UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

Visualización de volúmenes utilizando funciones de transferencia multidimensionales

Trabajo Especial de Grado presentado por
Br. Pedro Antonio Caicedo Vargas
ante la ilustre
Universidad Central de Venezuela

para optar al título de
Licenciado en Ciencias de la Computación

Tutor
Prof. Rhadamés Carmona

Caracas, Octubre de 2011

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por el Bachiller Pedro Antonio Caicedo Vargas C.I.: 18.392.552, con el título "Visualización de volúmenes utilizando funciones de transferencia multidimensionales", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 27 de Octubre de 2011 a la 1 y 30 PM, para que su autor lo defendiera en forma pública, en el Centro de Computación Gráfica, lo cual este realizó mediante una exposición oral de su contenido, y luego respondió satisfactoriamente a las preguntas que les fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente acta, en Caracas el 27 de Octubre de 2011, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Rhadamés Carmona.

_____
Prof. Rhadamés Carmona
(Tutor)

_____
Prof. Haydemar Nuñez
(Jurado Principal)

_____
Prof. Héctor Navarro
(Jurado Principal)

Resumen

El despliegue de volumen es una técnica de visualización que consiste en la proyección de las muestras del volumen en una imagen. Este proceso se divide en varias etapas: la reconstrucción de muestras, la clasificación y la composición. En particular, la clasificación es la etapa de mayor importancia, pues es aquí donde se asignan las propiedades ópticas a las muestras del volumen. En la clasificación, se enfatizan o no los diversos materiales presentes en el volumen. La manera más común de realizar la clasificación es mediante el uso de funciones de transferencia.

La implementación de la función de transferencia se realiza generalmente mediante una función lineal a trozos, en donde se le asigna a cada muestra posible del volumen un color y una absorción. Estas funciones unidimensionales no permiten la correcta caracterización de las fronteras entre los materiales. Para lidiar con este problema, se consideran otras variables al momento de clasificar el volumen. Dichas variables son el gradiente, su longitud, el vector normal, entre otros. Al aumentar la dimensión del dominio de la función de transferencia también aumentamos el conjunto de datos usados en la caracterización de zonas de interés, permitiéndonos entonces distinguir las fronteras entre los materiales.

En este trabajo se realiza una implementación de una función de transferencia multidimensional, expandiendo su dominio a través de la magnitud del gradiente y la segunda derivada direccional a lo largo del gradiente, estos cálculos fueron obtenidos por el método de reconstrucción de suma de convoluciones. Además se implementó una interfaz basada en *widget* para la edición de la función de transferencia. Las pruebas demuestran que las funciones de transferencia multidimensionales permiten mejorar la especificación de la clasificación en los volúmenes, pudiendo aislar partes del volumen de manera efectiva.

Palabras claves: Función de transferencia, clasificación, despliegue de volumen, métodos de reconstrucción.

Tabla de contenido

Introducción

Un volumen discreto por lo general es un conjunto de muestras adquiridas de un objeto físico. El despliegue del volumen consiste en proyectar directamente su conjunto de muestras hacia el plano imagen, para así visualizar el volumen sin necesidad de requerir de reconstrucciones poligonales intermedias[**1**].

En general no es útil observar el volumen en su totalidad, al contrario nos interesa observar subconjuntos del volumen, donde se encuentren las características de nuestro interés. Para ello, es necesario proveer una etapa de clasificación de las muestras del volumen que nos permita la manipulación de sus propiedades ópticas; de esta manera se pueden encontrar y resaltar las características o materiales de interés, mientras se ocultan las que no son de tanto interés. En cierta forma, podemos afirmar, que la utilidad del despliegue está determinada por la calidad de su clasificación, ya que en la clasificación es donde se determinan las propiedades ópticas del volumen para su visualización.

Debemos destacar que, para lograr obtener una zona de interés del volumen, existen principalmente dos posibilidades. La primera consiste en aislar la característica de interés por medio de herramientas de corte o segmentación, proceso que es complejo debido a la diversidad de formas que puede presentar una característica o material. La otra opción consiste en la manipulación directa de las propiedades ópticas mediante la función de transferencia.

La función de transferencia es la encargada de realizar el mapeo entre las muestras y las propiedades ópticas, llevando las muestras del domino físico original hacia el dominio espectral de colores y transparencias[**2**]. El dominio de dichas funciones puede estar representado bien sea por una sola dimensión (valor de las muestras) o por más de una dimensión (valor de las muestras, magnitud del gradiente, entre otros).

En este trabajo se propone la implementación de funciones de transferencia multidimensionales para la visualización de volúmenes. En dicha implementación se logra caracterizar de forma precisa las muestras en la clasificación.

Planteamiento del Problema

Generalmente, en la visualización de volúmenes se utilizan funciones transferencia unidimensionales para clasificar los datos del volumen; estas funciones comúnmente sustituyen los valores de los vóxeles por valores de emisión y absorción. Sin embargo, muchas veces estas funciones de transferencia no son suficientes en la caracterización, pues no permiten una correcta clasificación de las características de interés presentes en el volumen, las cuales suelen estar localizadas en las fronteras entre los materiales.

Es por ello, que para lograr una clasificación más precisa, se ha demostrado en trabajos previos que agregando otras variables (además de los valores de los vóxeles) al

dominio de la función de transferencia se pueden clasificar adecuadamente estas fronteras entre los materiales, logrando describir con mayor amplitud las características de interés que se deseen visualizar. Sin embargo, manipular varias variables para obtener los valores adecuados puede ser una tarea muy tediosa, que va a requerir de herramientas de edición sofisticadas.

Propuesta de solución

Las funciones de transferencias 1D presentan problemas en la caracterización de las muestras, por lo general presentes entre las fronteras de los materiales. De allí pues, que se plantea desarrollar un sistema que permita la manipulación de funciones de transferencia multidimensional 2D ó 3D y, de este modo tener más precisión en la clasificación.

Objetivo General

Implementar un sistema de despliegue de volúmenes que permita la edición de funciones de transferencia multidimensionales, para mejorar la diferenciación de los distintos materiales presentes en el volumen, y de las fronteras entre dichos materiales.

Objetivos Específicos

- Implementar el algoritmo de *Ray Casting* para el despliegue de volumen, utilizando ray casting basado en GPU.
- Extraer del volumen la información necesaria para la edición de la función de transferencia multi-dimensional.
- Realizar una interfaz que permita la visualización y manipulación de la función de transferencia, de manera usable e intuitiva.
- Definir un formato de archivo para cargar y guardar las funciones de transferencia obtenidas, asociándolas a un determinado volumen.
- Comparar las imágenes generadas con distintas funciones de transferencia unidimensionales y multidimensionales, en cuanto al tiempo de respuesta y a la efectividad en la separación de los materiales y sus fronteras.

Alcance de este trabajo

- Debido a que el despliegue está basado en ray casting acelerado por GPU, el tamaño del volumen y de los datos a utilizar no deben sobrepasar las capacidades de la GPU.
- Sólo se soporta volúmenes de 8-bits, ocasionando que cualquier volumen que este codificado a un valor distinto a 8-bits tenga que se recodificado.

El trabajo es presentado en cinco capítulos. En el Capítulo I se estudian los conceptos básicos sobre la visualización de volúmenes. Seguidamente en el Capítulo II se explican los

conceptos relacionados con las funciones de transferencia. En el Capítulo III se describe detalladamente la implementación de la función de transferencia, donde se indican sus estructuras de datos y el algoritmo general utilizado. En el Capítulo IV se realizan las pruebas pertinentes del sistema realizado, describiéndose los resultados obtenidos. Por último en el Capítulo V se presentan las conclusiones de la investigación y se proponen algunos trabajos futuros.

Capítulo I.

Marco Teórico

En la medicina existen herramientas que permiten la adquisición de datos físicos del cuerpo para su diagnóstico. Estas herramientas (Computed Tomography CT, Magnetic Resonance Imaging MRI, entre otros) capturan, por medio de procesos físicos, las densidades de los materiales que componen el cuerpo. Con esos datos, se forman un conjunto de muestras, típicamente representados por una malla regular conocida como volumen [**3**]. También, este conjunto de datos pueden provenir de simulaciones, mediciones, entre otros. Existen técnicas que permiten visualizar estos volúmenes. Algunas, generan geometría intermedia para permitir observar la superficie de interés [**4**]. Sin embargo, con estas técnicas solo podemos observar detalles específicos y no podemos observar el volumen general, con lo que se obvian datos importantes que nos podrían permitir una mejor panorámica para tomar decisiones acertadas. Otras técnicas, como el despliegue directo de volúmenes, sí permiten observar el volumen en su totalidad, y eliminar únicamente los materiales no relevantes para su estudio.

Específicamente, el despliegue directo de volumen se refiere a las técnicas que producen una imagen proyectando directamente los datos del volumen sobre el plano imagen [**1**]. Estas técnicas requieren un modelo óptico para simular la interacción de los datos del volumen con la luz; esto es, cómo el volumen de datos genera, refleja, dispersa y ocluye la luz.

En este capítulo se estudia el proceso de despliegue directo de volúmenes, y se presenta una técnica para su despliegue.

1.1 Vóxel y Volumen

Un vóxel es un conjunto S de datos (x, y, z, v), donde v representa el valor de alguna propiedad de los datos, mientras que su localización en el espacio 3D está dado por x, y, z. Las propiedades de los datos pueden estar representadas por un simple escalar, o por un conjunto de valores; en este último caso el valor de la muestra es multivaluada, incluyendo por ejemplo, color, intensidad, calor o presión. Incluso el valor v puede ser un vector, representando, por ejemplo, la velocidad de cada muestra del volumen[**5**]. En consecuencia, un volumen es una colección de vóxeles generados mediante alguna medición o simulación.

La manera como está estructurado internamente el volumen es importante, pues de esto depende su acceso y almacenamiento. Un volumen es representado por una malla, que de acuerdo a la topología fundamental presentada por Jonathan Shewchuck[**6**], estas mallas pueden ser estructuradas y no estructuradas.

Las mallas estructuradas, se consideran todas las que se pueden representar como un arreglo tridimensional de escalares y cuya conectividad queda implícita. Las mallas no estructuradas son más generales; las mismas pueden tener concavidades y requieren conectividad explícita entre celdas [**7**][**3**][**6**].

Existe dos formas de interpretar un vóxel (ver Figura 1.1); una es considerarlo como un cubo en cuyo caso el valor de muestra corresponde al centro del mismo, o verlo como una celda en el cual el valor de muestra v corresponde a un vértice de la malla de la celda [**4**].

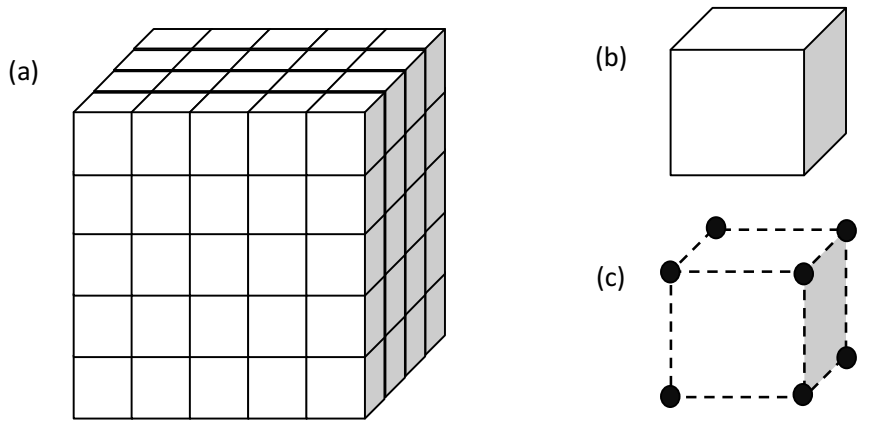


Figura 1.1: (a) una composición típica de volumen representado por un arreglo tri-dimendisional, mientras que (b) y (c) son las dos interpretaciones de vóxel; en (b) la interpretación como cubo y en (c) su interpretación como celda.

Los volúmenes pueden ser adquiridos de diferentes fuentes, tales como: los datos médicos, obtenidos por medio de escáneres de Tomografía Computarizada (CT) o Imágenes por Resonancia Magnética (MRI); los datos físicos, son los producidos por la Dinámica de Fluido Computacional (CFD); los datos sísmicos ó cualquier otro conjunto de datos representados como campos escalares tridimensionales. Cabe resaltar, que los volúmenes pueden ser generados sintéticamente por medio de técnicas procedurales[**8**], que son usadas por los científicos y programadores para desplegar objetos tales como: fluidos o gases, fenómenos naturales (la niebla, nubes y fuego), visualización molecular, explosiones y cualquier otro efecto utilizado en los juegos 3D de computadora.

1.2 Despliegue Directo de Volúmenes

El Despliegue Directo de Volúmenes (*Direct Volume Rendering*) [**9**][**10**] se describe como una técnica que permite visualizar un volumen discreto mediante la proyección directa de sus muestras, y sin necesidad de reconstruir geometrías intermedias. Durante la proyección de las muestras se simula la interacción de la luz con un medio semi-transparente representado por el volumen.

El despliegue directo de volúmenes es usado en distintas áreas de las ciencias y en el entretenimiento, principalmente como herramienta de diagnóstico en la medicina; igualmente en simulaciones en las ciencias naturales o simplemente para lograr gran realismo en las escenas de los videojuegos[**11**]. En el proceso de despliegue, el volumen es un arreglo de datos escalares extraídos de un espacio tridimensional y la interacción luz-volumen produce varios fenómenos (ver Figura 1.2):

- La absorción de la luz
- El esparcimiento (*scattering*) de la luz
- La emisión de la luz

El objetivo del despliegue directo de volúmenes no es simular todos estos fenómenos; por lo general, sólo se modela la propagación y el avance de la luz por el volumen. De esta manera, solo nos queda un modelo óptico en donde se toma en cuenta la emisión y absorción de la luz[**12**][**13**][**14**].

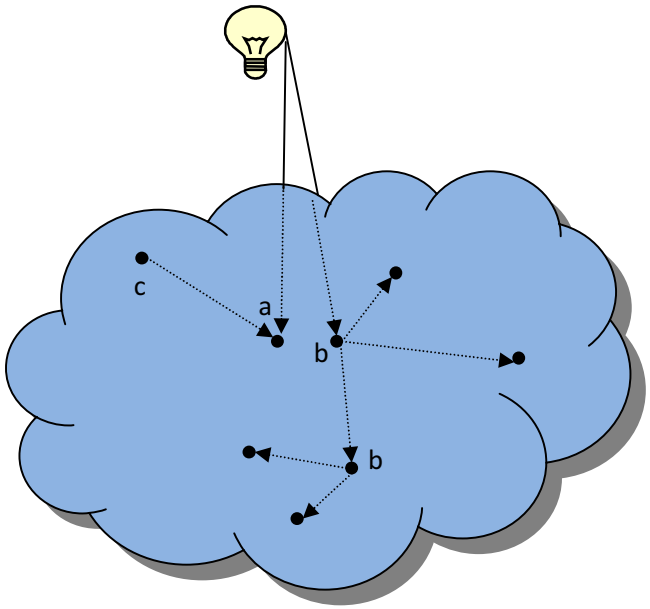


Figura 1.2: Fenómenos producidos por la interacción luz-volumen: (a) absorción, (b) *scattering*, y (c) emisión.

En el trabajo presentado por *Carmona*[**15**] se presenta detalladamente la derivación del modelo óptico Emisión-Absorción presentado por *Sabella*[**12**] y Williams *et al.*[**13**]. En este modelo el vector de visualización que atraviesa al volumen es parametrizado con λ en $[0, D]$, y el color resultante C se obtiene mediante la siguiente ecuación:

$$C = \int_0^D c(\lambda)\tau(\lambda)e^{-\int_0^\lambda \tau(\lambda')d\lambda'}\, d\lambda, \qquad [Ec.\,1.1]$$

En donde $c(\lambda)$, corresponde a la emisión, y $\tau(\lambda)$ a la absorción del volumen a una distancia λ (ver Figura 1.3). El factor $e^{-\int_0^\lambda \tau(\lambda')d\lambda'}$ correspondiente a la extinción de la luz, se puede interpretar como la transparencia $T(\lambda)$ del volumen a una profundidad o distancia λ. Basada en la transparencia, se puede calcular la opacidad α acumulada en la travesía del rayo a cualquier distancia λ, como:

$$\begin{aligned} &\alpha(\lambda) = 1 - T(\lambda) \\ \Rightarrow\ &\alpha(\lambda) = 1 - e^{-\int_0^\lambda t(\lambda')d\lambda'}. \end{aligned} \qquad [Ec.\,1.2]$$

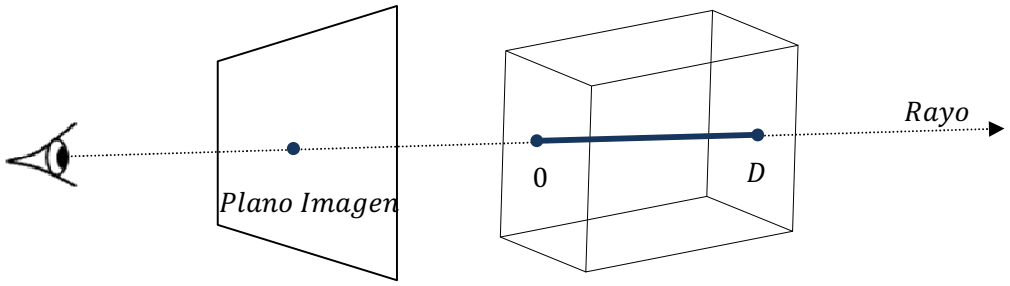


Figura 1.3: Para determinar el valor de un píxel en la imagen, se calcula la acumulación de color y opacidad a lo largo del rayo, desde que entra al volumen hasta que sale; barriendo λ desde 0 hasta D.

1.2.1 Pipeline de Visualización de Volúmenes

Levoy [**9**] y *Drebin et al.*[**10**] en su modelo de visualización de volumen, proponen una serie de etapas (pipeline) que permiten el despliegue del volumen. Por ser los modelos más usados en la visualización gráfica y científica, resumiremos las etapas del pipeline presentado en *Kniss* [**16**]:

a.- Pre-procesamiento: el volumen adquirido puede presentar ruido producto de las técnicas de muestreo empleadas por los sistemas de captura. Además, el volumen puede estar compuesto por una colección de imágenes independientes; por ejemplo, una colección de imágenes en formato *Windows Bitmap* (*.bmp*), o formato *DICOM* (*Digital Imaging and Communications in Medicine*)[**17**]. Es en esta etapa donde se realizan las conversiones de formato o se aplican filtros correctivos para reducir el ruido. Adicionalmente, se puede particionar el volumen, pre-calcular el vector gradiente por vóxel, etc., logrando entonces

obtener una representación correcta del volumen e información adicional, que luego se utiliza en las siguientes etapas del pipeline.

b.- Reconstrucción: Básicamente se trata de la interpolación entre las muestras que ocurre cuando se desea estimar los valores desconocidos entre los vóxeles o muestras conocidas. Comúnmente el hardware está optimizado para estimar las muestras inexistentes con filtros lineales para texturas 1D, bi-lineales para texturas 2D y tri-lineales para texturas 3D [**18**].

c.- Clasificación: Se refiere a la asignación de las propiedades ópticas a los vóxeles que componen el volumen. Es una de las etapas más importantes del pipeline de visualización de volúmenes, en donde se enfatiza las características de interés del volumen. Para valorar las características del volumen se acopla una función transferencia, y es por su manipulación donde se fijan las propiedades ópticas del volumen; por su importancia, este tema se aborda en el siguiente capítulo.

d.- *Shading*: El *shading* consiste en modular el color y el brillo del vóxel al considerar su interacción con una luz externa [**15**]. El modelo Blinn-Phong es muy utilizado en la visualización de volúmenes. Este modelo es estrictamente local y tiene tres componentes, el ambiental que representa la luz proveniente en todas las direcciones, el difuso que representa la luz reflejada (*scattered*) isotrópicamente en todas las direcciones, y el especular que representa la luz reflejada en dirección al visor para simular superficies pulidas.

e.- Composición: es la etapa final del pipeline y consiste en mezclar por medio de los operadores *under* ó *over* [**9**][**10**] , los colores modulados producto de la etapa de clasificación y *shading*.

1.2.2 Discretización de la Ecuación de Visualización de Volúmenes

Evaluar la ecuación 1.1 analíticamente acarrea demasiado cómputo. Además, en dicha ecuación no se tiene en cuenta el proceso de muestreo y clasificación del volumen. De acuerdo con Carmona[**15**], y centrando el análisis en la post-clasificación (ver Capítulo II), la integral de visualización de volumen se resume numéricamente a:

$$C \approx \sum_{i=0}^{n-1} \prod_{j=0}^{i-1} e^{-h\tau(S_j)}\, c(S_i)\left(1 - e^{-h\tau(S_i)}\right). \qquad \text{[Ec. 1.3]}$$

en donde h es la longitud fija entre muestras y n la cantidad de muestras.

Sea el color $c_i = c(S_i)$, y la opacidad $\alpha_i = 1 - e^{-h\tau(S_i)}$ (Figura 1.4); la ecuación queda simplificada a:

$$C \approx \sum_{i=0}^{n-1} \prod_{j=0}^{i-1} (1 - \alpha_j)\alpha_i c_i \qquad \text{[Ec. 1.4]}$$

$$C \approx \alpha_0 c_0 + (1 - \alpha_0)\alpha_1 c_1 + (1 - \alpha_0)(1 - \alpha_1)\alpha_2 c_2 + \cdots$$

$$\Rightarrow$$

$$+(1 - \alpha_0) \cdots (1 - \alpha_{n-2})\alpha_{n-1} c_{n-1}.$$

La derivación de esta ecuación se detalla en [**15**].

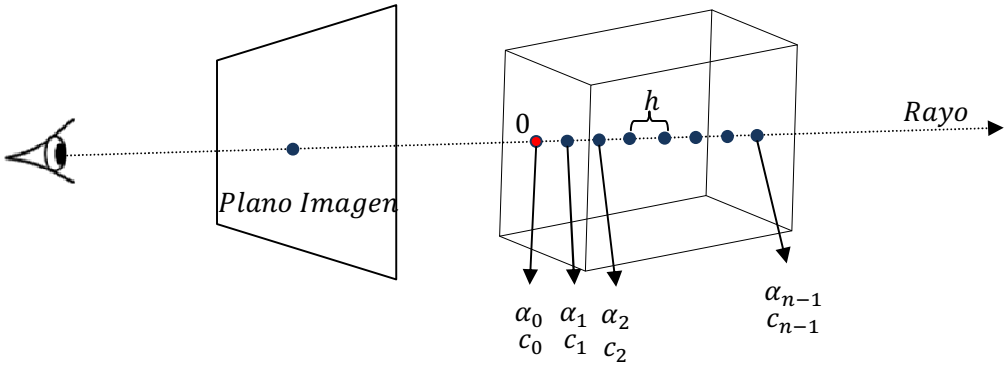


Figura 1.4: Para determinar el valor de un píxel, se componen la contribución de color y opacidad de las muestras equidistantes a lo largo de la travesía del rayo al pasar por el volumen. La distancia h generalmente es constante, y se asume constante también la emisión y absorción de cada segmento de longitud h.

Adicionalmente es necesario resaltar que la ecuación 1.4 es evaluada iterativamente a través de los operadores *over*/*under*. Estos operadores son utilizados para componer las muestras interpoladas, clasificadas y sombreadas [**9**][**10**]. Con el operador *under* se componen las muestras desde la más cercana a la más lejana (*front to back)* con respecto a la posición del ojo. Este operador se define como:

$$\begin{aligned} c &\coloneqq (1 - \alpha)\alpha_i c_i + c, \\ \alpha &\coloneqq (1 - \alpha)\alpha_i + \alpha, \end{aligned} \qquad \text{[Ec. 1.5]}$$

En cambio el operador *over* evalúa las muestras desde la más lejana a la más cercana con respecto a la posición del ojo (*back to front*), definiéndose como:

$$\begin{aligned} c &\coloneqq \alpha_i c_i + (1 - \alpha_i)c, \\ \alpha &\coloneqq \alpha_i + (1 - \alpha_i)\alpha, \end{aligned} \qquad \text{[Ec. 1.6]}$$

En los dos casos el c y α son inicializados en 0. Note que en el uso del operador *over*, el valor acumulado de α no es utilizado en la composición del color [**15**].

1.3 Algoritmos para el Despliegue de Volúmenes

Existen diversos algoritmos que permiten lograr la visualización de un volumen, por lo general estos algoritmos se caracterizan en dos grupos según la manera como utiliza el hardware gráfico. Los algoritmos que hacen uso de polígonos texturizados son principalmente acelerados por hardware (Textura 2D y 3D). Los demás algoritmos son técnicas basadas en software y generalmente son ray casting, splatting, y shear-warp. En la actualidad, el ray casting puede ser implementado por hardware, gracias a la programación de los GPUs.

A continuación se estudia el algoritmo de ray casting, y su implementación utilizando hardware gráfico.

1.3.1 *Ray Casting*

Ray casting es una técnica que consiste en el lanzamiento de un rayo por cada píxel de la imagen, partiendo desde la posición del ojo, y atravesando al volumen (ver Figura 1.5). En la travesía del rayo se muestrea el volumen, y con estas muestras se evalúa la ecuación de visualización del volumen [**9**].

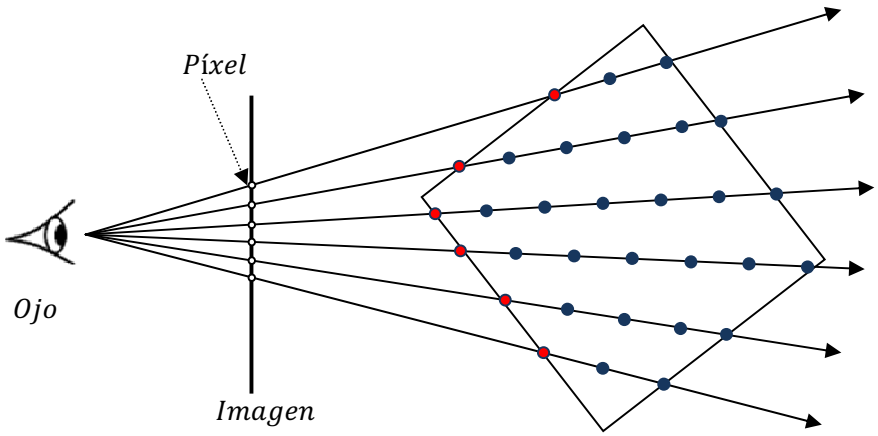


Figura 1.5: Idea general del *Ray Casting*

El rayo permite muestrear a intervalos regulares y obtener los valores escalares en esas posiciones. En caso de que el rayo atraviese un lugar del volumen en donde no se dispone de una muestra, se aproxima por medio de un filtro tri-lineal entre los 8 vecinos más cercanos de la muestra a reconstruir. Pueden utilizarse filtros más exactos que tomen en cuenta mayor cantidad de muestras, como los filtros gaussianos o bi-cúbicos [**10**].

Ahora bien, *Levoy*[**19**] presentó las dos optimizaciones siguientes: terminación temprana de rayos y salto de espacios vacíos. La primera optimización se logra componiendo el rayo en sentido *front to back* (desde la más cercana a la más lejana), en donde la travesía del rayo es truncada al alcanzar un umbral de opacidad determinado [**20**]; de esta manera se ahorra

tiempo de cálculo, pues más allá del umbral el aporte de color de las muestras restantes es nulo o despreciable. La segunda optimización complementa la primera puesto que la terminación temprana de rayos causa pocos efectos para objetos semi-transparentes. En la segunda optimización lo que se busca es saltar los segmentos de rayos que no aportan color a la ecuación, ahorrándose cálculos innecesarios.

Haciendo uso de una estructura de datos se realiza un seguimiento de las áreas del volumen donde la densidad es cero o muy cercana a cero. Si el área del volumen (generalmente un bloque) se considera vacío, se calcula el punto de salida del área y el rayo sigue su travesía, evitando el cálculo de composición de esa área. Por lo general estas áreas son bloques, donde se pre-calculan los valores de muestras mínimos y máximos, y dependiendo a estos valores se decide saltar o no un bloque [**21**].

Otra optimización es generar una visualización de poca resolución y con tasa de muestreo baja, en el momento que se está clasificando los vóxeles. A esto se lo conoce como refinamiento progresivo [**20**].

En la actualidad, el algoritmo de *ray casting* puede ser acelerado por hardware en tiempo real, gracias a la flexibilidad de los procesadores de fragmentos que permiten la ejecución de bloques de programa [**21**]. También es importante resaltar que el algoritmo de *ray casting* es considerado de orden de imagen, porque recorre la imagen en el mismo orden en que los píxeles están almacenados en memoria principal; debido que los rayos pueden atravesar el volumen en cualquier dirección, el algoritmo de *ray casting* no aprovecha la localidad espacial del volumen. Krüger and Westermann[**21**] proponen un algoritmo para realizar *ray casting* por GPU, en donde almacenan el volumen en una textura 3D para enviarcela al GPU. El algoritmo consta de 3 pasos:

1. Determinar el punto de entrada del rayo: El punto de entrada del rayo es obtenido desplegando en una textura 2D las caras delanteras del cubo unitario que representa el volumen en el espacio de textura (ver Figura 1.6a). Debido a que el vértice en el espacio textura es el mismo que en el espacio del color (pues comparten el mismo dominio), se le asignan como color sus coordenadas 3D. Luego se por rasterización se obtiene una textura del mismo tamaño del *viewport* (pantalla de visualización), en donde cada fragmento de la imagen representa el punto de entrada del rayo en espacio textura.

2. Determinar la dirección del rayo: En este paso se realiza el proceso anterior pero con las caras traseras del volumen (ver Figura 1.6b). Luego por medio de un programa de fragmento se muestrea la textura generada en el paso 1 en el pixel actual y se calcula la dirección del rayo normalizada. Esta información es almacenada en los componentes de color *RGB*. Por último, se calcula la longitud del rayo y se almacena en el canal alfa *A*.

3. Recorrido del rayo: Con el punto de entrada del rayo y su dirección obtenido de las dos textura generada en los pasos previos, se realiza N iteraciones avanzando de acuerdo a la distancia de muestreo, componiendo el color en cada iteración hasta, salir del volumen o haber alcanzado un nivel muy alto e opacidad. Al final del recorrido obtenemos el valor final del píxel en la imagen.

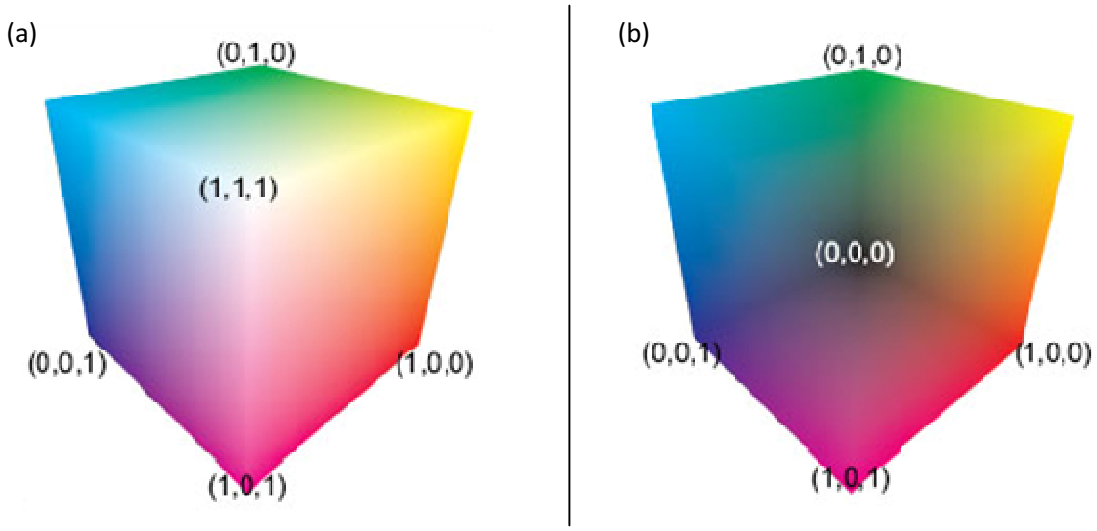


Figura 1.6: En (a) las caras delanteras del cubo unitario y en (b) sus caras traseras.

1.4 Iluminación

En el despliegue de volúmenes, para aumentar el efecto de profundidad y realismo, es necesario incorporar un modelo de iluminación. El *Shading* es el proceso encargado de calcular el modelo de iluminación a un área o píxel [**20**]. En el modelo de iluminación se simula la interacción existente entre la fuente de luz, la geometría y sus materiales.

Existen principalmente dos tipos de modelos de iluminación generales: El de iluminación local calcula la intensidad de la luz de algún lugar en la escena usando sólo información local, como la normal de la superficie y la distancia a cada fuente de luz [**22**]; en cambio, el de iluminación global, toma en cuenta más detalles como las sombras y la incidencia de la luz indirecta generada en el entorno [**20**].

Los modelos de iluminación local son simples, fáciles de evaluar y no requieren la complejidad computacional de la iluminación global [**23**]. El modelo de iluminación local más popular es el propuesto por *Phong* [**24**] y mejorado por Blinn denominado modelo de iluminación Blinn-Phong (*Blinn–Phong shading model*), en el cual se calcula la luz como una combinación lineal de tres términos diferentes: el ambiental, el difuso y el especular.

$$I_{phong} = I_{ambiental} + I_{difusa} + I_{especular}. \qquad [\text{Ec. } 1.7]$$

El término ambiental es constante, y representa la luz indirecta sobre parte o totalidad del objeto. Si no hay luz indirecta, la constante K_a se hace cero. Viene dado por:

$$I_{ambiental} = K_a \qquad \text{[Ec. 1.7.1]}$$

El término difuso se refiere a la cantidad de luz que es reflejada con igual intensidad en todas las direcciones en una superficie sin brillo. El término difuso es independiente de la dirección del observador; sólo depende del ángulo de incidencia ϕ entre la dirección de la fuente de luz y la normal de la superficie (ver Figura 2.13). El término difuso esta dado por:

$$I_{difusa} = I_p\, k_d\, \cos\phi = I_p\, k_d\, (\vec{l} \bullet \vec{n}). \qquad \text{[Ec. 1.7.2]}$$

donde I_p es la intensidad del recurso de luz, k_d es el coeficiente de reflexión difusa, $\vec{l}$ y $\vec{n}$ representan la dirección de la fuente de luz y el vector normal respectivamente.

Por último, el término especular es agregado para simular el brillo producido por el reflejo directo de la luz en la superficie, generando los llamados puntos de luz (*highlights*), los cuales son dependientes de la posición del observador. El término especular esta dado por:

$$I_{especular} = I_p\, k_s\, \cos\alpha^n = I_p\, k_s\, (\vec{h} \bullet \vec{n})^n. \qquad \text{[Ec. 1.7.3]}$$

$$\vec{h} = \frac{\vec{l} + \overrightarrow{\vec{v}_{Ojo}}}{\left|\vec{l} + \overrightarrow{\vec{v}_{Ojo}}\right|} \qquad \text{[Ec. 1.7.4]}$$

donde el vector $\vec{h}$ indica la dirección de máximo brillo, n representa el exponente de reflexión especular , α es el ángulo formado entre el vector normal $\vec{n}$ y el vector $\vec{h}$, y $\overrightarrow{\vec{v}_{Ojo}}$ es el vector de visualización (ver Figura 1.7).

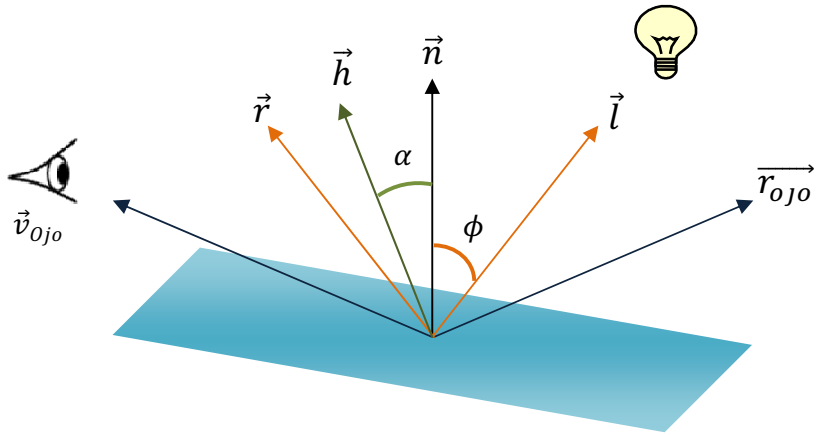


Figura 1.7: Vectores en el modelo de iluminación *Phong*.

2.5.1 Cálculo del Gradiente

El modelo de iluminación *Blinn-Phong*, usa los vectores normales para describir la forma del objeto y calcular la iluminación; pero en la visualización de volúmenes no se dispone de modelos geométricos en donde calcular el vector normal. Por tal motivo es necesario conseguir algún mecanismo que permita el cálculo de las normales, y así aplicar el modelo de iluminación por vóxel.

Afortunadamente el vector normal se puede obtener por medio del vector gradiente normalizado. El vector gradiente es la derivada de primer orden de un volumen discreto (f) [**25**]. El vector gradiente es definido como:

$$df(x,y,z) = \left(\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}\right). \qquad \text{[Ec. 1.8]}$$

El vector gradiente para transformarlo a vector normal es necesario dividirlo por su modulo, y de esta manera llevarlo a su forma unitaria. El vector gradiente puede calcularse con diversas técnicas, continuación describiremos dos métodos que permiten obtener una aproximación de dicho vector.

2.5.1.1 Diferencias centrales.

Es la técnica más común para aproximar el vector gradiente en un conjunto de vóxeles y, se basa en el primer término de una expansión de Taylor. En la Ec. 1.9 $\nabla f(x)$ es el gradiente del vóxel x, equivalente en forma de ecuación a:

$$\nabla f(x) \;=\; \nabla f\left(x_i, y_j, z_k\right) \qquad \text{[Ec. 1.9]}$$

$$\begin{aligned}
\nabla f_x\left(x_i, y_j, z_k\right) &\approx \frac{1}{2}\left(f\left(x_{i+1}, y_j, z_k\right) - f\left(x_{i-1}, y_j, z_k\right)\right) \\
\nabla f_y\left(x_i, y_j, z_k\right) &\approx \frac{1}{2}\left(f\left(x_i, y_{j+1}, z_k\right) - f\left(x_i, y_{j-1}, z_k\right)\right) \\
\nabla f_z\left(x_i, y_j, z_k\right) &\approx \frac{1}{2}\left(f\left(x_i, y_j, z_{k+1}\right) - f\left(x_i, y_j, z_{k-1}\right)\right)
\end{aligned}$$

En la Fig. 1.8 se puede visualizar claramente las muestras que se usarían para obtener el gradiente de un vóxel (x_i, y_j, z_k) en el volumen.

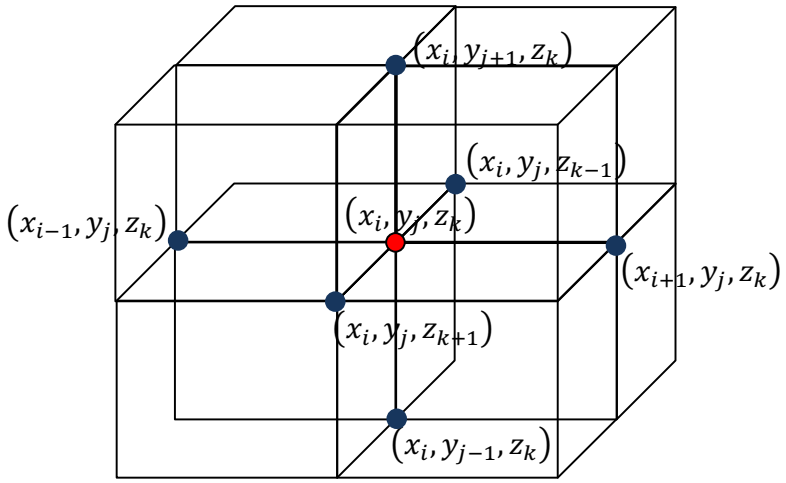


Figura 1.8: Muestras utilizadas en el cálculo del vector gradiente.

2.5.1.2 Filtros usando la expansión de Taylor de la suma de convoluciones

La aproximación del vector gradiente usando el método anterior no es suficiente para ciertas aplicaciones como el cálculo de la curvatura ó cuando se necesita más precisión en la clasificación de los vóxeles , en estos casos se necesita más precisión en la reconstrucción de sus derivadas. Möller *et al.*[**26**] proponen un método que permite diseñar filtros de reconstrucción utilizados para la interpolación o aproximación de las derivadas de una función, manipulando más de un criterio de exactitud, con mínimo error numérico y manteniendo buenas propiedades espectrales. La única hipótesis que requiere este método es que la función original (representada por las muestras discretas) sea una función suave y pertenezca al espacio funcional C^n (espacio de las funciones continuas).

Möller *et al.*[**26**] para reconstruir una función continua $f(k)$ o sus derivadas a partir de un conjunto de muestras $f[k]$, convoluciona $f[k]$ con un kernel de filtro continuo w. El filtro w puede ser un filtro de interpolación o un filtro derivativo; se puede denotar esta operación por $f_r^w(t)$. Formalmente esto puede ser escrito como:

$$f_r^w(t) = \sum_{k=-\infty}^{\infty} f[k] \cdot w(\frac{t}{T} - k), \qquad \text{[Ec. 1.10]}$$

donde T es la distancia constante existente entre las muestras $f[k]$. Ahora se puede expandir $f[k] = f(kt)$ dentro de una serie de Taylor de $N+1$ términos cerca de t. La expansión de la serie de Taylor para una muestra sería:

$$f[k] = \sum_{n=0}^{N} \frac{f^{(n)}(t)}{n!}(kT - t)^n + \frac{f^{(N+1)}(\mathcal{E}_k)}{(N+1)!}(kT - t)^{(N+1)} \qquad \text{[Ec. 1.10]}$$

donde $f^{(n)}(t)$ es la $n\text{-}th$ derivada de f y el error $\mathcal{E}_k \in [t, kT]$.

Sustituyendo la expansión de la serie de Taylor dentro de la suma de convolución de la ecuación 1.10, tendríamos una representación alternativa de la muestra t.

$$f_r^w(t) = \sum_{n=0}^{N} a_n^w(\tau) f^{(n)}(t) + r_{N,i}^w(\tau)$$

$$a_n^w(\tau) = \frac{T^n}{n!} \sum_{k=-\infty}^{\infty} (k-\tau)^n w(\tau - k) \qquad \text{[Ec. 1.12]}$$

$$r_{N,i}^w(\tau) \leq \left(\max\left(f^{(N+1)}(\varepsilon)\right)\right) |a_{N+1}^w(\tau)|\,, \quad \varepsilon \ \epsilon[\,(i-M)T, (i+M)T]$$

ó

$$r_N^w(\tau) \approx a_{N+1}^w(\tau) f^{(N+1)}(t)$$

donde τ se elige de manera que $t = (i+\tau)T$ con $0 \leq \tau < 1$, mientras que i es un entero. Note que los coeficientes de error derivados sólo dependen de la compensación de τ para la muestra más cercana, es decir, son periódicas a la distancia entre muestras T. Para más detalles, consultar [**27**].

La caracterización del proceso de filtrado en la ecuación 1.12 impone cuatro criterios diferentes para una buena reconstrucción de la $k\text{-}th$ derivada. Primero, se requiere que todos los a_n^w sean cero para todo $n<k$. Segundo, tenemos que normalizar por a_k^w con el fin de reconstruir la derivada real en oposición a un múltiplo de la misma. Además de determinar el mayor N, de tal manera que a_N^w sea cero, se pueda determinar el comportamiento del error asintótico del filtro para disminuir la distancia entre muestras T. Finalmente el error r (resto) nos da una indicación del error absoluto de ese filtro. Para tener una visión más detallada de diseño de filtro usando este método ver[**27**].

Al utilizar el método anterior obtenemos un filtro w, el cual está compuesto por coeficientes de polinomios (generalmente) que cumplieron las condiciones previas tomadas en la elaboración del filtro. Ahora bien, para obtener un filtro numérico necesitamos muestrear el filtro w, para poder usarlo en la interpolación o en el cálculo derivativo. En dicho proceso de muestreo del filtro w, se deben tener en cuenta dos condiciones: Primero, la cantidad de puntos a muestrear debe ser impar y separados a distancia constante θ. Segundo, cada punto tiene que ser clasificado según el pedazo de filtro W en el cual le corresponda; a estos pedazos o sub filtros de W los llamaremos w_k . Esta última condición nos permite saber que sub filtro w_k evaluar a cada punto, pues tenemos que transformar cada punto al espacio $[0,1]$ en donde están definidos los sub-filtros w_k y una vez allí realizar el muestreo del filtro de forma satisfactoria. Todo este proceso se explica con el siguiente ejemplo:

Supongamos que tenemos un filtro W de primera derivada definido dentro del intervalo $[-2,2]$ compuesto por 4 sub filtros polinomiales w_k; cada w_k abarca una unidad del intervalo y están definidos por los siguientes polinomios: $w_{-2} = \frac{1}{2}\tau^2$, $w_{-1} = -\frac{3}{2}\tau^2 + \tau + \frac{1}{2}$, $w_0 = \frac{3}{2}\tau^2 - 2\tau$ y $w_1 = -\frac{1}{2}\tau^2 + \tau - \frac{1}{2}$. Los sub filtros w_k se encuentran definidos en el intervalo $[0,1]$ obligándonos a trasladar cada unidad de filtro w_k al espacio que le corresponden en el filtro w; esto puede verse claramente en la figura 1.9b en donde se puede notar que cada w_k ocupa exactamente una unidad para formar el filtro W.

Armado el filtro W tenemos ahora que realizar el muestreo del filtro para obtener el Kernel. Como se dijo el número de muestras debe ser impar (para generar un kernel que se pueda centrar) además de tener una distancia constante de muestro. En este ejemplo vamos a calcular 9 muestras en el intervalo donde está definido el filtro W ([-2,2]), ocasionando entonces que la distancia de muestreo θ sea igual a 0,5. Sabiendo ya la distancia de muestreo θ se procede a calcular el valor de las 9 muestras empezando por el límite inferior -2 en donde se encuentra definido el filtro W. El muestreo se puede realizar de dos formas distintas: La primera forma consiste en armar el filtro W colocando cada sub filtro w_k en el pedazo de espacio que le corresponde en el filtro W, originando un traslado de los sub filtros polinomiales w_k para luego muestrear. La segunda forma consiste en trasladar cada uno de los puntos de muestreo del espacio original del filtro W al sub espacio $[0,1]$ en el que están definidos los sub filtros w_k sabiendo ya en cual w_k evaluar cada punto. La segunda manera de muestrear es la que usaremos por ser sencilla y practica de aplicar. Entonces los 9 puntos muestreados a una distancia de 0,5 serían los siguientes:

$$P = \begin{bmatrix} \mathrm{P}_1 \\ \mathrm{P}_2 \\ \mathrm{P}_3 \\ \mathrm{P}_4 \\ \mathrm{P}_5 \\ \mathrm{P}_6 \\ \mathrm{P}_7 \\ \mathrm{P}_8 \\ \mathrm{P}_9 \end{bmatrix} = \begin{bmatrix} -2 \\ -1{,}5 \\ -1 \\ -0{,}5 \\ 0 \\ 0{,}5 \\ 1 \\ 1{,}5 \\ 2 \end{bmatrix}$$

Ahora revisamos que sub filtro le corresponde a cada punto, arrojándonos como resultado que al sub filtro w_{-2} le corresponde los puntos P_1 y P_2, a w_{-1} los puntos P_3 y P_4, a w_0 los puntos P_5y P_6, y por último a w_1 le corresponde los puntos P_7, P_8 y P_9 (ver figura 1.9b). Sabiendo esto se procede a trasladar los puntos al espacio respectivo de cada sub filtro para luego evaluarlos. El traslado de los puntos se logra restando el límite inferior LI_k de cada sub filtro w_k definido en el filtro W (ver figura 1.9a) por el punto a evaluar P_i, es decir $P_i' = P_{\ i} - LI_k$ en donde i representa la cantidad de puntos, en este caso $i = 9$. En nuestro ejemplo los puntos trasladados P' serían los siguientes:

$$P' = \begin{bmatrix} \mathrm{P}_1' \\ \mathrm{P}_2' \\ \mathrm{P}_3' \\ \mathrm{P}_4' \\ \mathrm{P}_5' \\ \mathrm{P}_6' \\ \mathrm{P}_7' \\ \mathrm{P}_8' \\ \mathrm{P}_9' \end{bmatrix} = \begin{bmatrix} 0 \\ 0{,}5 \\ 0 \\ 0{,}5 \\ 0 \\ 0{,}5 \\ 0 \\ 0{,}5 \\ 1 \end{bmatrix}$$

Ahora si podemos evaluar cada punto trasladado P' en su w_k respectivo para obtener nuestro kernel ϑ de primera derivada, en este ejemplo se evaluaría de la siguiente manera:

$$P' = \begin{bmatrix} \mathrm{P}_1' \\ \mathrm{P}_2' \\ \mathrm{P}_3' \\ \mathrm{P}_4' \\ \mathrm{P}_5' \\ \mathrm{P}_6' \\ \mathrm{P}_7' \\ \mathrm{P}_8' \\ \mathrm{P}_9' \end{bmatrix} = \begin{bmatrix} 0 \\ 0{,}5 \\ 0 \\ 0{,}5 \\ 0 \\ 0{,}5 \\ 0 \\ 0{,}5 \\ 1 \end{bmatrix}, \qquad \text{Filtro } W = \begin{bmatrix} w_{-2} \\ w_{-1} \\ w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} \tau^2 \\ -\frac{3}{2}\tau^2 + \tau + \frac{1}{2} \\ \frac{3}{2}\tau^2 - 2\tau \\ -\frac{1}{2}\tau^2 + \tau - \frac{1}{2} \end{bmatrix}$$

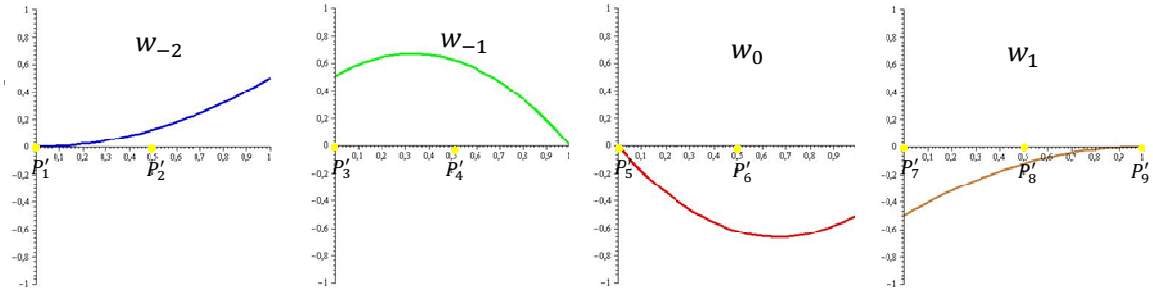
Evaluamos los P_i' en los w_k correspondientes y obtenemos el kernel ϑ.

$$\vartheta = \begin{bmatrix} w_{-2}(\mathrm{P}_1') \\ w_{-2}(\mathrm{P}_2') \\ w_{-1}(\mathrm{P}_3') \\ w_{-1}(\mathrm{P}_4') \\ w_0(\mathrm{P}_5') \\ w_0(\mathrm{P}_6') \\ w_1(\mathrm{P}_7') \\ w_1(\mathrm{P}_8') \\ w_1(\mathrm{P}_9') \end{bmatrix} = \begin{bmatrix} \mathrm{P}_1'^2 \\ \mathrm{P}_2'^2 \\ -\frac{3}{2}\mathrm{P}_3'^2 + \mathrm{P}_3' + \frac{1}{2} \\ -\frac{3}{2}\mathrm{P}_4'^2 + \mathrm{P}_4' + \frac{1}{2} \\ \frac{3}{2}\mathrm{P}_5'^2 - 2\mathrm{P}_5' \\ \frac{3}{2}\mathrm{P}_6'^2 - 2\mathrm{P}_6' \\ -\frac{1}{2}\mathrm{P}_7'^2 + \mathrm{P}_7' - \frac{1}{2} \\ -\frac{1}{2}\mathrm{P}_8'^2 + \mathrm{P}_8' - \frac{1}{2} \\ -\frac{1}{2}\mathrm{P}_9'^2 + \mathrm{P}_9' - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} (0)^2 \\ (0{,}5)^2 \\ -\frac{3}{2}(0)^2 + 0 + \frac{1}{2} \\ -\frac{3}{2}(0{,}5)^2 + 0{,}5 + \frac{1}{2} \\ \frac{3}{2}(0)^2 - 2(0) \\ \frac{3}{2}(0{,}5)^2 - 2(0{,}5) \\ -\frac{1}{2}(0)^2 + 0 - \frac{1}{2} \\ -\frac{1}{2}(0{,}5)^2 + 0{,}5 - \frac{1}{2} \\ -\frac{1}{2}(1) + 1 - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0{,}125 \\ 0{,}5 \\ 0{,}625 \\ 0 \\ -0{,}625 \\ -0.5 \\ -0{,}125 \\ 0 \end{bmatrix}$$

El kernel ϑ representa a un kernel de tamaño 9 de primera derivada y sería el siguiente:

$$\vartheta = \begin{bmatrix} 0 \\ 0{,}125 \\ 0{,}5 \\ 0{,}625 \\ 0 \\ -0{,}625 \\ -0.5 \\ -0{,}125 \\ 0 \end{bmatrix}$$

a) Sub filtros w_k definidos en $[0,1]$



b) Filtro W definido en $[-2,2]$

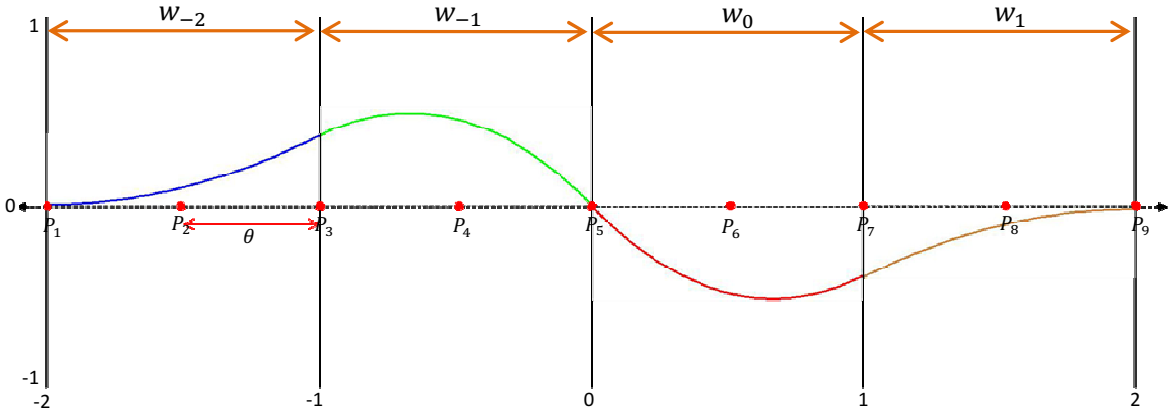


Figura 1.9: En la figura se muestra el filtro *w* y sus sub filtros w_k que lo componen, además de mostrar los puntos P_i y los puntos trasladados P_i' .

Como se vio en el ejemplo al muestrear el filtro W generamos un kernel o filtro unidimensional discreto el cual puede ser un kernel de interpolación o derivativo.

Obtenido el kernel unidimensional procedemos a componer el filtro final tridimensional, el cual usaremos al interpolar o derivar el volumen; para lograr dicho fin es necesario conocer de antemano qué tipo de reconstrucción de desea aproximar, pues en el proceso de construcción del filtro final 3D se necesitan a lo sumo dos kernels unidimensionales de acuerdo al Filtro 3D que deseemos construir, ya sea un Filtro 3D de interpolación o derivativo. En el caso que necesitemos construir un filtro 3D de interpolación sólo nos basta conocer el kernel

unidimensional de interpolación; en cambio sí queremos construir un filtro 3D derivativo necesitamos además del kernel de interpolación el kernel unidimensional de la derivada respectiva que deseamos reconstruir.

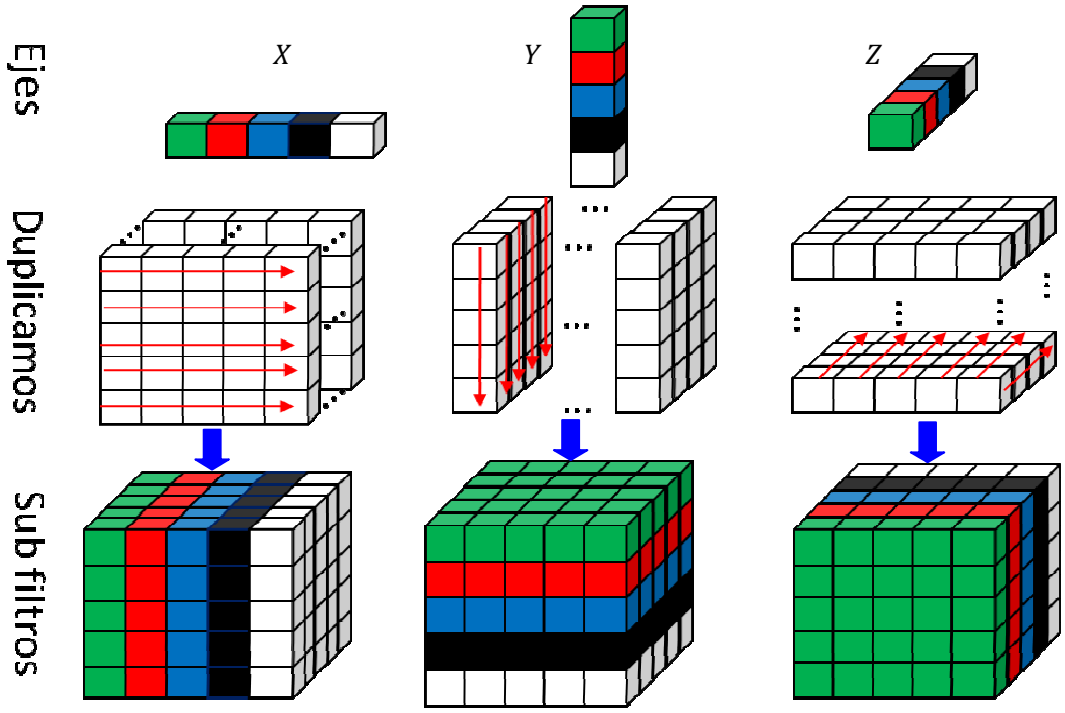


Figura 1.10: En la figura de arriba hacia abajo se puede observar el proceso de construcción de un filtro 3D según su eje, partiendo de un kernel unidimensional.

El proceso de construcción del filtro final 3D se resume en dos pasos:

a. Se construyen tres sub filtros tridimensionales los cuales representan los 3 ejes de coordenadas (x, y, z), utilizando los kernels unidimensionales obtenidos del muestreo del filtro w; estos sub filtros 3D son construidos duplicando el kernel unidimensional en dirección al eje correspondiente hasta formar un cubo (ver figura 1.10). El kernel unidimensional seleccionado por cada eje dependerá del filtro final 3D que se desea construir; es decir, si desearamos construir un filtro 3D para generar la primera derivada parcial con respecto a Y $(\frac{df}{dy})$, construiríamos tres sub filtros pero usando diversos kernels unidimensionales, para el sub filtro del eje X y Z utilizaríamos el kernel de interpolación mientras que para sub filtro del eje Y utilizaríamos el kernels unidimensional de primera derivada (ver figura 1.11b).

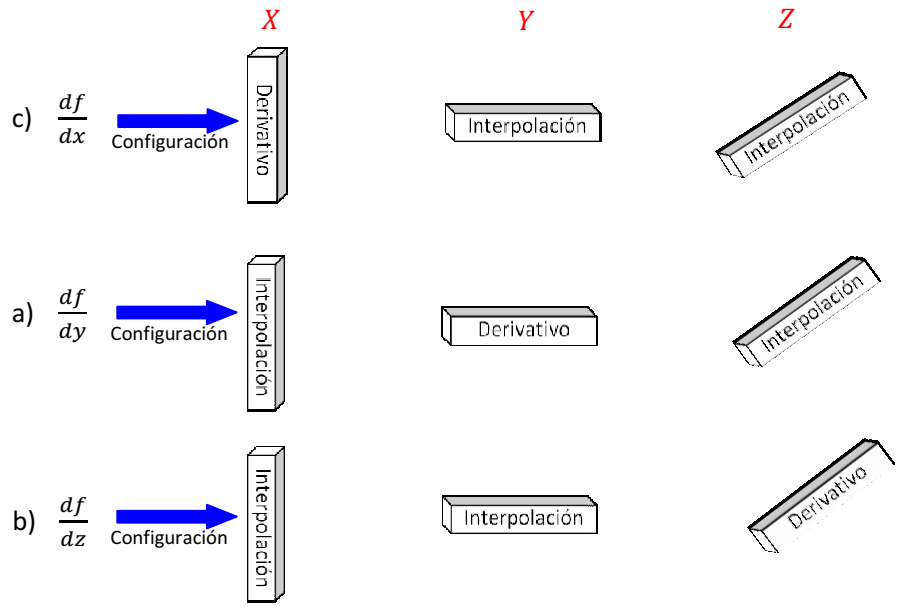


Figura 1.11: En la figura se pude observar la configuración de los kernels unidimensionales de interpolación o derivativo para obtener los filtros 3D de primera derivada parcial por cada eje.

b. Se realiza una multiplicación punto a punto de los tres sub filtros (A^x, A^y, A^z) y de esta manera obtendríamos el filtro Final R. Es decir, $R_{i,j,k} = A^x_{i,j,k} \cdot A^y_{i,j,k} \cdot A^z_{i,j,k}$ donde i, j y k denotan las dimensiones de los filtros (ver figura 1.12).

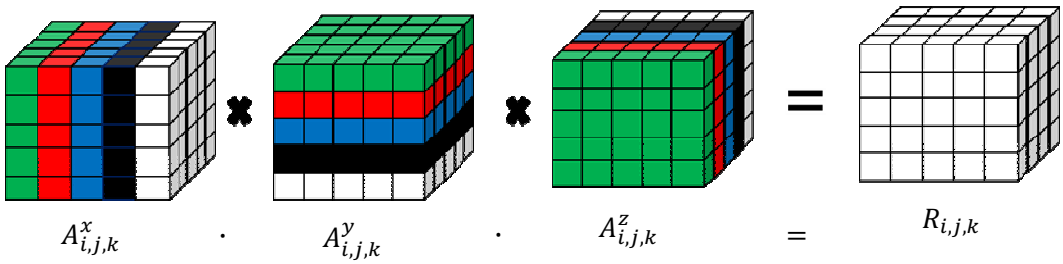


Figura 1.12: En la figura observa la multiplicación de los sub filtros A para obtener el filtro final R.

Los métodos de reconstrucción estudiados nos permiten obtener del volumen sus derivadas parciales. Estas derivas parciales pueden utilizarse para varios propósitos en el despliegue de volúmenes, como en la iluminación y la clasificación. El proceso de clasificación se describe en el siguiente capítulo, especificando en detalle la clasificación con funciones de transferencias.

Capítulo II

Funciones de Transferencia

En la visualización de volumen es primordial: clasificar, separar, o resaltar distintas zonas de interés que están parcial o totalmente obscurecidas dentro del volumen. De aquí la utilidad del despliegue de volumen. Para un observador, sea un médico o científico, es de vital importancia observar del volumen la zona de interés con total claridad.

Debido a la gran importancia de la clasificación, se han desarrollado bastantes investigaciones en esta área, permitiendo aislar con claridad la característica de interés para su correcta visualización[**23**][**28**][**29**][**30**][**31**][**32**]. Esto generalmente es alcanzado a través del uso de funciones de transferencia.

En el presente capítulo, se estudia los distintos tipos de funciones de transferencia, así como los mecanismos que orientan la configuración de la función de transferencia, ayudando al usuario a encontrar la función de transferencia más adecuada a su necesidad.

2.1 Clasificación

La utilidad de la visualización de volumen está dada por la capacidad de descubrir y enfatizar características de interés contenidas en él, así como ocultar los componentes que son de poco interés. A este proceso se le conoce como clasificación.

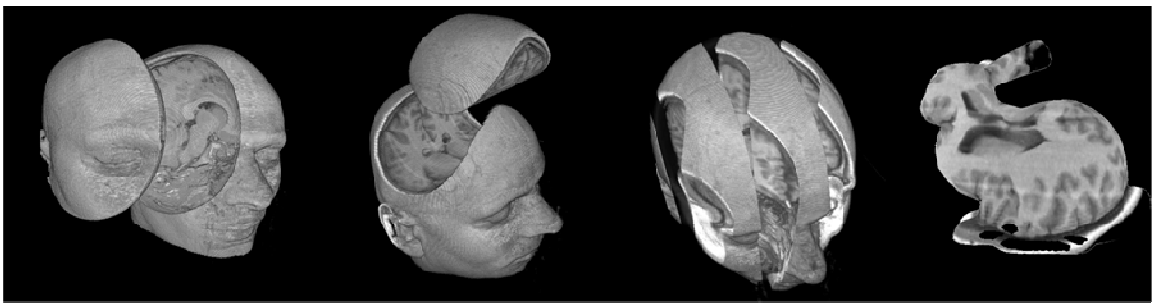


Figura 2.1: Imágenes de volúmenes clasificados por medio de planos y primitivas de corte [**33**].

Hay diferentes formas de realizar la clasificación. *Weiskopt et al.* [**33**] sugieren el uso de planos y primitivas de cortes (ver Figura 2.1), proporcionando herramientas para borrar o desplazar los componentes del volumen que son de poco interés.

En otro sentido, a través de la manipulación de las propiedades ópticas podemos enfatizar las características de interés u ocultar las que no lo son. Mediante la segmentación se puede lograr este objetivo. En este caso, el volumen puede ser particionado en sus estructuras específicas que lo conforman; así, se le asigna distintas propiedades ópticas (opacidad y color principalmente) a las distintas partes del volumen [**15**].

La manera más común de realizar esta asignación de las propiedades ópticas es mediante el uso de funciones de transferencia. Las funciones de transferencia son las encargadas de realizar el mapeo entre los vóxeles y las propiedades ópticas. En la sección 2.2 se estudia a detalle las funciones de trasferencia y se muestran los trabajos realizados sobre el tema.

Tzeng et al.[**34**] proponen un sistema inteligente donde se permite la aplicación de funciones de clasificación de alta dimensión, liberando al usuario de operar el sistema de visualización en el dominio de la función de transferencia. Por medio de una interfaz basada en dibujo (*Painting-based interface*), permite que el usuario seleccione directamente en el dominio de los datos las zonas de mayor y menor interés. Con esta información el sistema usa técnicas de inteligencia artificial para lograr la clasificación e inmediatamente refresca en la interfaz el volumen clasificado (ver Figura 2.2). Este enfoque permite la reutilización de la clasificación en volúmenes similares.

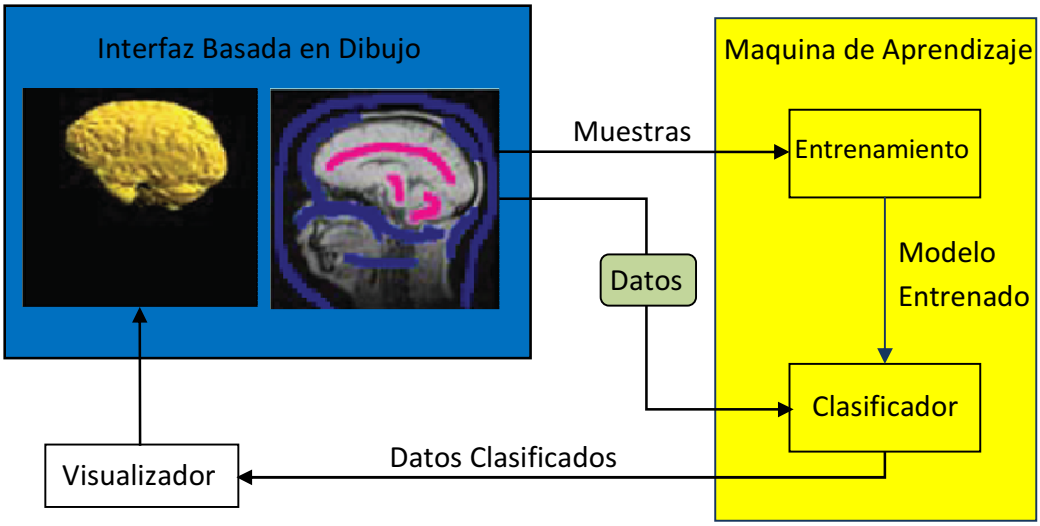


Figura 2.2: Proceso de clasificación usando la interfaz basada en dibujo[**34**].

En la segmentación y en las funciones de transferencia la clasificación se basa en la manipulación de las propiedades ópticas. Estas propiedades ópticas deben de una u otra forma ser asignadas al volumen para lograr su clasificación. Existen tres formas de realizar esta tarea las cuales son: pre-clasificación, post-clasificación y la clasificación pre-integrada.

2.1.1 Pre-clasificación y Post-clasificación

La asignación de las propiedades ópticas elegidas se puede realizar antes o después de interpolar los vóxeles. El orden en el cual es realizada esta asignación afecta directamente la calidad de la imagen e incrementa el cómputo.

La pre-clasificación asigna las propiedades ópticas (obtenidas de la función de transferencia) antes de interpolar los vóxeles [**35**]. Esta forma de clasificación reduce las altas frecuencias presentadas en la función de transferencia, produciendo una imagen de baja calidad, pero permite un despliegue rápido del volumen. En cambio en la post-clasificación, se aplica la función de transferencia, luego de interpolar los vóxeles, produciendo mejores resultados en funciones de transferencia lineales. En la Figura 2.3 se observa claramente el resultado de las dos técnicas para la componente roja $r(s)$ de la tupla de color $RGBA$ ($R =$ *red*/rojo, $G =$*green*/verde, $B =$*blue*/azul y $A =$*alpha*/alfa que indica el nivel de transparencia).

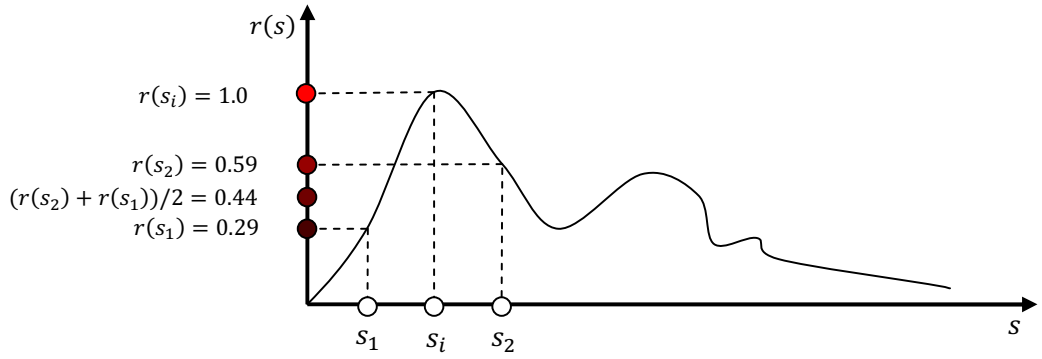


Figura 2.3: Ejemplo gráfico del uso de pre-clasificación versus post-clasificación. En la pre-clasificación se realiza la interpolación entre las muestras clasificadas s_1 y s_2 originando una intensidad de 0.44, atenuando la intensidad real, la cual se puede observar en la post-clasificación donde se clasifica s_i después de su interpolación entre las muestras s_1 y s_2 generando una intensidad de 1.0 para el color rojo.

2.1.2 Clasificación Pre-integrada

Los tipos de clasificación anteriores presentan problemas cuando la función de transferencia no es suave, es decir, cuando la función de transferencia presenta altas frecuencia. Aunque con la post-clasificación se obtienen resultados superiores que con la pre-clasificación, es necesario aumentar considerablemente la tasa de muestreo en el rayo cuando la función de transferencia tiene altas frecuencias. Esto acarrea un aumento considerable en el tiempo de cómputo. De lo contrario la visualización será errónea, produciendo bandas en vez de una superficie continua[**35**].

La clasificación pre-integrada soluciona este problema. Permite clasificar segmentos de rayos en vez de una muestra individual. La clasificación pre-integrada, consiste en clasificar los segmentos de los rayos ubicados entre cada par de muestras s_f y s_b, pre-calculando la integral de la función de transferencia entre cada par de muestras posible (ver Figura 2.4a) y almacenando dicho resultado en una textura 2D (ver Figura 2.4b) donde las coordenadas de acceso (x, y) corresponden a las muestras (s_f, s_b) [**35**].

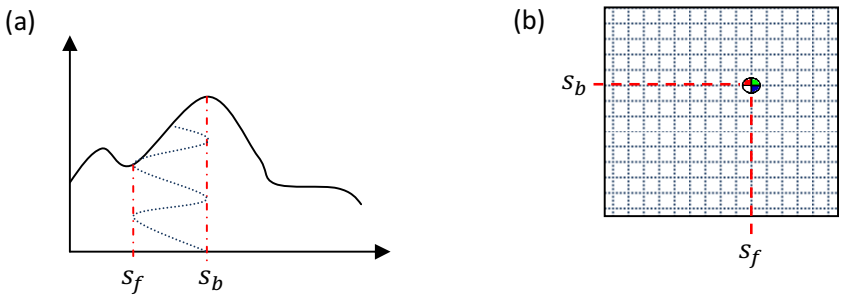


Figura 2.4: Clasificación pre-integrada: (a) representa el intervalo entre un par de muestras en la función de transferencia, (b) representa la textura que almacena las integrales pre-calculadas y son accedidas usando las muestras que delimita el intervalo de integración de la función de transferencia.

2.2 Funciones de Transferencia (FT)

La función de transferencia es una función continua (generalmente lineal definida a trozos) responsable de determinar las propiedades ópticas de los vóxeles (ver Figura 2.5) [**30**][**2**]. En otras palabras, la función de transferencia es la encargada de realizar el mapeo entre los vóxeles y las propiedades ópticas, llevando los vóxeles del domino físico original hacia el dominio espectral de colores y transparencias[**2**].

Nosotros podemos ver entonces, a la función de transferencia como una aplicación del producto cartesiano de las características $(S_1, S_2, \dots, S_n)$ del conjunto de vóxeles S hacia el producto cartesiano de las propiedades ópticas O:

$$f: S_1 \times S_2 \times \cdots \times S_n \rightarrow O_1 \times O_2 \times \cdots \times O_m \qquad [Ec.\,2.1]$$

Los valores de n y m son pequeños, y determinan la dimensionalidad del dominio (n) y rango (m) de la función de transferencia; además, nos permiten distinguir los distintos tipos de funciones de transferencias.

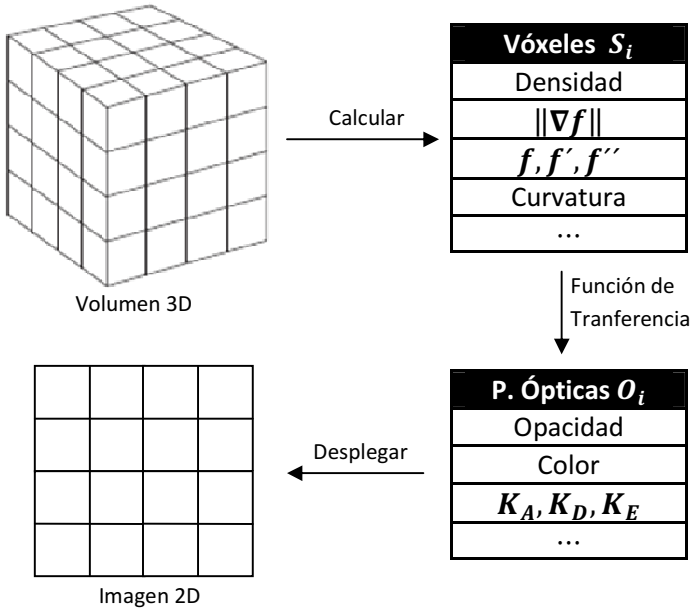


Figura 2.5: Proceso general de clasificación, del volumen se obtiene información (dominio) y utilizando esta información se procede a la asignación de las propiedades ópticas (rango) a través de la función de transferencia, para luego generar el despliegue.

A continuación presentamos diferentes tipos y variantes de funciones de transferencia utilizados en el despliegue de volumen.

2.2.1 Funciones de transferencias de una dimensión (1D)

Es el tipo de función de transferencia más simple, donde el dominio de la función está dado por la densidad de los vóxeles S y el rango es la opacidad α, es decir $f(S) = \alpha$ [**36**]. En el despliegue del volumen, las propiedades ópticas del vóxel se determinan por los valores de opacidad especificados en la función de transferencia. Sólo características importantes pueden recibir alta opacidad, así, no suelen ser obstruidas por las regiones de poco interés (ver Figura 2.6).

El principal rol que juega la opacidad, es hacer inteligible la visualización del volumen. A este particular tipo de función de transferencia se le conoce como función de opacidad [**36**].

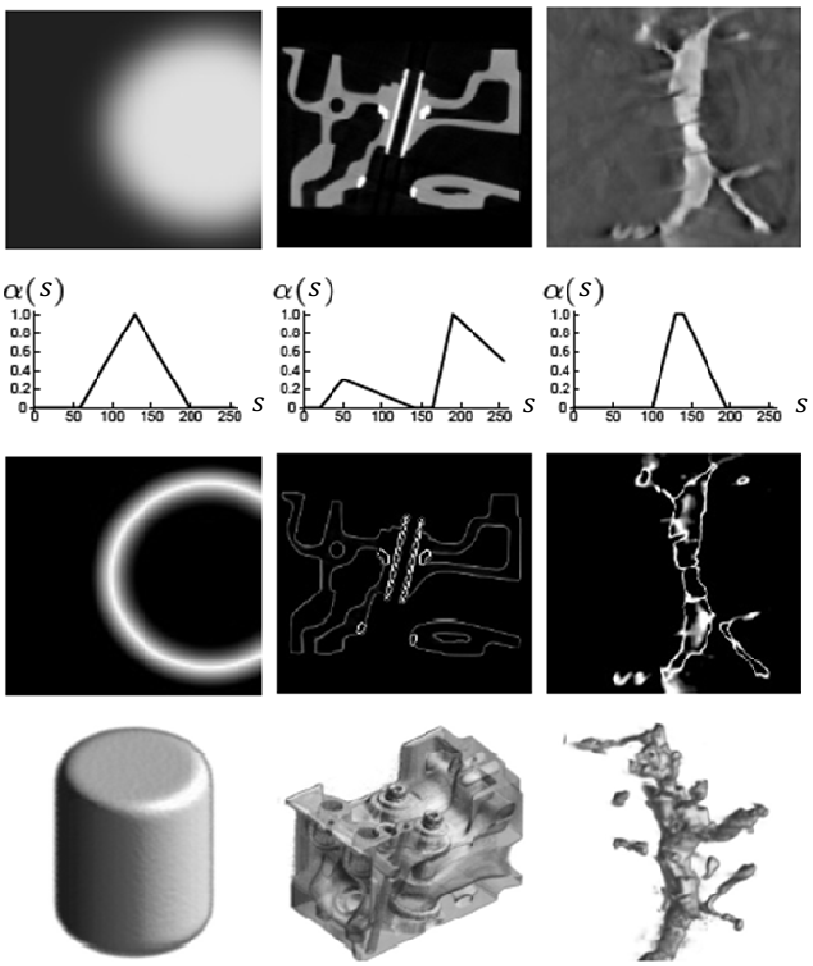


Figura 2.6: De arriba hacia abajo se pueden observar las muestras originales, las funciones de opacidad utilizadas, las muestras clasificadas y por último el despliegue de los volúmenes[**36**].

Las funciones de opacidad pueden ser generalizadas a diferentes tipos de funciones de transferencia, aumentando la dimensión del rango de la función. La dimensión del rango comúnmente incluye color, permitiendo distinguir visualmente las transiciones entre las estructuras. En general, cualquier propiedad óptica puede ser representada y compuesta en el hardware gráfico, pudiéndose incluir en la dimensión del rango de la función de transferencia propiedades como: opacidad, emisión, *scattering*[**37**], parámetros del *shading*, texturizado [**38**] e incluso el índice de refracción [**39**]. Al incluir la opacidad y emisión, el rango de la función de transferencia es una tupla RGBA, en donde RGB representa la emisión o color, y A representa su opacidad.

Note que aunque en estas funciones de transferencia se puede aumentar la dimensionalidad del rango, presentan siempre una única dimensión en su dominio, dado por la densidad de los vóxeles. Por eso también se le conocen como funciones unidimensionales (1D).

2.2.2 Funciones de transferencia multidimensionales

Las funciones de transferencia multidimensionales son aquellas que presentan más de una dimensión en su dominio. Así, para estas funciones, el valor de n es mayor a uno (ver Ec. 2.1). Estas funciones permiten clasificar las muestras basadas en una combinación de valores. Múltiples valores de los vóxeles tiende a incrementar la probabilidad de aislar unívocamente una característica en el dominio de la función, proporcionando un gran vocabulario para expresar las diferencias entre las estructuras contenidas en el volumen [**31**]. Estos valores, son los ejes de la función de transferencia y representan las distintas dimensiones presentes en el dominio de la función.

Existen diferentes propuestas para expandir la dimensionalidad del dominio de la función de transferencia. La manera más común de expandir la función de transferencia es usando la magnitud del gradiente $\|\nabla f\|$ como una segunda dimensión en la función de transferencia [**31**][**40**][**9**]. La magnitud del gradiente es una cantidad escalar, la cual nos describe la tasa de cambio local del vóxel, es decir, nos permite observar la rapidez con la cual cambian los valores con respecto a sus vecinos (valores adyacentes), pudiéndose distinguir las regiones homogéneas y las regiones de transición. Esto fue visto por *levoy*[**9**] en 1988, donde propuso dos estilos de función de transferencia, en ambos usando la magnitud del gradiente como segunda dimensión [**9**].

Los dos estilos de FT que presenta *levoy*[**9**] son FT de dos dimensiones, utilizando el valor de los vóxeles y la magnitud del gradiente en sus dimensiones, aunque los dos estilos presentan las misma dimensiones estas son utilizadas para diferentes propósitos. La primera variante de FT tiene como propósito desplegar las fronteras entre los materiales, fronteras que son de mucho interés en los datos médicos. Mientras que en la segunda variante de FT el propósito es permitir el despliegue de los contornos de superficie de forma coherente entre la diversidad de datos que componen el volumen.

Kindlmann et al.[**40**] usan además de la magnitud del gradiente, la segunda derivada direccional a lo largo del vector gradiente, para lograr la generación semi-automática de funciones de transferencia de una y dos dimensiones. Ellos utilizando el valor de los vóxeles, la magnitud del gradiente y la segunda derivada direccional a lo largo del vector gradiente, componen un volumen el cual utilizan para visualizar los límites entre los materiales de una manera más exacta. Luego *Kniss et al.*[**31**] basándose en este enfoque proponen la creación de una función de transferencia multidimensional utilizando el volumen de clasificación propuesto por *Kindlmann et al.*[**40**] en la generación semi-automática de funciones de transferencia 1D y 2D. Esta variante de función de transferencia la explicaremos con detalle más adelante en el documento. En cambio, *Lum y Ma* [**41**] usan el vector gradiente para generar una función que permite modular la iluminación entre los vóxeles, la cual llamaron función de transferencia de iluminación (*Lighting Transfer Function*).

Otros trabajos, usan otros tipos información derivada de los datos para lograr distinguir las características en el volumen de acuerdo a la particularidad de su forma y curvatura.

Hladuvka et al. [**30**] basándose en la geometría diferencial, usa las direcciones principales (vectores tangentes) y las curvaturas principales para describir la vecindad de cualquier punto en una superficie regular. De esta manera, para un punto específico P de una superficie regular, las direcciones principales $\overrightarrow{s_1}$ y $\overrightarrow{s_2}$ dan una idea de donde la superficie se inclina más o se inclina menos respectivamente. La correspondiente medida cuantitativa es expresada por las curvaturas principales k_1 y k_2, los cuales son números reales y representan el cambio de la normal en esas direcciones. *Hladuvka et al.* [**30**] proponen el uso combinado de los valores k_1, k_2 para enfatizar o eliminar clases de formas específicas, por medio de su elección en la función de transferencia. Esto proporciona flexibilidad en la distinción de formas.

Diferentes investigaciones expanden el dominio de la función de transferencia, agregando una nueva dimensión para denotar la importancia de los objetos. Esto es logrado a través de la asignación de la importancia por parte del usuario con ayuda del diagrama de dispersión (*scatterplot*), modulando la opacidad de acuerdo a la importancia [**42**]. Esto es típicamente utilizado en la visualización ilustrativa para resaltar las características de interés sin necesidad de quitar todo el contexto.

Zhou et al.[**43**]*,* para cumplir con el requerimiento anterior, proponen expandir la dimensionalidad del dominio de la función de transferencia usando la información de la posición del vóxel. Con esto no solo se logra resaltar las características de interés sin alterar el contexto, sino que también se puede usar como herramienta de corte en el volumen (ver Figura 2.7).

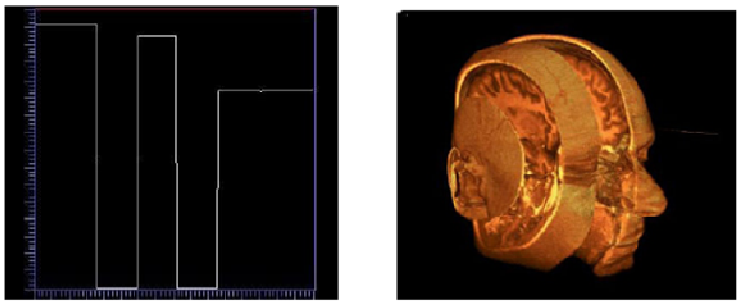


Figura 2.7: Función de transferencia usada para cortar el volumen propuesta por *Zhou et al*. [**43**]. La izquierda es la configuración de la función de corte a lo largo del volumen y la derecha el resultado sobre el volumen.

2.2.2.1 Función de Transferencia Tridimensional

Al aumentar la dimensión del dominio en la función de transferencia, también se incrementa la posibilidad de aislar la zona de interés, pues se incrementa el vocabulario para expresar las diferencias entre las estructuras contenidas en el volumen.

Kniss et al.[**31**] proponen una función de transferencia multidimensional de tres dimensiones, para mejorar la especificación de las características de interés en la clasificación de los vóxeles. Además, demostraron la aplicación de las funciones de transferencia multidimensionales para dos clases de datos distintos: para datos escalares y datos

multivariables. En este documento sólo hacemos mención de funciones de transferencias para datos escalares (vóxeles).

La primera medida derivada de los valores de los vóxeles es el gradiente. El gradiente es un vector que describe la dirección de mayor cambio. Además el gradiente normalizado es usado en el *shading* del volumen. La magnitud del gradiente nos describe la tasa de cambio local de los datos escalares (vóxeles). En ese sentido, va a representar la segunda dimensión del dominio de la función de transferencia tridimensional.

Para unificar nomenclatura *Kniss et al*.[**31**] usaron f' para referirse a la magnitud del gradiente de f, donde f es la función escalar que representa los vóxeles:

$$f' = \|\nabla f\| \qquad [Ec. 2.2]$$

Este valor como segundo eje en el dominio de la función de transferencia permite discriminar entre regiones homogéneas (bajo magnitud del gradiente) y regiones de transición (alto magnitud del gradiente). Esto puede ser visto claramente en la Figura 2.8. En la Figura 2.8a observamos el histograma 1D basado en los valores de los vóxeles, en donde se indican los tres materiales básicos en el *Chapel Hill CT Head*; aire (A), tejidos blandos (B), y hueso (C). En la Figura 2.8b vemos un histograma 2D producto de la mezcla de los valores de los vóxeles y la magnitud del gradiente. Algunos materiales son relativamente homogéneos, y en estos casos, la magnitud del gradiente es pequeña. Estos materiales pueden ser vistos en las regiones A, B y C del histograma 2D de la Figura 2.8b. En cambio, los límites entre los materiales son los arcos; la frontera entre aire y tejido blando (D), la frontera entre tejido blando y hueso (E) y la frontera entre aire y hueso (F). Estos límites entre los materiales pueden ser divididos usando una función de transferencia 2D basada en los valores de los datos y la magnitud del gradiente.

En la Figura 2.8c se puede visualizar un despliegue de volumen con los diversos materiales etiquetados. Cuando se trabaja con funciones de transferencia 1D, cada valor del vóxel engloba muchas características del volumen. De esta manera, ciertos detalles del volumen no pueden ser clasificados con una función de este tipo.

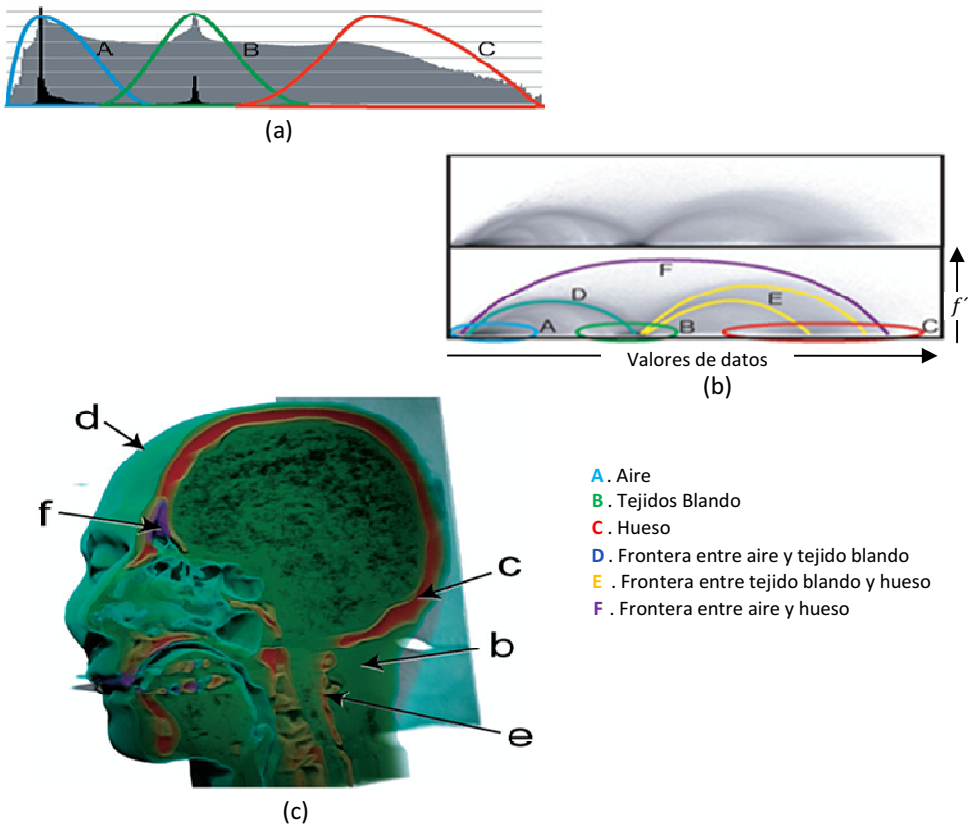


Figura 2.8: En la imagen (a) muestra un histograma 1D donde se observa los materiales básicos. En la imagen (b) se puede observar un histograma 2D compuesto por la primera derivada y las muestras, donde se observa los materiales básicos y las fronteras existentes entre ellos. Estos materiales y sus fronteras se pueden ver con claridad en el volumen de la imagen (c), que representa al *Chapel Hill CT Head* [**31**].

Frecuentemente, los arcos que definen la frontera entre los materiales en una FT 2D suele superponerse. En algunos casos esta superposición impide que los materiales puedan ser claramente aislados en la FT. Este efecto puede ser visto en la región circular del histograma 2D valor escalar/magnitud del gradiente del CT de un diente humano en la Figura 2.9a. La frontera dentina/fondo comparten algunos rangos de valores de datos y magnitudes de gradiente con porciones de pulpa/dentina (E) y la frontera entre fondo/esmalte (H). Entonces la frontera entre fondo/esmalte (H) es enfatizado en la FT 2D, ocasionando que las fronteras (E) y (H) sean erróneamente coloreadas en el despliegue de volumen (ver Fig. 2.9c).

Una segunda medida derivada de los vóxeles, permitiría más precisión en la especificación de fronteras complejas como la mencionada en el párrafo anterior. *Kniss et al*.[**31**] proponen el uso de la segunda derivada direccional a lo largo de la dirección del gradiente, la cual es una medida muy exacta pero es computacionalmente costosa. Esta solución incluye el uso de la matriz *hesiana* (**H**), la cual contiene las segundas derivadas parciales de f. El valor f'' denota la segunda derivada y está dada por la siguiente ecuación:

$$f'' = \frac{1}{\|\nabla f\|^2}(\nabla f)^{\mathrm{T}}\mathbf{H_f}\nabla f \qquad [Ec.\,3.3]$$

También se puede usar el *laplaciano* ó el *gradiente de la magnitud del gradiente* para aproximar la segunda derivada [**40**].

Un ejemplo de la distinción entre las fronteras de los materiales (utilizando la segunda derivada), puede ser visto en la Figura 2.9b, donde los límites (E), (F) y (G) no están sobrepuestos. Así, los materiales de interés pueden ser correctamente aislados.

En la Figura 2.9d se observa la imagen generada después de usar el histograma de la Figura 2.9b (que contiene la información de la segunda derivada) para clasificar los vóxeles.

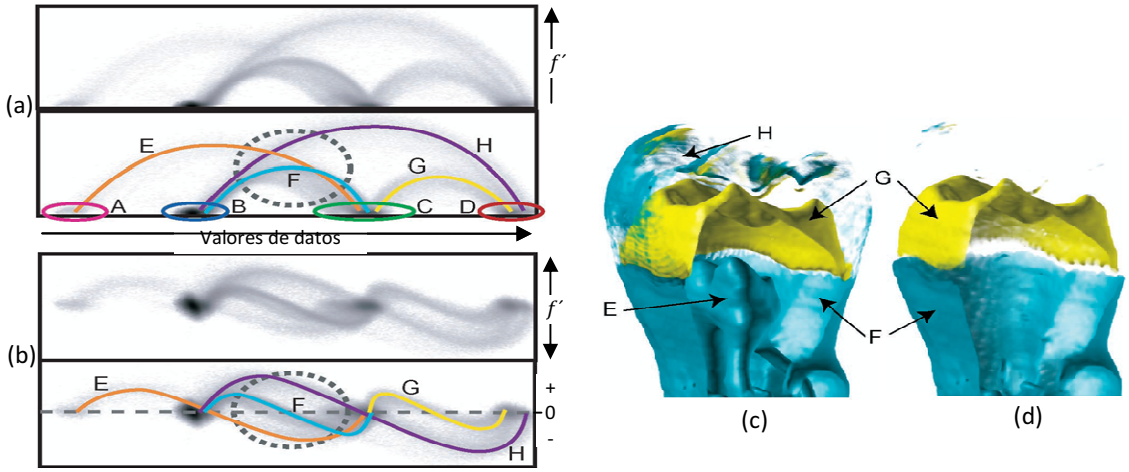


Figura 2.9: En la imagen (a) se muestra un histograma 2D producto de la primera derivada f' y los valores de los datos, donde se observa en el circulo punteado la superposición entre las fronteras de los materiales. En (b) se observa un histograma 2D producto de los datos y la segunda derivada f'', y se puede notar claramente en la línea punteada como las fronteras entre los materiales son divididas. En (c) se observa un despliegue erróneo al querer clasificar las fronteras G y F utilizando (a). En (d) se observa el despliegue producto de la correcta clasificación de G y F utilizando (b).

2.2.3 Diseño de las FT

La gran flexibilidad de configuración que posee la función de transferencia tiene ventajas y desventajas [**36**]. La principal ventaja, es que nos permite ser muy expresivos en la visualización del volumen, pues la imagen puede ser representada tomando en cuenta una gran variedad de aspectos en los datos. El problema es navegar en ese mar de variabilidad para poder encontrar una buena función de transferencia que nos permita representar las zonas de interés en el volumen. El método más común para lograr esta especificación de la función de

transferencia es el frustrante proceso de ensayo y error. Existen al menos tres razones por lo cual este método es frustrante [**36**]:

I. Las funciones de transferencia poseen un enorme número de grados de libertad. Es allí donde el usuario puede perderse. Incluso en una simple función 1D, todos los puntos de control agregados poseen dos grados de libertad.
II. Usualmente las interfaces para la configuración de la FT, no restringen o guían al usuario por el conjunto de datos. Esta falta de orientación hace que el usuario entre en un modo iterativo de ensayo y error, a la medida que visualiza el despliegue de acuerdo a los parámetros seleccionados en la TF.
III. Las funciones de transferencia son inherentemente no espaciales, en el sentido que su asignación de opacidad y color no incluyen la posición en el espacio como una variable en el dominio. Esto es frustrante si el usuario tiene interés en una característica de interés que es espacialmente localizada pero no distinguible en la TF.

El principal objetivo de la visualización de volumen es lograr una imagen, en donde se distinga con claridad la zona de interés. El método de ensayo y error consume mucho tiempo, además de generar frustración.

Afortunadamente, investigaciones recientes han explorado el problema del diseño de la función de transferencia, y han desarrollado nuevas interfaces para la especificación de la misma. Los métodos existentes que atacan el problema de diseño en las funciones de transferencias, se caracterizan por la forma como logran orientar la construcción de estas funciones. Los dos métodos existentes son los basados en la imagen y los basados en los datos, los cuales explicaremos a continuación.

2.2.3.1 Especificación de la FT basada en los datos

El principio de la especificación de la FT basada en los datos, es que la información derivada del conjunto de datos puede orientar al usuario hacia la configuración apropiada de la función de transferencia, o en realidad puede limitar el espacio de búsqueda de la función de transferencia en la interfaz de usuario hacia un subconjunto más prometedor [**36**].

Debido a que las funciones de transferencia son independientes de la posición, la información usada para su generación suele ser también independiente de la posición. Esto explica porque muchos métodos de especificación basados en los datos se fundamentan en los histogramas y su análisis, ya que los histogramas por su propia naturaleza acumulan información de manera independiente de la posición [**36**].

Muchos métodos basados en los datos para la especificación de la FT están realmente dirigidos a la determinación de isovalores para la extracción de isosuperficies. La determinación de isovalores es un problema similar a la especificación de la función de transferencia, es decir la eficacia de la visualización dependen absolutamente de los parámetros configurados, los cuales son aplicados uniformemente en todo el volumen.

Los métodos que ayudan a determinar isovalores son de mucha utilidad para la especificación de las funciones de opacidad, ya que proporcionan información sobre los valores de mayor importancia para mostrar la estructura de un conjunto de datos. Entonces en vez de asignar la máxima opacidad, como representando una isosuperficie, podemos dar un rango de opacidades distintas a una gama más amplia de valores, en función de su importancia relativa [**36**].

Una importante técnica para la determinación de isovalores es el espectro de contorno [**28**]. El espectro de contorno consiste en un conjunto de métricas calculadas sobre el volumen. Estas métricas muestran características importantes de una isosuperficie, tales como área de la superficie, volumen y la integral del gradiente. Sobre estos parámetros se definen un conjunto de funciones, que proporciona una herramienta útil para mejorar la consulta iterativa del conjunto de datos. Lo más importante es que dichas funciones se integran en la misma interfaz que se utiliza para establecer el isovalor. Al proporcionar una representación compacta y visual de los indicadores evaluados en el rango de isovalores posible, el usuario basándose en su necesidades puede decidir que isovalor usar (ver Figura 2.10).

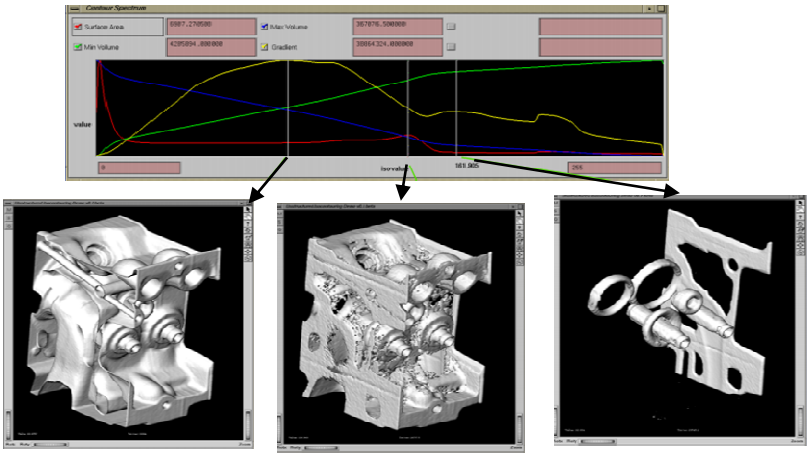


Figura 2.10: Interfaz de visualización de funciones y distintas visualizaciones de acuerdo al isovalor seleccionado[**28**].

El método semiautomático de *Kindlmann y Durkin* [**40**] se basa en el análisis de un histograma de tres dimensiones (*Histogram Volume*) que recoge la correlación existente en el volumen, entre el valor de la muestra, la magnitud del gradiente y la segunda derivada direccional a lo largo de la dirección del gradiente (ver Figura 2.11). Dichas correlaciones son extraídas en un mapa de distancias (*distance map*). Este mapa registra las relaciones entre los valores de las muestras y la proximidad a los límites entre materiales (ver Figura 2.12). Usando el mapa de distancia, el usuario puede interactivamente experimentar con diferentes configuraciones [**44**].

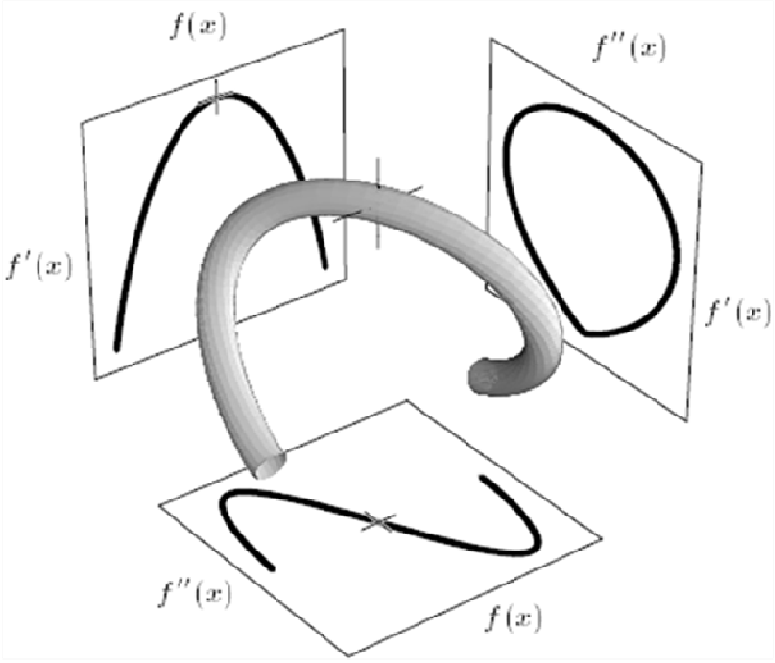


Figura 2.11: Relaciones entre el conjunto de datos f , la magnitud del gradiente f' y la segunda derivada direccional a lo largo de la dirección del gradiente f'' [**44**].

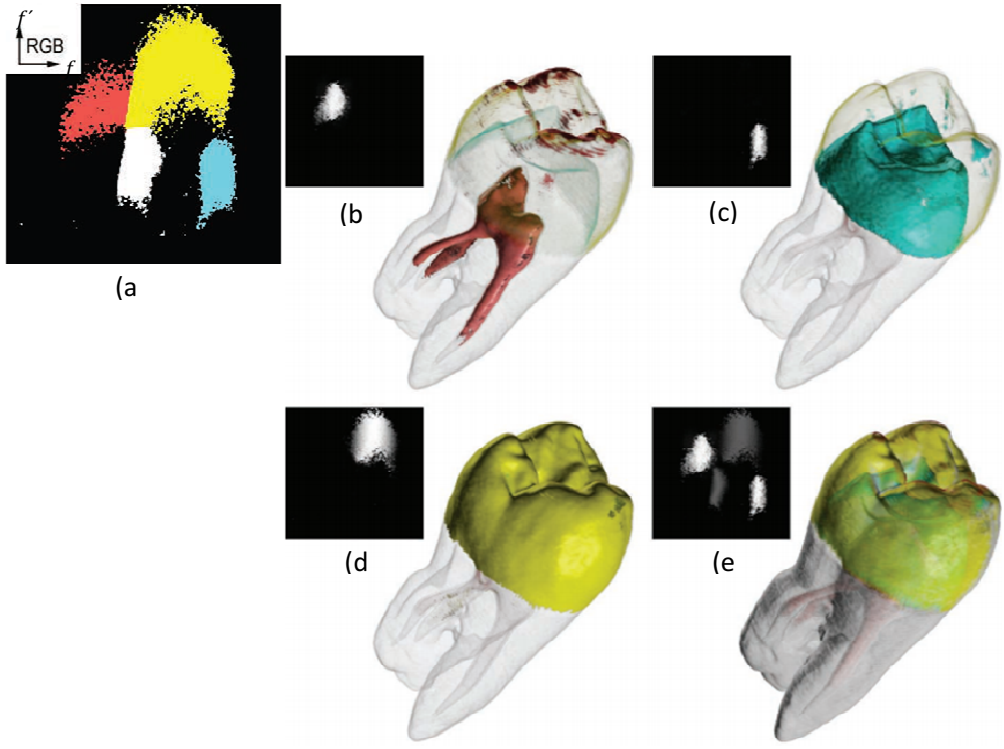


Figura 2.12: En (a) los distintos límites entre los materiales, y en (b), (c), (d), (e) distintas configuraciones de opacidades para los distintos límites entre materiales[**44**].

2.2.3.2 Especificación de la FT basada en la imagen

Los métodos basados en la imagen usan imágenes obtenidas de visualizaciones de volúmenes previas para navegar por el espacio de las funciones de transferencia. Este es un poderoso método para crear una interfaz más intuitiva, que permite al usuario encontrar directamente el mejor despliegue, sin tener que manipular directamente la función de transferencia.

El primer trabajo que uso este método fue el propuesto por *He et al.* [**29**], en donde se usaron algoritmos genéticos para producir una adecuada función de transferencia. Ellos proponen un sistema que genera aleatoriamente (o pre-define) un conjunto de funciones de transferencias, desplegando pequeñas imágenes de cada una de estas configuraciones de funciones de transferencia (ver Figura 2.13). Visualizando estos despliegues el usuario escoge los pre-despliegues del volumen de su mayor agrado, en base a estas selecciones una nueva población de funciones de transferencias son estocásticamente generadas. Este proceso es iterativo, hasta que el usuario encuentre una función de transferencia de su preferencia.

Alternativamente, métricas como entropía, varianza y energía son utilizadas en el proceso, como objetivo en una función de aptitud para evaluar las imágenes desplegadas sin orientación humana, y el proceso eventualmente converge en una función de transferencia que maximiza la función de aptitud.

El método tiene éxito en la generación de buenas representaciones, y libera al usuario de tener que editar manualmente la función de transferencia [**36**].

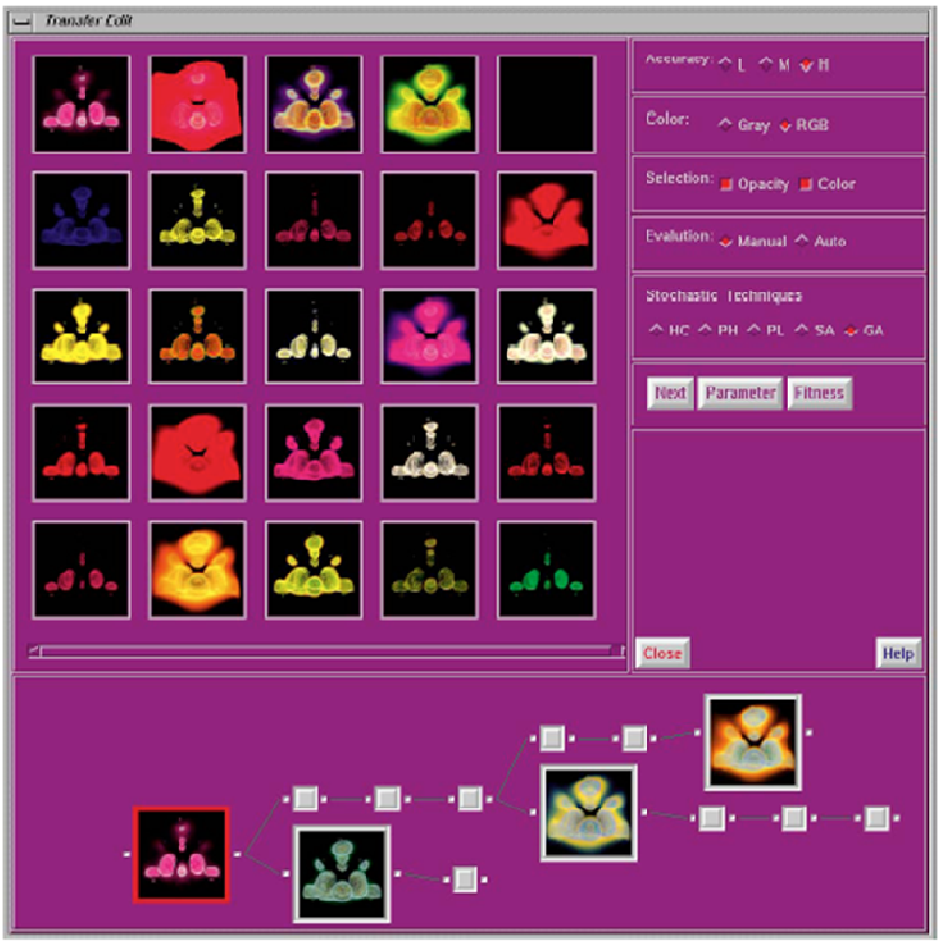


Figura 2.13: Interfaz usada para guiar al usuario, a través del uso de algoritmos genéticos [**29**].

El diseño de galería presentado por *Mark et al.*[**45**] aborda el problema de "*ajuste de parámetros*" en computación gráfica, con aplicaciones en problemas como: la ubicación de la fuente de luz para despliegue, control de movimiento para la animación articulada de figuras, y funciones de transferencia 1D en el despliegue de volumen[**36**]. Crea una interfaz intuitiva que abarca el espacio de todas las funciones de transferencia posibles, basándose en el análisis automatizado y el diseño de imágenes desplegadas (ver Figura 2.14). El diseño de galería de *Mark et al.* parametriza el espacio de todas las funciones de transferencia posibles y estocásticamente muestra los despliegues obtenidos según sus configuraciones . Si bien esto puede ser un proceso que consume tiempo, es totalmente automatizado [**23**].

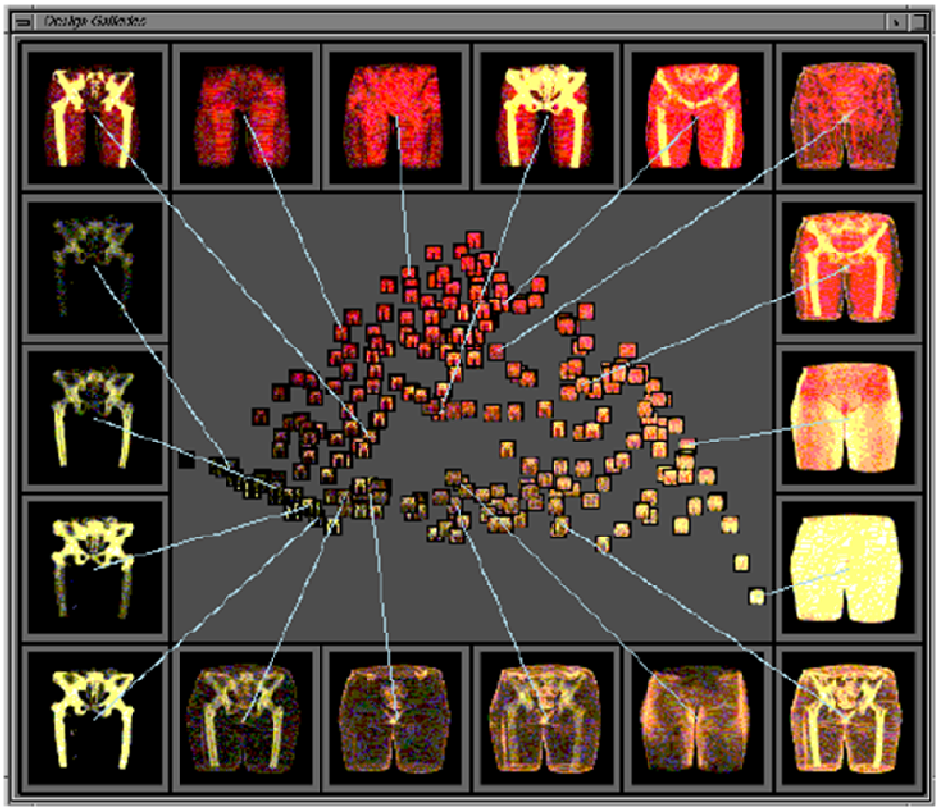


Figura 2.14: Interfaz del diseño de galería para la función de transferencia presentado por Mark et al.[**45**].

Konig y Groller [**46**] han utilizado un enfoque diferente para organizar las imágenes desplegadas, de tal forma de facilitar la búsqueda de buenas funciones de transferencia. La idea es mostrar pequeñas imágenes de rendering, ordenándolas según su relación con los espacios de las muestras escalares, color y opacidad. Además, proporcionan un conjunto de herramientas para cada espacio, permitiendo a los usuarios con diferentes niveles de experiencias elegir un enfoque adecuado según sus necesidades. Este tipo de editor puede verse en la Figura 2.15.

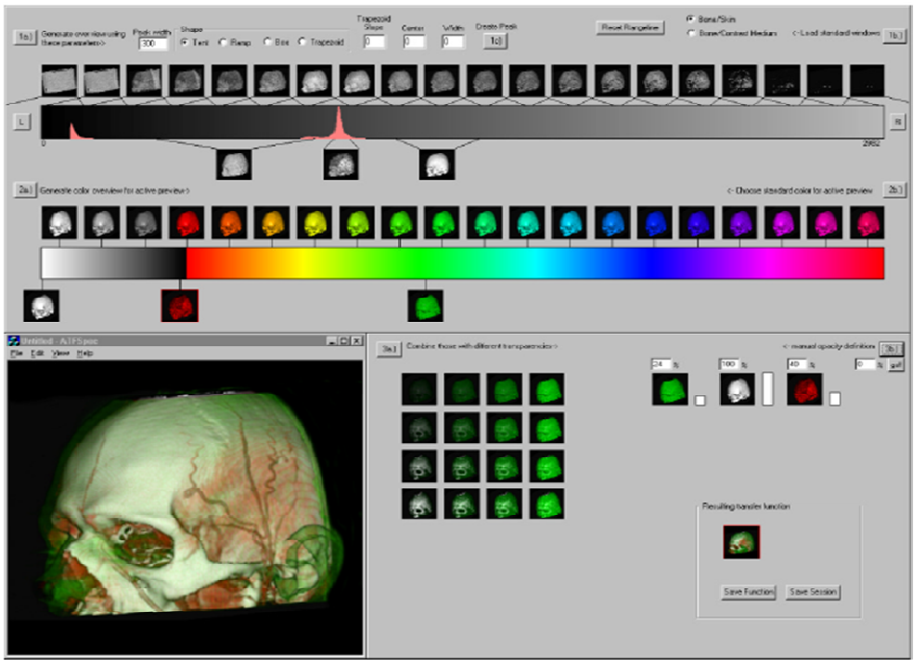


Figura 2.15: Interfaz de la función de transferencia guiada por iconos[**46**]

Kniss et al.[**32**][**36**] proponen un uso combinado de los métodos de especificación de la función de transferencia (basado en la imagen y datos), el cual llamaron *"dual-domain interaction"* (interacción dual de dominios). Ellos orientan la configuración de la FT relacionando la exploración del dominio espacial del volumen con su interacción en el dominio de la FT. A través del uso de un pincel 3D de exploración o haciendo clic en una imagen de corte, la función de transferencia es actualizada mostrando en su dominio los puntos consultados en la exploración espacial del volumen. Esto permite observar las relaciones existentes de los datos explorados entre el dominio espacial del volumen y el domino de la función de transferencia (ver Figura 2.16). Con estas relaciones, se obtiene una clasificación intuitiva en el dominio de la función de transferencia.

A fin de especificar estas herramientas de exploración y otras de clasificación, *Kniss et al.*[**32**][**36**] definen una variedad de widgets que permiten la manipulación directa de los datos, y de esta forma lograr la configuración de las propiedades ópticas del volumen.

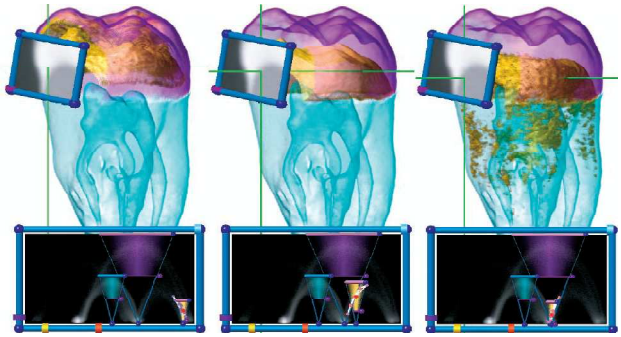


Figura 2.16: interacción dual de dominios (*Dual-domain interaction*).

En el siguiente capítulo se presenta los detalles de diseño e implementación del sistema desarrollado.

Capítulo III

Diseño e Implementación

El sistema se basa principalmente en el trabajo presentado por *Kniss et al*.[**31**] (presentado en el capítulo anterior sección 2.2.2.1). En este capítulo se describen los requerimientos del sistema que se tomaron en cuenta en la etapa de diseño. Además, se describen los detalles de implementación.

3.1 Detalles de Diseño

En la etapa de diseño de realizó el modelado de la aplicación para comprender el sistema que posteriormente se desarrollaría. Para modelar, analizar y diseñar sistemas orientados a objetos se utiliza el lenguaje de modelado UML (*Unified Modeling Language*)[**47**]. UML ofrece diversos diagramas de modelado en los que se encuentran los diagramas de casos de usos y los diagramas de clases los cuales se utilizaron para el diseño del sistema. Se realizaron los diagramas de casos de usos para el levantamiento de requerimientos de la interfaz de usuario y el diagrama de clases para modelar la estructura del sistema orientado a objetos.

3.1.1 Diagramas de casos de usos

A continuación se presentan los casos de usos modelados para el desarrollo del sistema, en tres niveles de especificación.

3.1.1.1 Casos de Usos –Nivel 0

En este caso de uso se modela el sistema a nivel general (ver Figura 3.1). El sistema solo poseerá un tipo de actor llamado usuario, el cual interactuará con el sistema de despliegue de volúmenes.

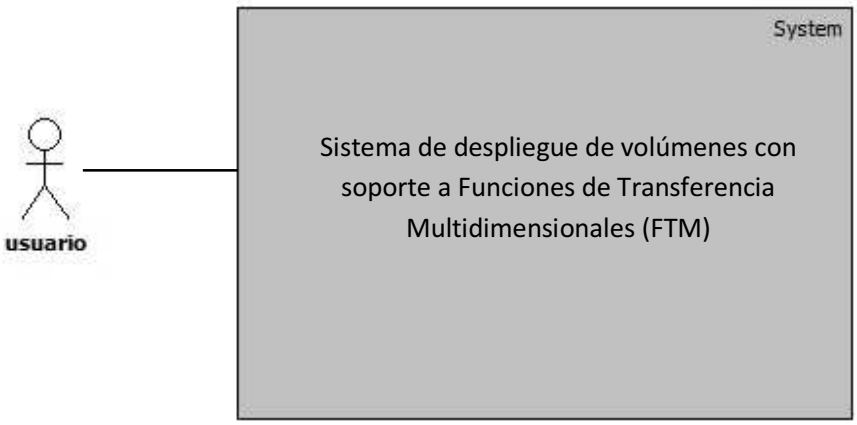


Figura 3.1: Casos de uso del sistema, Nivel 0.

3.1.1.2 Casos de Usos –Nivel 1

En este nivel de caso de uso se describe las interacciones generales del actor usuario con los componentes del sistema (ver Figura 3.2). La descripción detallada de los casos de uso se expone en la Tabla 3.1, en donde se indica los elementos que interactúan con cada caso de uso.

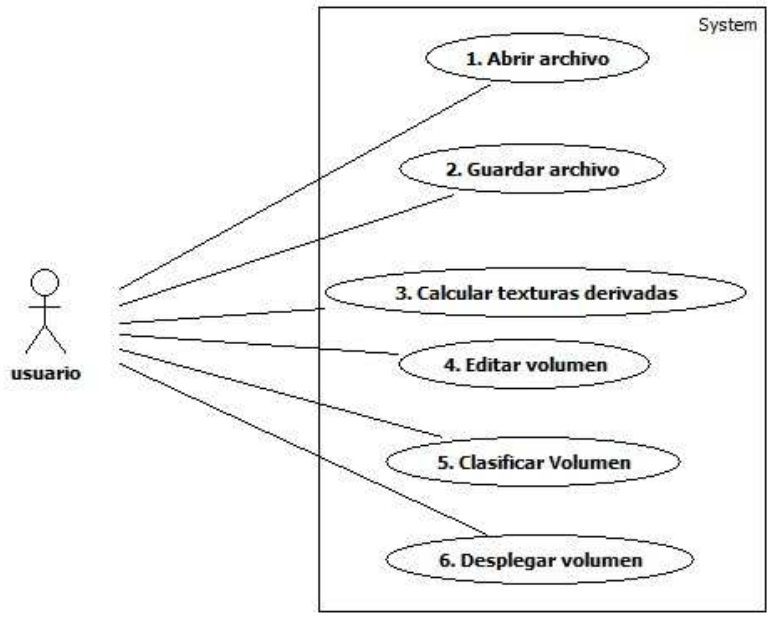


Figura 3.2: Casos de uso del sistema, Nivel 1.

Tabla 3.1: Casos de uso del sistema, Nivel 1.

CASOS DE USO – NIVEL 1	
CASO DE USO 1	
Nombre	Abrir archivo.
Actor	Usuario.
Descripción	Permite al usuario abrir los archivos compatibles necesarios para el uso del sistema.
Pre-condición	El usuario debe poseer acceso al sistema.
Flujo Básico	- El usuario ejecuta el sistema. - Se solicita un nombre de archivo. - Se verifica la existencia y valides del tipo de archivo. - De tratarse de un archivo válido se procede a la carga del archivo en el sistema, de lo contrario, se notifica al usuario que el archivo no es válido o no existe (según corresponda).
Post-condición	El usuario puede utilizar todas las funcionalidades del sistema sí el tipo de archivo cargado corresponde a un volumen (*.PVM*).
CASO DE USO 2	
Nombre	Guardar archivo.
Actor	Usuario.
Descripción	Permite al usuario guardar las configuraciones que realizó en el sistema para su posterior utilización.
Pre-condición	Debe haber previamente un archivo compatible cargado en el sistema.
Flujo Básico	- El usuario ingresa el nombre del archivo. - Se verifica la existencia del nombre del archivo, si este nombre existe, se notifica su existencia para que el usuario pueda cambiar el nombre del archivo ó decida sobrescribir el archivo.

	- Con el nombre de archivo se procede a generar el archivo o, a sobrescribirlo según sea el caso.
Post-condición	El archivo es generado y almacenado en memoria segundaria.
CASO DE USO 3	
Nombre	Calcular texturas derivadas.
Actor	Usuario.
Descripción	Permite al usuario (re)calcular las derivadas usadas en la clasificación, pudiendo este mejorar la precisión de dichas derivadas (las derivadas son almacenadas en texturas).
Pre-condición	Debe haber un archivo (volumen) previamente cargado.
Flujo Básico	- El usuario selecciona el filtro a utilizar. - Utilizando el filtro se recorre el volumen, obteniendo las derivadas. - Luego, el resultado anterior es normalizado y almacenado en texturas.
Post-condición	Se actualizaron las derivadas usadas en el sistema.
CASO DE USO 4	
Nombre	Editar volumen.
Actor	Usuario.
Descripción	El usuario puede manipular las características como: escala, distancia de muestreo, entre otras, cargadas por defecto en el volumen.
Pre-condición	Debe haber un archivo (volumen) previamente cargado.
Flujo Básico	- El usuario selecciona la propiedad. - Se edita la propiedad. - Se ordena al sistema actualizar las propiedades del volumen.
Post-condición	Obtenemos los nuevos valores de características del volumen.
CASO DE USO 5	
Nombre	Clasificar volumen.
Actor	Usuario.
Descripción	El usuario manipula la función de transferencia y decide que parte del volumen visualizar, además de asignarle diversos colores a las distintas partes del volumen.
Pre-condición	Debe haber un archivo (volumen) previamente cargado.
Flujo Básico	- El usuario selecciona la dimensionalidad de la función de transferencia (1D, 2D o 3D). - Luego manipula los puntos de control, pudiendo agregar o eliminar los puntos de control. También puede editar los puntos de control existentes, pudiendo modificarle su ubicación, el color y opacidad. - Utilizando los puntos de control se construye la textura de clasificación.
Post-condición	Texturas de clasificación actualizadas.
CASO DE USO 6	
Nombre	Desplegar volumen.
Actor	Usuario.
Descripción	El usuario visualiza el volumen con todas transformaciones aplicadas a él.
Pre-condición	Debe haber un archivo (volumen) previamente cargado.
Flujo Básico	- El usuario luego de editar las características del volumen, la función de transferencia o incluso las derivadas, ordena al sistema recargar las nuevas texturas y características actualizadas. - Utilizando las nuevas texturas se obtiene un nuevo volumen transformado. - Se despliega el volumen transformado.
Post-condición	Visualización del volumen actualizada.

3.1.1.3 Casos de Usos –Nivel 2

En este nivel de especifican la funcionalidades ofrecidas por el sistema al autor usuario, las cuales pueden visualizarse en la Figura 3.3. Cada una de estas funcionalidades se detalla en la Tabla 3.2.

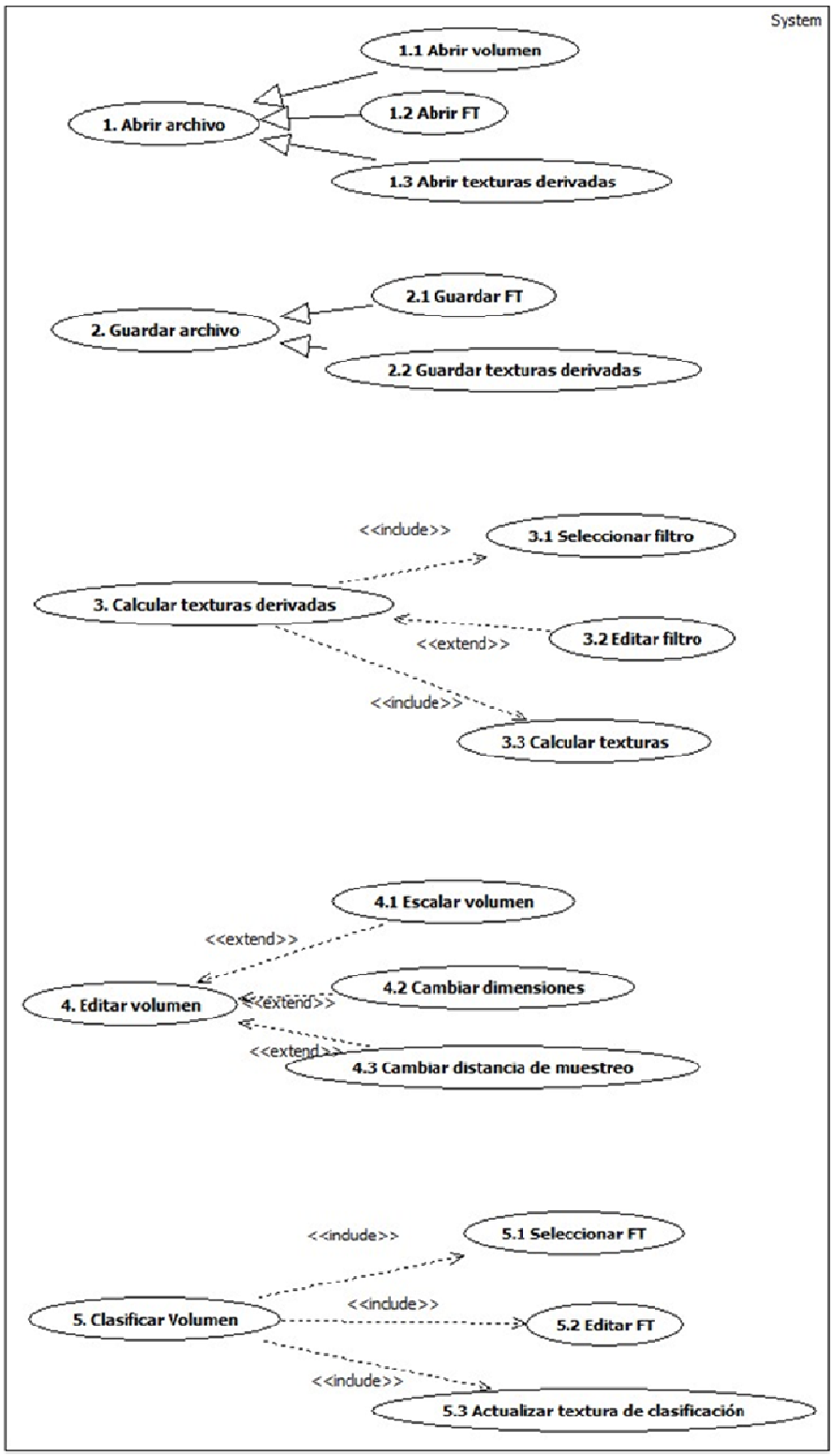


Figura 3.3: Casos de uso del sistema, Nivel 2.

Tabla 3.2: Casos de uso del sistema, Nivel 2.

CASOS DE USO – NIVEL 2	
CASO DE USO 1.1	
Nombre	Abrir volumen.
Actor	Usuario.
Descripción	Permite al usuario abrir un archivo de tipo volumen (.PVM).
Pre-condición	El usuario debe poseer acceso al sistema.
Flujo Básico	El usuario selecciona un archivo de este tipo y luego se produce la carga.
Post-condición	El usuario puede utilizar las funcionalidades del sistema en su totalidad.
CASO DE USO 1.2	
Nombre	Abrir FT.
Actor	Usuario.
Descripción	Permite al usuario abrir un archivo de tipo FT, permitiendo cargar configuraciones previas de las funciones de transferencia.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	El usuario selecciona un archivo de este tipo y luego se produce la carga.
Post-condición	Se actualizan las funciones de transferencia de acuerdo al archivo.
CASO DE USO 1.3	
Nombre	Abrir texturas derivadas.
Actor	Usuario.
Descripción	Permite al usuario abrir las derivadas de un volumen almacenadas en texturas.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	El usuario selecciona un archivo de este tipo y luego se produce la carga.
Post-condición	Se actualizan las derivadas del volumen de acuerdo al archivo.
CASO DE USO 2.1	
Nombre	Guardar FT.
Actor	Usuario.
Descripción	Permite al usuario guardar las configuraciones realizadas a las funciones de transferencia.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	El usuario ingresa el nombre del archivo y luego se guardan todas las configuraciones de las funcione de transferencia en un archivo.
Post-condición	Las configuraciones de las funciones de transferencia son almacenadas en un archivo.
CASO DE USO 2.2	
Nombre	Guardar texturas derivadas.
Actor	Usuario.
Descripción	Permite al usuario guardar las derivadas del volumen.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	El usuario ingresa el nombre del archivo y luego se guardan todas las texturas de las derivadas del volumen en un archivo.
Post-condición	Las texturas de las derivadas son almacenadas en un archivo.
CASO DE USO 3.1	
Nombre	Seleccionar Filtro.
Actor	Usuario.
Descripción	El usuario selecciona el tipo de filtro a utilizar en el cálculo de las derivadas.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	El usuario de acuerdo a la lista de filtro selecciona el filtro deseado.
Post-condición	Filtro seleccionado.
CASO DE USO 3.2	
Nombre	Editar filtro.
Actor	Usuario.
Descripción	De acuerdo al filtro seleccionado se permite la edición de dicho filtro, pudiendo el usuario ingresar el filtro de su preferencia.
Pre-condición	Se debe haber seleccionado un filtro previamente.
Flujo Básico	Una vez seleccionado el filtro si este lo permite puede cambiar los valores del kernel que compone el filtro.

Post-condición	El usuario obtiene un filtro editado.
CASO DE USO 3.3	
Nombre	Calcular texturas.
Actor	Usuario.
Descripción	El usuario le ordena al sistema que proceda a calcular las derivadas.
Pre-condición	Se debe haber seleccionado un filtro previamente, además de tener cargado el volumen.
Flujo Básico	Se calcula las derivadas del volumen usando el filtro seleccionado y son almacenadas en texturas.
Post-condición	Texturas de las derivadas actualizadas.
CASO DE USO 4.1	
Nombre	Escalar volumen.
Actor	Usuario.
Descripción	Edita las variables responsables de realizar el escalado del volumen.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	El usuario selecciona cada una de las variables y las edita.
Post-condición	Variables de escalado actualizadas.
CASO DE USO 4.2	
Nombre	Cambiar dimensiones.
Actor	Usuario.
Descripción	Permite el usuario cambiar las dimensiones del cubo de visualización.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	El usuario selecciona cada una las variables y las edita.
Post-condición	Dimensiones del cubo de visualización actualizados.
CASO DE USO 4.3	
Nombre	Cambiar distancia de muestreo.
Actor	Usuario.
Descripción	Permite al usuario modificar la distancia en que se muestrea el volumen.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	El usuario selecciona la variable y la edita.
Post-condición	Variable de muestreo actualizada.
CASO DE USO 5.1	
Nombre	Seleccionar FT.
Actor	Usuario.
Descripción	El usuario selecciona el tipo de función de transferencia a trabajar.
Pre-condición	Debe haber un volumen previamente cargado.
Flujo Básico	Ubica el tipo de función de transferencia y la selecciona.
Post-condición	Función de transferencia seleccionada.
CASO DE USO5.2	
Nombre	Editar FT.
Actor	Usuario.
Descripción	El usuario edita los puntos de control que componen la función de transferencia.
Pre-condición	Debe haber seleccionado la función de transferencia.
Flujo Básico	El usuario selecciona un punto de control y realiza las transformaciones permitidas según función de transferencia, pudiendo editar su color, transparencia o posición en todas las funciones de transferencia. Además, se puede editar el degradado, tamaño y tipo de figura en funciones de transferencia multidimensionales.
Post-condición	Puntos de control transformados.
CASO DE USO 5.3	
Nombre	Actualizar volumen de clasificación.
Actor	Usuario.
Descripción	Se actualiza el volumen de clasificación usando los nuevos puntos de control transformados.
Pre-condición	Se editaron los puntos de control de la función de transferencia.
Flujo Básico	Utilizando los nuevos puntos de control se actualiza la textura de clasificación.
Post-condición	Textura de clasificación actualizada.

3.1.2 Diagramas de clases

Luego de definir los requerimientos de la aplicación, se plantea un sistema orientado a objeto para implementarlos. El diagrama de clases de este sistema se muestra en la Figura 3.4.

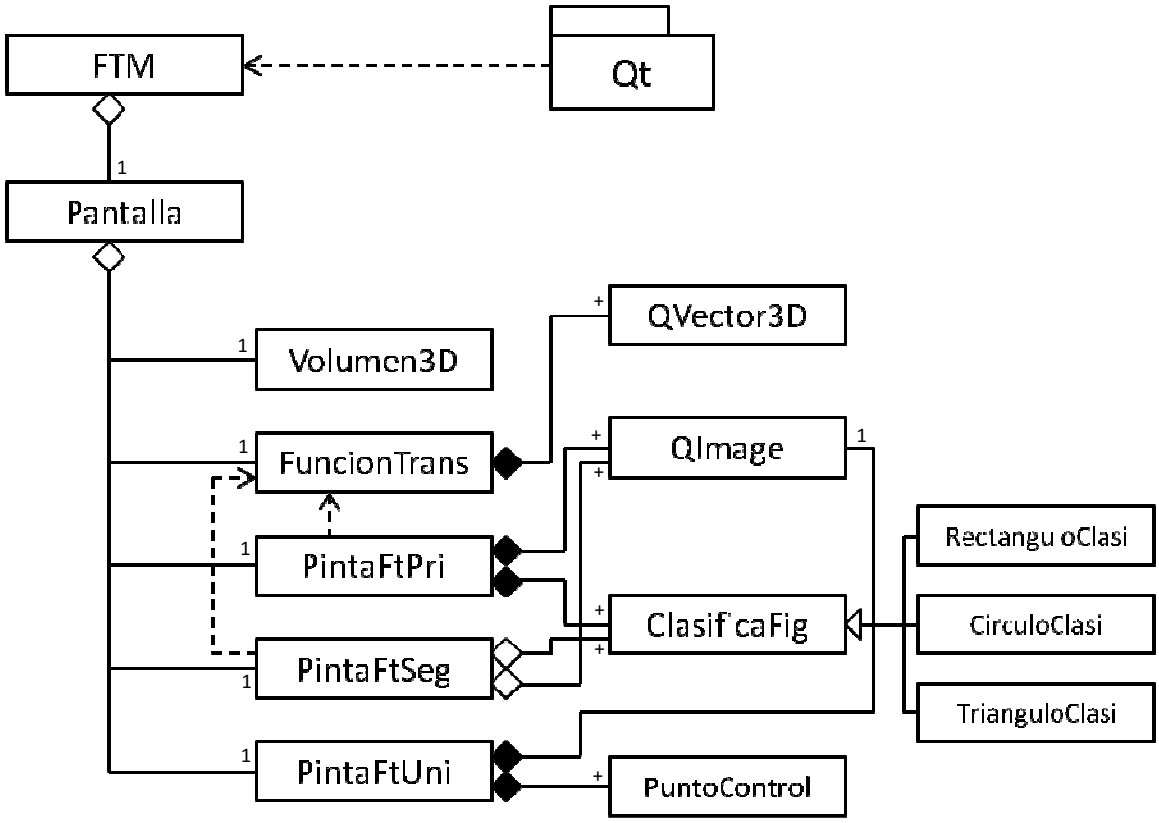


Figura 3.4: Diagrama de clases del sistema.

A continuación se describen las clases principales del sistema:

FTM: Es la clase encargada de manejar la interfaz en el sistema. Para lograr tal propósito utiliza la biblioteca *Qt* (definida en la siguiente sección). En Qt se reciben los eventos generados; estos eventos son entonces capturados por la clase *FTM* y respondidos con invocaciones a métodos de la clase *Pantalla*.

Pantalla: Es la clase principal encargada del despliegue, además es la responsable de la lógica general del sistema. En esta clase se encuentran contenidas las estructuras principales como el volumen, las distintas estructuras de funciones de transferencia y las estructuras encargadas de los distintos cálculos de derivadas.

Volumen3D: Esta clase contiene el volumen principal, su respectiva textura y datos generales como escala, dimensiones, entre otros.

FuncionTrans: Es la clase encargada de contener todas las estructuras necesarias para calcular las distintas derivadas usando diversos métodos.

PintaFtPri: Esta clase es la encargada soportar la edición de una función de transferencia 2D, también es la encargada de contener y construir el volumen de clasificación y sus figuras de clasificación. Esta clase y las dos siguientes tienen el prefijo pinta porque literalmente se encargan de pintar el punto de control (según la dimensión que representan) en la función de transferencia.

PintaFtSeg: Esta clase soporta la edición de la tercera dimensión de la función de transferencia, y almacena dicha manipulación en el volumen de clasificación y las figuras de clasificación contenidas en *PintaFtPri* al cual tiene acceso.

PintaFtUni: es la clase encargada de contener todo el soporte que permita la manipulación de una función de transferencia 1D.

ClasificaFig: Es una clase abstracta que es implementada en las clases: *RectanguloClasi*, *CirculoClasi* y *TrianguloClasi*. Esta clase permite estandarizar el comportamiento en general de las tres clases anteriores, aunque cada clase implementa a su manera los métodos impuestos por la clase padre. Esta clase es utilizada para contener los puntos de control en funciones de transferencia 2D y 3D.

PuntoControl: Es una clase encargada de contener los puntos de control para las funciones de transferencia 1D.

Las clases *QImage* y *QVector3D* son clases propias de la biblioteca base *Qt*. La primera clase permite la manipulación de imágenes y, la segunda clase provee la manipulación de un vector en el espacio tridimensional.

3.2 Detalles de implementación

Luego del diseño del sistema se procedió a su implementación. A continuación se detallan aspectos de la implementación del sistema.

3.2.1 Recursos de software

En la implementación se utilizó el entorno de desarrollo Microsoft Visual Studio 2008 y bajo el lenguaje de programación C++ con las librerías: Qt, OpenGL®,Glew y OpenMP.

Qt [**48**] es una biblioteca multiplataforma utilizada principalmente para el desarrollo de interfaces gráficas de usuario, aunque también provee estructuras de datos y métodos que permiten realizar una gran cantidad de acciones como métodos para acceder a datos mediante SQL (*Structured Query Language* o lenguaje de consulta estructurado), gestión de hilos, entre otros. En nuestra implementación se utilizó Qt para realizar la interfaz de usuario y algunas estructuras de datos en la lógica general.

OpenGL® (*Open Graphics Library*)[**49**] es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D, utilizado principalmente para generar la visualización del volumen en la aplicación.

GLEW (*OpenGL Extension Wrangler Library*)[**50**] es una librería multiplataforma escrita en C/C++, destinada a ayudar en la carga y consulta de extensiones de OpenGL®; es utilizada en la aplicación para facilitar el manejo del OpenGL®.

Por último se utilizó OpenMP para la paralelización de los ciclos *for* en los algoritmos implementados. OpenMP[**51**] es un API definido por los principales desarrolladores de software y hardware en conjunto, soportado por los lenguajes de programación C/C++ y Fortran para la programación multiproceso de memoria compartida.

OpenMP permite paralelizar secciones de código sobre la base del modelo de ejecución Fork/Join. Todos los programas en OpenMP empiezan con un hilo maestro. El hilo maestro se ejecuta secuencialmente hasta que encuentra la sección paralela; en dicho punto el hilo maestro crea (Fork) una cierta cantidad de hilos que permitirán paralelizar la sección paralela. Las declaraciones del programa que pertenezcan a la sección paralela se ejecutan por los distintos hilos en paralelo. Cuando los distintos hilos completan su ejecución, estos son sincronizados y desactivados (o destruidos), quedando solamente activo el hilo maestro el cual continua con la ejecución secuencial del código hasta que encuentre otra sección paralela en donde se vuelve a repetir el proceso, este proceso puede verse gráficamente en la Figura 3.5.

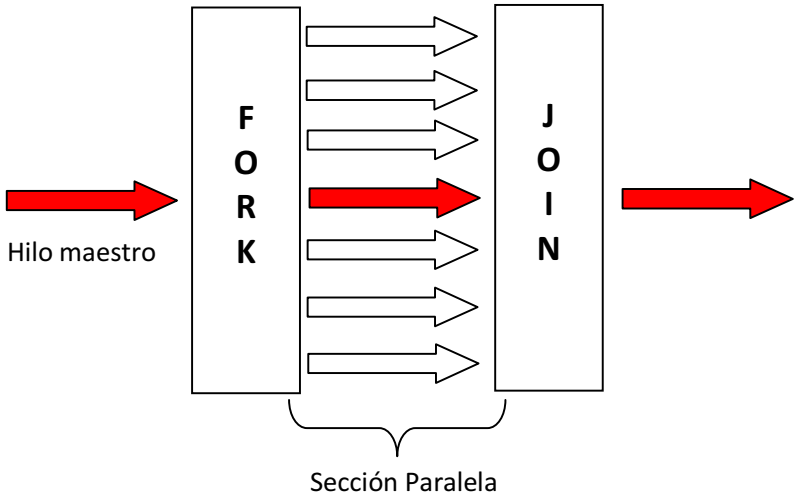


Figura 3.5: Modelo de ejecución Fork/Join.

3.2.2 Estructura de datos

Como se pudo observar en los diagramas de casos de uso, el sistema básicamente necesita estructuras de datos que permitan la extracción de las derivadas, la clasificación y el despliegue del volumen. De acuerdo a lo anterior, se diseñó el diagrama de clases con soporte a

estos requerimientos iniciales, de tal forma que las clase *FuncionTrans* es la encargada de manejar el cálculo y almacenamiento de las derivadas; la clasificación según sea la dimensionalidad de la función de transferencia son manejadas por la clases *PintaFtPri*, *PintaFtSeg* y *PintaFtUni*; la etapa de despliegue la cual necesita los resultados de las clases anteriores es manejada por la clase padre *Pantalla* (ver figura 3.3).

A continuación se especifican cada una de las estructuras, las cuales representan a las clases mencionadas, en donde se detalla su estructura interna y las consideraciones que se tuvieron en cuenta para satisfacer tanto la problemática que se ataca (aumentar la dimensionalidad del dominio de la función de transferencia de uno a dos y tres para lograr mejorar la clasificación del volumen), como los requerimientos de diseño.

Una de las consideraciones más importantes es que los datos pasados a la tarjeta gráfica (en nuestra implementación) fueron datos con precisión de 8-bits (con valores de 0 a 255) y en el cálculo de las derivadas, se necesitaba contar con toda la precisión posible para minimizar el error. De tal forma que, muchas veces se tenían que poseer dos copias de los cálculos derivados del volumen, una con la máxima precisión posible y otra con precisión de 8-bits.

La clase *FunctionTrans* es la encargada de almacenar todos los datos necesarios para lograr la expansión de la dimensionalidad del dominio en la función de transferencia. Por lo tanto, es aquí donde se encuentran contenidas la magnitud del gradiente y la segunda derivada; estas estructuras son almacenadas por con precisión flotante para los dos casos. Además, también se encuentran en esta clase los vectores gradiente almacenados como QVector3D (clase de Qt que representa un punto en el espacio 3D). En esta clase *FunctionTrans* también se encuentran contenidos los vectores normales, magnitud del gradiente y la segunda derivada con precisión de 8-bits como apuntadores a enteros sin signos (32-bits, 24 bits para los tres componentes del vector normal y 8 bits para la primera derivada) y caracteres sin signos (8-bits para la segunda derivada), utilizado esto por la Tarjeta Gráfica.

El incrementar la dimensionalidad del dominio de la FT trajo como consecuencia el incremento de los grados de libertad que presentan los puntos de control en la FT, pasando de simples puntos a figuras planas o volumétricas. Para soportar su representación se creó una estructura de datos llamada *ClasificaFig*. *ClasificaFig* es una clase abstracta que permite representar los puntos de control para FT de dos o tres dimensiones; Con esta clase se soporta la edición de dichos puntos de control, pudiéndose con esta estructura cambiar el tipo de figura (rectángulo *RectanguloClasi*, circulo *CirculoClasi* y triangulo *TrianguloClasi*), editar el tamaño y color.

La clase *PintaFtPri* es la responsable de contener: una lista de tipo *ClasificaFig* utilizando un *QVector* (soportado por la biblioteca Qt) y el volumen que representa la función de transferencia multidimensional, él cual está representado por una lista de imágenes planas por medio de un *Qvector* de *QImage* (estructura de datos de Qt para las imagenes). La clase *PintaFtSeg* por su parte tiene apuntadores a estos dos vectores, lo cual le permite modificar sus características en caso de estar trabajando con un FT de tres dimensiones. En otro sentido, la

clase *PintaFtUni* es la encargada de manejar todo lo concerniente a las funciones de transferencia de una sola dimensión, proveyendo en sí, las estructuras de datos para almacenar los puntos de control y la textura de una dimensión pasada a la tarjeta gráfica; utilizando para ello un *QVector* de tipo *QPair* de *int* y *QColor,* para los puntos de control y un *QImage* para almacenar la textura unidimensional.

Para contener las clases principales se utiliza la clase principal *Pantalla*, la cual además de contener estas estructuras, es la encargada de la comunicación con la tarjeta gráfica para generar el despliegue del volumen.

3.2.3 Algoritmo pseudo formal e implementación

Una vez cargado el volumen, el algoritmo general se puede resumir principalmente en 3 etapas:

1. Calcular o reconstruir las derivadas del volumen.
2. Clasificar el volumen.
3. Desplegar el volumen.

A continuación explicaremos los algoritmos más relevantes implementados en cada una de las etapas.

3.2.31. Calcular o reconstruir las derivadas del volumen.

Esta etapa es la encargada de pre procesar el volumen para obtener las derivadas, normales e histogramas de dispersión, pero la obtención de las derivadas es su tarea primordial. Los datos que deben calcularse (o reconstruirse) son la magnitud del gradiente (información de la primera derivada) y la segunda derivada direccional a lo largo de la dirección del gradiente (información de la segunda derivada). En el caso de la primera derivada sólo existe una fórmula para calcularse, mientras que en el caso de la segunda derivada existen tres fórmulas distintas (ver Tabla 3.3). Estas fórmulas para evaluarlas necesitamos las derivadas parciales de primer o segundo orden según sea el caso. Por lo tanto necesitamos de algún método de reconstrucción que nos permita obtener estos datos del volumen para evaluar las fórmulas. Los métodos de reconstrucción implementados fueron el método de diferencias centrales y el método de la expansión de Taylor de la suma de convoluciones basado en filtros, los cuales se describen a continuación.

Tabla 3.3: Formulas para calcular las derivadas.

Primera derivada f'	Magnitud del gradiente $\sqrt{\left(\frac{df}{dx}\right)^2 + \left(\frac{df}{dy}\right)^2 + \left(\frac{df}{dz}\right)^2}$

Segunda derivada f''	Gradiente del gradiente $\frac{1}{\|\nabla f\|}\nabla(\|\nabla f\|)\nabla f$	Laplaciano $\frac{d^2f}{dx^2}+\frac{d^2f}{dy^2}+\frac{d^2f}{dz^2}$	Matriz Hessiana $\frac{1}{\|\nabla f\|^2}(\nabla f)^{\mathrm{T}}\mathbf{H_f}\nabla f$

El método de reconstrucción de diferencia centrales consiste en restar las muestras adyacentes de un vóxel (*i,j,k*) en cada eje, y dividir su resultado entre 2. Un ejemplo de este método puede verse en el algoritmo 1.1 en donde se aproxima la primera derivada parcial en el eje x de un vóxel considerando las condiciones de frontera, en las otras derivadas el algoritmo es similar cambiando solo el eje en donde es aplicado.

```
//Algoritmo para obtener la primera derivada parcial de 'x' (gx)
int gx;
//obtenerMuestra() es una función que retorna el valor de la muestra
//en una posición del volumen.
si (i>0) entonces
  si (i<ancho-1) entonces
       gx = (obtenerMuestra(i+1,j,k) - obtenerMuestra(i-1,j,k))/2.0
  sino
       gx = obtenerMuestra(i,j,k) - obtenerMuestra(i-1,j,k);
sino
  gx = obtenerMuestra(i+1,j,k) - obtenerMuestra(i,j,k);
```

Algoritmo 1.1: Diferencia centrales para el cálculo de una derivada parcial de primer orden.

En caso de que se necesite reconstruir una derivada parcial de segundo orden para una muestra se requieren los vectores gradientes de las muestras adyacentes, pues ahora al momento de aplicar el método de diferencia centrales se muestrean los vectores gradientes, para obtener alguna de derivada parcial de segundo orden. En el algoritmo 1.2 se observa la manera de realizar el cálculo de $(\frac{d^2f}{dxy})$, teniendo ya pre-calculado los vectores gradiente.

```
//Algoritmo para obtener la derivada parcial de 'x' con respecto a 'y' (gxy)
int gxy;

//Primero se obtiene el gradiente de la muestra anterior y porterior,
//luego se accede al valor del vector gradiente en el eje y
//obtenerGradiente() es una función que retorna un vector 3D('x','y','z')
//de la muestra en la posición requerida.
if (i>0) entonces
   if (i<ancho-1) entonces
       gxy = (obtenerGradiente(i+1,j,k).y() –
              obtenerGradiente (i-1,j,k).y())/2.0;
   sino
        gxy = obtenerGradiente(i,j,k).y() - obtenerGradiente (i-1,j,k).y();
sino
   gxy = obtenerGradiente (i+1,j,k).y() - obtenerGradiente(i,j,k).y();
```

Algoritmo 1.2: Diferencia centrales para el cálculo de una derivada parcial de segundo orden.

El segundo método de reconstrucción que se utilizó fue el método de la expansión de Taylor de la suma de convoluciones basado en filtros. Para aplicar este método se necesita conocer de antemano al menos dos kernels de reconstrucción unidimensionales, el kernel de

interpolación y primera derivada. Adicionalmente se requiere el kernel de segunda derivada, sí se desea reconstruir la segunda derivada parcial de una muestra. Como se explicó en la sección 1.4.1.2 del capítulo I, con los kernels unidimensionales se construye un filtro 3D según la derivada parcial que se desea reconstruir. El proceso de construcción se define en el algoritmo 1.3, en donde se muestra una función llamada *calculaSubKer* que retorna un dato de tipo *long long*. La función *calculaSubKer* recibe cuatro parámetros, de los cuales los tres primeros parámetros representan los kernels de acuerdo a su eje, mientras que el último parámetro representa el filtro3D donde se guardaran los resultados. Entonces los tres primeros parámetros representa la configuración de filtro que se requiere reconstruir, es decir que para reconstruir un filtro de primera derivada para x, se debe pasar como primer parámetro el kernel derivativo de primera derivada (representando el eje x en la función) mientras que en los dos parámetros siguientes se pasa el mismo kernel de interpolación (representando al eje y e z en la función), y de esta forma obtendríamos un filtro 3D de primera derivada parcial para x. Esta función retorna el valor absoluto de la suma de los componentes del filtro 3D.

```
//función para la construccción del filtro
funcion calculaSubKer( *KerX, *KerY, *KerZ, *KerSal): long long
    //variable para acumular la suma
    long long acumulaSuma = 0;
    //recorrido por z
    Para o->0 hasta tamano(KerZ)-1
      //recorrido por y
      Para p->0 hasta tamano(KerY)-1
         //recorrido por
         Para q->0 hasta tamano(KerX)-1
            //se multiplica los KerX, KerY y KerZ para obtener el filtro 3D
            (*KerSal)[q][p][o] = (*KerX)[q]*(*KerY)[p]*(*KerZ)[o];
            //se acumula el valor absoluto del punto calculado
            acumulaSuma += abs((long) ((*KerSal)[q][p][o]));

    retorna acumulaSuma;
```

Algoritmo 1.3: Función para el armado del filtro 3D.

Una vez obtenido el Filtro 3D se debe de aplicar este a todo el volumen para obtener su derivada. En dicho proceso se debe centrar el filtro 3D en cada muestra del volumen, para realizar la convolución digital. Centrado el filtro 3D se multiplica componente a componente (filtro-muestra), se acumulan estos resultados y, se normaliza el resultado final por la suma acumulada de del filtro 3D de reconstrucción, para obtener el valor de la derivada en cada muestra. En el algoritmo 1.4 se muestra el algoritmo que implementa este proceso.

```
//Suma acumulada del kernel(peso del filtro)
long long sumaAcumFiltro3D;
//derivada actual
double derAct;
//Matriz 3D donde se guardara la derivada
double volumenDer[ancho][alto][profundidad];
//Se arma el filtro 3D de acuerdo a la derivada a reconstruir
//Obtenemos la suma acumlada y armamos el filtro 3D
sumaAcumFiltro3D = calculaSubKer(&qkerX, &qkerY, &qkerZ,&filtro3D);
//Se obtiene la diferencia de kernel para el recorrido interno del filtro
```

```
int tamDifKer = tamano(qkerX)/2;
//Se recorre el volumen profundidad, alto y ancho
Para k->0 hasta profundidad-1
  Para j->0 hasta alto-1
    Para i->0 hasta ancho-1

      //Se recorre el kernel centradose en el muestra k,j,i
      Para o->(k-tamDifKer) hasta (k+tamDifKer)
        Para p->(j-tamDifKer) hasta (j+tamDifKer)
          Para q->(i-tamDifKer) hasta (i+tamDifKer)
            //Se evalua los límites del volumen
            si(o>=0 and p>=0 and q>=0 and
               o<profundidad and p<alto and q<ancho)entonces
               //Se multiplica la muestra con el filtro 3D
               derAct = filtro3D[q-i+tamDif1][ p-j+tamDif1][ o-k+tamDif1]
                        * obtenerMuestra(q,p,o);
      //Se normaliza la derivada obtenida
      //Se toma en cuenta la división entre cero
      si(sumaAcumFiltro3D!=0)entonces
        derAct /= sumakerDerX;
      //Se almacena la derivada
      volumenDer = derAct;
```

Algoritmo 1.4: Cálculo de la derivada utilizando el método de la expansión de Taylor de la suma de convoluciones basado en filtros.

Utilizando alguno de estos dos métodos de reconstrucción se obtienen las derivadas parciales y se evalúan las ecuaciones para obtener las dos derivadas finales.

Es importante señalar en esta etapa las instrucciones de OpenMP utilizadas para optimizar el cálculo de las derivadas en procesadores que dispongan de más de un núcleo de procesamiento. El recorrido que se debe hacer del volumen en el cálculo de las derivadas es costoso por la cantidad de ciclos que posee. Estos recorridos en el cálculo de las derivadas fueron paralelizados a través de OpenMP. En la aplicación se paralelizaron los ciclos *for* utilizando el esquema presentado en el Algoritmo 1.5. Los ciclos *for* paralelizados en los recorridos fueron siempre los más externos, aminorando así el costo administrativo extra que acarrea dividir y juntar los datos por hilo.

```
//Obtiene la cantidad de núcleo que poseee el procesador
int nroProcesadoresOpenMP = omp_get_num_procs();
//Activa la cantidad de hilos a usar, uno por núcleo
omp_set_num_threads(nroProcesadoresOpenMP);
//Configura la sección paralelizable, donde se pasan las varible privadas
//que se necesite para evitar inconsistencias en el resultado final
#pragma omp parallel private(MyVariable)
{
  //Expecifica el ciclo que se paralerizará
  #pragma omp for
  for(int i=0;i<N;i++)
  {

    //contenido del ciclo. Se ejecuta en paralelo
  }
}
```

Algoritmo 1.5: Instrucciones utilizadas para paralelizar los ciclos con OpenMP en C/C++.

3.2.3.2 Clasificar volumen

La etapa de clasificación consistió en proveer una interfaz de usuario para la edición de los puntos de control (puntos 1D, imagen 2D y volumen 3D), e implementar un algoritmo para la construcción del volumen de clasificación en funciones de transferencia 2D y 3D.

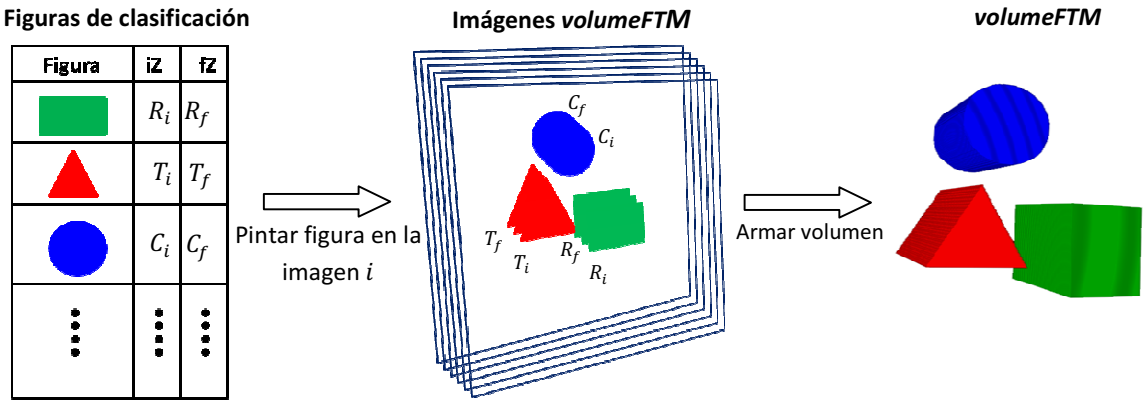


Figura 3.6: Armado del volumen de clasificación. Se observa de izquierda a derecha el procedimiento, el cual consiste dibujar las figuras en la imágenes si la figura esta en el rango de la imagen.

Un volumen se puede ver lógicamente como una lista de imágenes, en donde cada imagen se puede manipular de forma separada, pero juntas construyen el volumen. El volumen de clasificación es una lista de 256 imágenes 2D (256x256), en donde el tamaño de la lista corresponde a la profundidad del volumen y a la tercera dimensión de la función de transferencia, por su parte las dimensiones de las imágenes corresponden: el ancho a los valores de los vóxeles y el alto a la segunda dimensión de la función de transferencia. Entonces la construcción del volumen de clasificación utilizado por la FT 3D, se simplifica en dibujar en cada imagen del volumen de clasificación, las figuras de clasificación agregadas por el usuario verificando siempre los valores de la segunda derivada para cada figura de clasificación, pintando cada figura en el rango de imágenes permitido de acuerdo a su segunda derivada (ver Figura 3.6). Este proceso se describe en el Algoritmo 1.5.

```
//Arreglo de imágenes 2D(256x256)
QImage volumeFTM[256];
//Se iterar por la profundidad de la Segunda derivada (8-bit)
Para i=0 hasta 255
  //Se limpia la imagen de la profundidad i con ceros
  volumeFTM[i].limpiar(0);
  //Recorro el vector de figuras de calsificacion creadas
  Para j=0 hasta tamano(vectorDeFig)
    //trato la fig según la dimension de la FT
    si(nroDimension==3 and i<=vectorDeFig[j].iZ() and i>=vectorDeFig[i].fZ())
      vectorDeFig[j].pinta(&volumeFTM[i]);
    sino
```

```
      si(nroDimension==2)
        vectorDeFig[j].pinta(&volumeFTM[i]);
```

Algoritmo 1.5: Armado del volumen de clasificación.

La FT 2D es una simplificación de la FT 3D, por tanto su implementación se resume a ignorar la profundidad del volumen de clasificación, pitando todas las figuras de clasificación en la primera imagen de la lista de imágenes que conforma el volumen de clasificación.

3.2.3.3 Desplegar volumen

El despliegue del volumen fue realizado por medio de la técnica de *ray casting* acelerado por hardware (GPU). Esta técnica fue seleccionada por la calidad de las imágenes reproducidas y por el nivel de paralelismo que se logra, al utilizar explícitamente las capacidades del hardware gráfico.

Se utilizó el algoritmo general de *ray casting* acelerado por GPU con algunas modificaciones. La implementación del algoritmo general de *ray casting* acelerado por GPU fue proporcionada por el Centro de Computación Gráfica de la Universidad Central de Venezuela, en donde se desarrolló el sistema de visualización. El algoritmo de *ray casting* simple acelerado por GPU es logrado apoyándose de OpenGL® por medio de su extención de GLSL (*OpenGL® Shading Language*), la cual nos permite ejecutar bloques de códigos directamente en la GPU llamados *shader*. En el *shader* de fragmento para ser más específico, es donde se implementa el algoritmo general de ray casting explicado en la sección 1.3.1.

Las modificaciones que se realizaron al algoritmo general de *ray casting* fueron para que soportará la función de transferencia multidimensional (2D/3D). Las modificaciones fueron dos: la primera consistió en proporcionarle las texturas 3D (donde están almacenadas tanto la primera y segunda derivada) al hardware gráfico. La segunda modificación consistió en realizar el muestreo de las texturas (volumen y las derivadas) utilizando como índice una muestra a lo largo del vector de visualización. Estas tres muestras obtenidas son los índices usados para muestrear el volumen o textura de visualización; todo este proceso se ilustra en la Figura 3.7.

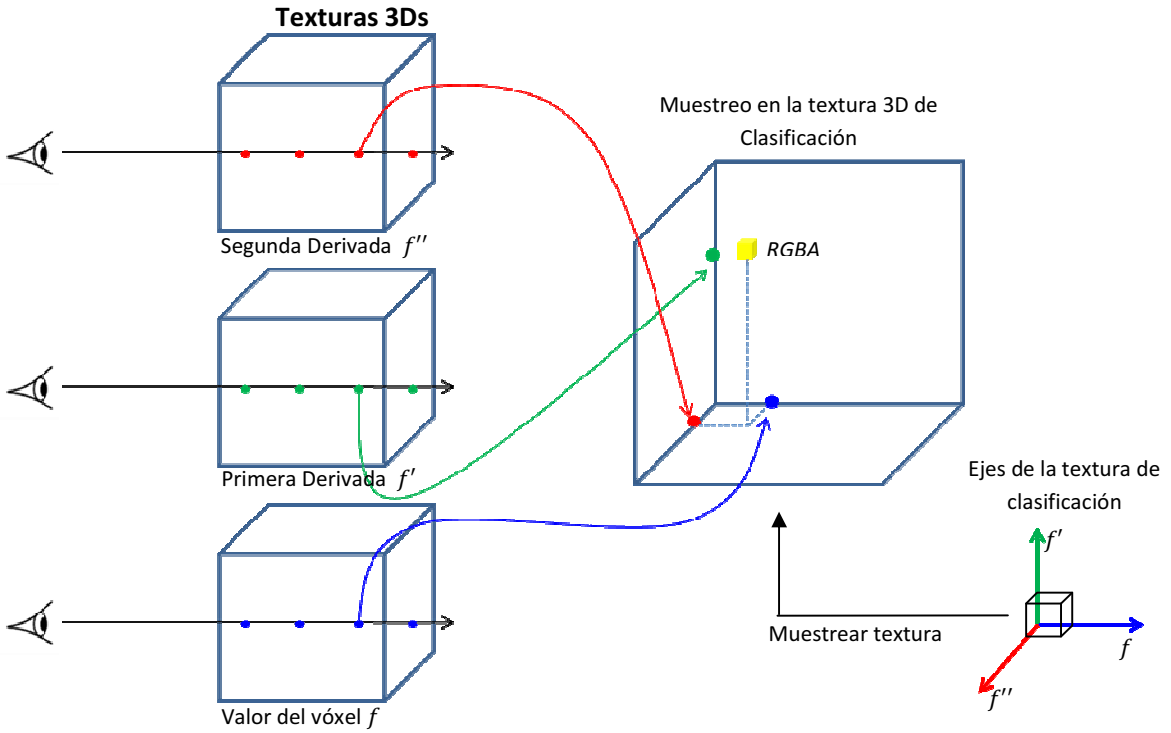


Figura 3.7: Proceso de muestreo de la textura de clasificación, utilizando como índice los muestreos previos de las texturas del volumen, primera y segunda derivada, obtenidos con la misma muestra del vector de visualización.

La muestra obtenida luego del la textura de clasificación es la usada por el algoritmo de *ray casting* en la composición del color. Estas muestras son tuplas *RGBA* las cuales describen el color *RGB* y la opacidad *A*.

3.2.4 Interfaz de usuario

El sistema dispone de una interfaz gráfica de usuario (GUI) basada en *widgets* implementados con la biblioteca *Qt*. La interfaz de usuario provee soporte a la edición de los puntos de control de las funciones de transferencia, elección del método de reconstrucción, cálculo de las derivadas y el despliegue del volumen.

La interfaz principal consta de un menú y un *widget* central, la cual se inicia al ejecutar el sistema. El menú consta de las opciones archivo y editar, donde cada opción es a su vez un menú desplegable; El *widget* central por su parte es el área de despliegue del volumen. La interfaz principal puede verse en la Figura 3.6.

El menú archivo provee las acciones para abrir y guardar archivos de acuerdo a los requerimientos diseñados en los casos de uso. Mientras que el menú editar provee todas las opciones para manipular el volumen. A continuación se explican las opciones principales que

representan la esencia del sistema; estas opciones permiten satisfacer los requerimientos originales diseñados en los casos de uso.

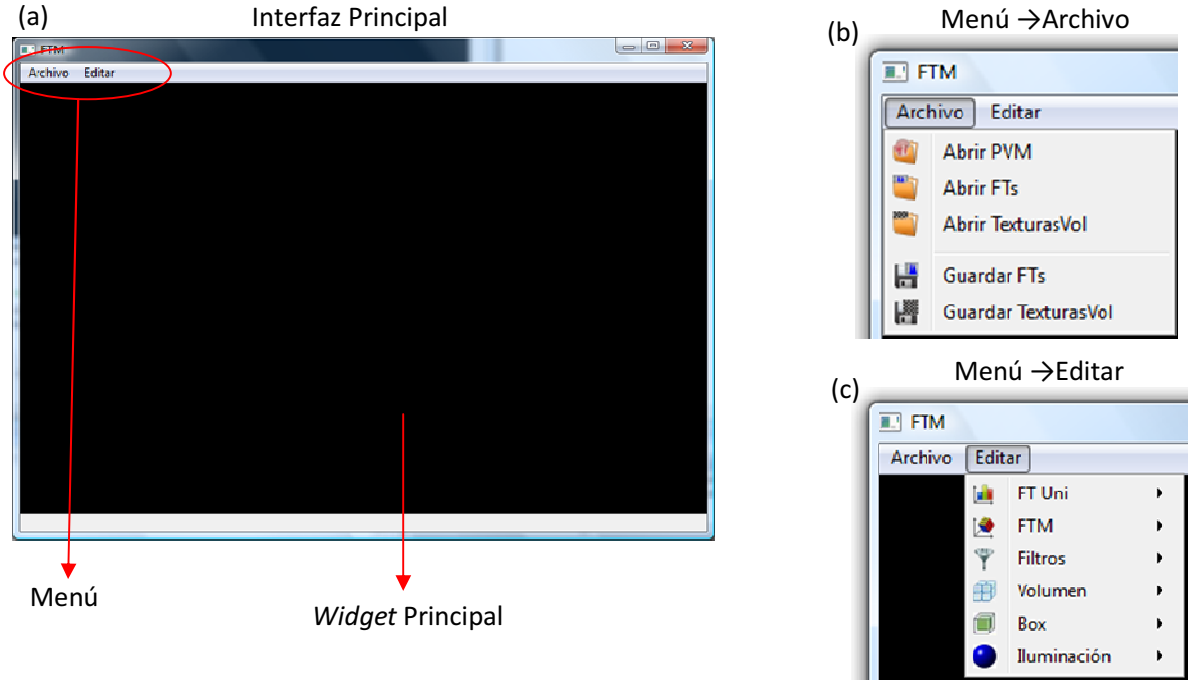


Figura 3.6: (a) Interfaz principal, (b) opciones del **menú → Archivo** y (c) opciones del **menú → Editar**.

La opción **Archivo → Abrir PVM** es la que nos permite la cargar del volumen en el sistema, la cual al presionarla se nos despliega una ventana de dialogo solicitando la ubicación del archivo, el archivo del volumen el cual debe ser de tipo *PVM*. Las otras opciones que provee el menú Archivo funciona de manera similar desplegando el mismo diálogo pero trata el archivo según la opción dada; estas opciones puede verse en los casos de uso abrir y guardar en la sección de diseño.

El menú Editar es más complejo pues desde este menú se acceden las interfaces para la edición de la Función de transferencia con una o más dimensiones, el ingreso del filtro y la edición general del volumen.

Para utilizar una función de transferencia unidimensional se diseñó una sub interfaz que permite su edición; para habilitarla tenemos que navegar por **Editar→FTUni**, y luego presionar **Mostrar Widget** (ver Figura 3.7a). La interfaz consta de un Widget de edición principal, en donde el usuario puede manipular los puntos de control. Además provee un botón que permite activar el uso de la función de transferencia unidimensional, y un segundo botón que permite activar la interfaz de edición de la función de transferencia multidimensional (2D-3D). Esta interfaz puede verse en la Figura 3.7b.

(a) **Editar→FTUni→Mostrar Widget** (b) **Interfaz FT unidimensional**

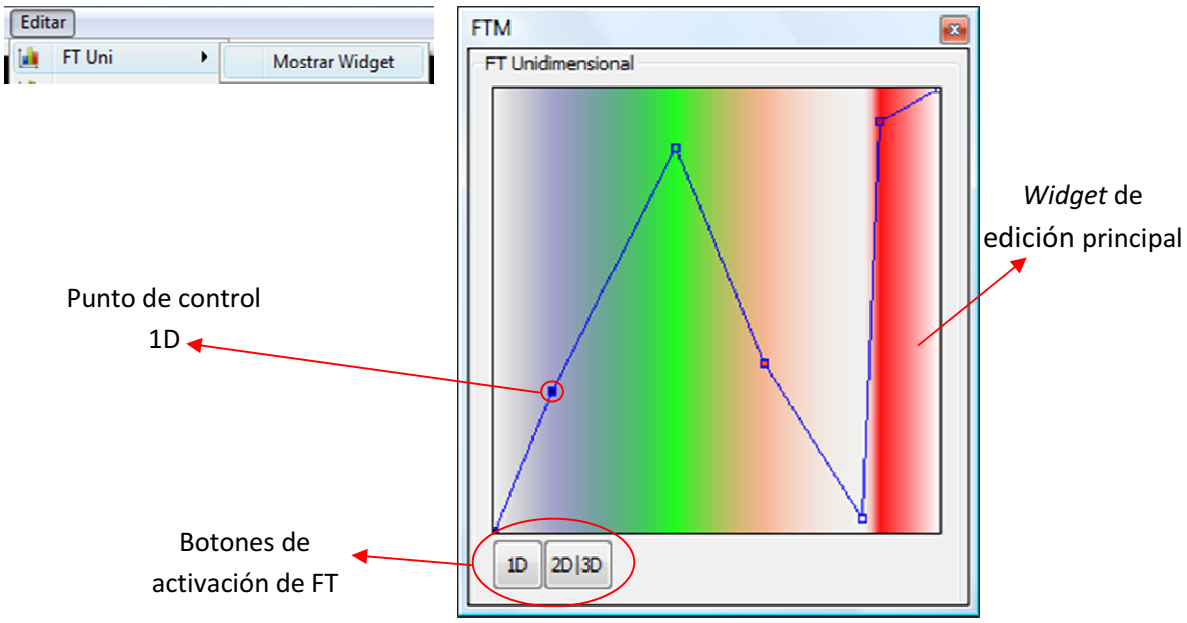


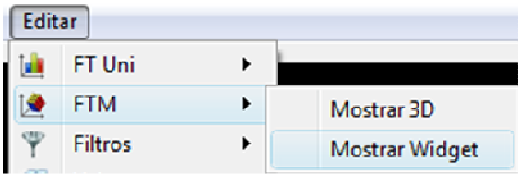
Figura 3.7:(a) Camino para activar la interfaz y en (b) la interfaz para la edición de la FT unidimensional.

En el caso que el usuario desee utilizar una función de transferencia multidimensional debe activar la interfaz de edición, presionando a la opción **Mostrar Widget** del camino **Editar→FTM** (ver Figura3.8a). Esta interfaz es un poco más compleja y posee tres componentes. El primer componente permite la edición de la función de transferencia multidimensional, el segundo componente permite listar las figuras de clasificación y el tercer componente permite la edición de las propiedades ópticas de la figura de clasificación; estos componentes pueden ser vistos en la Figura 3.8b. Cada uno de los componentes mencionados se desglosan a continuación:

a. El primer componente permite la edición de la FTM y está compuesto por dos widgets; en estos widgets se dibujan las figuras de clasificación (equivalente a los puntos de control en la FT 1D) utilizados para armar la función de transferencia multidimensional. Estos widget permiten generar la textura que representa la FTM (ver Figura 3.9a). El primer widget representa al plano formado por los ejes x e y, mientras que el segundo widget representa al plano formado por los ejes x y z. Con estos planos se construye el volumen de clasificación utilizando las figuras de clasificación pintados en ellos. Es importante recordar que los ejes x, y, z representan el valor de los vóxeles, la primera derivada y la segunda derivada respectivamente en la FTM. En la Figura 3.9b se puede observar el significado del cada widget. En el primer componente se permite activar el tipo de función de transferencia a utilizar (1D, 2D o 3D), en donde de acuerdo a la opción se modificará el primer componente ó nos mostrará otra interfaz. En el caso de activar la función de transferencia 2D ó 3D se mantendrá el primer componente, con la diferencia que en la 2D se ocultará el segundo widget encargado de la edición de la tercera dimensión. En el caso que el usuario seleccione la función de transferencia 1D,

se ocultará toda la interfaz de la FT multidimensional y se abrirá la interfaz encargada de la edición de la FT unidimensional la cual puede verse en la Figura 3.7b.

(a) **Editar→FTM→Mostrar Widget**



(b) **Interfaz FT Multidimensional**

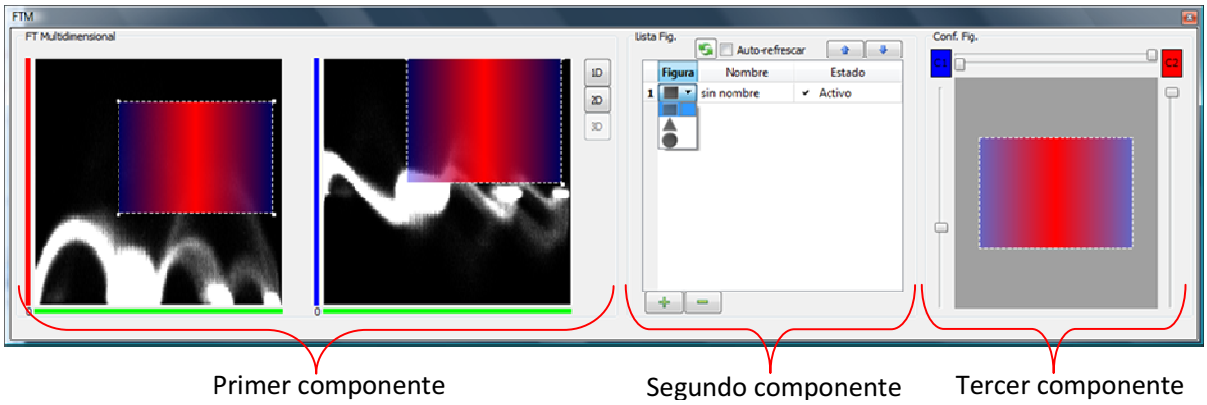


Figura 3.8: (a) Camino para activar la interfaz FT multidimensional y en (b) sus componentes.

b. El segundo componente es el encargado de administrar las figuras de clasificación pintadas en los widgets de la función de transferencia. Este componente está conformado por una tabla de figuras de clasificación en donde se puede cambiar el tipo de figura de clasificación, asignarle algún nombre a la figura así como activar o desactivar su utilización en la reconstrucción de la textura de la FTM. Además a esta tabla se proveen botones para eliminar figura, agregar figura, subir o bajar la figura en la tabla y un botón de actualización en cual permite observar los cambios realizados en el volumen con la FTM editada. Todos estos elementos que conforman el segundo componente de la interfaz de la FTM puede verse en la Figura 3.10a.

c. Por último el tercer componente tiene la tarea de editar las propiedades ópticas de la figura de clasificación. Este componente permite configurar el color y opacidad de la figura además de su degradado (ver Figura 3.10b).

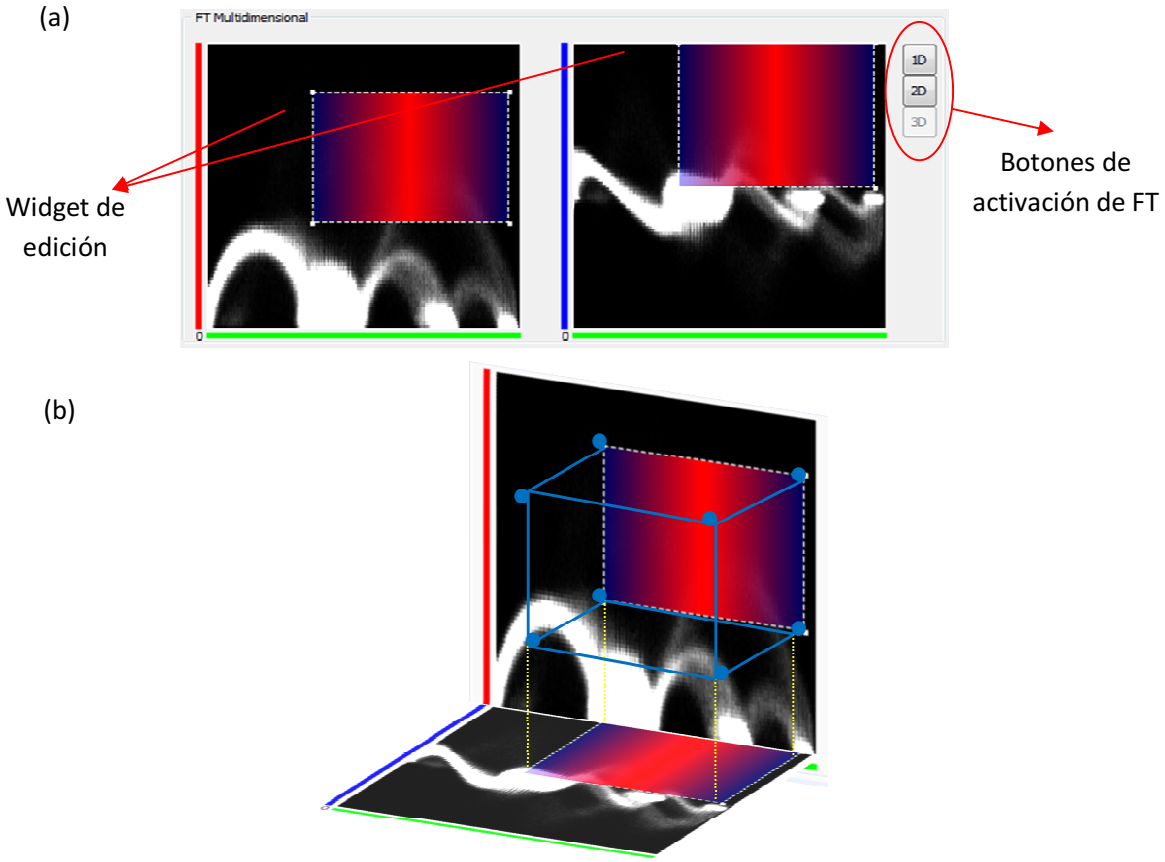


Figura 3.9: El primer componente de la interfaz de FT multidimensional, en (a) las características del componente y en (b) el significado gráfico de esta componente al armar la textura de clasificación.

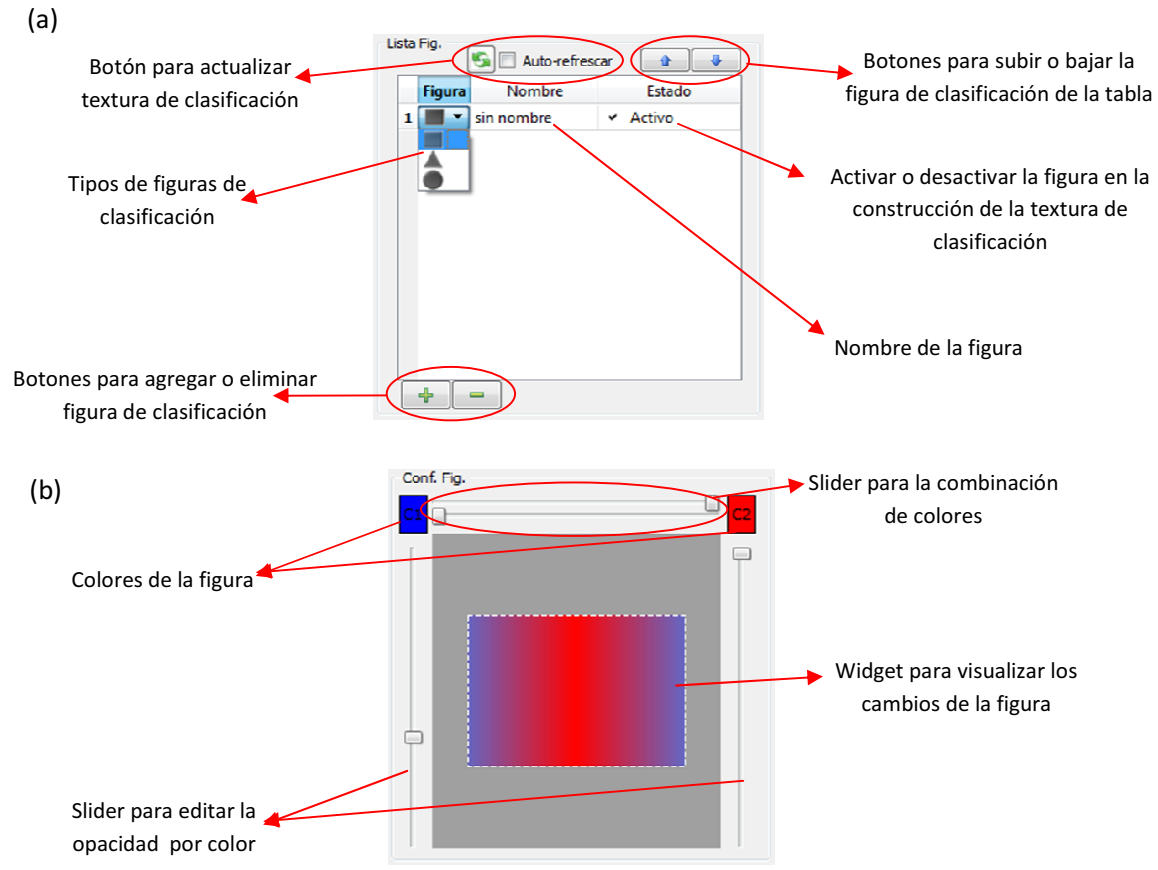
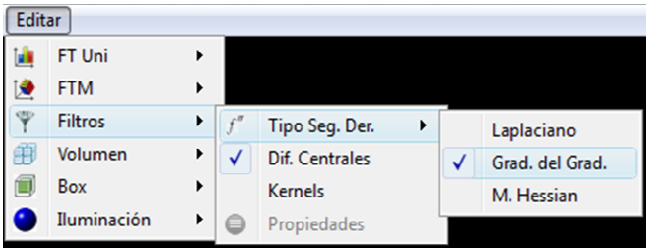


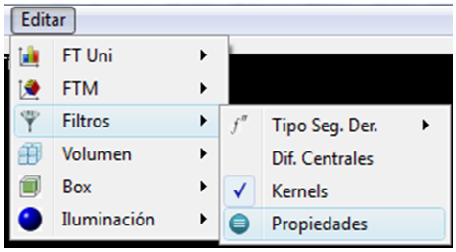
Figura 3.10: El segundo y tercer componente de la interfaz de FT multidimensional, en (a) y (b) respectivamente. Además se muestran las partes de cada componente.

Ahora bien, sí el usuario desea cambiar el método de reconstrucción o seleccionar alguna ecuación en particular de las tres permitidas para el cálculo de la segunda derivada (ver Tabla 3.3), el usuario dispone de un sub menú desplegable al navegar por **Editar→Filtros** del menú. En dicho sub menú podrá cambiar el tipo de método de reconstrucción (configurarlo si es el caso); también podrá seleccionar la ecuación de segunda derivada (ver Figura 3.11a). Para seleccionar el método de diferencia centrales el usuario debe presionar **Dif.Centrales** siguiendo el camino **Editar→Filtros**. Sí en cambio quiere escoger el método de la expansión de Taylor de la suma de convoluciones basado en filtros, debe activarlo presionando **Kernels** desde **Editar→Filtros**. Debido a la complejidad de este segundo método se creó una interfaz de configuración la cual se activa al darle a **Propiedades** desde el camino **Editar→Filtros**, si se encuentra activo el método (ver Figura 3.11b). En esta interfaz de configuración del método de la expansión de Taylor de la suma de convoluciones basada en filtros, se provee tres listas en el cual el usuario puede ingresar los kernel unidimensionales, así como un botón que de inicio al cálculo de las derivadas utilizando este método. Esta interfaz puede verse en la Figura 3.11c.

(a) **Editar→Filtros**



(b) **Editar→Filtros→Propiedades**



(c) **Interfaz para configurar el segundo método**

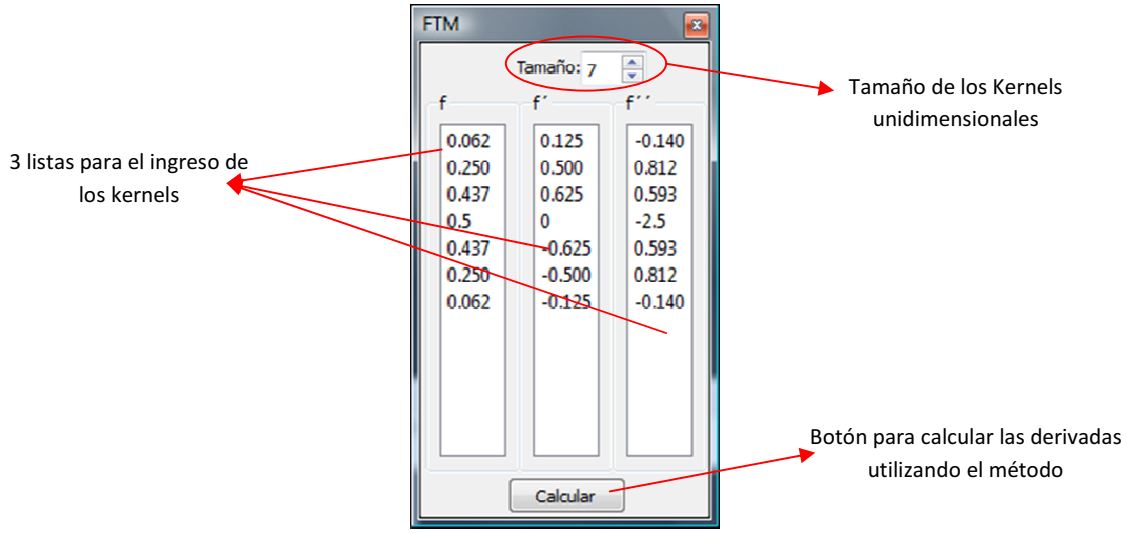


Figura 3.11: Sub menú filtros. En (a) todas las opciones permitidas del sub menú filtro, en (b) se muestra como activar la interfaz propiedades cuando el segundo método (**Kernels)** está activo y en (c) la interfaz que permite editar las propiedades del segundo método.

Una vez implementado el sistema, se tiene que poner a prueba para de esta forma evaluar su desempeño. Estas pruebas con sus respectivos resultados se presentan a continuación en el siguiente el capítulo.

Capítulo IV

Pruebas y Resultados

En este capítulo se muestran pruebas cualitativas y cuantitativas del sistema de rendering implementado, evaluando así los objetivos propuestos, sobre una plataforma basada en Windows.

4.1 Descripción del ambiente de pruebas.

El sistema requiere de un hardware especializado para su ejecución; para realizar las pruebas se utilizó un PC con procesador Intel(R) i3 540 3.70 GHz (2 núcleos/4 hilos), 4GB de memoria y un sistema operativo Windows 7 de 64 bits. La tarjeta gráfica es basada en NVidea, con chip GTX 470, 1280 MB de memoria, y un ancho de banda teórico de 113.9GB/segundo. Los volúmenes utilizados en las pruebas tienen el formato *.pvm* con precisiones de muestreo de 8 bits y 16 bits. Es importante aclarar que si la precisión del volumen es de 16 bits, dicho volumen en el proceso de carga es transformado a precisión de 8bits. Las características de los volúmenes son presentados en la Tabla 4.1.

Tabla 4.1: Características de los volúmenes utilizados en el ambiente de pruebas.

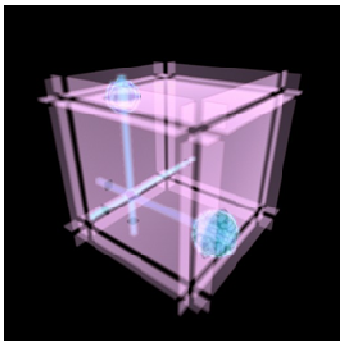
volumen	1	2	3
Dimensiones (*w*x*h*x*d*)	64x64x64	128x256x256	256x256x161
Precisión de muestreo	8bits	8bits	16bits
Tamaño en MB	0,25MB	8MB	20,125MB
Nro. de Vóxeles	262.144	8.388.608	10.551.296

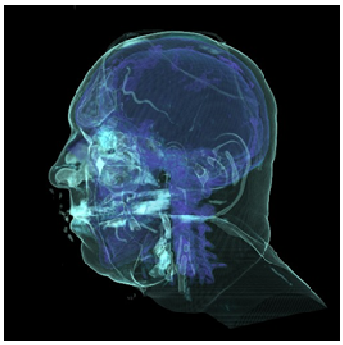
Los volúmenes escogidos representan diferentes formas físicas, los cuales pueden verse en la Figura 4.1. A continuación se describe cada volumen:

- Volumen **A**: Es un conjunto de objetos representados por varias barras cruzadas, bolas y platos.
- Volumen **B**: Es una tomografía computarizada de una cabeza humana, tomada del proyecto humano visible[**52**].
- Volumen **C**: Es una tomografía computarizada de un diente.

Cada uno de estos volúmenes fue obtenido de [**53**].

(a) (b)





(c)

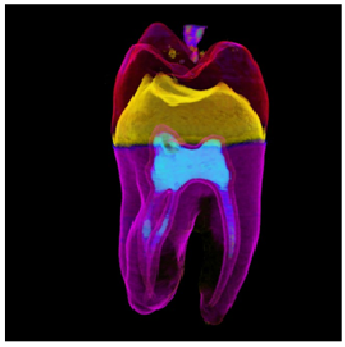


Figura 4.1: Volúmenes utilizados en los casos de pruebas.

4.2 Resultados cuantitativos.

En las mediciones de desempeño y calidad de los distintos tipos de funciones de transferencias, se eligieron las fórmulas y métodos de reconstrucción de mayor precisión. Estas son: la ecuación 3 de la Tabla 3.3 para el cálculo de la segunda derivada y el método suma de convoluciones basado en filtro para la reconstrucción de las derivadas. Luego se realizaron pruebas de rendimiento con cada volumen y tipo de función de transferencia. En la medición de los tiempos de respuesta se utilizaron funciones de medición de OpenMP para las pruebas del método de reconstrucción, mientras que para comparar los tiempos de respuestas de las distintas funciones de transferencia se utilizó la biblioteca *QTime* de Qt.

4.2.1 Elección de la ecuación de la segunda derivada y el método de reconstrucción

Las derivadas de un volumen entre más exacta sean más exacta podrá ser su clasificación, pues permite identificar y aislar mejor una característica en la función de transferencia multidimensional. Por su importancia se necesitó elegir la fórmula que proporcionará más precisión y el método de reconstrucción con más exactitud. Basándonos en el trabajo de *Kniss et al*.[**31**] se eligió la ecuación 3 de la Tabla 3.3 para calcular la segunda derivada (f''), la cual involucra la matriz *hessiana*, una matriz de segundas derivadas parciales. En cambio para la elección del método de reconstrucción, se realizaron pruebas de exactitud a un volumen representativo, para elegir el método más exacto. También, se realizaron pruebas de tiempos para evaluar el desempeño de los dos métodos. Por último se realizó un análisis costo/beneficio entre los tiempos de respuesta y la exactitud de la reconstrucción. Con base al

análisis anterior y el objetivo principal de la aplicación el cual es obtener la mayor precisión en la clasificación, se elige el método utilizado para evaluar el desempeño y calidad de las distintas funciones de transferencias. Es importante mencionar también, que los algoritmos de reconstrucción fueron optimizados utilizando OpenMP para la arquitectura mencionada anteriormente logrando aprovechar la máxima capacidad de computo del procesador (4 hilos); además por cada método se realizaron al menos 10 corridas de sus algoritmos de reconstrucción, para obtener luego un promedio de ellos.

En la prueba de reconstrucción se tomó el volumen **C**, pues presenta en su composición varios materiales y es referenciado en los trabajos presentados por [**31**][**40**]. En la Figura 4.2a y 4.2b se muestram los diagramas de dispersión obtenidos con los dos métodos.

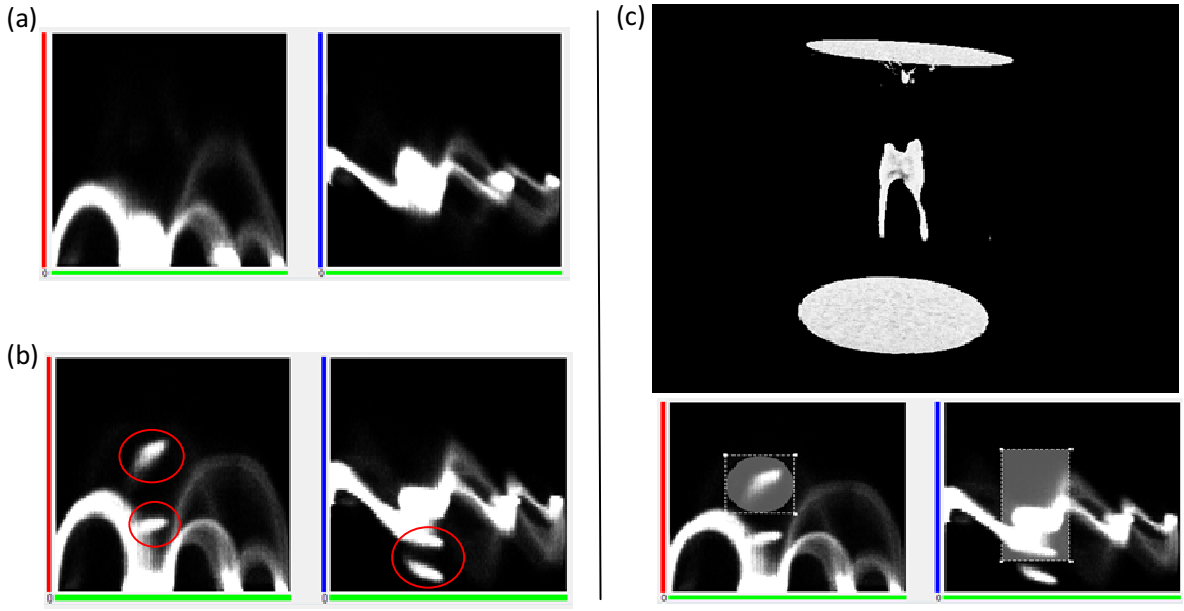


Figura 4.2: En (a) los diagramas de dispersión obtenido con el método de diferencias centrales, en (b) los diagramas de dispersión obtenidos con el método de suma de convoluciones, y en (c) despliegue del volumen C que señala la característica identificada en (b) que no se detecta en (a).

El método de diferencia centrales no permite identificar ciertos materiales que componen el volumen de prueba (ver Figura 4.2a), en cambio el método de suma de convoluciones presenta más precisión, pues muestran las parábolas un poco más definidas además de identificar mejor los materiales (ver Figura 4.2b). En la Figura 4.2c se puede observar un despliegue, el cual representa la característica que no pudo ser identificada con claridad por el método de diferencia centrales. Esta característica identificada representa la parte más interna del diente.

Aunque el método de reconstrucción de diferencia centrales sea poco preciso, su costo computacional es despreciable en comparación al método de suma de convoluciones, como puede ser visto en la Tabla 4.2.

Tabla 4.2: Tabla de los tiempos de respuestas de los dos métodos de reconstrucción.

Método de reconstrucción	Primera derivada (f') (Seg.)	Segunda derivada (f'') (Seg.)	Total (Seg.)
Dif. centrales	0.5312	1.0963	1.6275
Suma de convoluciones	182.6632	333.7489	516.4121

Debido a que se busca generar la mayor precisión posible en la clasificación, se escoge el método de suma de convoluciones para la reconstrucción de las derivadas por su notable exactitud sobre el método de diferencia centrales. Es importante mencionar que el método de diferencia centrales por su rapidez es una buena opción para clasificaciones menos precisas o para una etapa previa de exploración en la clasificación del volumen.

4.2.2 Pruebas de rendimiento en la clasificación

Las mediciones se hicieron en base al tipo de función de transferencia en cada uno de los volúmenes, midiendo la tasa de cuadros por segundos. Para un mismo volumen, se consideran tres funciones de transferencia en la prueba: 1D, 2D y 3D.

Tabla 4.3: Cuadros por segundos presentados por los distintos tipos de funciones de transferencia.

Volumen	Función de Transferencia		
	1D	2D	3D
A	59.66	49.43	31.03
B	26.69	19.35	15.55
C	29.62	23.32	15.16

En la Tabla 4.3 se observa en cada volumen que mientras la dimensión es menor, la tasa de cuadros por segundo es mayor. Este comportamiento ocurre por dos razones: La primera razón es el inherente incremento en la cantidad de muestreos necesarios a medida que se aumenta la dimensionalidad del dominio de la FT. La segunda razón es debido al nivel de especificación del volumen que por definición se puede obtener aumentando la dimensionalidad del dominio de las FT; esto significa que las FT que presentan menor dimensionalidad no permiten alcanzar el nivel de especificación al clasificar el volumen, que las FT de mayor dimensionalidad.

4.3 Resultados cualitativos

A continuación se presentan los resultados visuales de la aplicación con las distintas funciones de transferencias utilizadas en cada volumen. Cada volumen presenta entonces tres funciones de transferencia 1D, 2D y 3D. El objetivo de estas clasificaciones era aislar unívocamente dos características presentes en el volumen utilizando cada una de las funciones de transferencia. Con esto se muestra visualmente las limitaciones que presentan los diferentes tipos de funciones de transferencia.

4.3.1 Volumen A

En este volumen en primera instancia, utilizando la función de transferencia 3D se clasificaron dos características del volumen asignándole el color rojo a la primera característica y blanco a la segunda característica. Luego se buscaron funciones de transferencias 2D y 1D equivalentes. Debido a la sencillez del volumen es un ejemplo ideal para identificar las limitaciones que presentan las funciones de transferencia 1D y 2D con respecto a la 3D, en cuanto al grado de especificación de las características en la clasificación.

En la Figura 4.3 se observan las tres configuraciones de los puntos de control realizadas a las diferentes funciones de transferencia, con sus respectivos despliegues de volumen obtenidos con dichas configuraciones. Como se puede observar en la Figura 4.3 la clasificación realizada con al FT 3D no pudo ser igualada ni por la FT 2D, ni por la FT 1D. Esto es debido a que la segunda derivada permite diferenciar ciertos materiales y contornos que no pueden separarse al utilizar la información del volumen (f) y la información de la primera derivada (f'). También, es importante señalar que la FT 1D no pudo competir con la clasificación realizada con las FT 2D y 3D, debido a que la FT 1D solo toma el valor de los vóxeles (f) para realizar la clasificación.

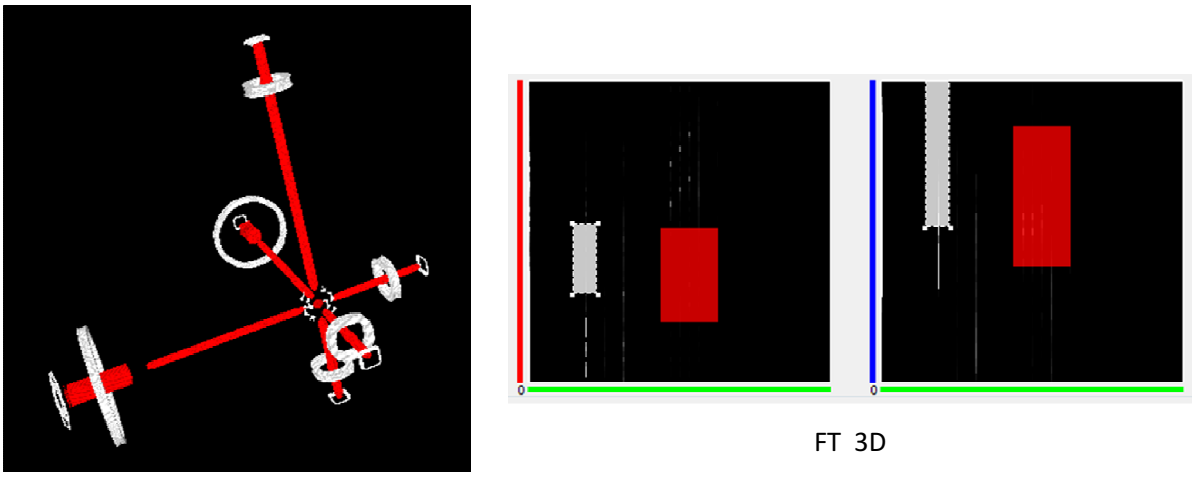


Figura 4.3: Los tres tipos de funciones de transferencia para el volumen 1, con sus respetivos despliegues.

4.3.2 Volumen B

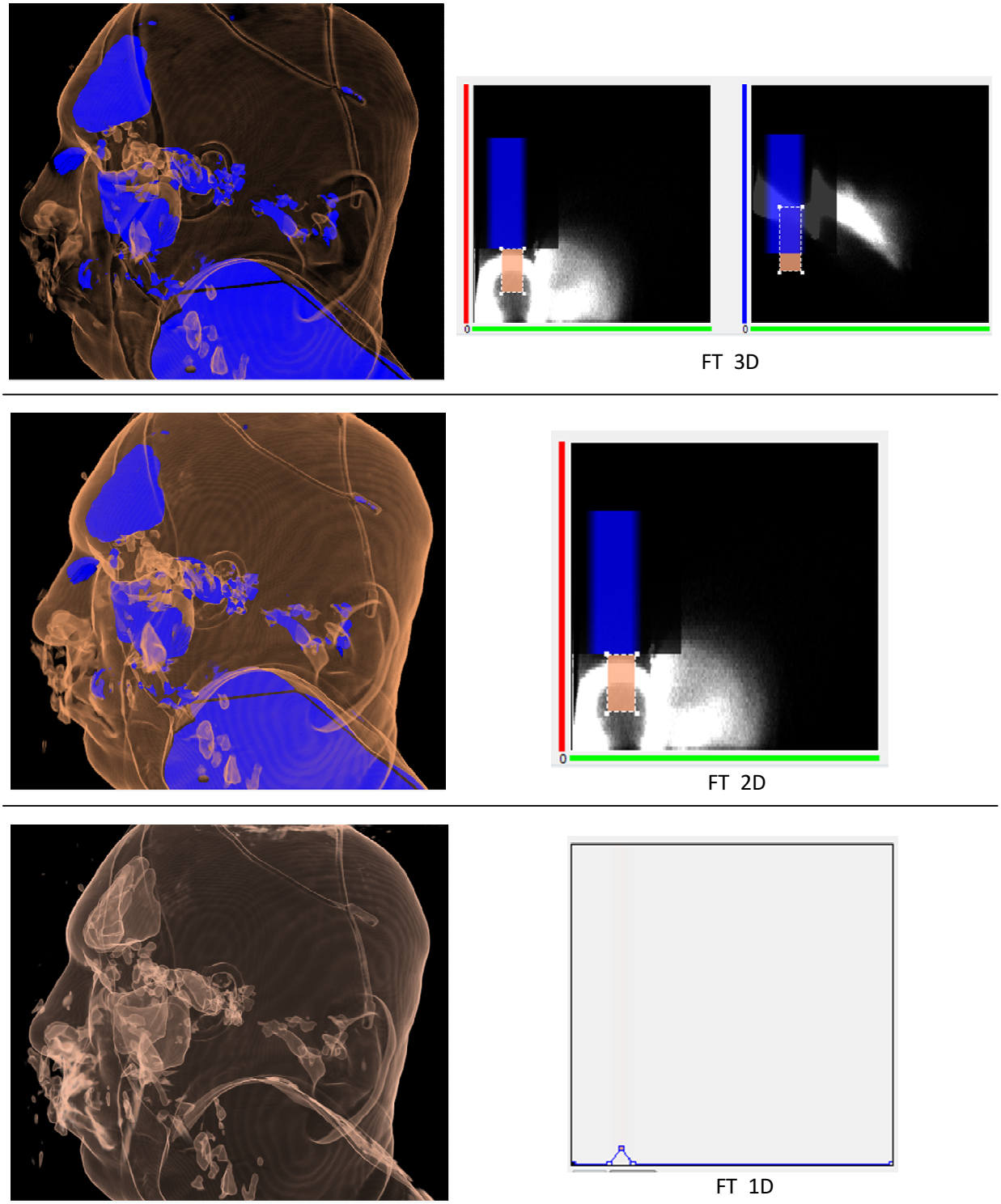


Figura 4.4: Los tres tipos de funciones de transferencia para el volumen 2, con sus respetivos despliegues.

En este volumen se pretende clasificar los senos nasales ubicados en la cabeza humana utilizando cada una de los distintos tipos de funciones de transferencia. Utilizando la FT 1D no

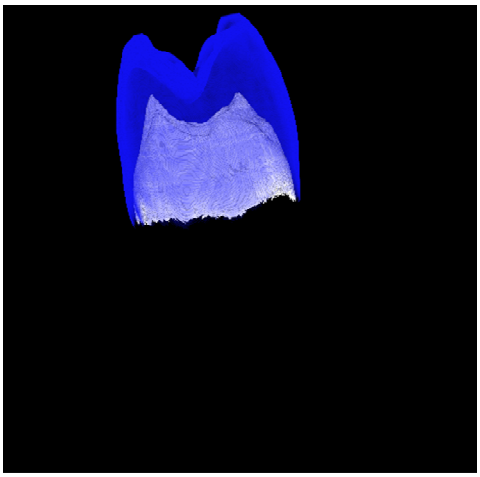
es posible realizar tal propósito (ver Figura 4.4), sólo se puede obtener una aproximación en el cual se puede visualizar los senos nasales pero no se puede aislar para poder asignarle otro color distinto. Caso contrario ocurre en las FT 2D y 3D en los cuales se puede caracterizar los senos nasales con el color azul producto de la expansión de su dominio.

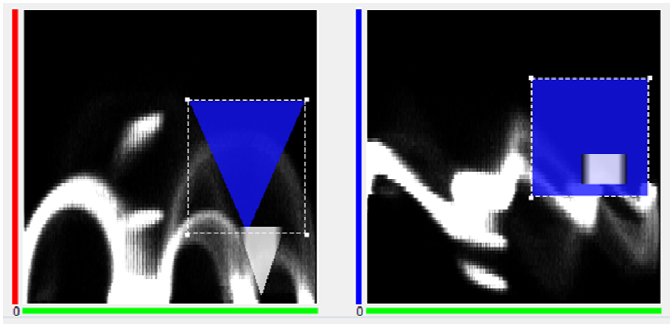
Las dos clasificaciones logradas por las FT 2D y 3D son muy parecidas, sólo un cambio de la intensidad de la piel es notable (ver Figura 4.4). Esta coincidencia es debido a la necesidad de ubicar el punto de control en la tercera dimensión de la FT 3D en donde se acumulan la mayor cantidad de datos para poder realizar la clasificación, caso contrario ocurre en la caracterización de la piel en donde el punto de control en la tercera dimensión se ubica solo en un subconjunto de los datos para los mismos valores de vóxeles. Por ello el cambio de intensidad de la piel entre las FT 2D y 3D. Esta ubicación de los puntos de control en los mismos valores de vóxeles, la gran altura (eje y) que presenta el punto de control y, además se encuentra ubicado en las zona del diagrama de dispersión donde se concentran la mayor parte de los datos (zonas de color blanco), ocasiona que la FT 3D se comporte como una FT 2D; debido a que prácticamente se ignora la tercera dimensión, pues para un rango de vóxeles en la tercera dimensión poseen todos el mismo valor.

4.3.3 Volumen C

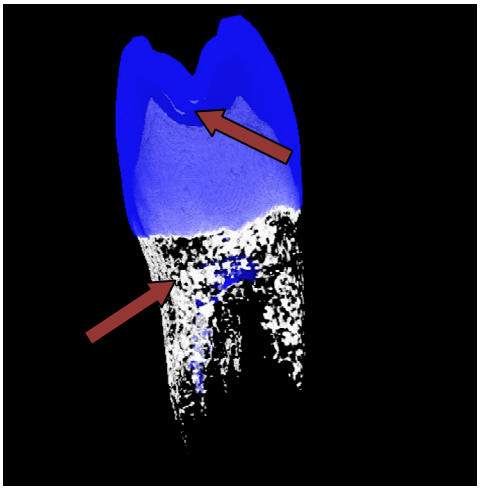
El volumen C presenta las fronteras entre materiales bien definidas, pudiendose notar esto en los histogramas de las FT 2D y 3D de la Figura 4.5. En este volumen se pretende separar las dos características clasificadas con la FT 3D, con las FT 2D y 1D. Con la FT 1D esto no es posible, solo se puede clasificar las dos características como un todo, además de contener otros materiales (ver Figura 4.5) que no son parte de las dos características inicialmente clasificadas con la FT 3D.

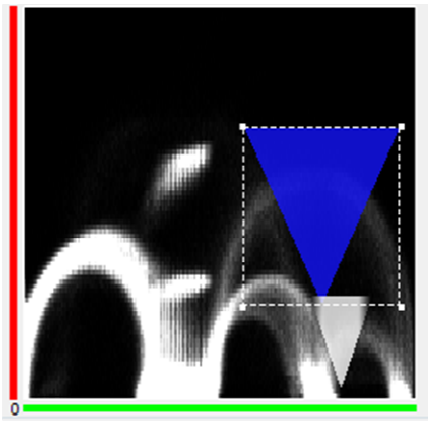
La clasificación de las dos características iniciales utilizando la FT 2D, no se pudieron clasificar de forma univoca (ver Figura 4.5). Esto es debido a que en la clasificación generada utilizando la FT 3D, sí se utiliza su tercera dimensión para separar las fronteras entre materiales que comparten los mismos valores en la segunda dimensión. Entonces la FT 2D no logra la correcta clasificación porque no puede separar estas fronteras complejas, tratándolas a todas por igual.



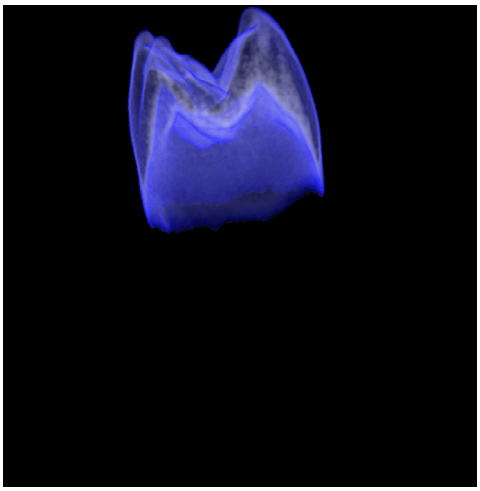


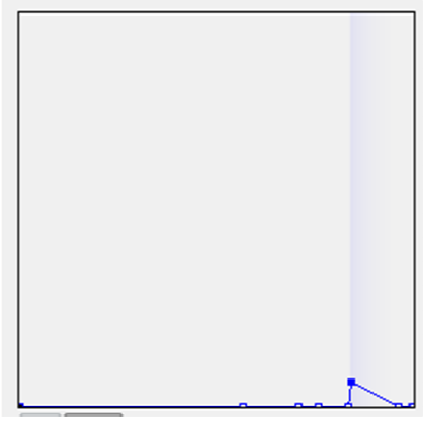
FT 3D





FT 2D





FT 1D

Figura 4.5: Los tres tipos de funciones de transferencia para el volumen 3, con sus respetivos despliegues.

4.4 Mediciones de memoria

En las mediciones de memoria se tomaron las estructuras datos que abarcaban la mayor cantidad de memoria en donde se encuentra el volumen, las derivadas, los vectores gradientes y el volumen que representa la FT 3D. Las mediciones de memoria para los casos de prueba pueden verse en la Tabla 4.4.

Tabla 4.4: Memoria utilizada en los casos de prueba.

Volumen	A	B	C
Tamaño del volumen (MB)	0.25 MB	8 MB	10.0625 MB
Memoria RAM necesaria (MB)	77.5 MB	496 MB	607.375 MB
Memoria GPU necesaria (MB)	65.5 MB	112 MB	124.375 MB

Se puede observar claramente en la Tabla 4.4 que existe una relación entre el tamaño del volumen y la cantidad de memoria necesaria de la aplicación. Es decir a medida que aumenta el tamaño del volumen también aumenta la cantidad de memoria necesaria tanto en memoria principal como en memoria de GPU. La razón es que por cada vóxel del volumen se necesita obtener y almacenar sus derivadas originando un incremento en la memoria. Esta es una de las desventajas de utilizar funciones de transferencia multidimensionales.

Una vez mostrada cada unas de las pruebas y resultados con sus análisis, se llegaron a unas conclusiones, las cuales se presentan el próximo capítulo. Además, se proponen algunos trabajos futuros producto de la investigación.

Capítulo V

Conclusiones y Trabajos futuros

En este trabajo se diseñó e implementó un sistema de visualización que permite la clasificación a través de la manipulación de funciones de transferencias de multidimensionales. El sistema cuenta con una interfaz que permite la edición de la funciones de transferencia a través de la manipulación de puntos de control (punto 1D o figuras 2D y 3D). Además, el sistema permite guardar y cargar las funciones de transferencias, así como también las texturas usadas en la expansión del dominio de la función de transferencia. Por otra parte, el sistema realiza la especificación de la función de transferencia basada en los datos mostrando para ello unos diagramas de dispersión calculados utilizando los valores de los vóxeles y las derivadas, lo cual permite orientar al usuario en la edición de la función de transferencia.

Las derivadas utilizadas por la función de transferencia multidimensional para la expansión de su dominio, se calcula por medio de dos métodos de reconstrucción en el sistema. Estos métodos fueron comparados para seleccionar el método de reconstrucción que permita más precisión en la reconstrucción para luego ser usado en las pruebas. El método de reconstrucción de suma de convoluciones demostró ser más preciso en la reconstrucción, aunque acarrea un costo computacional importante con respecto al método de diferencia centrales.

Utilizando las derivadas obtenidas por medio del método de reconstrucción de suma de convoluciones, se realizaron pruebas de tiempo de respuesta y detalle de la clasificación a los distintos tipos de funciones de transferencia. En dichas pruebas se demostró que las funciones de transferencia multidimensionales permiten aislar las características contenidas en el volumen con mayor precisión que las funciones de transferencia unidimensionales en donde no se pudo seleccionar de manera específica la característica deseada. También se observó una merma en el desempeño en comparación con las funciones unidimensionales; esto indica que muestrear múltiples textura en vez de una afecta los tiempos de respuesta del sistema. También, esta merma en el desempeño de las funciones de transferencia multidimensionales es debido al nivel de especificación que se alcanza en la clasificación a través de ellas, aislando unívocamente la característica, esto ocasiona que mejoras del algoritmo del *ray casting* como la terminación temprana del rayo no beneficie el despliegue realizado con estos tipos de funciones; caso contrario ocurre en las funciones de transferencia unidimensionales en donde no se pudo aislar unívocamente la característica, trayéndose consigo otros materiales pudiendo esto beneficiar la terminación temprana del rayo aplicada en el algoritmo de despliegue.

Las principales desventajas que acarrea el uso de funciones de transferencia multidimensionales son la introducción de una etapa de pre procesamiento en donde se deben calcular las derivadas, y el aumento considerable de la memoria al tenerse que almacenar estas derivadas para la implementación de la función de transferencia.

Como trabajos a futuro se propone realizar el cálculo de las derivadas en algún lenguaje paralelo provisto por el hardware gráfico (OpenCL o CUDA), en donde se permita reducir el tiempo de cálculo cuando se utiliza el método de reconstrucción de suma de convoluciones.

Se recomienda además aumentar el repertorio de figuras de clasificación utilizadas para editar la función de transferencia multidimensional, permitiendo el uso de otras figuras o pinceles para construir cualquier forma que pueda tener alguna frontera entre los materiales.

La edición de las funciones de transferencia multidimensionales, requieren de cierta experticia; así pues se recomienda desarrollar una técnica que permita la especificación de la función de transferencia de forma automática.

Referencias

[1] Nelson Max, "Optical Models for Direct Volume Rendering", in *Visualization in Scientific Computing, Springer*, 1995, pp. 35-40.

[2] Mauricio Cele López Belón, "Visualización Interactiva de Volúmenes", Universidad Nacional del Sur, Argentina, Tesis de Maestria 2007.

[3] Kenneth Moreland, "Fast High Accuracy Volume Rendering", University of New Mexico and Sandia National Laboratories, Tesis doctoral 2004.

[4] Cline Harvey and Lorensen Willian",Marching Cubes: A high resolution 3D surface construction algorithm," in *Computer Graphics*, 1987, pp. Vol. 21 – No. 4.

[5] Arie E Kaufman, "Introduction to Volume Graphics", Center for Visual Computing and Computer Science Department, University of New York , New York , 2000.

[6] Jonathan Richard Shewchuck, "Delaunay Refinement Mesh Generation", Carnegie Mellon University, Tesis de doctorado 1997.

[7] Nelson Max, Peter Williams, Claudio Silva, and Richard Cook, "Volume Rendering for Curvilinear and Unstructured Grids", in *Proceedings of ACM SIGGRAPH 2000*, 2000, pp. 53-61.

[8] D Ebert, F. K. Musgrave, D Peachey, K Perlin, and S Worley, "Texturing and Modeling: A Procedural Approach", in *Academic Press*, 1998.

[9] Marc Levoy, "Display of Surfaces from Volume Data", in *IEEE Computer Graphics & Applications*, 1988, pp. 29–37.

[10] R. A. Drebin, L Carpenter, and P Hanrahan, "Volume Rendering", in *ACM Computer Graphics (SIGGRAPH 88 Proceedings)*, 1988, pp. 65-74.

[11] Sheng Guo, Cage Lu, and Xiaochang Wu, "Dynamic Volumetric Cloud Rendering for Games on Multi-Core Platforms", Software and Services Group, Intel Corporation, 2010.

[12] Paolo Sabella, "A rendering algorithm for visualizing 3D scalar fields", in *Proc. ACM SIGGRAPH 88, Computer Graphics*, 1988, pp. 51–58.

[13] Peter Williams and Nelson Max, "A volume density optical model", in *Proc. Workshop on Volume Visualization 92, Computer Graphics*, 1992, pp. 61–68.

[14] Phillippe G Lacroute, "Fast Volume Rendering Using Shear-Warp Factorization of the Viewing

Transformation", Universidad de Standford, Reporte Técnico 1995.

[15] Rhadamés Carmona, "Visualización Multi-Resolución de Volúmenes de Gran Tamaño", Universidad Central de Venezuela, Caracas, Tesis Doctoral 2010.

[16] Joe M Kniss, "Interactive Volume Rendering Techniques", Departamento de Ciencias de la Computación, Universidad de Utah, Tesis de Maestría 2002.

[17] (2011, Enero) Digital Imaging and Communications in Medicine. [Online]. http://medical.nema.org/

[18] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner, *OpenGL® Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*, 3rd ed.: Addison-Wesley Professional, 1999.

[19] Marc Levoy, "Efficient ray tracing of volume data", in *ACM Trans. Graph.*, 1990, pp. 245–261.

[20] Jonsson Marcus, "Volume Rendering", Umea University, Suecia, Tesis de Maestría 2005.

[21] J Krüger and R Westermann, "Acceleration techniques for GPU-base Voume Rendering",in *Proc. Visualization '03*, Washington, 2003, pp. 287-292.

[22] Lisa Marie Sobierajski, "Global Illumination Models for Volume Rendering",The State University of New York, New York, Tesis de Doctorado 1994.

[23] Markus Hadwiger, Joe M Kniss, Klaus Engel, and Christof Rezk-Salama, "High-Quality Volume Graphics on Consumer PC Hardware", SIGGRAPH, San Antonio, Notas de curso n° 42 2002.

[24] Bui Phong Tuong, "Illumination for Computer Generated Pictures", University of Utah, ACM 1975.

[25] M.J Bentum, B.B.A. Lichtenbelt, and T Malzbender, "Frequency analysis of gradient estimators in volume rendering", in *IEEE Transactions on Visualization and Computer Graphics*, Piscataway, NJ, USA , 1996, pp. 242 - 254.

[26] Torsten Möller, Klaus Mueller, Yair Kurzion, Raghu Machiraju, and Roni Yagel, "Design Of Accurate And Smooth Filters For Function And Derivative", Mississippi State University, Department of Computer Science, Mississippi,.

[27] Torsten Möller, Raghu Machiraju, Klaus Mueller, and Roni Yagel, "Evaluation and Design of Filters Using a Taylor Series Expansion", in *IEEE Transactions on Visualization and Computer Graphics*, 1997, pp. 184-199.

[28] Chandrajit L Bajaj, Valerio Pascucci, and Daniel R Schikore",The Contour Spectrum," in *Proceedings Visualization ´97*, 1997, pp. 167–173.

[29] Taosong He, Lichan Hong, Arie Kaufman, and Hanspeter Pfister, "Generation of Transfer Functions with Stochastic Search Techniques", in *Proceedings Visualization 96*, 1996, pp. 227–234.

[30] Jirí Hladuvka, Andreas Konig, and Eduard Groller, "Curvature-Based Transfer Functions for Direct Volume Rendering", in *Spring Conference on Computer Graphics 2000*, 2000, pp. vol. 16,58–65.

[31] Joe Kniss, Gordon Kindlmann, and Charles Hansen, "Multidimensional Transfer Functions for Interactive Volume Rendering", in *IEEE Transactions on Visualization and Computer Graphics*, 2002, pp. 270-285.

[32] Joe Kniss, Gordon Kindlmann, and Charles Hansen, "Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets", , 2001.

[33] Daniel Weiskopf, Klaus Engel, and Ertl Thomas, "Volume Clipping via Per-Fragment Operations in Texture-Based Volume Visualization", in *Proc. of IEEE Visualization'02*, Germany, 2002, pp. 93-100.

[34] Fan-Yin Tzeng, Eric B Lum, and Kwan-Liu Ma, "An Intelligent System Approach to Higher-Dimensional Classification of Volume Data", in *IEEE Transactions on Visualization and Computer Graphics*, 2005, pp. 1-11.

[35] Klaus Engel, Martin Kraus, and Thomas Ertl, "High Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading", in *Proc. Siggraph/Eurographics Workshop on Graphics Hardware*, California, 2001, pp. 9-16.

[36] Gordon Kindlmann, "Transfer Functions in Direct Volume Rendering: Design, Interface, Interaction", in *Scientific Computing and Imaging Institute*, Utah, 2002.

[37] Wolgang Krueger, "The application of transport theory to visualization of 3-D scalar data fields. ", in *Computers in Physics*, 1991, pp. 397–406.

[38] Zhi-Pei Liang and Paul Lauterbur, "Principles of Magnetic Resonance Imaging", in *IEEE Press*, New York, 2000.

[39] David Rodgman and Min Chen, "Refraction in Discrete Ray Tracing", in *Proceedings Volume Graphics 2001*, 2001, pp. 3–17,403.

[40] Gordon Kindlmann and James W Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering", in *IEEE Symposium On Volume Visualization*, 1998, pp. 79-86.

[41] Eric Lum and Kwan-Liu Ma, "Lighting Transfer Functions Using Gradient Aligned Sampling", in *Proceedings of IEEE Visualization 2004*, 2004, pp. 289–296.

[42] Ivan Viola, Armin Kanitsar, and Eduard Groller, "Importance-Driven Feature Enhancement in

Volume Visualization", in *IEEE Transactions on Visualization and Computer Graphics*, 2006, pp. 95-106.

[43] Jianlong Zhou, Andreas Doring, and Klaus D Tonnies, "Distance Transfer Function Based Rendering", University of Magdeburg, Germany, Reporte Técnico 2004.

[44] Hanspeter Pfister et al., "The Transfer Function Bake-Off", in *IEEE Computer Graphics and Applications*, 2001, pp. 16–22.

[45] J Marks et al., "Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation", in *Proceedings IEEE Visualization 1996*, 1996, pp. 389-400.

[46] Andreas Konig and Eduard Groller, "Mastering Transfer Function Specification by Using VolumePro Technology", in *Spring Conference on Computer*, 2001, pp. 279–286.

[47] Object Management Group. (2011, Septiembre) Unified Modeling Language. [Online]. http://www.uml.org/

[48] Nokia. (2011, Septiembre) Qt. [Online]. http://qt.nokia.com/

[49] OpenGL. (2011, Septiembre) Open Graphics Library. [Online]. http://www.opengl.org/

[50] OpenGL. (2011, Septiembre) The OpenGL Extension Wrangler Library. [Online]. http://glew.sourceforge.net/

[51] OpenMP. (2011, Septiembre) The OpenMP® API specification for parallel programming. [Online]. http://openmp.org/

[52] U.S. National Library of Medicine. (2011, Septiembre) Visible Human Project. [Online]. http://www.nlm.nih.gov/research/ visible/visible_human.html

[53] Volvis.org. (2011, Septiembre) The Volume Library. [Online]. http://www9.informatik.uni-erlangen.de/External/vollib/