) > 13$. El valor 13 se obtiene del cociente entero del tamaño de la vecindad(27) con dos. Y así generar el mapa binario

que se muestra a continuación:

$$mapaBinario = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{matrix} \tag{4.4}$$

4.1.2.3. Dependencia lineal mediante el wroskiano

Al seleccionar los valores absolutos más grandes de los coeficientes correspondientes de cada volumen fuente, generan dos grandes desventajas: la existencia de una alta sensibilidad al ruido y la producción de un efecto de emborronamiento. Para eliminar estos efectos indeseables es necesaria la aplicación de una regla de fusión que aunque trabaje a nivel pixel considere una vecindad que ayuda a detectar si ese pixel pertenece a alguna estructura de interés.

Así con el propósito de establecer la existencia de un patrón sin realizar una extracción de características previa al proceso de fusión, se optó por una regla que considera como contexto una región pequeña alrededor del voxel de interés para evaluar la dependencia lineal local. Este criterio basado en el algebra lineal ha sido reportado en la literatura en la detección de cambios en el análisis de imágenes[14] y en esquemas de fusión de imágenes que emplean la transformada wavelet y la transformada curvelet[4, 6].

El fundamento de esta técnica se basa en un modelo empírico de vector de imágenes y la aplicación de los conceptos de dependencia e independencia lineal. Este modelo fue propuesto en [14], donde los conceptos mencionados se aplican para decidir si ha ocurrido un cambio o no entre dos imágenes.Dada una imagen, el modelo de vector se ilustra en la Figura 4.2. Cada pixel con su área de vecindad es la región de soporte de cada vector correspondiente. El pixel central de cada región de soporte es posteriormente reemplazado por el vector correspondiente. La región de soporte puede tener varios tamaños, por ejemplo 3x3, 5x5 o 7x7.

Una manera simple y rigurosa para determinar la dependencia lineal es el determinante de Wronskiano, el cual está definido para funciones pero puede aplicarse también a vectores[22].

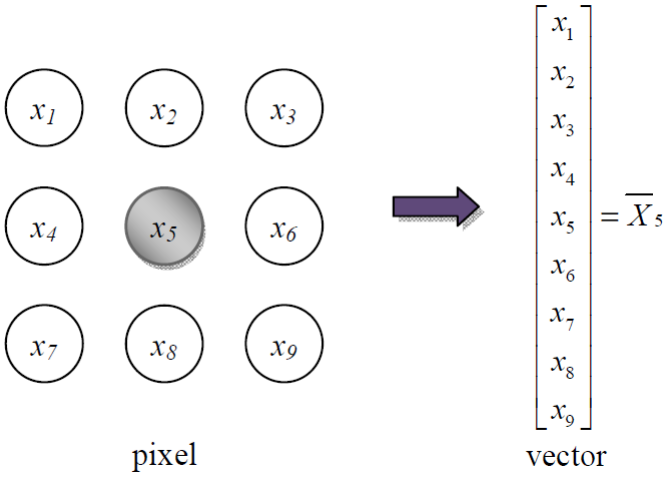


FIGURA 4.2: Construcción de los vectores a partir de una región de 3x3 en el modelo vector

Para el caso del esquema de fusión se considera una vecindad de 26 voxeles y en extensión a los esquemas encontrados en [6, 14], la prueba de dependencial lineal consiste en evaluar los voxeles dentro de un cubo pequeño de $w_s w_s w_s$, si los voxeles son linealmente independientes, no hay estructuras importantes en esa región; por el contrario, si son dependientes esto indica la existencia de un patrón que debe ser conservado en el volumen fusionado. Aplicando el determinante de Wronskiano, la dependencia de un cubo centrado en el voxel (i, j, k) se describe como

$$D_A(i,j,k) = \sum_{m=i-w_s}^{i+w_s} \sum_{n=j-w_s}^{j+w_s} \sum_{o=k-w_s}^{k+w_s} L_A^2(m,n,o) - L_A(m,n,o) \tag{4.5}$$

donde $L_A(m, n, o)$ es el coeficiente de primer orden rotado del volumen A en la posición (m, n, o). La regla es seleccionar los coeficientes de detalle con la dependencia más alta y se expresa como:

$$L_F(i,j,k) = \begin{cases} L_A(i,j,k) & if \quad D_A(i,j,k) \geq D_B(i,j,k) \\ L_B(i,j,k) & if \quad D_A(i,j,k) < D_B(i,j,k) \end{cases} \tag{4.6}$$

4.2. Programación del algoritmo

La fusión se realiza con un método llamado 'Fusionate' que requiere como valores de entrada porcentaje de combinación de los coeficientes de baja frecuencia del volumen fuente A y B, elección de método de selección de altas frecuencias así como los volúmenes

de coeficientes de los volumenes fuente A y B cuyas condiciones son tener el mismo tamaño. La primera etapa de este algoritmo consiste en realizar la transformada Hermite a dos niveles de resolución cuyo resultado será una estructura que almacene el coeficiente proyectado de primer orden "Mh'(una estructura por nivel) y una estructura que contenga los 53 volumenes de coeficientes para los dos volumenes fuente(las primeras 26 estrucuturas corresponden a los volúmenes de coeficientes del primer nivel de descomposición y las siguientes 27 estructuras corresponden al segundo y último nivel de descomposición). El primer nivel de descomposición no utilizará el coeficiente 000.

La fusión de los coeficientes inicia con la selección de elementos de baja frecuencia, es decir L000, ya sea tomando sólo los valores de los coeficientes del volumen fuente A o del volumen fuente B, o eligiendo el porcentaje que requiere proporcionar el usuario y la fusión de estructuras de los demás coeficientes de acuerdo a lo siguiente: se genera un volumen binario de decisión(formado de ceros y unos)mediante una función 'FusionCAF', el cual recibe como argumentos las estructuras de volúmenes Mh de cada volumen fuente y un método; esta última opción define la forma de obtener dicho volumen binario el cual es aplicado a cada par de estructuras de volúmenes de coeficientes correspondiente.

Una vez realizada la fusión, los coeficientes se almacenan en una estructura para efectuar la reconstrucción del volumen mediante la transformada inversa de Hermite la cual se explica en la Sección 3.5

A continuación se muestra el pseudocódigo del algoritmo para la DHT de segundo orden en tres dimensiones, cabe mencionar que este pseudocódigo considera la rotación de coeficientes.

continua...

Una vez generados los coeficientes de la transformada Hermite para los volúmenes fuente se efectua el esquema de fusión cuyo pseudocódigo se presenta en el pseudocódigo ??

La función FusionCAF define la forma de obtener un mapa binario el cual es aplicado a cada par de matrices de coeficientes correspondientes. El pseudocódigo para la fusión de coeficientes de altas frecuencias se muestra en el pseudocódigo ??

Algorithm 2 Calcula la transforma discreta Hermite Parte 1

Require: volumenFuente y sus dimensiones rows,columns,slides

1: $rows = rows + 2$
2: $columns = columns + 2$
3: $slides = slides + 2$
4: Almacenar nuestro volumen fuente en una nueva estructura 'd' (un arreglo tridimensional de valores flotantes de dimensiones [rows][columns][slides])
5: Defino variables auxiliares de tipo entero
6: $r2 = rows - 2$
7: $c2 = columns - 2$
8: $s2 = slides - 2$
9: $nRows2, nColumns2, nSlides2$
10: **for** $level = 0$ to 1 **do**
11: Defino una estructura MC de 18 elementos
12: Defino una estructura MR de [9] por [rows] elementos
13: Defino una estructura MT de [36] por [rows] por [columns] elementos
14: Defino una estructura L de [62] elementos
15: $nRows2 = tomaEnteroMenor(r2 * 0{,}5 + 1)$

Algorithm 3 DHT parte 2

16: $nColumns2 = tomaEnteroMenor(c2 * 0{,}5 + 1)$
17: $nSlides2 = tomaEnteroMenor(s2 * 0{,}5 + 1)$
18: $r2 = nRows2$
19: $c2 = nColumns2$
20: $s2 = nSlides2$
21: Define la estructura donde se almacena los coeficientes 'C' [level][nRows2][nColumns2][nSlides2]
22: $sCounter = 0$
23: **for** $s = 0$ to $s < slides$ **do**
24: $rCounter = 0$
25: **for** $r = 0$ to $r < rows$ **do**
26: $cCounter = 0$
27: **for** $c = 0$ to $c < columns$ **do**
28: L[0] = d[r][c][s] + MR[0][c]
29: L[1] = d[r][c][s] - MR[0][c]
30: L[2] = L[0] + MC[0]
31: L[3] = L[0] - MC[0]
32: L[4] = L[1] + MC[1]
33: L[5] = L[1] - MC[1]
34: L[6] = L[2] + MT[0][r][c]
35: L[7] = L[2] - MT[0][r][c]
36: L[8] = L[3] + MT[1][r][c]
37: L[9] = L[3] - MT[1][r][c]
38: L[10] = L[4] + MT[2][r][c]
39: L[11] = L[4] - MT[2][r][c]
40: L[12] = L[5] + MT[3][r][c]
41: L[13] = L[5] - MT[3][r][c]
42: $aux1 = 14$
43: **for** $aux2 = 1$ to $aux2 < 9$ **do**
44: L[aux1] = L[aux2+5] + MR[aux2][c]
45: L[aux1+1] = L[aux2+5] - MR[aux2][c]
46: **end for**
47: $aux1 = 30$
48: **for** $aux2 = 2$ to $aux2 < 18$ **do**
49: L[aux1] = L[aux2 +12] + MC[aux2]
50: L[aux1+1] = L[aux2+12] - MC[aux2]
51: **end for**
52: MR[0][c] = d[r][c][s]
53: MC[0] = L[0]
54: MC[1] = L[1]
55: MT[0][r][c] = L[2]
56: MT[1][r][c] = L[3]
57: MT[2][r][c] = L[4]
58: MT[3][r][c] = L[5]
59: **for** $aux1 = 1$ to $aux1 < 9$ **do**
60: MR[aux1][c] = L[aux1+5]
61: **end for**
62: **for** $aux1 = 2$ to $aux1 < 18$ **do**
63: MC[aux1] = L[aux1+12]
64: **end for**

Algorithm 4 DHT parte 3

65: **if** r, c, s son impares **then**

66: $C[level] \Rightarrow Coef000[rCounter][cCounter][sCounter] = L[30] + MT[4][r][c]$

67: $C[level] \Rightarrow Coef001[rCounter][cCounter][sCounter] = L[30] - MT[4][r][c]$

68: $C[level] \Rightarrow Coef002[rCounter][cCounter][sCounter] = L[34] - MT[8][r][c]$

69: $C[level] \Rightarrow Coef010[rCounter][cCounter][sCounter] = L[31] + MT[5][r][c]$

70: $C[level] \Rightarrow Coef011[rCounter][cCounter][sCounter] = L[31] - MT[5][r][c]$

71: $C[level] \Rightarrow Coef012[rCounter][cCounter][sCounter] = L[35] - MT[9][r][c]$

72: $C[level] \Rightarrow Coef100[rCounter][cCounter][sCounter] = L[32] + MT[6][r][c]$

73: $C[level] \Rightarrow Coef101[rCounter][cCounter][sCounter] = L[32] - MT[6][r][c]$

74: $C[level] \Rightarrow Coef102[rCounter][cCounter][sCounter] = L[36] - MT[10][r][c]$

75: $C[level] \Rightarrow Coef020[rCounter][cCounter][sCounter] = L[39] + MT[13][r][c]$

76: $C[level] \Rightarrow Coef021[rCounter][cCounter][sCounter] = L[39] - MT[13][r][c]$

77: $C[level] \Rightarrow Coef022[rCounter][cCounter][sCounter] = L[43] - MT[17][r][c]$

78: $C[level] \Rightarrow Coef110[rCounter][cCounter][sCounter] = L[33] + MT[7][r][c]$

79: $C[level] \Rightarrow Coef111[rCounter][cCounter][sCounter] = L[33] - MT[7][r][c]$

80: $C[level] \Rightarrow Coef112[rCounter][cCounter][sCounter] = L[37] - MT[11][r][c]$

81: $C[level] \Rightarrow Coef200[rCounter][cCounter][sCounter] = L[48] + MT[22][r][c]$

82: $C[level] \Rightarrow Coef201[rCounter][cCounter][sCounter] = L[48] - MT[22][r][c]$

83: $C[level] \Rightarrow Coef202[rCounter][cCounter][sCounter] = L[52] - MT[26][r][c]$

84: $C[level] \Rightarrow Coef120[rCounter][cCounter][sCounter] = L[41] + MT[15][r][c]$

85: $C[level] \Rightarrow Coef121[rCounter][cCounter][sCounter] = L[59] + MT[33][r][c]$

86: $C[level] \Rightarrow Coef122[rCounter][cCounter][sCounter] = L[59] - MT[33][r][c]$

87: $C[level] \Rightarrow Coef210[rCounter][cCounter][sCounter] = L[49] + MT[23][r][c]$

88: $C[level] \Rightarrow Coef211[rCounter][cCounter][sCounter] = L[60] + MT[34][r][c]$

Algorithm 5 DHT parte 4

89: $C[level] \Rightarrow Coef212[rCounter][cCounter][sCounter] = L[60] - MT[34][r][c]$
90: $C[level] \Rightarrow Coef220[rCounter][cCounter][sCounter] = L[57] + MT[31][r][c]$
91: $C[level] \Rightarrow Coef221[rCounter][cCounter][sCounter] = L[61] + MT[35][r][c]$
92: $C[level] \Rightarrow Coef222[rCounter][cCounter][sCounter] = L[61] - MT[35][r][c]$
93: $cCounter = cCounter + 1$
94: **end if**
95: **for** $aux = 30$ to $aux < 62$ **do**
96: MT[aux-26][r][c] = L[aux]
97: **end for**
98: **end for**
99: **if** r es impar **then**
100: $rCounter = rCounter + 1$
101: **end if**
102: **end for**
103: **if** s es impar **then**
104: $sCounter = sCounter + 1$
105: **end if**
106: **end for**
107: Obten Mh de C con la función proyección 3D a 1D
108: Guarda los volúmenes de coeficientes en el primer nivel de la estructura C
109: $rows = nRows2 + 1$
110: $columns = nColumns2 + 1$
111: $slides = nSlides2 + 1$
112: El volumen fuente ahora es el volumen coeficiente 000 de tamaño [rows][columns][slides]
113: **end for**

Algorithm 6 Esquema de Fusión

Require: C_{nivel} {Combinación de coeficientes de baja frecuencia}
$nivel = 0$
$C_0(000) = CVolA_{nivel}(000) * Porcentaje + CVolB_{nivel}(000) * (1 - Porcentaje)$
{Combinación de coeficientes de alta frecuencua}
for $l = 1$ to $l = 0$ **do**
$mapaBin = FusionCAF(nivel, renglon, columna, corte, metodo)$
for $coef = 0$ to $coef = 26$ **do**
for $dim = 0$ to $dimTotal - 1$ **do**
$C_l(coef)_{dim} = (mapaBin_{dim} \cdot CVolA_l(coef)_{dim}) + ((1 - mapa) \cdot CVolB_l(coef)_{dim})$
end for
end for
end for
$VolumenFusion = IDHT(C)$

Algorithm 7 FusionCAF

Require: $MhA, MhB, metodo$
  if metodo = seleccion del máximo **then**
    $Mapa = MhA > MhB$
  end if
  if metodo = máximo absoluto **then**
    $Mapa = |MhA| > |MhB|$
  end if
  if metodo = max verificacion consistencia **then**
    Agrega ceros en las tres dimensiones de MhA y MhB
    for $dim = 0$ to $dimensionTotal$ **do**
      Almacena los valores de la vecindad del voxel en turno.Genera una copia de MhA,MhB.Ordena los 26 valores del voxel y guarda el mayor en la copia.Genera una estructura 'A,B' del tamaño de MhA o MhB
      $mapa_{dim} = ((\sum_{i=0}^{26} A_i * 1) > (\sum_{i=0}^{26} B_i * 1)) > 13$
    end for
  end if
  if metodo = dependencia lineal **then**
    Agrega ceros en las tres dimensiones de MhA y MhB
    for $dim = 0$ to $dimensionTotal$ **do**
      Almacena los valores de la vecindad del voxel en turno en 'A,B'
      $mapa_{dim} = (\sum_{i=0}^{26} A_i^2 - A_i) \geq (\sum_{i=0}^{26} B_i^2 - B_i)$
    end for
  end if

Capítulo 5

Medical Volume Fusion

El programa desarrollado implementa una solución al problema planteado por el Dr. Boris Escalante Ramírez y lo planteado en [26] de fusión de volúmenes reconstruidos a partir de imágenes médicas. Fue codificado usando Visual Studio 2010 en C++. Este programa fue desarrollado en una computadora con las siguientes características: GPU: Nvidia Quadro FX5800 CPU:Intel Xeon @3.33GHz(2 procesadores) RAM: 12GB y Sistema Operativo: Windows 7(64 bits). El ambiente de usuario final es el equipo mencionado anteriormente y como parte de los entregables del Proyecto IXTLI Fusión de datos 3D

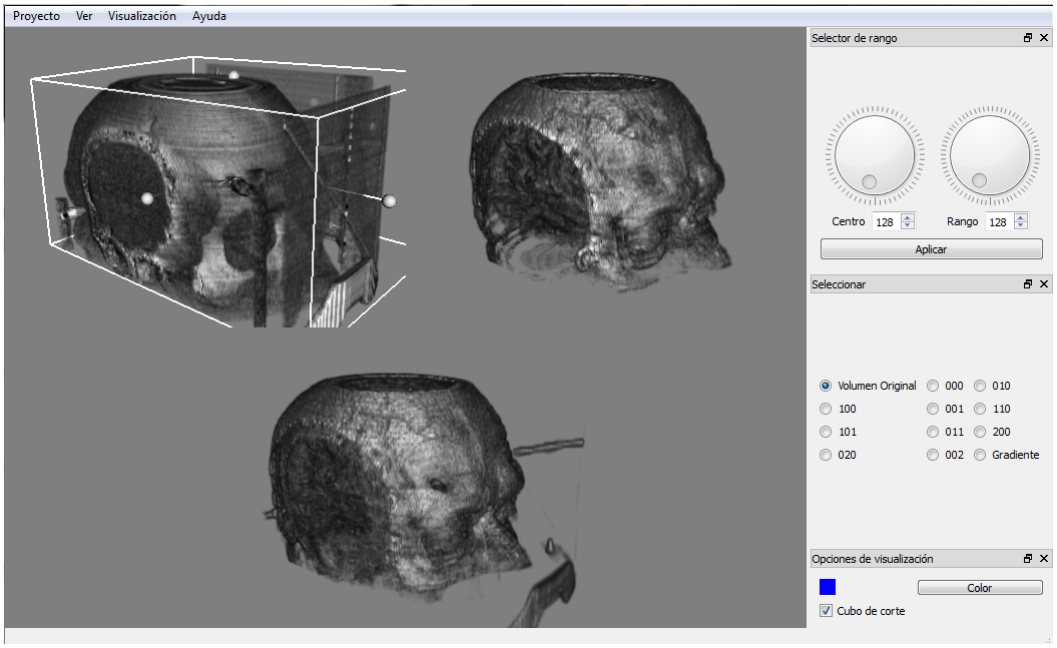


FIGURA 5.1: Programa Medical Volume Fusion en ejecución

en imagenología médica con clave IX100610[1] este programa tiene a futuro mostrarse en el observatorio IXTLI.

Se tomó como base el primer programa para visualización de volúmenes y realización de la transformada Hermite discreta DHT realizado por el Departamento de Visualización de la Dirección General de Cómputo y de Tecnologías de información y comunicación. Este programa permitía cargar un volumen. La interfaz permitía reconstruir un volumen a partir de una secuencia de imágenes médicas, y también a partir de formatos NRRD y NIFTI para guardar volúmenes. Ya reconstruido el volumen, el programa permite filtrar niveles de gris y asignar mapas de colores preestablecidos. Además permite realizar cortes al volumen para visualizar zonas de intéres particulares y visualizar distintos coeficientes resultados de la transformada Hermite.

5.1. Método de desarrollo

Para el desarrollo de la aplicación se siguió la siguiente metodología:

- Búsqueda de librerías open-source (o con licencia GNU-GPL) basadas en C++ para visualización científica.
- Aprender el manejo y programación de la libreria seleccionada
- Efectuar la carga de tres volúmenes en la misma ventana y realizar manipulaciones en los volúmenes
- Interactuar con la librería Qt[12].Generar una interfaz y cargar los tres volúmenes.
- Implementar la transformada discreta multiresolución orientada Hermite y efectuar pruebas.
- Implementar la transformada inversa discreta multiresolución orientada Hermite y efectuar pruebas.
- Programar el esquema de fusión propuesto: selección del máximo y máximo absoluto, selección del máximo con verificación de consistencia y dependencia lineal.
- Prueba de fusión con los datos CT - RMregCT y RM - PETregRM.

Todo el proceso anterior se procedió bajo una enfoque de desarrollo de software iteractivo e incremental(para ir añadiendo las funcionalidades necesarias). Se eligió el lenguaje de programación C++ por la rapidez y versatilidad en la ejecución de los programas desarrollados en este lenguaje. Se eligió un enfoque de clases para la implementación de los algoritmos y del proceso de visualización. Esto para tener la posibilidad a futuro de reutilización de código en futuras aplicaciones.

5.1.1. Virtualization toolKit VTK

La librería de visualización científica VTK(Visualization toolkit)[2]es un *open-source*, portable, multiplataforma, un sistema de software orientado a objetos para computación gráfica e interacción 3D, visualización y procesamiento de imágenes. Implementado en C++, VTK soporta los lenguajes de programación Tcl, Python y Java. VTK es un sistema de propósito general que lleva varios años en constante desarrollo (desde 1993). Tiene la compatibilidad con otras librería para manejo de imágenes médicas ITK que es de intéres para futuros proyectos a partir del trabajo aquí mostrado. Una prime-

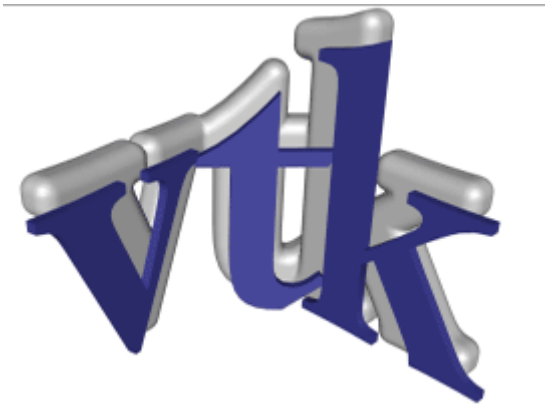


Figura 5.2: VTK

ra experiencia con el despliegue de imágenes consiste en desplegar una sencilla imagen plana(como en Matlab). Mientras que hay algunos problemas, como la orientación relativa(renglón/columna vs columna/renglón) y la posición con respecto al origen, este es intuitiva y bastante sencillo procedimiento

5.1.1.1. El pipeline de visualización

El procesamiento de datos encausado(pipeline) de VTK transforma los datos en formas que pueden ser desplegados por un subsistema de gráficos definidos previamente o usando primitivas gráficas que se pueden encausar. Por ejemplo podemos desear leer un conjunto de puntos desorganizados, crear una malla poligonal por medio de una triangulación, cuando se despliega la malla usando una renderización poligonal(superficie).

El pipeline(encausamiento) o red de visualización se construye con la conexión de procesos y se compone de los siguientes elementos:

- *Data processing segment*(Segmento de procesamiento de datos) Consiste en datos fuente y filtros de datos. Estos objetos dan al usuario acceso directo a los datos.
- *Rendering segment*(Segmento de renderizado)El segmento de rendering es esencialmente un wrapper de OpenGL. En este segmento del pipeline, el usuario no tiene acceso directo a los datos, esto es una forma de que sea más eficiente los gráficos en el hardware. El segmento de rendering consta de mappers, actors, renderers y render windows.

En la Figura 5.3 se muestra el proceso para desplegar en una pantalla usando la librería VTK De acuerdo a la Figura 5.3 se comienza con obtener nuestros datos fuente, posteriormente se pueden realizar transformaciones con los *filtros* que es un objeto que toma las entradas y opera sobre de ellas para generar una nueva salida. Dentro del proceso de visualización un objeto que efectua la conexión entre el segmento de procesamiento de datos y el segmento de renderización son los mappers. Ya que estos mappers transforman nuestros datos fuente en algo que pueda ser renderizado.

Los Actors (actores)son objetos OpenGL, se encargar de ajustar y controlar las propiedades de apariencia de la manifestación física de los objetos que son renderizados en la escena.Por lo general controlan la transparencia y el mapeo de color. El render window es la ventana actual en el que opera el despligue en pantalla y finalmente el renderer es donde los datos actuales se dibujan en pantalla. Dentro de un render window podemos contener varios renderers.

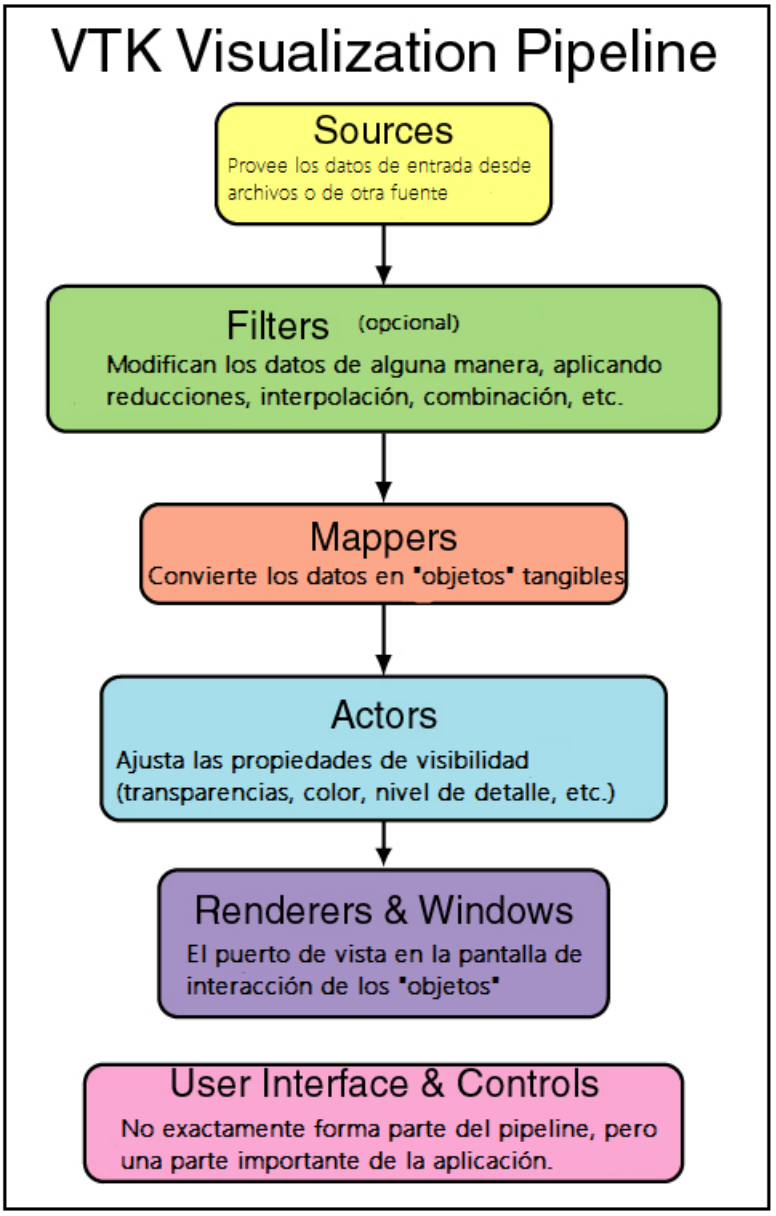


FIGURA 5.3: Pipeline de Visualización de VTK[27]

5.1.2. Qt framework

Qt es un framework(marco de trabajo) que desarrolló la compañia Trolltech en 1994 en Oslo,Noruega. En 2008 Nokia adquirió los derechos de este framework y continuo con su desarrollo. En 2011, Digia una compañia de software adquiere la licencia comercial de Qt de Nokia. Posteriormente en agosto de 2012 Digia adquiere Qt de Nokia. Se eligió este framework para diseñar la interfaz por que es multiplataforma(se puede migrar a otros sistemas operativos), proporciona una API(Application programming interface)con una orientación clara; conserva el *look and feel* del sistema donde se esta ejecutando, es

FIGURA 5.4: Qt code less, create more, deploy everywhere

decir, se adapta al diseño de interfaz del sistema operativo y está implementado en C++. Además se eligió el lenguaje C++ por que la mayoría de motores gráficos están escritos en ese lenguaje y por cuestión de rendimiento.

A diferencia de WinApi (la interfaz de programación de Windows), GLUI,wxWidgets, GTK+ entre otras; Qt tiene un mecanismo de comunicación entre las funciones y los elementos de la interfaz conocido como *Signals and slots*. Cuando se produce un evento en un objeto, éste puede emitir una señal. Esta señal se puede conectar con un slot(método de otro objeto)que corresponde a esa señal. Las clases que emitan señales o contengan slots deben indicar que son clases hijas de OBJECT(Objetos Qt) usando la directiva Q_OBJECT. Este mecanismo es similar al manejador de elementos que se utiliza en Java.

5.1.3. Boost libraries

Boost es in conjunto heterogéneo de utilidades que tienen su origen en una iniciativa de algunos miembros del cómite de estandarización de C++, aunque posteriormente han recibido aportaciones de múltiples autores. Comprenden un extenso catálogo que incluye los siguientes tópicos: string and text processing; containers; iterators; algorithms; function objects and higher-order programming; generic programming; template metaprogramming; preprocessor metaprogramming; concurrent programming; math and numerics; correctness and testing; data structures; input/output; inter-language support; memory; parsing; miscellaneous. El proceso de selección para que una librería sea admitida, es público y bastante estricto. En realidad esta colección es una especie de

antesala oficiosa del Estándar, ya que algunas de sus componentes son analizadas y evaluadas por los miembros del comité como candidatas a ser incluidas en futuras revisiones del estándar C++.

La librería de interés es la de matrices multidimensionales. Estás boost.multiarray proveen un n-dimensional arreglo con métodos y de gran utilidad para no reservar la memoria a utilizar(a través de la función *malloc* de las librerías estándar de C/C++).

5.1.4. Tratamiento de las imágenes

Esta aplicación carga imágenes en formato DICOM o RAW, pero no está implementado el modulo para generar un volumen VTK con formato bmp ya que ese el tipo de extensión que se tenían de los ejemplos a fusionar, por lo que requirió un tratamiento a las imágenes para que visualizaran en la aplicación. Mediante el uso de un convertidor de imágenes llamado dicom2[7] se transformaron en archivos RAW que VTK puede reconocer. Este software fue desarrollado y mantenido por Sébastien Barré.

5.1.5. Arquitectura del sistema

En general, existe existe una clase llamada Mainwindow que se encarga de generar y configurar todos los elementos de la interfaz. Existe un widget llamado QVTKWidget definido en una plantilla XML que crea los elementos gráficos de la ventana. De dicho widget, se obtiene la ventana usada en el pipeline de VTK (vtkRenderWindow),a la que se añaden los elementos para el proceso de dibujo. (vtkRenderer, vtkInteractor, vtkViewport,vtkCamera). Encontramos que vtkImageData representa el volumen reconstruido en 3D, (los voxels), entonces en las variables correspondientes a cada volumen es donde realizamos las modificaciones para mostrar los resultados. Se construyen matrices multidimensionales usando la librería Boost, copiando el contenido de estas matrices y después son procesados los datos que representan las tonalidades de grises que forman a la imagen por los algoritmos de la transformada Hermite y efectuando el esquema de fusión propuesto se obtiene una matriz que representa al volumen sintético o fusionado para finalizar con la transformada inversa de esa matriz y transformarla con el pipeline de VTK en un volumen que pueda mostrarse en pantalla.

La interfaz de usuario se compone de las siguientes partes:

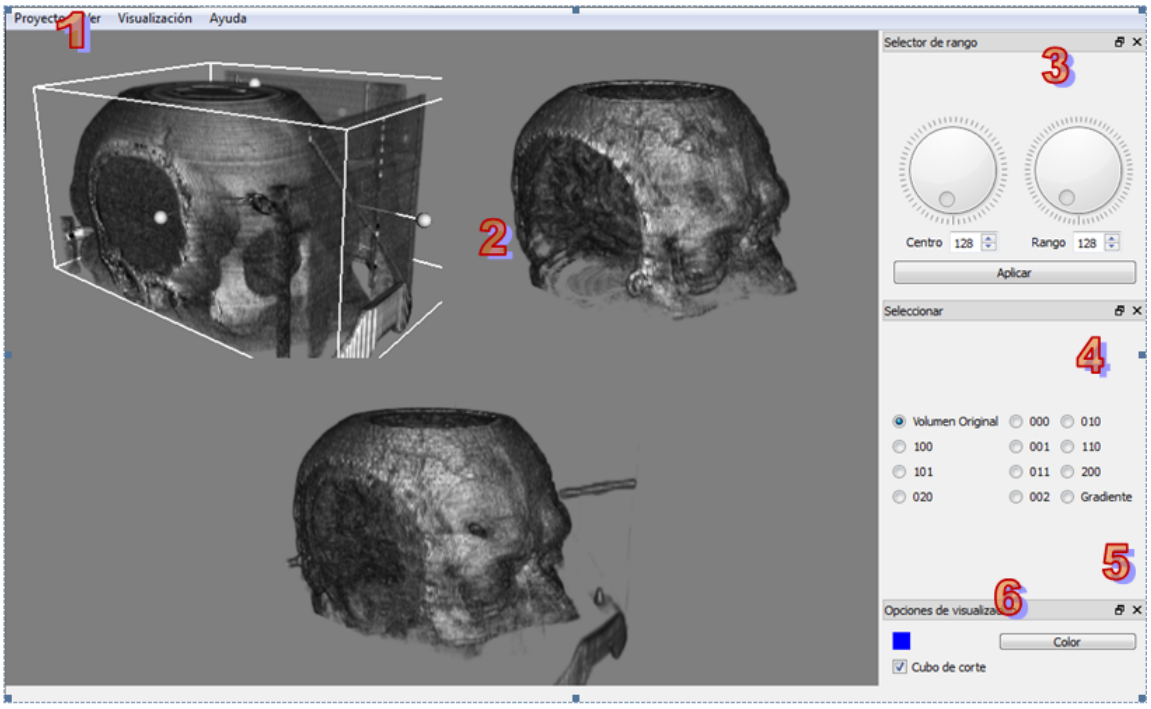


FIGURA 5.5: Componentes de la interfaz

1. Barra de menú.
2. Ventana de visualización de los volúmenes.
3. Selector de rango.
4. Transformada Hermite y la selección de coeficientes para visualización.
5. Botón que muestra una ventana de diálogo para elegir otro color de fondo a la ventana de visualización.
6. Un checkbox que habilita o deshabilita el cubo de corte que se presentará en el volumen mostrado en el borde superior izquierdo.

Barra de menú Contiene los menús proyecto, ver, visualización y ayuda, los cuales a su vez contienen otras opciones. En el menú proyecto se tiene las opciones de cargar datos, cargar proyecto, fusionar. Por último este menú tiene la opción de Salir del sistema. La acción de cargar datos muestra una serie de ventanas las cual no

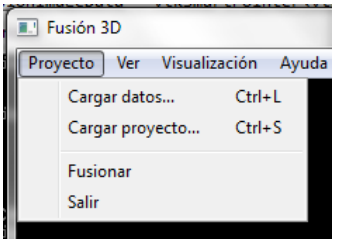


FIGURA 5.6: Opciones del menú proyecto

irán guiando para llenar valores que se ne-
cesitan para cargar nuestros datos. De acuerdo al formato (DICOM, RAW) serán distintas ventanas las que irán saliendo.

Para subir datos en formato DICOM, sólo se requiere el directorio en donde están las imágenes que formaran el volumen A y volumen B (1 y 2) y ponerle un nombre a dicho proyecto. Para subir datos en formato RAW disponemos de las siguientes ventanas

FIGURA 5.7: Wizard para cargar datos en formato DICOM

(Figura 5.7): Para subir datos en RAW (Figura 5.8) hay que poner la ubicación del

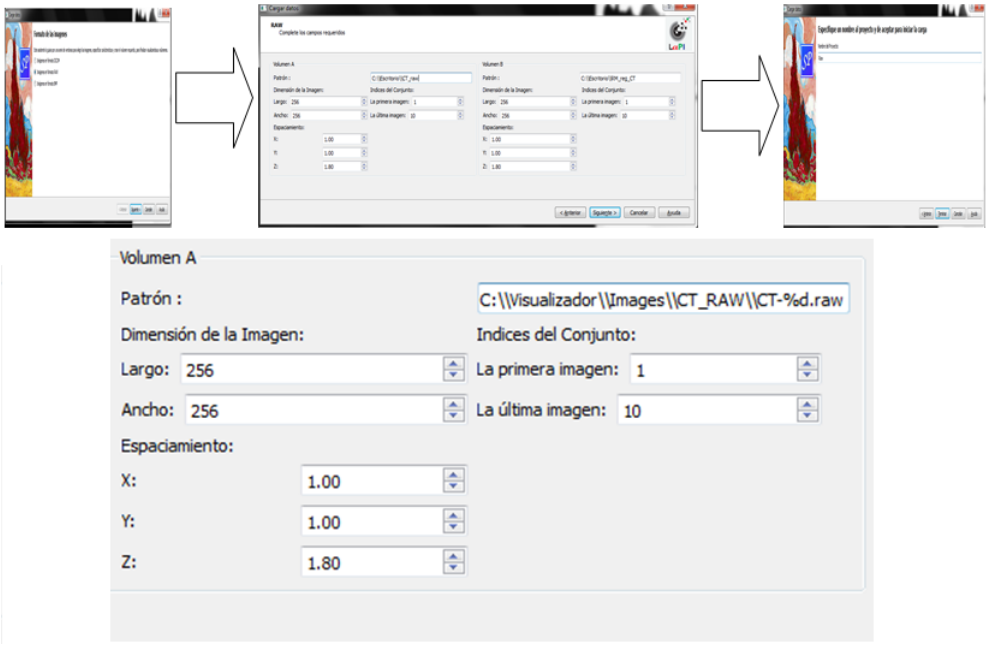


FIGURA 5.8: Wizard para cargar datos en formato RAW

conjunto de imágenes en la caja indicada por 'Patrón' de manera que cada '\\' sea doble.

Por ejemplo si el directorio es '$C : \backslash Images \backslash CT$' se pondrán '$C : \backslash\backslash Images \backslash\backslash CT$', si las imágenes estan en donde se encuentra la aplicación, sólo se indica '$Images \backslash\backslash CT$' .

Además hay que colocar el nombre del lote de imágenes, ejemplo, si mis imágenes son ct1.raw, ct2.raw, ct3.raw,..., ct81.raw esto se debe de colocar 'ct %d.raw' donde se sustituye el número por la palabra %d para finalmente colocar en la caja indicada por Patrón '$Images\\CT\\ct\,\%d.raw$'.

La sección que indica 'Dimensión de la Imagen' debe de colocar en valores positivos enteros el tamaño en pixeles de la imagen. En 'Espaciamiento' hace más larga la dimensión por separar los voxeles interiores determinado por una constante. Y finalmente 'Indices del conjunto' es colocar un número entero por los que empieza el lote de imágenes, si tenemos ct1, ct1, ct3,. . . ,ct81 la primera imagen es 1 y la última es 81 (Figura 5.8).

Después tenemos el menú 'ver' (figura 5.9) donde se tienen las opciones selector de rango, seleccionar y opciones de visualización. Este menú tiene un checkbox que indica si la ventana flotante esta mostrándose en la ventana principal o no. Este checkbox se puede seleccionar o quitar.

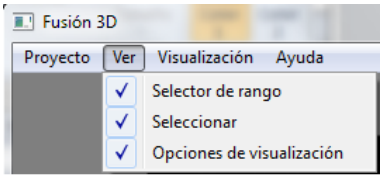


FIGURA 5.9: Opciones del menú Ver

El menú visualización tiene una acción llamada definir segmentos y color. La ventana de segmento hace una segmentación por usuario determinando que rangos de nuestra escala de gris que forman la imagen vamos a poner un color. Se pueden crear desde una a 7 clases distintas cada una con rangos definidos y un color por cada límite del rango determinado por el botón selecciona color.

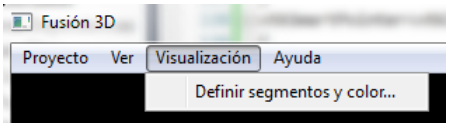


FIGURA 5.10: Opciones del menú visualización

El menú Ayuda contiene las opciones 'Acerca de' y 'Sobre Qt'(Figura 5.12). Estas dos son ventanas de diálogo que dan una breve descripción de lo que hace la aplicación y con que fue diseñada la interfaz gráfica, es decir, los créditos del Qt framework.

Ventana de visualización de los volúmenes Es la sección de nuestra ventana principal de la aplicación en dónde se mostrará los datos de las imágenes médicas como volumen. En esta área de visualización se visualizan las manipulaciones a los volúmenes

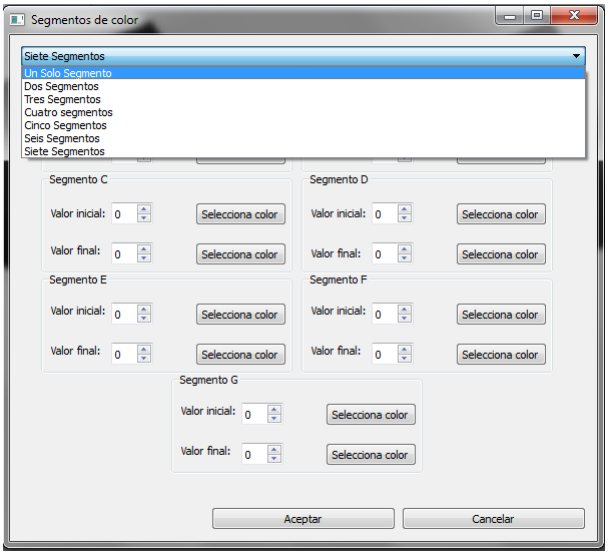
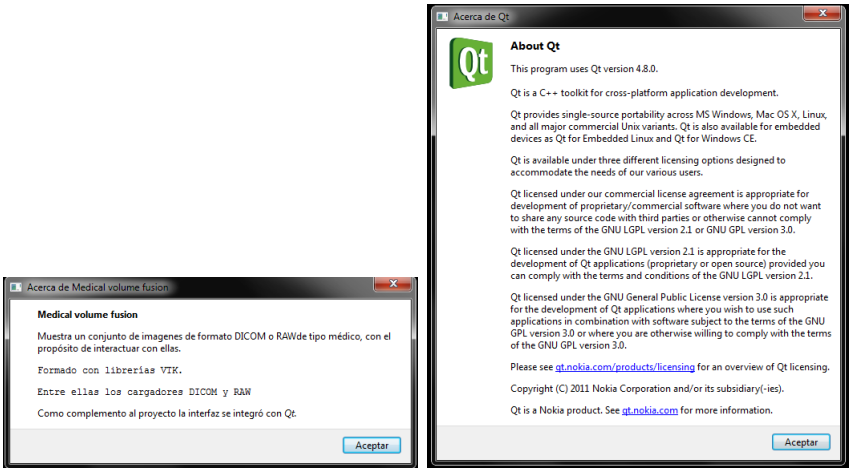


FIGURA 5.11: Ventana de segmentos de color



(a) (b)

FIGURA 5.12: Ventanas que forman parte del menú 'Ayuda'.(a)Sobre la aplicación. (b)Créditos de qt respecto a la interfaz gráfica.

.

así como las transformaciones: Hermite Discreta y Fusión de volúmenes. Para la transformada Hermite se calcula el volumen que corresponde al coeficiente seleccionado por el radio button. Sólo para los volúmenes A y B (1 y 2) La fusión de volúmenes combina la información de dos o más fuentes de manera que el resultado conserve las características más importantes de los datos originales de forma integral. En esta aplicación solo se podrán fusionar dos volúmenes del mismo tamaño.

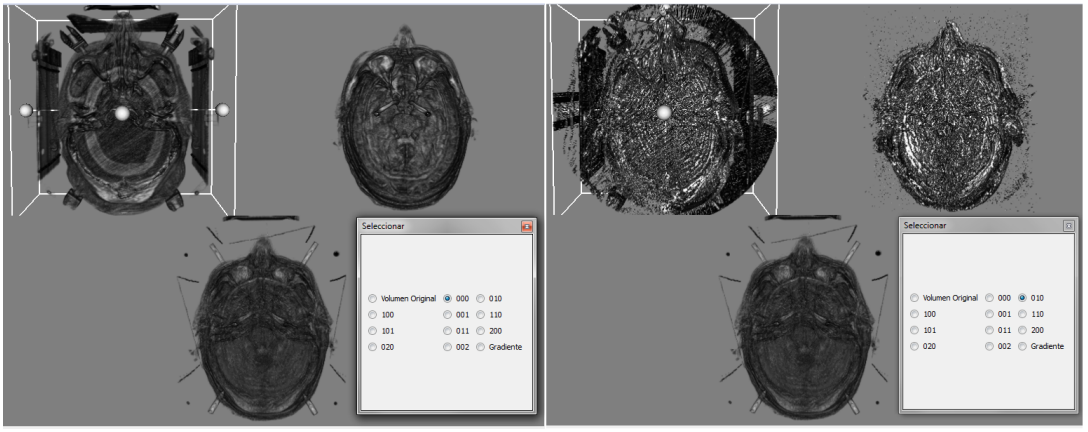


FIGURA 5.13: Vista de la transformada Hermite de un volumen de Tomografía vista desde el zenit, del lado izquierdo se seleccionó el coeficiente 000 mientras que para el lado derecho se seleccionó el coeficiente 010

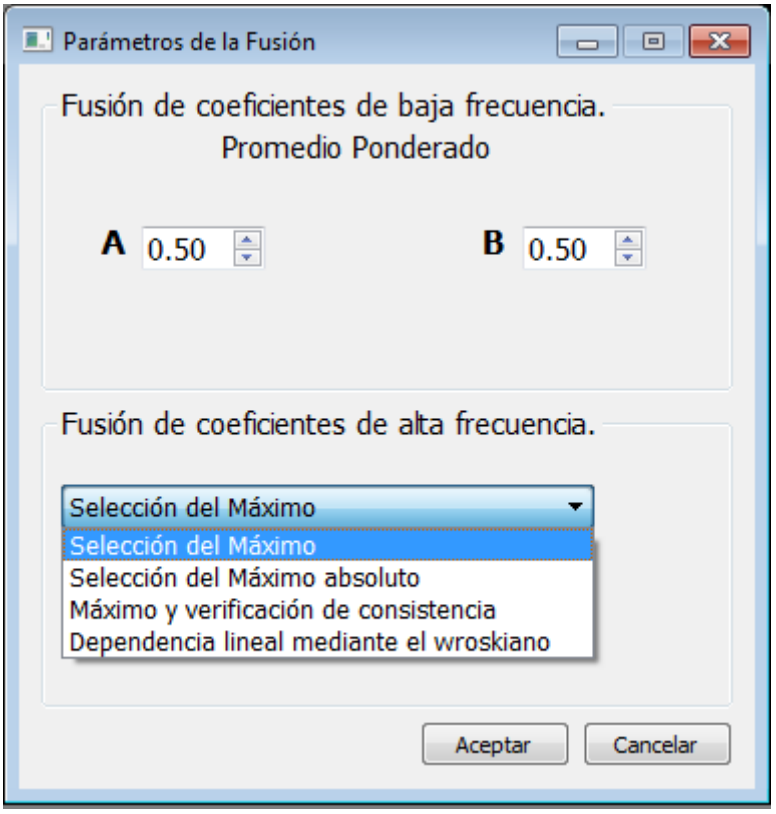


FIGURA 5.14: Ventana que muestra las opciones de fusión de los dos volúmenes de acuerdo al esquema propuesto en el Capítulo 4.

Selector de rango Esta opción consiste en agrandar una región del histograma correspondiente al volumen que se está observando con el propósito de que puedan apreciarse más detalles existentes en el volumen. El selector esta formado por dos diales, es decir, dos círculos graduados; dos spinbox o una caja donde se puede elegir un número de determinado rango de valores; y un botón de 'aplicar'.

Como se observa en la figura anterior, la selección del centro y ancho del histograma se indica de forma gráfica, la idea es que pueda modificarse con los 'spinbox' y con los diales o perillas graduadas; donde puede mostrarse un volumen totalmente transparente (no visible) y un volumen totalmente sólido (un recuadro muy oscuro).

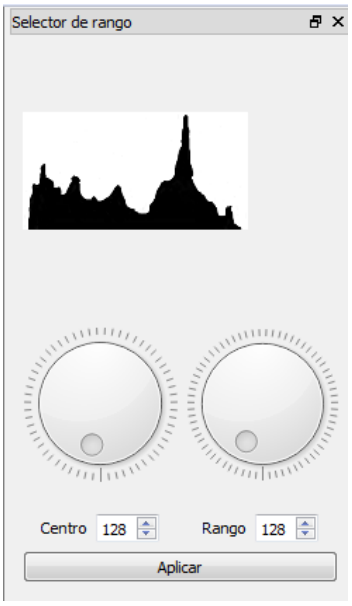


FIGURA 5.15: Selector de rango

Botón del color de fondo En esta sección se configura el color de fondo de la escena, mostrando que color es el actual y al presionar el botón 'color' se mostrará un cuadro de diálogo para seleccionar un color de un paleta determinada del sistema operativo.

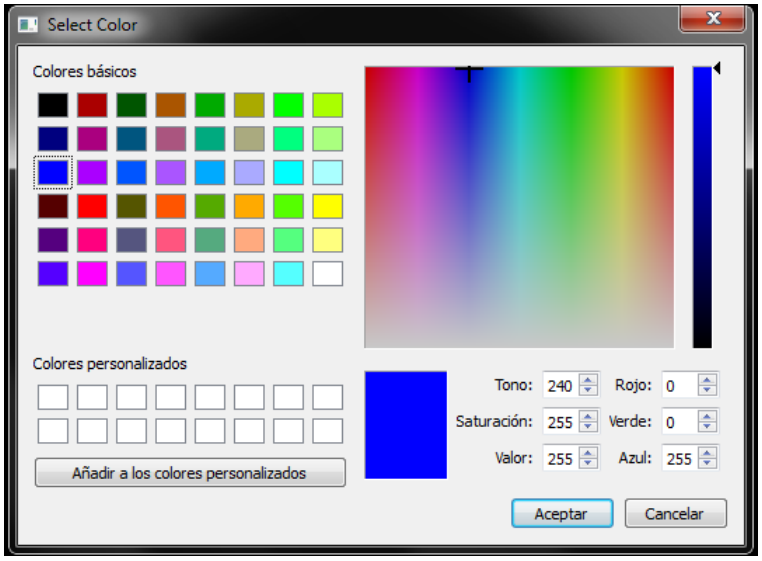


FIGURA 5.16: Ventana de selección de color

Checkbox que habilita el cubo de corte El cubo de corte es un widget de VTK para manejar un plano de corte en cualquiera de los tres ejes (sagital, axial, zenit) también para manejar un región determinada del volumen. Con el ratón se puede girar y mover el plano. Si hacemos un recorte de región, el cubewidget representa la región que nos interesa dejando visible sólo esa parte del volumen.

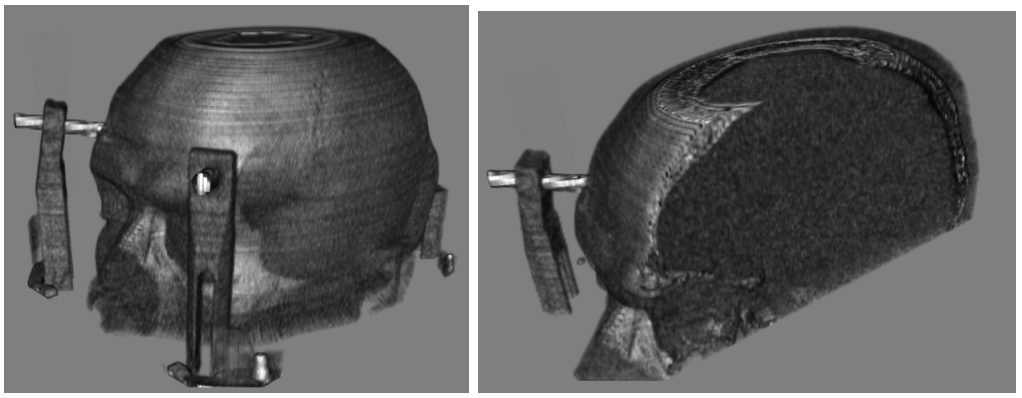
Capítulo 6

Resultados

El principal objetivo de implementar los algoritmos de la transformada Hermite, la transformada inversa y el esquema de fusión se completaron satisfactoriamente, sin embargo el costo computacional fue alto para la visualización en porción de n_2 y de la transformada Hermite en proporción $log_2 n$. Al momento de efectuar una carga de datos, la aplicación debe efectúar la transformada Hermite durante la espera de visión en pantalla lo que lleva al usuario a esperar un minuto máximo para visualizar los volúmenes formados. De igual manera sucede al efectuar la fusión de volúmenes, el usuario tiene que esperar máximo un minuto para observar los resultados del proceso.

Dado que no se cuenta con una base de datos de imágenes médicas registradas, se partió de un conjunto de dos pares de imágenes tridimensionales previamente registradas[26], las cuales se describen a continuación, además de mostrar una vista de cada volumen.

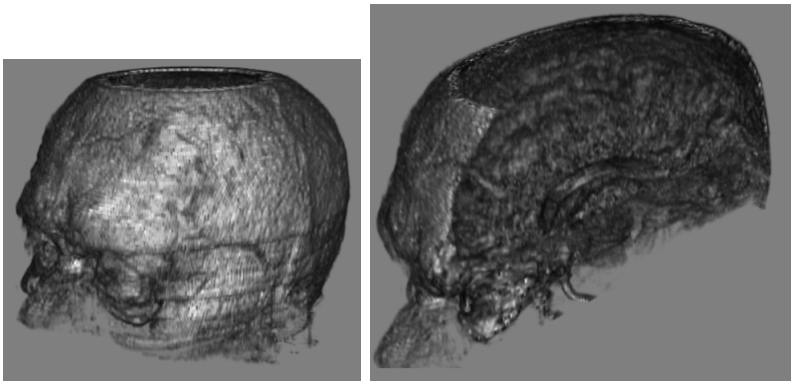
- **CT/MR** Esta par de volúmenes consiste en secuencias de imágenes de tomografía computarizada y resonancia magnética en formato RAW tratado con el software dicom2. Cada secuencia consta de 89 imágenes de 320 por 320 pixeles a 72dpi a 8 bits por pixel. Las imágenes fueron proporcionadas por el Instituto de Óptica del Consejo Superior de Investigaciones Científicas en España[11]Figuras6.1 y 6.2
- **MRI/PET** Esta par de volúmenes está conformado por secuencias de imágenes de resonancia magnética y tomografía por emisión de positrones en formato RAW tratado con el software dicom2. Cada secuencia consta de 100 imágenes de 256 por 256 pixeles a 72dpi de 8 bits por pixel. Al igual que el caso anterior, este conjunto



(a) (b)

FIGURA 6.1: Volumen que se formó a partir del conjunto CT

.



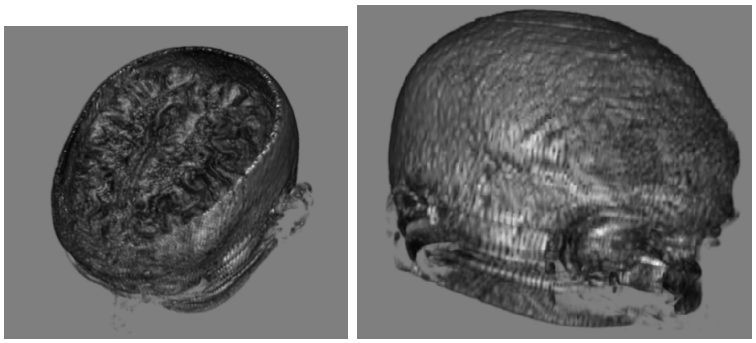
(a) (b)

FIGURA 6.2: Volumen que se formó a partir del conjunto MR

.

de imágenes fue proporcionado por el Instituto de Óptica del Consejo Superior de Investigaciones Científicas en España[11].Figuras6.3 y 6.4

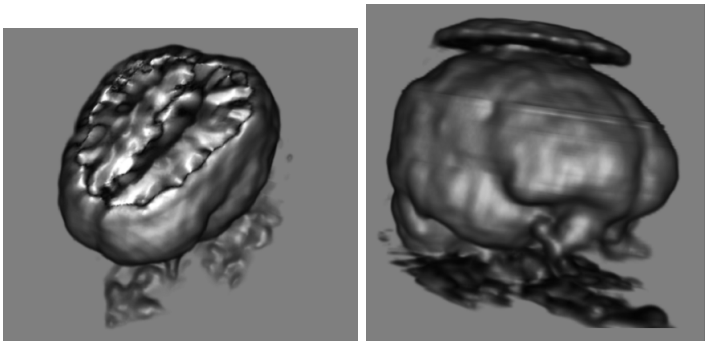
Una de las manipulaciones que se puede efectuar a los volúmenes es la de recortar mediante un cubo de corte. Este widget (un actor de VTK) permite explorar por el interior del volumen formado. En la Figura vemos como se muestra el interior del volumen RMI (un poco en el detalle de los glóbulos oculares) Cabe resaltar que la transformada Hermite (para el proceso de fusión no el de visualización) utilizan sólo dos niveles de descomposición. Esto es por qué se realizaron pruebas en [26] con los volúmenes y en general entre los niveles 2 y 3 no existía una diferencia visual significativa en el resultado, no así en el tiempo de ejecución ya que aumentar el número de niveles de resolución implica un mayor costo computacional en proporción de $log_2 n$



(a) (b)

FIGURA 6.3: Volumen que se formó a partir del conjunto MRI

.



(a) (b)

FIGURA 6.4: Volumen que se formó a partir del conjunto PET

.

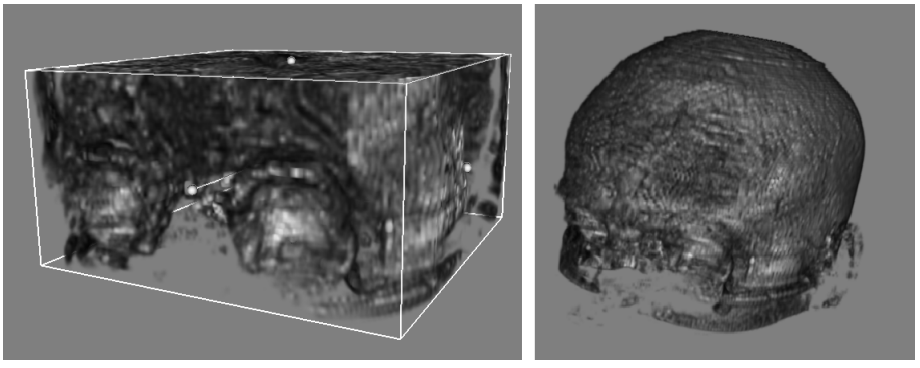


FIGURA 6.5: El cubo de corte muestra una sección de lo que es el volumen de la cabeza.

La visualización estereoscopia en el Observatorio IXTLI tuvo más vicisitudes al momento de despliegue de datos, esto por que al realizar el renderizado de los tres volúmenes en pantalla también efectuaba dos renders mas a cada uno de los volúmenes para generar la parte trasera del despligue y la parte delantera y formar así la visualización estereoscópica activa con el sistema de crystal eyes de Real3D.

Capítulo 7

Conclusión

En este trabajo se ha presentado lo que comienza a ser un motor de visualización para una aplicación de visualización de imágenes médicas más robusta. Las visualizaciones con los volumenes CT/MR tienen una mejores características que las de modalidad MRI/PET que se aprovecharon en las pruebas que se efectuaron en el Observatorio IXTLI. Con ello se plantean nuevos objetivos y que quedarán fuera lo delimitado en esta tesis pero que podría lograrse en nuevos trabajos entre estas nuevas metas se encuentran:

- Implementar algoritmos de segmentación: k-means, fuzzy C-means.
- Guardar los volúmenes fusionados en otro conjunto de imágenes. Esto por que la aplicación sólo me genera un objeto visible en mi pantalla del resultado de la fusión de mis volúmenes pero no se desarrollo la manera de almacenarlos en conjuntos de imágenes. Pero en posible de hacer. Con ello se plantea un nuevo problema que es obtener nuevas imágenes a partir de mi volumen cambiando el plano de corte
- Lograr que se carguen volúmenes en otros formatos.
- Mejorar la iluminación
- Interacción con el mouse para las transformaciones del volumen
- Mejorar el algoritmo implementándolo en paralelo.
- Migrarlo a otro sistema (no estar casado con Windows)

Apéndice A

Código clase principal

```
Mainwindow::Mainwindow(QWidget *parent, Qt::WFlags flags): QMainWindow(parent,
    flags)
{
  /* !\note El constructor configura las distintas funciones que crean los
    elementos de la pantalla.
    Posteriormente inicializa el renderer, renderWindow y la camera para hacer 3
    puertos de vista y
    una sola pantalla.
  */
  this->setupUi(this);
  this->setWindowTitle(tr("Medical Volume Fusion"));
  this->setCentralWidget(qVTK1);
  setActions();
  setupMenuBar();
  dockWinRangeSelector();
  dockCoefSelector();
  dockSetBackgroundColor();
  this->statusBar()->showMessage("..Cargando interfaz..", 12000);
  cubeActive = false;

  this->volume1Active = false;
  this->volume2Active = false;
  this->volumeFActive = false;

  OpacityMean = 128;
  OpacityRange = 128;
  ColorSegment c = ColorSegment(  0,//minGrayLevel
                  255,//maxGrayLevel
                  0.0,//rMin
```

mainwindow.cpp

```
                    0.0,//rMin
2                   0.0,//gMin
                    0.0,//bMin
4                   1.0,//rMax
                    1.0,//gMax
6                   1.0);//bMax
  Colors.push_back(c); //Se mete "c" en el vector de Colors
8 }

10 void Mainwindow::setActions()
  {
12   /*!
       \note Esta funcion inicializa las acciones de los menu's. Esta seccion es
       exclusiva de la interfaz
14   */
     acDataLoad = new QAction(tr("Cargar datos..."),this);
16   acDataLoad->setShortcut(tr("Ctrl+L"));
     connect(acDataLoad,SIGNAL(triggered()),this,SLOT(slotDataLoad()));
18   acLoadProject = new QAction(tr("Cargar proyecto..."),this);
     ...
20 }
  void Mainwindow::setupMenuBar()
22 {
     /*!
24     \note Esta funcion es exclusiva de la interfaz y se encarga de asignar las
       acciones en los menu's
     */
26   menu = menuBar()->addMenu(tr("&Proyecto"));
     menu->addAction(acDataLoad);
28   menu->addAction(acLoadProject);
     menu->addSeparator();
30   menu->addAction(acFuse);
     menu->addAction(tr("&Salir"),this,SLOT(close()));
32   ...
  }
34 void Mainwindow::slotDataLoad()
  {
36   /*!
       Genera una instancia de la clase ProjectWizard y muestra el wizard que cargar
        los volumenes.
38   */
     ProjectWizard *lw = new ProjectWizard(this);
40   lw->show();
     connect(lw,SIGNAL(fileReady(const QString)),this,SLOT(slotFileReady(const
       QString)));
42 }
```

```
44 void Mainwindow::slotFileReady(const QString &file)
   {
46   /*!
       Aqui se hace el parser para leer un txt que creo el Wizard para obtener los
       valores para la carga de las imagenes
48     Y una vez que identifico que tipo de archivo leyo, manda a llamar a la
       funcion LoadVolume con los respectivos parametros.
     */
50   QFile data(file);
     if (!data.open(QFile::ReadOnly | QFile::Text))
52   {
           QMessageBox::warning(this, tr("Recent Files"),
54                                tr("Cannot read file %1:\n%2.")
                 .arg(data.fileName())
56                               .arg(data.errorString()));
      return;
58    }
     QString line;
60   QTextStream out(&data);
     while(!out.atEnd())
62   {
       line = out.readLine();
64     if(line == "dicom")
       {
66       for(int i=1;i<3;i++)
         {
68         line = out.readLine();
           directory = line.toStdString();
70         /*!
             Aqui llamo a la funcion para cargar el volumen dicom i( 1 o 2)
72           los valores de directory e i van a cambiar en cada iteracion.
             \param directory es un StdString declarado globalmente.
74           \param i es el volumen a cargar el primero(1) o segundo (2).
           */
76         LoadVolume(directory,i);
         }
78     }
       else if(line == "raw")
80     {
         for(int j=1;j<3;j++)
82       {
           line =out.readLine();
84         filePattern = line.toStdString();
           line = out.readLine();
86         sizeX = line.toInt();
           line = out.readLine();
```

```
        sizeY = line.toInt();
        line = out.readLine();
        dataSpacingX = line.toFloat();
        line = out.readLine();
        dataSpacingY = line.toFloat();
        line = out.readLine();
        dataSpacingZ = line.toFloat();
        line =out.readLine();
        firstImageIndex = line.toInt();
        line = out.readLine();
        lastImageIndex = line.toInt();
        /*!
          Aqui se manda a llamar la funcion para raw y las variables son globales
     las cuales cambian en cada ciclo.
        */
        LoadVolume(filePattern,dataSpacingX,dataSpacingY,dataSpacingZ,sizeX - 1,
    sizeY - 1,firstImageIndex,lastImageIndex,j);
        this->statusBar()->showMessage(tr("..Cargando volumenes.."), 180);
      }
    }
    break;
  }
  data.close();
  /*! La instancia de vtk para los renderers

  */
  vtkCamera* camera;

  f = new Fusion(imageDatV1, imageDatV2, 1);
  this->statusBar()->showMessage(tr("..Realizando Transformada Hermite de los
    Volumenes.."), 18000);
  f->CreateDHT();
  //renderer v1
  rendererV1 = vtkSmartPointer<vtkRenderer>::New();
  propOriginal = vtkSmartPointer<vtkVolumeProperty>::New();
  qVTK1->GetRenderWindow()->AddRenderer(rendererV1);
  camera = rendererV1->GetActiveCamera();
  rendererV1->SetViewport(0,0.5,0.5,1);

  volume1 = vtkSmartPointer<vtkVolume>::New();
  mapperV1 = vtkSmartPointer<vtkVolumeRayCastMapper>::New();

  vtkVolumeRayCastCompositeFunction *compositeFunction =
    vtkVolumeRayCastCompositeFunction::New();
  compositeFunction->SetCompositeMethodToClassifyFirst();

  mapperV1->SetVolumeRayCastFunction(compositeFunction);
```

```
44  mapperV1->SetInput(f->GetImageDataV1(0));
    volume1->SetMapper(mapperV1);
46  mapperV1->SetSampleDistance(0.1);

48  rendererV1->AddVolume(volume1);
    rendererV1->SetBackground(0.5,0.5,0.5);
50  rendererV1->ResetCamera();
    qVTK1->GetRenderWindow()->Render();
52  volume1Active = true;
    propOriginal = volume1->GetProperty();
54  this->statusBar()->showMessage(tr("..Mostrando volumen 1.."), 30000);
    //renderer v2
56  rendererV2 = vtkSmartPointer<vtkRenderer>::New();
    qVTK1->GetRenderWindow()->AddRenderer(rendererV2);
58  rendererV2->SetActiveCamera(camera);
    rendererV2->SetViewport(0.5,0.5,1,1);
60
    volume2 = vtkSmartPointer<vtkVolume>::New();
62  mapperV2 = vtkSmartPointer<vtkVolumeRayCastMapper>::New();


64
    mapperV2->SetVolumeRayCastFunction(compositeFunction);
66  mapperV2->SetInput(f->GetImageDataV2(0));
    volume2->SetMapper(mapperV2);
68  mapperV2->SetSampleDistance(0.1);

70  rendererV2->AddVolume(volume2);
    rendererV2->SetBackground(0.5,0.5,0.5);
72  rendererV2->ResetCamera();
    qVTK1->GetRenderWindow()->Render();
74  volume2Active = true;
    this->statusBar()->showMessage(tr("..Mostrando volumen 2.."), 30);
76
    rendererF = vtkSmartPointer<vtkRenderer>::New();
78
      qVTK1->GetRenderWindow()->AddRenderer(rendererF);
80  rendererF->SetViewport(0.0,0.0,1,0.5);
    rendererF->SetBackground(0.5,0.5,0.5);
82
    box = vtkBoxWidget::New();
84  box->SetInteractor(qVTK1->GetInteractor());
    box->SetPlaceFactor(1.01);
86
    vtkSmartPointer<vtkImageData> input = f->GetImageDataV1(0);
88
    box->SetInput(input);
```

mainwindow.cpp

```
{
  /*!
    Metodo exclusivo de la interfaz. Crea una ventana de dialogo con el mensaje
    sobre la aplicacion.
  */
  QMessageBox::about(this,
    tr("Acerca de Medical volume fusion"),
    message);
}
void Mainwindow::setSliceCube(bool checked)
{
    /*!
      Aqui hay que habilitar el vtkBoxWidget y sino existe hay que crearlo (o
    sino se crea desde slotFileReady()
    */
  if(this->volume1Active && this->volume2Active)
  {
    if(checked || 0x49)
    {
      box->EnabledOn();
      cubeActive = true;
    }
    else
    {
      vtkSmartPointer<vtkTransform> identity = vtkSmartPointer<vtkTransform>::New
    ();
      identity->Identity();
      box->SetTransform(identity);
      vtkPlanes *planes = vtkPlanes::New();
      box->GetPlanes(planes);
      mapperV1->SetClippingPlanes(planes);
      mapperV2->SetClippingPlanes(planes);
      if(this->volumeFActive == true)
        mapperF->SetClippingPlanes(planes);
      planes->Delete();
      box->EnabledOff();
      cubeActive = false;
    }
  }
}
void Mainwindow::slotLoadProject()
{
  /*!
    Metodo exclusivo de la interfaz, abre una ventana y cargar el proyecto
    seleccionado.
  */
  QString fileName
```

```
            = QFileDialog::getOpenFileName(this, tr("Abrir proyecto"),".","*.txt");
45      if (fileName.isEmpty())
            return;
47    this->slotFileReady(fileName);
    // std::cout<<"SlotLoadProject"<<std::endl;
49 }
  void Mainwindow::dockWinRangeSelector()
51 {
    /*!
53    Metodo exclusivo de la interfaz
    */
55  QDockWidget *dock = new QDockWidget(tr("Selector de rango"),this);
    dock->setAllowedAreas(Qt::LeftDockWidgetArea | Qt::RightDockWidgetArea);
57  addDockWidget(Qt::RightDockWidgetArea, dock);
    RangeSelector *dwRangeS = new RangeSelector(dock);
59  dock->setWidget(dwRangeS);
    dock->setHidden(false);
61  dock->setEnabled(true);
    dock->adjustSize();
63  view->addAction(dock->toggleViewAction());
    connect(dwRangeS,SIGNAL(mean_Range(int,int)),this,SLOT(slotSetGrayInterval(int,
       int)));
65 }

67 void Mainwindow::dockCoefSelector()
  {
69  QDockWidget *dock = new QDockWidget(tr("Seleccionar"),this);
    dock->setAllowedAreas(Qt::LeftDockWidgetArea | Qt::TopDockWidgetArea);
71  addDockWidget(Qt::RightDockWidgetArea,dock);
    SelectorCoefDHT *dhtCoef = new SelectorCoefDHT(dock);
73  dock->setWidget(dhtCoef);
    dock->setHidden(false);
75  dock->setEnabled(true);
    dock->adjustSize();
77  view->addAction(dock->toggleViewAction());
    connect(dhtCoef,SIGNAL(sendOption(const QString)),this,SLOT(slotLoadCoefDHT(
      const QString)));
79 }

81 void Mainwindow::slotFusion(float percentA, float percentB, int highF)
  {
83  /*!
      Aqui se hacen todas las llamadas, instanciaciones, que involucren a la Fusion
85  */
    if(!volumeFActive)
87  {
      vtkCamera* camera = rendererV1->GetActiveCamera();
```

```
89      vtkSmartPointer<vtkVolumeProperty> prop = vtkSmartPointer<vtkVolumeProperty
        >::New();
        prop->SetAmbient(0.5);
91      prop->SetDiffuse(0.5);
        prop->SetSpecular(0.5);
93      prop->SetSpecularPower(20.0);
        prop->SetIndependentComponents(false);
95      prop->SetInterpolationTypeToLinear();
        prop->ShadeOn();
97      rendererF = vtkSmartPointer<vtkRenderer>::New();
        qVTK1->GetRenderWindow()->AddRenderer(rendererF);
99      rendererF->SetActiveCamera(camera);
        rendererF->SetViewport(0.0,0.0,1,0.5);
101
        volumeF = vtkSmartPointer<vtkVolume>::New();
103     mapperF = vtkSmartPointer<vtkVolumeRayCastMapper>::New();

105     vtkVolumeRayCastCompositeFunction *compositeFunction =
        vtkVolumeRayCastCompositeFunction::New();
        compositeFunction->SetCompositeMethodToClassifyFirst();
107     mapperF->SetVolumeRayCastFunction(compositeFunction);
        f->CreateVolumeFusion();
109     mapperF->SetInput(f->Fusionate(percentA, percentB, highF));
        this->statusBar()->showMessage(tr("..Generando el volumen fusionado.."),
        6000);
111     volumeF->SetMapper(mapperF);
        mapperF->SetSampleDistance(0.2);
113     rendererF->AddVolume(volumeF);
        rendererF->SetBackground(0.5,0.5,0.5);
115     rendererF->ResetCamera();
        qVTK1->GetRenderWindow()->Render();
117     volumeFActive = true;

119     vtkBoxWidgetCallback *callback = vtkBoxWidgetCallback::New();
        callback->SetVolumeMapper(mapperF);
121     box->AddObserver(vtkCommand::InteractionEvent, callback);
    }
123 else
    {
125   mapperF->SetInput(f->Fusionate(percentA,percentB,highF));//esto es para no
      recrear todos los renderers, cuando se haga la fusion con otras opciones...
    }
127 }
void Mainwindow::slotSetGrayInterval(int opacityMean, int opacityRange)
129 {
    /*!
```

```
    Aqui se hacen todas las llamadas, instanciaciones, para CreateProperty esta
    funcion
    va a contener a opacityMean que es la media del valor de gris elegido por el
    usuario
    opacityRange el tamanio del rango, hay que dividir la mitad de ese valor y
    sumar o restar
    para obtener el minimo y el maximo.
  */
  OpacityMean = opacityMean;
  OpacityRange = opacityRange;

  SetPropertyToActiveVolumes();
}

void Mainwindow::slotSetRGBSegment(std::vector<ColorSegment> *colors)
{
  /*!
    Aqui se hacen todas la llamadas, instanciaciones para leer el vector de
    ColorSegment
    a partir del vector "colors"
  */
  Colors = *(colors);

  SetPropertyToActiveVolumes();
}

void Mainwindow::SetPropertyToActiveVolumes()
{

  vtkSmartPointer<vtkVolumeProperty> property = CreateProperty( OpacityMean,//
    mean
                                    OpacityRange, //range
                                    Colors);
  if(this->volume1Active)
  {
    this->volume1->SetProperty(property);
  }
  if(this->volume2Active)
  {
    this->volume2->SetProperty(property);
  }
  if(this->volumeFActive)
  {
    volumeF->SetProperty(property);
  }
}

```

```
void Mainwindow::slotChangeBackground()
{
  /*!
    Metodo exclusivo de la interfaz para asignar el color de fondo del renderer.
  */
  QColorDialog winColor;
  QColor col = winColor.getColor(Qt::blue,this);

  if(col.isValid() && winColor.Accepted)
  {
    emit backgroundChanged(col);
    label->setPalette(QPalette(col));
    label->setAutoFillBackground(true);
    vtkRendererCollection *renderers = qVTK1->GetRenderWindow()->GetRenderers();
    vtkCollectionSimpleIterator sit;
    renderers->InitTraversal(sit);
    for(int k=0;k<3;k++)
    {
      vtkRenderer *r = renderers->GetNextRenderer(sit);

      if(r != NULL)
      {
        r->SetBackground(col.redF(),
                 col.greenF(),
                 col.blueF());
      }
    }
  }

}
void Mainwindow::dockSetBackgroundColor()
{
  /*!
    Metodo exclusivo de la interfaz. Crea la ventana flotante que contiene el
    boton para
    cambiar el color de fondo.
  */
  QDockWidget *dockB = new QDockWidget(tr("Opciones de visualizacion"),this);
  dockB->setAllowedAreas(Qt::LeftDockWidgetArea | Qt::BottomDockWidgetArea);
  QWidget *window = new QWidget(dockB);
  QGridLayout *grid = new QGridLayout(window);
  QPushButton *pushButtonColor = new QPushButton(tr("Color"),window);
  QCheckBox *boxWidgetState = new QCheckBox(tr("Cubo de corte"),this);
  label = new QLabel(window);
  label->setMaximumSize(16,16);
  label->setPalette(QPalette(Qt::blue));
```

mainwindow.cpp

```
  label->setAutoFillBackground(true);
  grid->addWidget(label,0,0);
  grid->addWidget(pushButtonColor,0,1);
  grid->addWidget(boxWidgetState,1,0);
  window->setLayout(grid);
  dockB->setWidget(window);
  dockB->setHidden(false);
  dockB->setMaximumSize(280,80);
  connect(pushButtonColor,SIGNAL(clicked()),this,SLOT(slotChangeBackground()));
  connect(boxWidgetState,SIGNAL(toggled(bool)),this,SLOT(setSliceCube(bool)));
  addDockWidget(Qt::RightDockWidgetArea, dockB);
  view->addAction(dockB->toggleViewAction());
}
void Mainwindow::slotShowRGBSegment()
{
  /*!
    Este metodo es exclusivo de la interfaz. Crea una instancia de RGBSegment la
    ventana
    que pide cuantos segmentos de color va a usar el usuario.
  */
  RGBSegment *rgbS = new RGBSegment();
  connect(rgbS,SIGNAL(segment(std::vector<ColorSegment> *)),this,SLOT(
    slotSetRGBSegment(std::vector<ColorSegment> *)));
  rgbS->show();
}
void Mainwindow::slotLoadCoefDHT(const QString coef)
{
  /*!
    Aqui vamos a hacer las llamadas, instanciaciones, etc. correspondientes a la
    visualizacion
    de los Coeficientes de DHT cada numero ya indica que coeficiente es el que va
     a mostrar en pantalla.
    La opcion 12 regresa al volumen original.
  */
  if(coef == "000")
  {
    if(volume1Active)
      mapperV1->SetInput(f->GetImageDataV1(1));

    if(volume2Active)
      mapperV2->SetInput(f->GetImageDataV2(1));

  }
  ...
  else
  {
    QMessageBox::warning(this, tr("ERROR"),
```

```
                                tr("Opcion no valida."));
45    }

47 }
   void Mainwindow::LoadVolume(std::string name,int op)
49 {

51    vtkSmartPointer<vtkDICOMImageReader> dicomReader = vtkSmartPointer<
        vtkDICOMImageReader>::New();
      dicomReader->SetDirectoryName(name.data());
53    dicomReader->Update();

55    if(op == 1)
      {
57    ...
      }
59    else if(op == 2)
      {
61      /*!
        Aqui voy a cargar el volumen 2 tipo dicom
63      con los parametros
        name Es un std::string que tiene la direccion absoluta del conjunto de
        imagenes
65      */
        imageDatV2 = dicomReader->GetOutput();
67      imageDatV2->Update();
      }
69    else
        QMessageBox::warning(this,tr("Advertencia"),tr("No es una opcion valida"));
71 }
   void Mainwindow::LoadVolume(std::string name, float spaceX, float spaceY, float
       spaceZ,int imaX,int imaY,int first, int last,int op)
73 {

75    if(op==1)
      {
77      /*!
        Aqui voy a cargar el volumen 1 tipo raw
79      con los parametros
        name  Es un std::string que tiene el patron para cargar raw
81      spaceX  es un float que representa el tamanio del voxel al momento de crearse
         el volumen sobre el eje X.
        spaceY  es un floar que representa el tamanio del voxel al momento de crearse
         el volumen sobre el eje Y.
83      spaceZ  es un floar que representa el tamanio del voxel al momento de crearse
         el volumen sobre el eje Z.
        imaX  es un int que representa el tamanio en pixeles de largo de la imagen.
85      imaY  es un int que representa el tamanio en pixeles de ancho de la imagen.
```

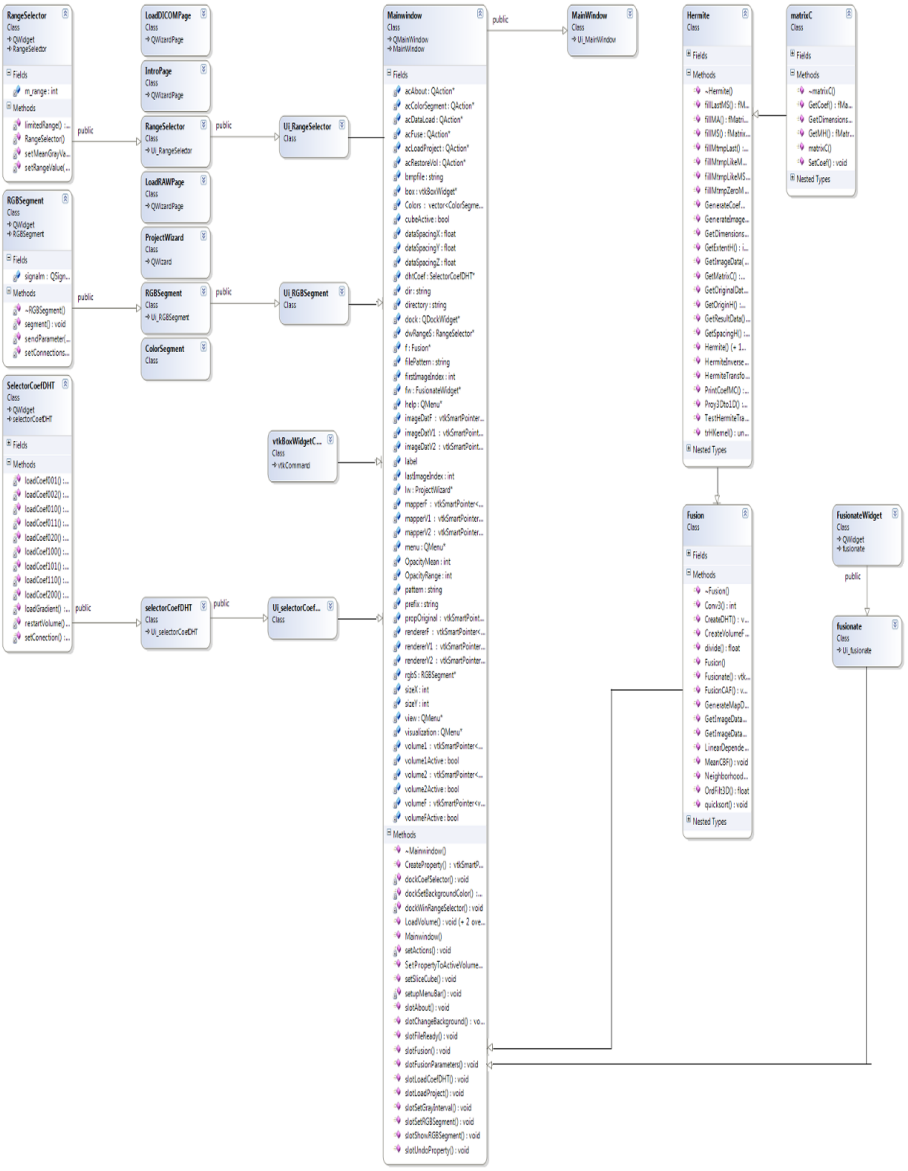
```
      first es un int que representa el numero indicado en el nombre de la imagen
      para comenzar la carga.
87    last  es un int que representa el numero indicado en el nombre de la imagen
      para terminar la carga.
      */
89    imageDatV1 = vtkSmartPointer<vtkImageData>::New();
      vtkSmartPointer<vtkImageReader2> r = vtkSmartPointer<vtkImageReader2>::New();
91    r->SetDataSpacing(spaceX, spaceY, spaceZ);
      r->SetFilePattern(name.data());
93    r->SetDataExtent(0, imaX, 0, imaY, first, last);
      r->SetDataOrigin(0.0, 0.0, 0.0);
95    r->SetDataScalarTypeToUnsignedChar();
      r->SetDataByteOrderToLittleEndian();
97    r->UpdateWholeExtent();
      imageDatV1 = r->GetOutput();
99    imageDatV1->Update();
  }
101 else if(op ==2)
  {
103 ...
  }
105 else
    QMessageBox::warning(this,tr("Advertencia"),tr("No es una opcion valida"));
107 }

109 vtkSmartPointer<vtkVolumeProperty> Mainwindow::CreateProperty(int opacityMean,
      int opacityRange, std::vector<ColorSegment> colorSegments)
{
111   vtkSmartPointer<vtkVolumeProperty> prop = vtkSmartPointer<vtkVolumeProperty>::
      New();
      vtkSmartPointer<vtkColorTransferFunction> colorFun = vtkSmartPointer<
      vtkColorTransferFunction>::New();
113   vtkSmartPointer<vtkPiecewiseFunction> opacityFun = vtkSmartPointer<
      vtkPiecewiseFunction>::New();

115   for(unsigned int i=0; i<colorSegments.size(); i++)
      {
117     colorFun->AddRGBSegment(colorSegments[i].minGrayLevel, colorSegments[i].
      redMin, colorSegments[i].greenMin, colorSegments[i].blueMin,
                  colorSegments[i].maxGrayLevel, colorSegments[i].redMax,
      colorSegments[i].greenMax, colorSegments[i].blueMax );
119   }

121   opacityFun->AddSegment( opacityMean - 0.5*opacityRange, 0.0,

123               opacityMean + 0.5*opacityRange, 1.0 );

125   prop->SetAmbient(0.1);
```

```
    prop->SetDiffuse(0.9);
127 prop->SetSpecular(0.4);
    prop->SetSpecularPower(10.0);
129 prop->SetIndependentComponents(false);
    prop->SetColor( colorFun );
131 prop->SetScalarOpacity( opacityFun );
    prop->SetInterpolationTypeToLinear();
133 prop->ShadeOn();


135

    return prop;
137 }

139 void Mainwindow::slotFusionParameters()
  {
141 if(this->volume1Active && this->volume2Active)
    {
143   FusionateWidget *fw = new FusionateWidget(this);
      fw->show();
145   connect(fw,SIGNAL(methodsValue(float,float,int)),this,SLOT(slotFusion(float,
      float,int)));
    }
147 else
    {
149   QMessageBox::warning(this,tr("Advertencia"),tr("Para fusionar se necesita
      cargar dos volumenes."
        "Vaya al menu Proyecto->Cargar datos y eliga el conjunto de datos a
      visualizar."
151     "Si ya habia generado un proyecto y conoce la ubicacion del archivo, vaya a
       Proyecto->Cargar Proyecto." ));
    }
153

  }
```

mainwindow.cpp

Apéndice B

Diagrama de clases



Bibliografía

[13] HEARN Donald and BAKER M.Pauline. *Gráficos por computadora con OpenGL*. Number 3. Pearson, 2006. ISBN 978-84-205-3980-5.

[15] DOUGHERTY Geoff. Digital image processing for medical applications. *Cambridge University Press*, 2009.

[19] VAQUERO L.J.J. and DESCO M.M. Más de un siglo de imagen médica. *Arbor: Ciencia, pensamiento y cultura*, (698):337–365, 2004.

[3] Real 3D, 2013. URL http://www.reald.com/content/crystal-eyes-4.aspx.

[27] John T.Bell. Virtualization toolkit tutorial, 2004. URL http://www.cs.uic.edu/~jbell/CS526/Tutorial/Tutorial.html.

[26] CRUZ-TECHICA Sonia. Fusión de volúmenes generados a partir de imágenes médicas utilizando la transformada hermite. Master's thesis, 2011.

[24] ROMERO-HERNÁNDEZ Oscar and TAPIA-PATLÁN Enrique Ramón. Desarrollo de una aplicación para visualización, manipulación y segmentación de volúmenes, 2007.

[23] BANKMAN Isacc N. *Handbook of Medical Imaging. Processing and Analysis*. Academic Press, 2000.

[18] HADWIGER M. PREIM B. RITTER F. VILANOVA A. ZACHMANN G. KLEIN J., FRIMAN O. and BARTZ D. Visual computing for medical diagnosis and treatment. *Computers and Graphics 33*, 4:554–565, 2009.

[28] VisionARC, 2012. URL http://www.visionarc.es/tecnologias2.html.

[16] MARTENS Jean-Bernand. The hermite transform-theory. *Speech and Signal Processing 38*, 9:1595–1606, 1990.

[25] HOPSON M.P. RILEY K.F. and BENCE S.J. *Mathematical Methods for Physics and Engineering.* Cambridge University Press, 2002.

[20] ABRAMOWITZ M. and SETEGUN I. *A handbook of mathematical functions.* Dover, 1965.

[5] LOPEZ-CALOCA A.A. *Técnicas Avanzadas de Fusión de Imágenes.* PhD thesis, Universidad Nacional Autónoma de México, 2007.

[29] FREEMAN W. and ADELSON E. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, (9):891–906, Septiembre 1991.

[8] ESCALANTE-RAMÍREZ Boris and SILVÁN-CÁRDENAS J.L. Advanced modeling of visual information processing. a multi-resolution directional-oriented image transform based on gaussian derivatives. *Signal Processing: Image Communication 20*, (9-10):891–906, 2005.

[9] ESCALANTE-RAMÍREZ Boris and SILVÁN-CÁRDENAS J.L. The multiscale hermite transform for local orientation analysis. *IEEE Transactions on Image Processing 15*, (5):1236–1253, 2006.

[17] SILVÁN-CÁRDENAS J.L. Compresión de imágenes basada en modelos gaussianos de percepción visual. Master's thesis, Universidad Nacional Autónoma de México,Facultad de Ingeniería, 2002.

[21] HASHIMOTO M. and SKLANSKY J. Multiple-order derivatives for detecting local image characteristics. *Comput.Vision Graph*, 1(39):28–55, 1987.

[10] MUÑOZ-TORRES Juan Carlos. Fusión de imágenes médicas a través de la transformada hermite. Master's thesis, Universidad Nacional Autónoma de México, Facultad de Ingeniería, 2005.

[14] DURUCAN E. and EBRAHIMI T. Change detection and background extraction by linear algebra. *Proceeding of the IEEE*, 10(89):1368, 2001.

[6] KUMAR A. AGUILAR-PONCE R., TECPANECATL-XIHUITL J. and BAYOUMI M. Pixel-level image fusion scheme based on linear algebra. pages 2658–2661. ISCAS 2007, IEEE Internacional Symposium on Circuits and Systems, 2007.

[4] MAHYARI A. and YAZDI M. A novel image fusion method using curvelet transform based on linear dependency test. Internacional Conference on Digital Image Processing.

[22] TENENBAUM Morris and POLLARD Harry. *Ordinary Differencial Equations*. Dover, 1963.

[1] Convocatoria 2010 para el fortalecimiento de la docencia, 2010. URL http://www.ixtli.unam.mx/images/pdf/ListaProyIXTLI2010.pdf.

[12] Digia. Qt, 2011. URL http://qt.digia.com/.

[2] Virtualization toolkit vtk, 2011. URL http://vtk.org/.

[7] Sébastien Barré. Dicom2: How to, August 2013. URL http://www.barre.nom.fr/medical/dicom2/how-to.html.

[11] S.Instituto de óptica CSIC. Image processing and vision modeling group. URL http://www.iv.optica.csic.es/. Última visita Junio 2011.