Estudio y Diseño: Renderizado Volumétrico con AMILab

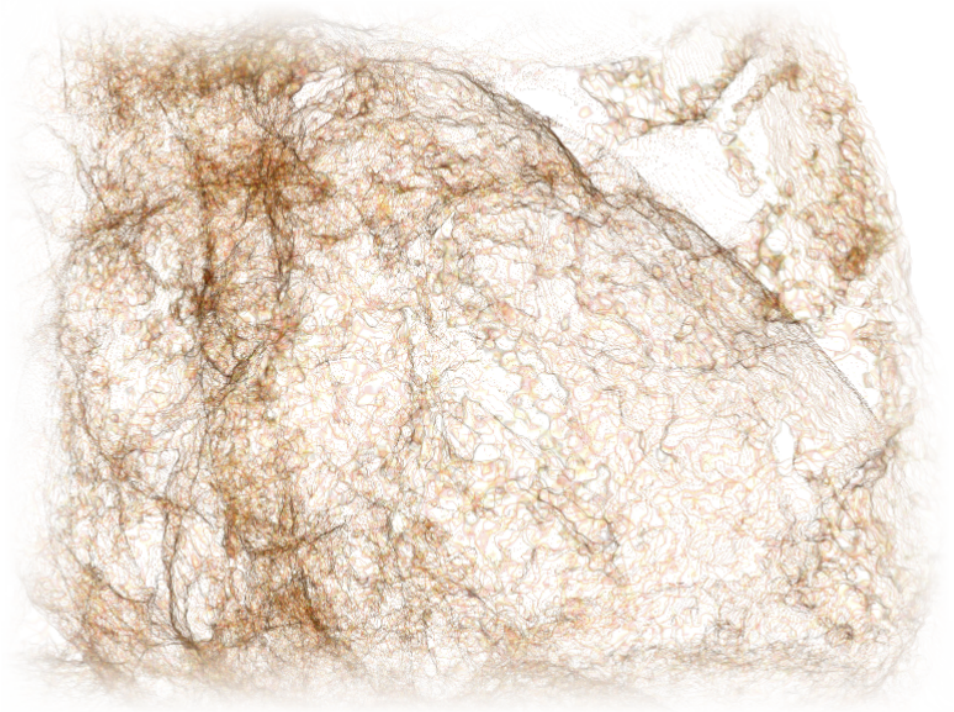
Carlos Falcón Torres

Trabajo Fin de Máster Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI)

Antonio Carlos Domínguez Brito

Karl Krissian

Al equipo de desarrollo de AMILab.
Gracias.



Índice general

Índice de figuras

Capítulo 1

Introduccción

Las técnicas de renderizado representan un gran activo en el desarrollo de los gráficos por computador. Esto es debido al amplio campo de aplicación e investigación existente, y más aún si tenemos en cuenta que toda naturaleza es por definición volumétrica y está definida en un dominio multidimensional. El renderizado volumétrico ha sido y es de interés para las aplicaciones científicas y/o de ingeniera que requieren la visualización de conjuntos de datos tridimensionales. Los casos más comunes de aplicación los encontramos en el campo de:

- La medicina, donde se emplea el renderizado como una herramienta decisiva en el diagnóstico, la visualización y el estudio del cuerpo humano.
- El cine, utilizándose para la creación de efectos especiales como nubes, explosiones, fuego, humo, etc.
- El diseño y análisis de maquinaria para la búsqueda de defectos.
- El análisis sísmico y el estudio de los fenómenos que ocurren en la corteza terrestre.
- La aeronáutica donde se emplean para el estudio de la aerodinámica de los aviones.
- En la Física y en la Química utilizados para la visualización de las nubes electrónicas y la estructura de los átomos y materiales.

La formulación clásica para el renderizado volumétrico fue propuesto por Levoy en 1988 [14]. Este modelo es una versión muy simplificada de la teoría del transporte en la luz en la que solo se utilizaba la emisión, la absorción y el sombreado (*shading*) superficial. Este modelo ha sido una referencia a lo largo de los años y con el paso del tiempo se adaptó a las técnicas de renderizado volumétricos por hardware, Cabral, B. & Foran [3], y Wilson, O. & Gelder en 1994 [36], y posteriormenye por C. RezkSalama & K.Enge, en el 2000 [21]. En la actualidad la mayoría de los esfuerzos en las técnicas de renderizado se centran en este campo, en el que se intentan explotar el 100 % de los recursos hardware.

El objetivo principal de este Trabajo Fin de Máster (TFM) se centra en el uso de las técnicas de representación de volúmenes como medio de visualización de resultados, principalmente en el campo de las imágenes médicas, que es la principal área de conocimiento del Grupo de Imagen, Tecnologa Médica y Televisión (GIMET) [33] que forma parte del Centro de Tecnologías de la Imagen [31]. En el planteamiento seguido se han fijado una serie de objetivos:

- Estudio del estado del arte de los métodos de renderizado volumétrico. (60 horas)
- Familiarización con el software AMILab. (90 horas)
- Programación de una interfaz de renderizado volumétrico para AMILab que permita combinar varios volúmenes en una misma escena, establecer un sistema de configuraciones predefinidas, cambiar en tiempo real las vistas de proyeccin y, los componentes de la función de transferencia (210 horas)
- Generación de la documentación y memoria. (90 horas)

Esta memoria se encuentra organizada en 5 capítulos tal y como exige el reglamento interno del máster de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI):

- Introdución, en este primer capítulo se realiza una pequeña introducción sobre la temática y se comentan los objetivos propuestos.
- Contextualización, en este capítulo se realiza un estudio de los antecedentes y estado actual de la materia.

- Fundamentos Teóricos y Desarrollo, en este capítulo se definen los conceptos básicos a manejar, se describen las técnicas utilizadas y, la interfaz y algoritmos implementados.
- Resultados, este capítulo está dedicado a describir los resultados
- Conclusiones y Líneas futuras, por último se describirán las conclusiones y líneas futuras de este TFM.

Capítulo 2

Contextualización

2.1. AMILab

AMILab [27] es un software multiplataforma para el procesado digital de imágenes que cuenta con visualizacidores 2D y 3D. Se trata de un software libre y gratuito, bajo licencia LGPL [15] y disponible para su descarga en `amilab.org`. El software actualmente está orientado tanto para el uso académico, como para la investigación, especialmente dentro del campo de tratamiento de imágenes médicas.

AMILab engloba varias funcionalidades dentro de un solo software, permitiendo crear y combinar nuevas funcionalidades con los ya existentes. Todo ello gracias a su potente lenguaje de *scripts* orientado a objetos. A la hora de diseñar y desarrollar AMILab se han tomado como referencia softwares de la talla de: 3D Slicer, Osirix, Matlab, Octave, SCIRun, Mevislab, Gimp, entre otros. Cada uno de estos programas proporcionan diferentes características que AMILab trata de incorporar. Destacando las herramientas de visualización 2D, 3D y series temporales, los algoritmos de procesado de imágenes, entre los que se encuentran algunos propios desarrollados por el investigador Karl Krissian, y la posibilidad de usar un lenguaje de *scripts* estilo a python, matlab y/o perl. Otro aspecto a destacar en AMILab son sus herramientas para la creación de interfaces gráficas de manera fácil e intuitiva sin necesidad de compilar el código.

En general, es difícil encontrar un software que satisfaga todas las necesidades, por lo que AMILab permite que cada usuario personalice el software a su gusto, bien modificando su diseño, y/o añadiendo nuevas funcionalidades.

AMILab cuenta con un manual de ayuda online y un equipo de desarrollo que atiende rápidamente las consultas y/o sugerencias.

2.1.1. Estructura de AMILab

El diseñado y estructura de AMILab ha ido evolucionado a lo largo de su desarrollo, mejorando tanto en su aspecto como en sus funcionalidades, esto se debe gracias a la aparición de nuevas tecnologías y herramientas. Entre las características más importantes de AMILab destacan los siguientes aspectos:

- La estabilidad de su núcleo, desarrollado con lenguajes orientados a objetos, C++ y Python. La implementación actual cuenta con una estructura multihilos en la que se han utilizando las librerías Pthreads [2] y Boost C++ [29], minimizándose tanto los tiempos de ejecución como de compilación.

- Posee un sistema de *wrapping* (empaquetado) automático, Figura 2.2, que hace que actualmente AMILab cuente con la contribución de más de 500 clases de las librerías VTK [23], ITK [17] y wxWidgets [24] entre otras.

- Cuenta con su propio intérprete y gramática realizados con las herramientas Flex [32] y Bison [28] .

Para conseguir la autimatización del proceso de compilación de AMILab y contribuir a que el software siga una política multiplataforma de manera sencilla se hace uso de un complejo y entramado sistema de makefiles utilizando la herramienta CMake [30]

En la Figura 2.1 puede verse una descripción aproximada de la arquitectura actual del software AMILab. En la *Capa de Middleware* se encuentran los paquetes y subsistemas de la tecnología que son usados en la implementación. En la *Capa Genérica de Aplicación* están los paquetes de servicio y aquellos que son muy genéricos, o de los que dependen bastantes subsistemas o paquetes. Destacar que en el caso de *wrapping* automático, este se produce por demanda. Es decir, no se produce sistemáticamente a lo largo de todas y cada una de las clases que forman parte de la librería, sino que se hace solamente para las indicadas en el fichero *classes.txt* de la configuración del *wrapping* de cada una de ellas. Por último, en la *Capa Específica de Aplicación* están los paquetes y subsistemas que forman parte de la aplicación. Por

simplicidad, en esta última capa se ha intentado hacer una abstracción del total de paquetes y subsistemas que existen en realidad dentro del software.

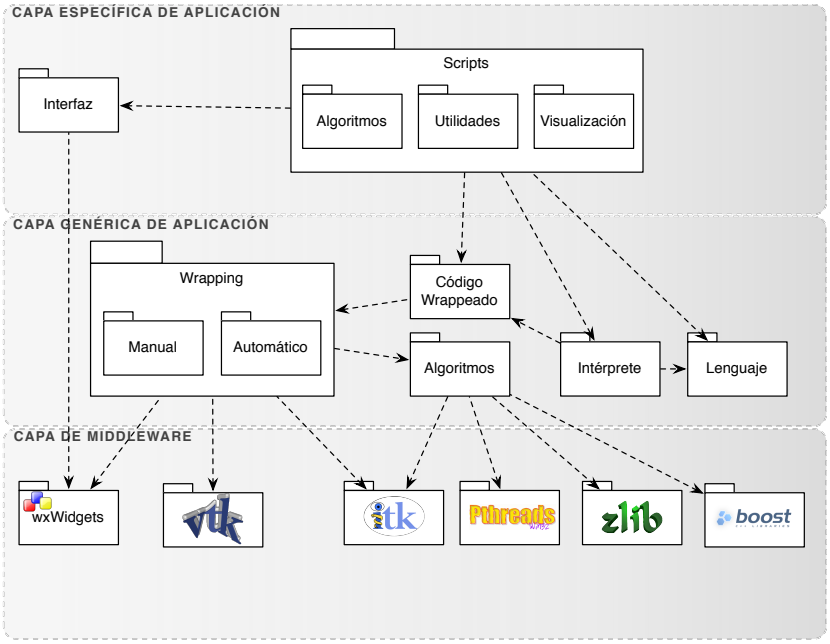


Figura 2.1: Arquitectura de AMILab

2.1.2. Alternativas a AMILab

Dentro de la problemática en la que se engloba este TFM, las principales alternativas al renderizado volumétrico dentro de AMILab son las incluidas por los softwares:

- 3D Slicer: 3D Slicer [18] es una plataforma flexible que puede ser fácilmente extendida para permitir el desarrollo tanto de herramientas interactivas como por lotes. Posee funcionalidades para el registrado de imágenes, procesado de tractografía, cuenta con una interfaz para dispositivos externos con soporte guiado por imagen, renderizado volumétrico por GPU, entre otras. 3D Slicer tiene una organización modular que permite la adición de nuevas funcionalidades de forma sencilla,

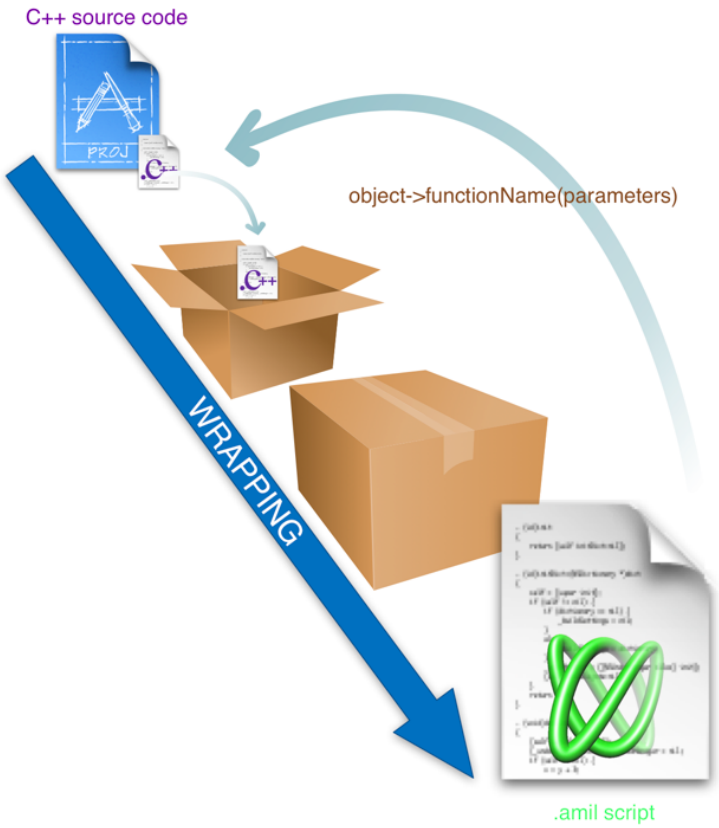


Figura 2.2: Sistema de *wrapping* en AMILab

además provee un número de características genéricas que no están disponibles en las herramientas de la competencia. 3D Slicer está distribuído bajo una licencia BSD no restrictiva.

- Osirix: Osirix [22] es un software gratuito desarrollado en ObjectiveC sobre la plataforma Macintosh usando el entorno de desarrollo GNUstep. Se encuentra disponible al público en el sitio web de Apple Inc. Osirix ha sido específicamente diseñado para la navegación y visualización de imágenes multimodales y multidimensionales: Visor 2D, Visor 3D, Visor 4D (series 3D con dimension temporal) y visor 5D (series 3D con dimensiones temporales y funcionales). El Visor 3D ofrece todos los modos de renderizado modernos: Reconstrucción multiplanar,

renderizado de supercie, renderizado volumétrico y proyección de máxima intensidad. Osirix incorpora una estación DICOM (Digital Imaging and Communication in Medicine)[5] de sistema de archivado y transmisión de imágenes para imágenes médicas y un paquete software de procesado de imágenes para investigación médica (radiología e imágenes nucleares), imágenes funcionales, imágenes 3D, microscopía confocal e imágenes moleculares. Los visualizadores biomédicos pueden usar este software para observar conjuntos de datos anatómicos y extraer información visual de referencia [16].

2.2. Renderizado Volumétrico [37]

La visualización de volumenes es un método con el que se obteniene información de interes a partir de un conjunto de datos volumétricos al aplicarse técnicas de gráficos interactivos y/o imágenes. Entre dichas técnicas encontramos la representación de volumenes, el modelado, la manipulación y el dibujado. Los volúmenes 3D son entidades con información en su interior, pero no necesariamente son superficies y bordes, sino que también pueden ser grandes representaciones geométricas. Estas se suelen obtener mediante muestreo, simulación o técnicas de modelado. Como por ejemplo, secuencias de cortes 2D obtenidos en una resonancia magnética (MRI), de una tomografía computarizada (CT) o desde una tomografía por emisión de positrones (PET), estas son reconstrucciones 3D de modelos volumétricos cuyo objetivo es llegar a ser una buena representación de estos para que un experto pueda realizar diagnósticos a partir de ellos, y establecer un plan de tratamiento o de cirugía. Esta tecnología también se utiliza a menudo en la industria, con el objetivo de inspección de materiales o piezas mecánicas. En el campo de la biología, con el uso de microscopios como fuente de los datos para el estudio de la morfología de las estructuras biológicas. Otras aplicaciones las encontramos en los campos de cálculo, dentro del estudio de la dinámica de fluidos computacional, donde los resultados de las simulaciones se realizan en supercomputadores y a menudo se visualizan como un volumen de datos para realizar su análisis y verificación.

Recientemente, el área de renderizado de volúmenes ha vuelto a cobrar interés en la comunidad, así como en las aplicaciones informáticas tipo CAD (Diseño asistido por computador) y simuladores de vuelos, debido a la aparición de nuevas técnicas. Con los años se han desarrollado numerosas técnicas

para representar datos volumétricos, yendo desde los métodos de representación de primitivas hasta los métodos de aproximación a superficies sin el uso de información de estas. Cuando se visualiza un volumen con las técnicas anteriores, siempre se perdía la referencia de una dimensión al representar los volúmenes. Para evitar esto las nuevas técnicas de renderizado tratan de mostrar todos los datos 3D en una única imagen 2D. El renderizado volumétrico aporta más información y ofrece una mejor calidad que el renderizado con superficies, pero a costa de aumentar la complejidad del algoritmo y el consiguiente tiempo de computación. Para mejorar esta problemática surgieron varios métodos de optimización, tanto software, como hardware.

2.2.1. Técnicas de renderizado volumétrico

El renderizado volumétrico es el proceso de creación de una imagen 2D directamente de los datos 3D. A diferencia de los métodos de renderizado con superficies, las técnicas de renderizado volumétrico operaran con muestras de datos reales, sin generar una representación geométrica intermedia. Las técnicas de renderizado más comunes son las de ordenación de objeto, las de ordenación de imagen o las híbridas. La técnica de ordenación de objetos es una técnica de renderizado basada en un esquema de asignación donde el volumen de datos se proyecta sobre el plano de la imagen desde el final. En los algoritmos de ordenación de imágenes, el esquema de mapeo se utiliza cuando los rayos son emitidos por cada píxel en el plano de la imagen a través de los datos para determinar el valor final del píxel. En las técnicas basadas en el dominio el volumen de datos es transformado en un dominio alternativo, como la compresión, frecuencia, entre otros, para luego generar una proyección directamente de ese dominio.

Ordenación de imagen

La técnica de ordenación de imagen se centra en el plano de la imagen como punto de partida del algoritmo. La idea principal es que haya por cada píxel de la imagen final un rayo en la escena. El rayo es muestreado siguiendo la evaluación de la ecuación 2.1. La técnica de renderizado volumétrico fue propuesta por Levoy [14] siendo esta una buena representación de este enfoque. En la figura 2.3 se ilustra el concepto. La complejidad computacional

de estos algoritmos se rige por el número de píxeles de la imagen final.

$$I_\lambda = \sum_{k=1}^{M} C_\lambda(s_k)\alpha(s_k) \prod_{i=1}^{k-1} (1 - \alpha(s_i)) \tag{2.1}$$

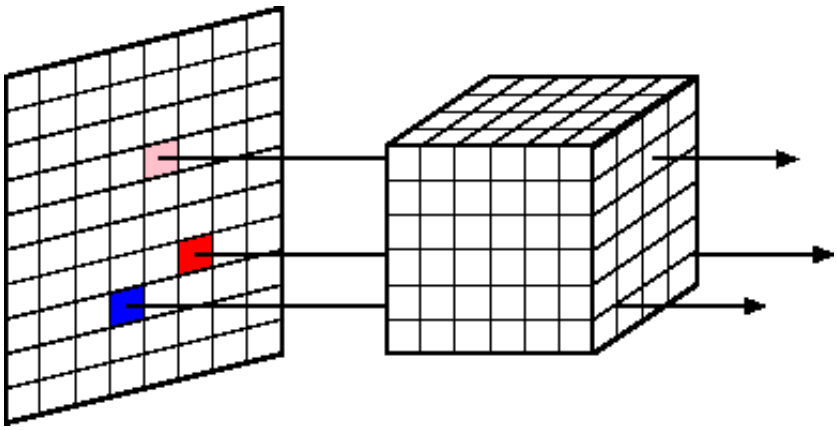


Figura 2.3: Representación de la técnica de ordenación de imagen

Ordenación de objeto

La información del volumen es el centro de interés en el método de ordenación de objeto. La operación de renderizado se lleva a cabo por cada vóxel del conjunto de datos. Un ejemplo de este tipo de representación es el *Splatting* introducida por Westover [35]. Cada vóxel es visto como una partícula y se proyecta en el plano de la imagen, donde se crea una huella de acuerdo a su color y la opacidad. Los vóxeles más cercanos al plano de la imagen se mezclan con los más alejados. Con esto se logra una aproximación a la ecuación de renderizado simplificada. En este tipo de técnicas el acceso a la memoria alineada puede suponer una importante aceleración en el tiempo de procesamiento, en comparación con la de renderizado volumétrico. La complejidad computacional de los métodos de ordenación de objeto es determinada por el número de vóxeles del volumen. En la Figura 2.4 se muestra la representación de este método.

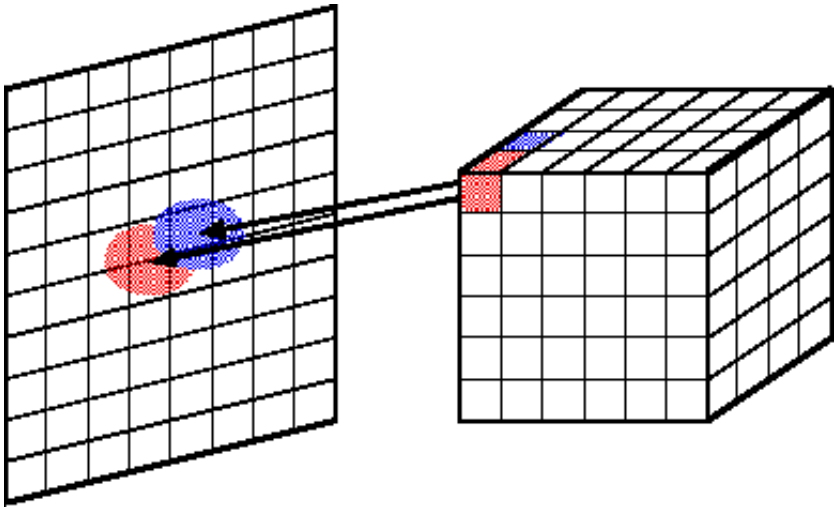


Figura 2.4: Representación de la técnica de ordenación de objeto

Híbridas

Los métodos anteriores presentan distintas ventajas e inconvenientes, es por ello que nacen las técnicas de ordenación híbridas combinando las ventajas de cada uno y creando un algoritmo de procesamiento rápido y preciso. La factorización de corte-deformación propuesta por Lacroute y Levoy [12] es el algoritmo más rápido basado puramente en software de renderizado volumétrico. La idea conceptual es transformar el volumen en un sistema de coordenadas intermedia llamado "corte espacial del objeto". La definición de este espacio es que todos los rayos de visión son paralelos a los ejes de coordenadas. En la Figura 2.5 se muestra la transformación para la proyecciń paralela. Las líneas horizontales representan los cortes de los datos que son atravesados por los rayos. Un conjunto de estos segmentos representan el volumen almacenado por cada eje de coordenadas. El conjunto más perpendicular a la dirección de la vista se selecciona y se transforma con el fin de establecer la vista de los rayos perpendicular a los cortes, pudiendo ser mezclados en una imagen intermedia de atrás a adelante. El paso de deformación elimina las distorsiones en la imagen intermedia y realizando rotaciones en torno al eje de visión. Este proceso se representa en la Figura 2.5. La razón por la que se considera una técnica híbrida es porque los cortes de los vóxeles del volumen se forman en la imagen intermedia, que se considera una operación

de ordenación de objeto. Después para cada píxel de la imagen final se aplica una correspondencia de posición con la imagen intermedia, calculándose el color final del pxel. Este último paso de la deformación pertenece al método de ordenación por componete.

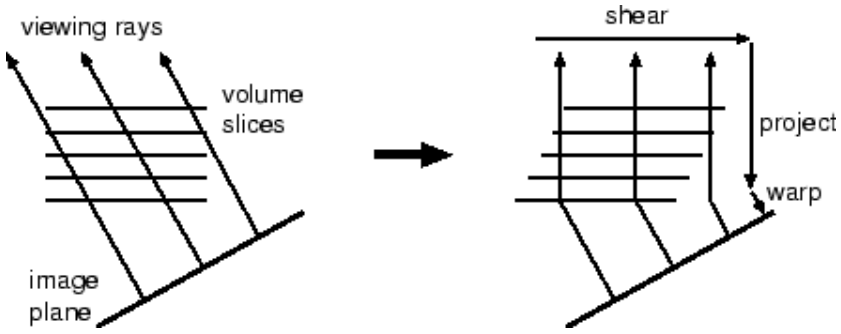


Figura 2.5: Esquema de la factorización de corte-deformación

Mapeo de texturas

A medida que ha ido mejorando el hardware dedicado a gráficos el mapeo de texturas se ha vuelto una técnica interesante. Uno de los primeros en usar este tipo de técnicas fue Cabral [3]. Al principio solo se soportaban las texturas 2D. El concepto es muy similar a la factorización de corte-deformación. El volumen se divide en un conjunto de cortes para cada eje de coordenadas. El grupo de corte más perpendicular al eje de visión es el elegido para la representación, esto se hace mediante la asignación de la información sobre un conjunto de triángulos que poseen la misma configuración geométrica que de los cortes que representan. Por último se vuelca la transformación de los triángulos y el mezclado dentro del buffer principal del hardware. La metodología de la técnica se muestra en la Figura 2.6. La aparición de las tarjetas gráficas que cuentan con texturas en 3D han revolucionado la metodología. El volumen de datos ahora se almacena en la memoria interna del propio hardware, y en caso de que un triángulo intersecte la textura 3D, el plano de corte se proyecta en el triángulo. Para utilizar esta técnica en el renderizado volumétrico, se deben de dibujar un montón de cortes paralelos al plano de proyección de la intersección del volumen. Una vez más la transformación geométrica y la composición se realiza en hardware. La configuración para

los diferentes ángulos se muestran en la Figura 2.7.

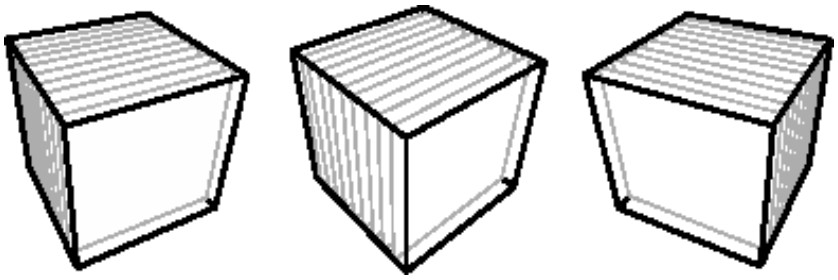


Figura 2.6: Aproximación de mapeo 2D

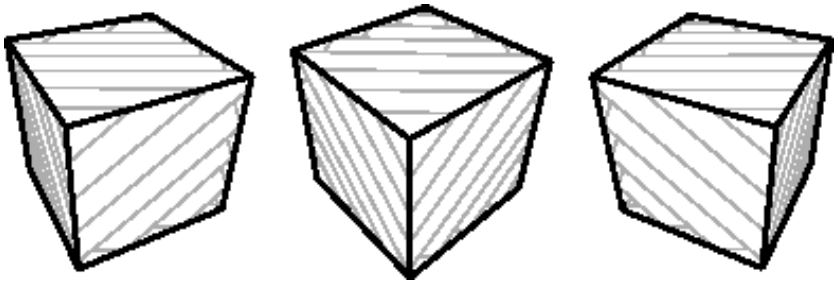


Figura 2.7: Aproximación de mapeo 3D

Capítulo 3

Fundamentos Teóricos y Desarrollo

3.1. Conceptos

3.1.1. Vóxel

Vóxel (Píxel volumétrico), Figura 3.1, es una manera de representar objetos volumétricos como bitmaps tridimensionales en lugar de como vectores. Es decir, es la unidad cúbica que compone un objeto tridimensional, análogamente a lo que es un píxel en 2D. Estos no suelen tener su posición (x, y, z) explícitamente codificada junto con sus valores sino que se deduce basándose en su posición con respecto a otros vóxeles en el archivo de datos.

Un vóxel representa un subvolumen cúbico con un valor constante en su interior. Este valor es igual al de un grid/píxel de la representación original de los datos volumétricos. Los límites de un vóxel están exactamente en el medio de los grids vecinos. Los conjuntos de datos de un vóxel tienen una resolución limitada ya que el único dato preciso de un vóxel está en el centro de cada celda.

Suponiendo que los datos de muestreo de un vóxel se distribuyan como una una señal de banda limitada, se pueden realizar las reconstrucciones exactas de puntos de datos entre los vóxeles de la muestra con un filtro de paso bajo. Se pueden hallar aproximaciones visualmente aceptables a este filtro mediante una interpolación polinómica como la tri-cúbica.

Los puntos y los polígonos son a menudo representados explícitamente por las coordenadas de sus vértices, a diferencia de los píxeles y los vóxeles. Una

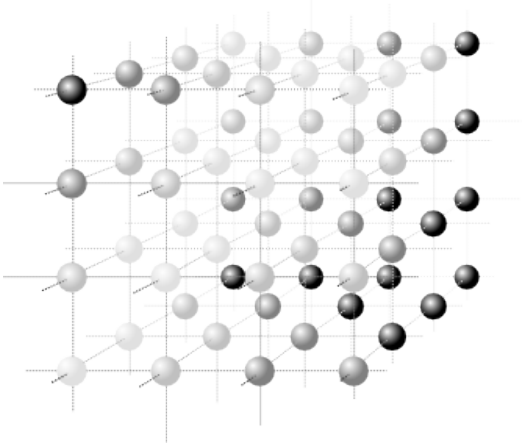


Figura 3.1: Representación tridimensional de vóxeles (cortesía de wikipedia)

consecuencia directa de esta diferencia es que los polígonos son capaces de representar estructuras 3D sencillas, con muchos espacios rellenos de forma homogénea, mientras que los vóxeles representan mejor espacios regularmente distribuidos que no están rellenos de forma homogénea.

3.1.2. Tomografía Axial Computarizada (TAC/CT)

Se trata del método más antiguo de escaneo médico tridimensional, Figura 3.2. Consiste en la imagen de un corte o sección de un objeto computarizada a partir de una serie de imágenes de rayos X. Con la TAC, las imágenes de rayos X son tomadas desde diferentes ángulos alrededor del sujeto de prueba. Los datos de las imágenes son combinados y reconstruidos sección por sección a datos tridimensionales gracias a operaciones matemáticas.

3.1.3. Imagen por Resonancia Magnética (IRM)

Este método es uno de los escanéos médicos más modernos, utiliza el fenómeno de la resonancia magnética para obtener información sobre la estructura y composición del cuerpo a analizar, Figura 3.3. Funciona al medir señales por protones dentro de los átomos de hidrógeno. A veces los conjuntos de volúmenes de datos son combinados. Por ejemplo, dos técnicas diferentes, IMA y TAC, pueden ser usadas en el mismo objeto. Así, los conjuntos de

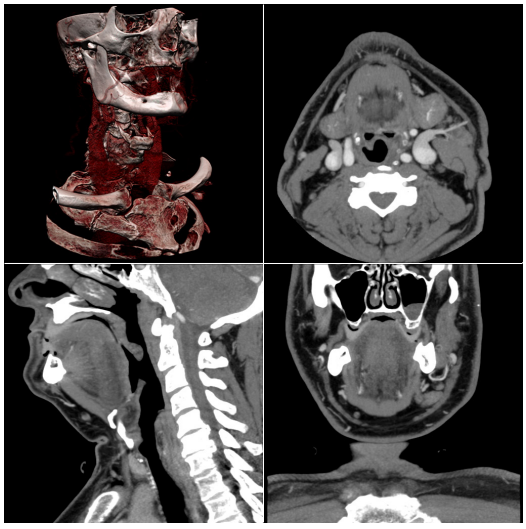


Figura 3.2: Tomografía Axial Computarizada (cortesía de wikipedia)

datos son combinados para mostrar los resultados más importantes de cada método.

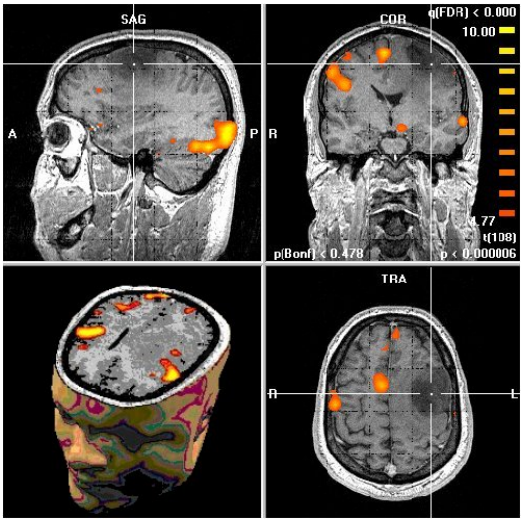


Figura 3.3: Resonancia Magnética

3.1.4. DICOM

DICOM (Digital Imaging and Communications in Medicine)[5] es el estándar de la industria de transferencia de imágenes radiológicas y de otro tipo de información médica entre los equipos. DICOM permite la comunicación digital entre sistemas de equipos de diagnóstico y terapéuticos de los distintos fabricantes.

Dicha conectividad es importante al tener en cuenta la relación costo-eficacia en la atención de los pacientes. Los usuarios de este sistema pueden proporcionar servicios desde sus propias instalaciones o desde cualquier región gracias al sistema en red. Otra ventaja es la compatibilidad del formato con los nuevos equipos y sistemas. Como resultado, las imágenes se pueden capturar y comunicar con mayor rapidez, y a su vez los especialistas pueden hacer un diagnóstico más rápido, mejorando las decisiones de los tratamientos a realizar. El estándar DICOM 3.0 ha evolucionado desde la versión 1.0 (1985) y la 2.0 (1988) desarrollado por la American College of Radiology (ACR) y la National Electrical Manufacturers Association (NEMA). Para apoyar este nuevo estándar el RSNA Electronic Communications Committee comenzó a trabajar con la ACR-NEMA MedPacs en 1992.

Los ficheros DICOM corresponden a la parte 10 del estándar DICOM, en el que se describe como almacenar información de imágenes médicas en un medio extraíble. Generalmente es obligatorio incluir también los metadatos de la imagen. DICOM restringe los nombre de los ficheros a nombres de 8 carácteres. Del nombre del fichero no debe de extraerse ninguna información, lo que suele ser una fuente común de problemas con contenidos creados por desarrolladores que no han leído la especificación cuidadosamente. Se trata de un requerimento histórico para mantener compatibilidad con antiguos sistemas existentes. También es obligatorio la presencia de un directorio de contenido, el fichero DICOMDIR, que proporciona un índice e información de resumen para cada uno de los ficheros DICOM del contenido. La información del DICOMDIR proporciona substancialmente más información sobre cada fichero, de manera que hay menos necesidad de nombres de fichero con significado. En la página del software OsiriX [6] se pueden encontrar una gran colección abierta y gratuita de ejemplos de ficheros DICOM.

3.1.5. VTK

VTK [23] es un sistema de software libre para la realización de gráficos 3D por computador, procesamiento de imagen y visualización. VTK consiste en una biblioteca de clases de C++ y varias capas de interfaz interpretadas como Tcl/Tk, Java, y Python. Kitware, cuyo equipo creó y sigue ampliando el Kit de herramientas, ofrece apoyo profesional y servicios de consultoría para VTK. Esta librería soporta una amplia variedad de algoritmos de visualización como: escalar vector Euclides, tensor, textura y métodos volumétricos; y avanzadas técnicas de modelado como: modelado implícito, reducción de polígonos, suavizado de malla (*mesh smoothing*), corte, contorneado y triangulación de Delaunay. VTK tiene un amplio marco de visualización de la información, cuenta con un conjunto de *widgets* de interacción 3D, soporta el procesamiento en paralelo y se integra con diversas bases de datos de herramientas GUI como Qt y Tk. VTK es multiplataforma y se ejecuta en plataformas Linux, Windows, Mac y Unix. VTK también incluye soporte auxiliar de *widgets* de interacción 3D, anotación bi y tridimensional y computación paralela. En su núcleo VTK es implementado como un conjunto de herramientas de C++, exigiendo a los usuarios crear aplicaciones combinado varios objetos. El sistema soporta ajuste automatizado del núcleo de C++ en Python, Java y Tcl, para el que también se puedan escribir aplicaciones VTK utilizando estos lenguajes interpretados.

3.1.6. Rendering

Las técnicas de renderizado volumétrico directo generan imágenes desde datos 3D sin extraer las superficies geométricas de los datos [13]. Estas técnicas utilizan un modelo de mapas de datos que definen las propiedades ópticas, tales como el color y la opacidad [19]. Durante el renderizado estas propiedades se acumulan a lo largo de cada rayo de visión formando una imagen con los datos.

Aunque el conjunto de datos se interpreta como una función continua en el espacio, para efectos prácticos, esta se representa por un conjunto de vectores. Cada vóxel corresponde a una posición en el espacio de datos y tiene uno o más valores de datos asociados. Mientras que los valores intermedios se obtienen por interpolación de los datos de los vóxeles vecinos. Este proceso se conoce como reconstrucción y juega un papel importante en la prestación de volumen y en las aplicaciones de procesamiento.

En esencia, el papel del modelo óptico es describir cómo interactúan las partículas en el volumen con la luz. El modelo más utilizado supone que el volumen se compone por partículas que simultáneamente emiten y absorben luz. Los modelos más complejos incorporan iluminación local y sombras volumétricas, que cuenta para los efectos de dispersión de la luz. Los parámetros ópticos son especificados por los valores dados, o bien se calculan a partir de la aplicación de una o más funciones de transferencia . El objetivo de la función de transferencia en las aplicaciones de visualización es destacar y clasificar las características de interés en los datos.

Las imágenes son creadas por el volumen de muestreo a lo largo de todos los rayos de visión y la acumulación de las propiedades ópticas como resultado, un ejemplo de ello se muestra en la Figura 3.4. El modelo de emisión-absorción, el color y la opacidad acumulada son calculados acordes a la ecuación 3.1, donde C_i y A_i son el color y la opacidad asociada a la función de transferencia del valor de los datos de la muestra i.

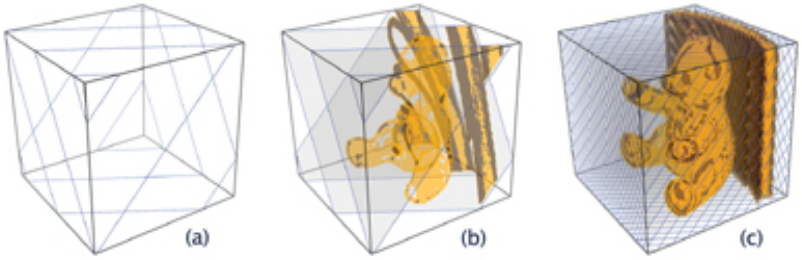


Figura 3.4: Ejemplo de composición y muestreo de un volumen (Cortesía de NVidia)

$$
\begin{aligned}
C &= \sum_{i=1}^{n} C_i \prod_{j=1}^{i-1} (1 - A_j) \\
A &= 1 - \prod_{j=1}^{n} (1 - A_j)
\end{aligned}
\tag{3.1}
$$

La opacidad, representada por A_i, aproxima la absorción, y la ponderación de la función color, C_i, hace lo mismo con la emisión y absorción a lo

largo del segmento del rayo entre las muestras i e i+1. Para la componente de color, el producto de la suma representa la cantidad de luz emitida en la muestra i que es atenuada antes de llegar al ojo. Esta fórmula es válida para evaluar la clasificación de las muestras a lo largo del rayo de visión y el cálculo del color acumulado, C, y la opacidad, A, de una manera iterativa. La ecuación 3.1 es una aproximación numérica continua del modelo óptico que presenta una buena aproximación y un renderizado de calidad. Las técnicas de renderizado basadas en textura realizan pasos de muestreo y composición de grupos de primitivas geométricas 2D dentro del volumen, como se muestra en la Figura 3.4. A cada primitiva se le asigna un coordenada de la textura, para después *rasterizarla* y mezclarla dentro de la memoria de atrás hacia adelante o viceversa. En la etapa de sombreado, las coordenadas de textura interpoladas se utilizan para obtener una textura, dado que los valores interpolados actúan como coordenadas de la textura dentro de la función de transferencia.

Función de transferencia

El papel de la función de transferencia es hacer hincapié en las características de los datos mediante las propiedades ópticas. Las funciones de transferencia más simples y utilizadas son aquellas con una sola dimensión, y que ellas mismas mapean el rango de valores de los datos, color y opacidad. Por lo general, estas funciones de transferencia se implementan con tablas de búsqueda de texturas. Cuando la tabla de búsqueda se construye, el color y la opacidad se asignan generalmente por separado en la función de transferencia. Para una correcta prestación, los componentes de color deben ser multiplicados por los de opacidad, ya que el color se aproxima a la emisión y absorción de los rayos dentro de un segmento. El diseño de la función de transferencia es un proceso iterativo que requiere de la perspectiva del conjunto de datos subyacente. Parte de la información debe ser proporcionada por el histograma del volumen, definiendo el rango de valores a destacar. Mientras que la opacidad asignada depende de la frecuencia de muestreo. Por ejemplo, cuando se utilizan menos cortes, la opacidad tiene que ser ampliada, de modo que la intensidad global de la imagen sigue siendo la misma. La ecuación 3.2 se utiliza para corregir la función de opacidad cuando el usuario cambia la frecuencia de muestreo s de la frecuencia de muestreo de referencia s_0:

$$A = 1 - (1 - A_0)^{s_0/s} \tag{3.2}$$

No obstante para controlar la calidad y la velocidad del renderizado, se ha de jugar con el numero de cortes, con el ratio s/s_0

Iluminación

Los modelos de iluminación son usados para mejorar la apariencia de los objetos. Los modelos más sencillos usan aproximaciones locales para definir la intensidad de la luz reflejada por la superficie de un objeto. La aproximación más común es el modelo de Blinn-Phong, que calcula la intensidad reflejada en función de la superficie normal, la dirección, y la intensidad, I_L, de la fuente de luz puntual, y los coeficientes de ambiente, difuso, especular y brillo: k_a, k_d, k_s , y n:

$$I = k_a + I_L k_d(\overrightarrow{l} \cdot \overrightarrow{n}) I_L k_s(\overrightarrow{h} \cdot \overrightarrow{n})^n \tag{3.3}$$

La intensidad calculada se utiliza para modular los componentes de color de la función de transferencia. Por lo general, la ecuación 3.3 se evalúa en el segmento de sombreado que requiere información por cada píxel. En las aplicaciones de renderización, el vector gradiente normalizado se utiliza como la superficie normal. Por desgracia, el gradiante no está bien definido en regiones homogéneas del volumen. Por lo que se suele usar, el modelo de Blinn-Phong basado en la frecuencia, donde solo las regiones con magnitudes altas estarán sombreadas [10]. Mientras que la iluminación local hace caso omiso de las contribuciones indirectas de luz, sombras y otros efectos a nivel global.

Composición

Para evaluar la ecuación de renderizado (3.4) de manera eficientemente, las muestras se tienen que ordenar de atrás hacia delante, y el color y la opacidad acumulados deben calcularse iteratívamente. Un solo paso del proceso de composición se conoce como el operador Over:

$$\begin{aligned} \widehat{C}_i &= C_i + (1 + A_i)\widehat{C}_{i+1} \\ \widehat{A}_i &= A_i + (1 + A_i)\widehat{A}_{i+1} \end{aligned} \tag{3.4}$$

Donde $\widehat{C}_i$ y $\widehat{A}_i$ son el color y la opacidad obtenidos del fragmento i visto por el rayo, obteniendo el segmento de sombra y el color acumulado desde el final del volumen. Si las muestras se ordenan de delante hacia atrás, el

operador utilizado es el *Under*:

$$\begin{aligned} \widehat{C}_i &= (1 + A_i)C_i + \widehat{C}_{i+1} \\ \widehat{A}_i &= (1 + A_i)A_i + \widehat{A}_{i+1} \end{aligned} \tag{3.5}$$

Donde $\widehat{C}_i$ y $\widehat{A}_i$ son el color y la opacidad acumulada desde el frente del volumen.

Las ecuaciones de composición 3.4 y 3.5 son fáciles de implementar con hardware especifico para realizar la transparencia. Para el operador Over, el factor de mezcla de origen se establece en 1 y el factor de mezcla de destino se establece en 1-alfa destino, mientras que para el operador Under el factor de origen es 1-alfa y el factor del destino 1. Por otra parte, si el hardware permite la lectura y la escritura del mismo búfer, la composición se puede realizar en la fase de sombreado mediante la proyección de fragmentos de los vértices del polígono sobre el rectángulo de visualización. Para poder combinar estructuras opacas dentro de volúmenes, estas se deben de dibujar antes que el volumen, para que así la prueba de la profundidad pueda desechar los fragmentos que están dentro de los objetos. El operador Under requiere dibujar la geometría y el volumen en búfers de color por separado y combinarlos al final. En este caso, los valores de profundidad de la geometría se utilizan para el descarte de los fragmentos en el paso del renderizado.

3.1.7. *Ray Casting*

Ray Casting [14] es una técnica simple y directa capaz de realizar renderizado volumétrico directo (DVR) de alta calidad, y viene definida por la ecuación 2.1. Comparte muchas características con el Ray Tracing [8]. En Ray Tracing miles de rayos transportadores de energía viajan a partir de una fuente luminosa hacia los objetos de la escena, haciéndolos visibles al promover una dinámica de transferencia de energía radiante. Los objetos reflejan parte de esa energía, de manera difusa y especular, y esta finalmente llega al ojo del observador donde se percibe como colores. Análogamente, el *Ray Casting* emplea rayos explícitamente para transportar energía radiante en su recorrido a través del volumen de datos.

De todos los algoritmos de renderizado volumétrico, *Ray Casting* es una de las técnicas más estudiadas y usadas a lo largo de los años. El objetivo

principal del *Ray Casting* es hacer un buen uso de la información tridimensional sin tratar de imponer una estructura geométrica. Solucionando una de las limitaciones más importantes de las técnicas de extracción de superficies, esto se consigue gracias a la proyección de finas capas en el espacio de adquisición. Este es el principal fallo en las técnicas de extracción de superficie, principalmente en las imágenes médicas. Otro problema que suele surgir en los métodos de renderizado ocurre cuando se intentan representar datos de fluidos y otras materias que requieran de cierta transparencia en su modelación, donde la mayoría de los métodos falla, esta limitación no ocurre cuando se utiliza *Ray Casting*.

La mayoría de las implementaciones de renderizados volumétricos que usan *Ray Casting* están basados en el modelo de Blinn/Kajiya [9], Figura 3.5, que se apoya en la ecuación de composición, 2.1. En el espacio de la imagen los rayos son emitidos desde el punto de vista del usuario a través del plano sobre el volumen. A lo largo de su camino, los datos se van definiendo según las esquinas de cada muestreo formando vóxeles. El periodo de muestreo suele ser el mismo entre dos puntos de la muestra.

Una muestra se calcula a partir de la interpolación trilineal dentro de un conjunto de ocho vóxeles, tal y como se muestra en la Figura 3.6, que serán clasificados de acuerdo a la función de transferencia. Si la muestra aporta una contribución al rayo, el gradiente normal se calcula sobre la base de una interpolación trilineal de las diferencias centrales normalizadas de los ocho vóxels que contiene el punto de muestreo. Finalmente, la muestra está compuesta con la anterior muestras de los rayos a lo largo de la trayectoria del rayo. Para terminar, se realiza una composición con la muestra anterior a lo largo del camino del rayo.

La implementación más rápida de este método se logra mediante la combinación de varias técnicas, como la terminación temprana de rayos, la descomposición en *octree* y el muestreo adaptativo [20]:

- La terminación temprana de rayos es una técnica que puede ser utilizado si los rayos se desplazan de delante hacia atrás. Simplemente termina el recorrido del rayo, si la opacidad acumulada por el rayo está por encima de un umbral determinado.

- La descomposición en *octree* es una técnica de jerárquica espacial que permite hacer un recorrido rápido del espacio vacío, con el consiguiente ahorro de tiempo en el recorrido y cálculo de las interpolaciones trilineal.

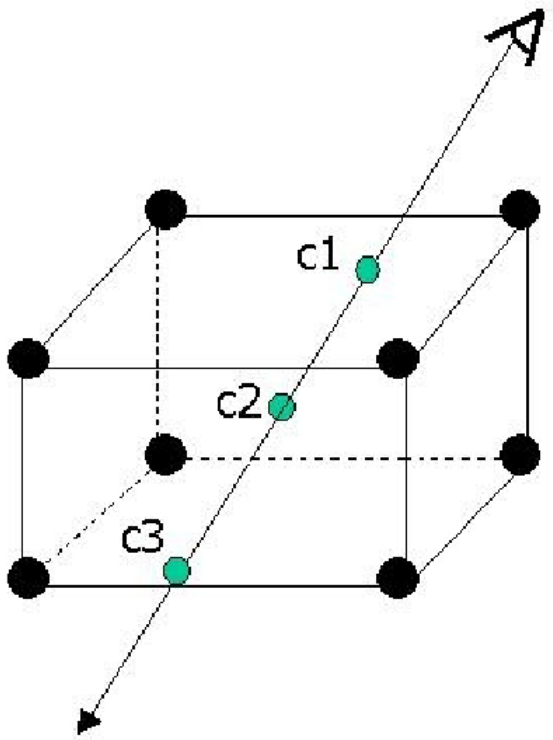


Figura 3.5: Modelo de un rayo (Cortesía de [37])

- El muestreo adaptativo trata de minimizar el trabajo mediante el aprovechamiento de las partes homogéneas del volumen, para cada porción de la imagen, un rayo atraviesa los vértices del rectángulo de la selección y recursivamente se reducen las reparticiones dentro de este cuadrante si la diferencia en el valor de píxel de la imagen es más grande que un umbral.

Aunque estas técnicas de aceleración mejoran significativa los tiempos de procesamiento, la técnica de *Ray Casting* estaba lejos de ser una técnica interactiva. Sin embargo, eran usadas por la alta calidad de sus salidas. Es por ello que Krueger et al [11], Weiler et al [34], y Bernardon et al [1] implementaron algoritmos de *Ray Casting* para hardware gráfico escritos como *shaders* que se ejecutan directamente en GPU (Unidad de procesamiento gráfico). GPU *Ray Casting* es un algoritmo multi-paso en el cual se trazan

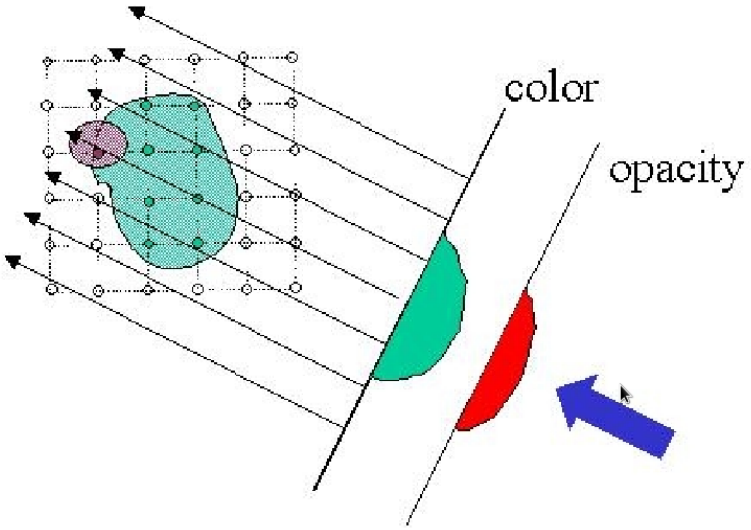


Figura 3.6: *Ray Casting* sobre un conjunto de datos con proyección paralela y uniforme (Cortesía de [37])

rayos en el primer paso y se componen colores en el segundo sin usar la CPU. Recientemente, Stegmaier et al [25] muestran cómo aprovechar las capacidad de iteración de las tarjetas gráficas en estado del arte para codificar el GPU *Ray Casting* como un algoritmo de un solo paso. El uso de estas técnicas se ha extendido en los últimos años debido a las mejoras de rendimiento de las tarjetas gráficas que erradican casi por completo la no interactividad de este método.

3.2. Desarrollo

El objetivo de este TFM es el desarrollo de un prototipo de renderizado volumétrico dentro del software AMILab, este debe cubrir las necesidades del grupo GIMET en sus tareas de investigación. Por tanto, el cometido del renderizado volumétrico estará centrado en la representación de muestreos, las representaciones numéricas y la voxelización (discretización de información 3D). Su aplicabilidad principalmente será en el conjunto de datos médicos, tales como resonancia magnética, tomografía computarizada o ultrasonido.

Actualmente hay varios proyectos fin de carrera (PFCs) que hacen uso parcial o total de las funcionalidades del renderizado volumétrico como herramienta de visualización, entre ellos se encuentran:

- La representación de los contornos con precisión sub-pixel en imágenes 3D.
- La representación de la segmentación de estructura vasculares en tomografía computarizada.
- La representación 3D a partir de los datos obtenidos al utilizar un aparato de ultrasonido.

Como se ha ido comentando, el renderizado volumétrico consiste en visualizar los datos basados en vóxeles en una imagen. Esto no es más que la proyección de un conjunto de datos unidimensionales (generalmente tridimensionales) en una imagen 2D, para tener un entendimiento de la estructura contenida dentro de los datos.

A partir de las herramientas y funcionalidades del propio software y las aportadas por la librería VTK se ha diseñado e implementado dentro de AMILab una estructura de clases para el desarrollo del renderizado volumérico. Como se ha descrito, AMILab es un software basado en *scripts* y por tanto muchas de sus funcionalidades se definen en este tipo de ficheros, y el renderizado volumétrico no es una excepción. A la hora de realizar el desarrollo del renderizado volumétrico, se ha optado por añadir una nueva rama dentro de la arquitectura de *scripts*, tal y como se muestra en la Figura 3.7.

La implementación del renderizado volumétrico se encuentra disponible únicamente dentro del *trunk* de AMILab, ya que actualmente se encuentra en fase de prueba. No obstante, las intenciones son de incorporar dicha funcionalidad en la próxima versión estable del software, con el objetivo de ofrecer sus funcionalidades a un mayor número de usuarios.

AMILab cuenta con la funcionalidad de renderizado volumetrico desde su versión 3.1.0, que era la antigua versión en desarrollo. En ella se podía trabajar con hasta dos volúmenes en una misma escena, y realizar modificaciones sobre los volúmenes en tiempo real. Actualmente, existe una nueva versión en desarrollo de AMILab que cuenta con una nueva interfaz de renderizado. Esta versión es más potente y amigable, en ella se conservan todas las funcionalidades de la anterior versión, y se le han añadido otras nuevas, como:

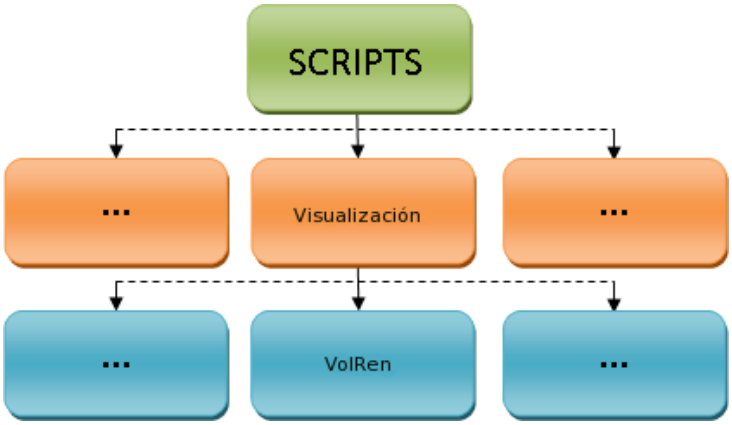


Figura 3.7: Representación de la arquitectura de *scripts* de AMILab

- La inclusión de una barra de herramientas.
- La inclusión de un sistema de configuración de curvas y escena.
- La inclusión de una interfaz de configuraciones rápidas.
- Y otros muchos cambios que han ayudado a conseguir una mayor usabilidad.

En la versión 3.1.1 de AMILab, que es la actual versión de desarrollo, se ha incorporado la posibilidad de herencia múltiple en los *scripts* lo que ha permitido la realización de *scripts* más potentes y desacoplados. La estructura de *scripts* del renderizado volumétrico hace uso de esta nueva funcionalidad haciendo más sencilla la jerarquía de los componentes.

3.2.1. Interfaz

La interfaz del renderizado volumétrico está pensada y diseñada para que su uso sea intuitivo y sencillo con el objetivo de que usuarios no expertos no tengan dificultades en su manejo. Esta se puede observar en la Figura 3.10, en ella podemos contemplar los principales componentes: el visor, la barra de herramientas, Figura 3.8, y las pestañas de configuración, Figura 3.9.

El visor es el principal componente de esta interfaz, se trata de un módulo independiente y desacoplado del resto de los *scripts*, lo que permite ser

utilizados por otros. Se trata de un visor 3D inteligente, ya que es capaz de configurar sus parámetros para aprovechar al máximo el sistema hardware. El visor está pensado y diseñado para que haga uso de funciones GPU siempre y cuando sea posible. Además el visor admite trabajar con n volúmenes o utilizar múltiples visores sobre un mismo volumen. Su desarrollado se ha llevado a cabo desde lenguaje *scripts* a partir de las funcionalidades ofrecidas por la librería VTK.

La barra de tareas, es una herramienta de acceso rápido a funcionalidades comunes y/o frecuentes. Únicamente será accesible cuando lo esté la interfaz del renderizado volumétrico. La inclusión en este nuevo diseño se hizo con el objetivo de facilitar la usabilidad del sistema, evitando la mezcla de funcionalidades en las pestañas de la interfaz. Algunas de sus funcionalidades son:

- Mostrar y ocultar el histograma del actual volumen.
- Mostrar y ocultar la ventana de configuraciones rápidas.
- Obtener una captura del visor.
- Y guardar el fichero de configuración actual.

Por último, tenemos un conjunto de pestañas, en las que englobamos las distintas funcionalidades del módulo. Estas se clasifican en tres tipos: *I/O, Volume y Scene*.

- Las funcionalidades de entrada y salida, son aquellas operaciones relativas al manejo previo de los volúmenes, las operaciones de cargas y reducción de tamaños de estos. Por defecto se reduce cada dimensión por la mitad, lo que implica una redución de 2^3 en el tamaño del volumen.
- Las funcionalidades sobre los volúmenes engloban operaciones de mezcla, configuración de luces, recortes y opciones para su pintado. Estas operaciones son independientes para cada uno de los volúmenes, por lo tanto los cambios que se realicen solo afectarán al volumen sobre el que se aplique.
- Por último tenemos las operaciones sobre la escena, que permiten modificar el color del fondo y el manejo de la cámara de la escena de forma relativa a la pocisión actual.

Una vez cargado un volumen, se tendrá acceso a todas las funcionalidades del renderizado, como se muestra en la Figura 3.11. Además se nos desplegará un nuevo módulo debajo del visor, en el cual se representa el histograma del volumen seleccionado y su correspondiente curva de control. Inicialmente se le asigna a cada volumen una curva de control sintética.

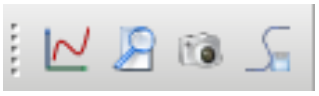


Figura 3.8: Barra de herramientas del renderizado volumétrico

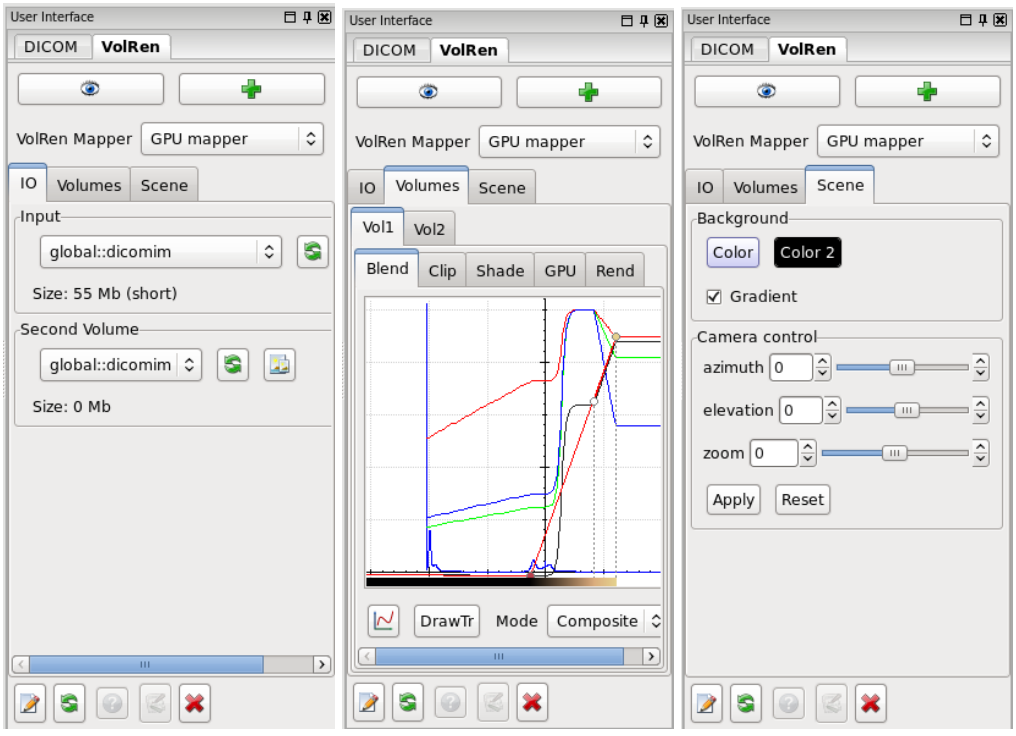


Figura 3.9: Pestañas de la interfaz del renderizado volumétrico

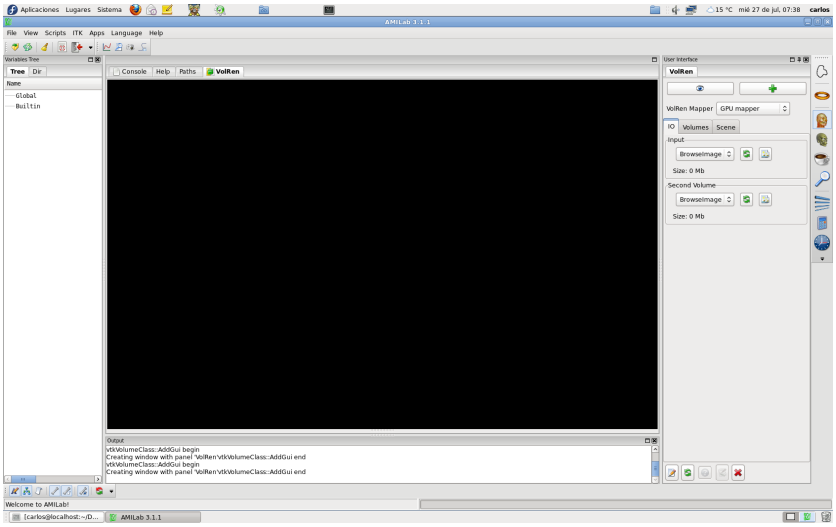


Figura 3.10: Interfaz de usuario del renderizado volumétrico

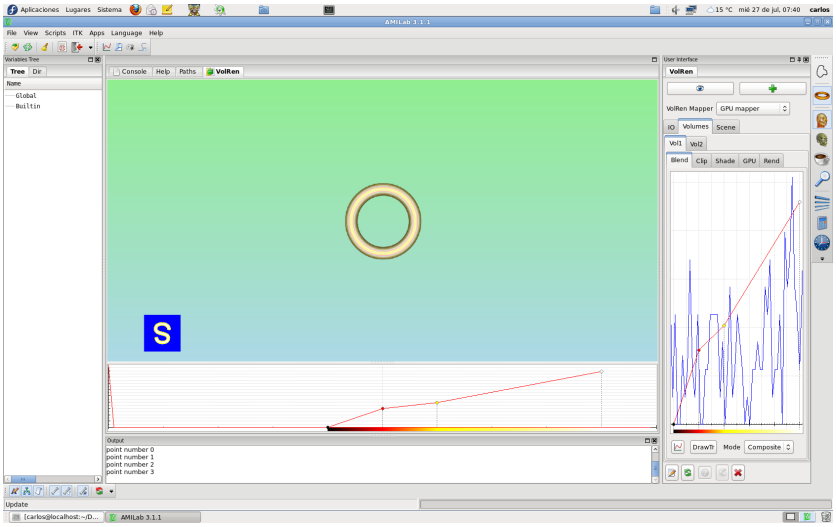


Figura 3.11: Interfaz de usuario del renderizado volumétrico una vez se ha cargado un volumen

Visor de voluménes

Los datos volumétricos que usaremos pueden ser generados a partir de dos fuentes, aquellos adquiridos por observación empírica o mediante simulación por computador. Estos últimos a su vez los subdividiremos en dos:

- Los obtenidos por escaneo del objeto (IRM, TAC)
- O los generados sintéticamente.

Dada la naturaleza de los volúmenes, la lectura de datos del visor se puede realizar desde dos vías, generando una imagen sintética o bien desde una especie de estructura DICOM de AMILab. Antes de realizar la carga de los volúmenes en el visor es recomendable tener en cuenta el tamaño de los datos a cargar y en caso de que se considere necesario realizar un escalado de los datos.

Una vez realizada la carga y la pertinente comprovación se podrán visualizar los volúmenes generados. La técnica de renderizado elegida a sido *Ray Casting* dado que es la metodología que más se ajusta a los requerimientos dados y la que consigue los resultados más realistas. Cabe destacar que el punto débil de esta técnica sigue siendo el tiempo de calculo pero hoy en día con una buena tarjeta gráfica esta problemática pasa a segundo plano. Dentro de las características del visor destacamos la posibilidad de elegir entre varios modelos de mapeado y otros tantos de mezclado, además de poder cambiar los parámetros de la función de transferencia y escena en tiempo real.

Como se ha comentado con anterioridad, la implemnetación se ha realizado haciendo uso de la librería VTK dentro del lenguaje de AMILab. A temas de interfaz, el visor es un componente exclusivamente de visualización por lo que no aporta funcionalidades a la interfaz por sí mismo. En la Figura 3.12 puede verse el componente, en el que se ha cargado un volumen y el *widget* de orientación.

Histograma y curvas de control

La interfaz de usuario es un componente importante del proceso de diseño interactivo. AMILab cuenta con un editor de curvas para especificar y modificar la función de transferencia a través de un conjunto de puntos de control. Este componente cuenta con dos estructuras básicas el histograma y la curva de control:

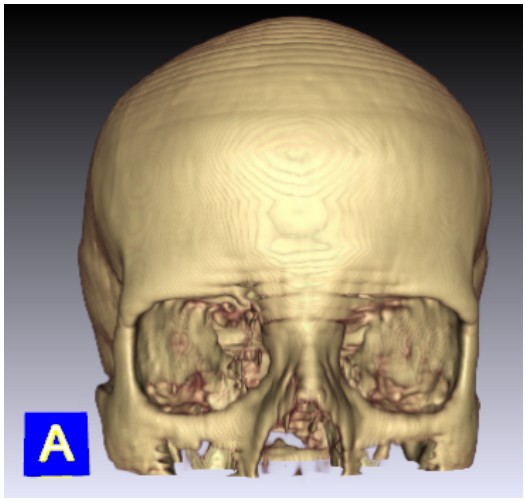


Figura 3.12: Estructura del visor

- El histograma es una herramienta que proporciona información sobre los datos del volumen. En el eje horizontal se representa la densidad de los datos, mientras que en eje vertical se define la frecuencia de los valores representados.
- La curva de control, es la estructura de representación de función de transferencia, con ella se puede definir y modificar dicha función de manera sencilla.

AMILab contaba ya con estos módulos, por lo que solo hubo que adaptarlo a las necesidades de la aplicación. El objetivo de dicho módulo es añadir a la interfaz una herramienta que facilite la configuración de las propiedades de los volúmenes por parte del usuario. La estructura del componente se muestra en la Figura 3.13 y como se puede observar en ella, incorpora varias funcionalidades interesantes:

- Añadir/Eliminar puntos.
- Cambiar el color.
- Modificar la intensidad.
- Y otras tantas funcionalidades.

Actualmente el módulo del histograma admite varias funciones de control sobre el histograma por lo que se está trabajando para manejar funciones de transferencia con más de un curva.

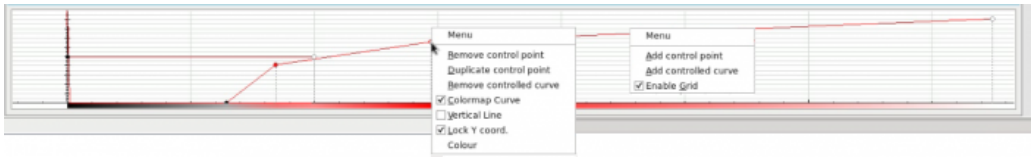


Figura 3.13: Módulo de representación del histograma

Estructura para la representación de curvas y escena

Los ficheros de configuración son meros ficheros de texto en los que se define la información codificada de la función de transferencia del volumen, además de algunos parámetros de configuración de la escena y del propio volumen. Su diseño está basado en las clases de VTK y su estructura es la representada en la Figura 3.14. Estos ficheros siguen la política de los ficheros DICOM donde la carpeta raíz aporta información sobre su categoría y el nombre asociado al fichero es meramente discriminatorio. Tal y como se puede observar en la Figura **??**, el fichero cuenta con una serie de comentarios que no pueden faltar en su estructura. Omitiendo estos, lo primero que se define en el fichero es el número de puntos con los que cuenta la curva de control. A continuación se definen el conjunto de puntos que es una tupla de 7 elementos, seguido por la parametrización de la luz del escenario, el modo de mezclado y si se desea normalizar la curva de control con respecto al volumen. Por último se definen los parámetos del fondo de la escena y a término de información global los ficheros cuentan con un campo de descripción. La creación de estos ficheros puede llegar a ser un poco tediosa, pese a tener una estructura bastante sencilla. No obstante, es posible crear nuevos ficheros desde la barra de herramientas, mediante la operación *Save current curve* que guarda la configuración del volumen actual, pudiendo ser esta editada mediante el ajuste de los parámetros de las distintas opciones de la interfaz.

```
# Points (IRGBoMS)
1..∞
-∞..+∞ 0..1 0..1 0..1 0..1 0..1 0..1
...
# Property shadeon/off Ambient Diffuse Specular SpecularPower
opacityUnitDistance Blendmode Normalize
0/1 0..1 0..1 0..1 1.. 0..1 0/1 0/1
# Background RGB RGB UseGradientBackground
0..1 0..1 0..1 0..1 0..1 0/1 0/1
# Description
A brief description
```

Figura 3.14: Estructura del fichero de configuración

Ventana de previsualización

Una de las funcionalidades más importantes del actual sistema de renderizado la encontramos en este componte, Figura 3.15. La vista previa es una colección de pre-sets, un pre-set es un fichero de representación de curva y escena siguiendo la estructura mostrada en la Figura 3.14, que permite acelerar los ajustes de las configuraciones de los volúmenes. Estos archivos se encuentran definidos en la carpeta CURVES dentro del directorio del renderizado volumétrico. El proceso para añadir una nueva categoría es sencillo, basta con crear una nueva carpeta dentro del directorio y definir nuevos pre-sets en su interior.

En la Figura 3.15 se muestra la estructura del componente, como se puede observar, es bastante básica e intuitiva. En la parte superior del componente se pueden seleccionar una de las distintas categorías mediante el *combobox*. Para aplicar un pre-set, basta con seleccionarlo, para ello se debe presionar el botón izquierdo del ratón sobre la configuración deseada, al realizar dicha acción el nombre del pre-set elegido se pondrá de color rojo, y pulsar el botón *Apply*, haciendo que el volumen seleccionado rote. Inmediatamente se aplicará dicha configuración al volumen actual y se cerrará la ventana de previsualización.

Este módulo puede llegar a utilizarse para otras funcionalidades, como por ejemplo mostrar múltiples vistas de un volumen con el objetivo de estudiarlo en más detalle. Actualmente esta opción no se encuentra activa desde la

interfaz, por lo que hay que ejecutarla desde la consola de AMILab.

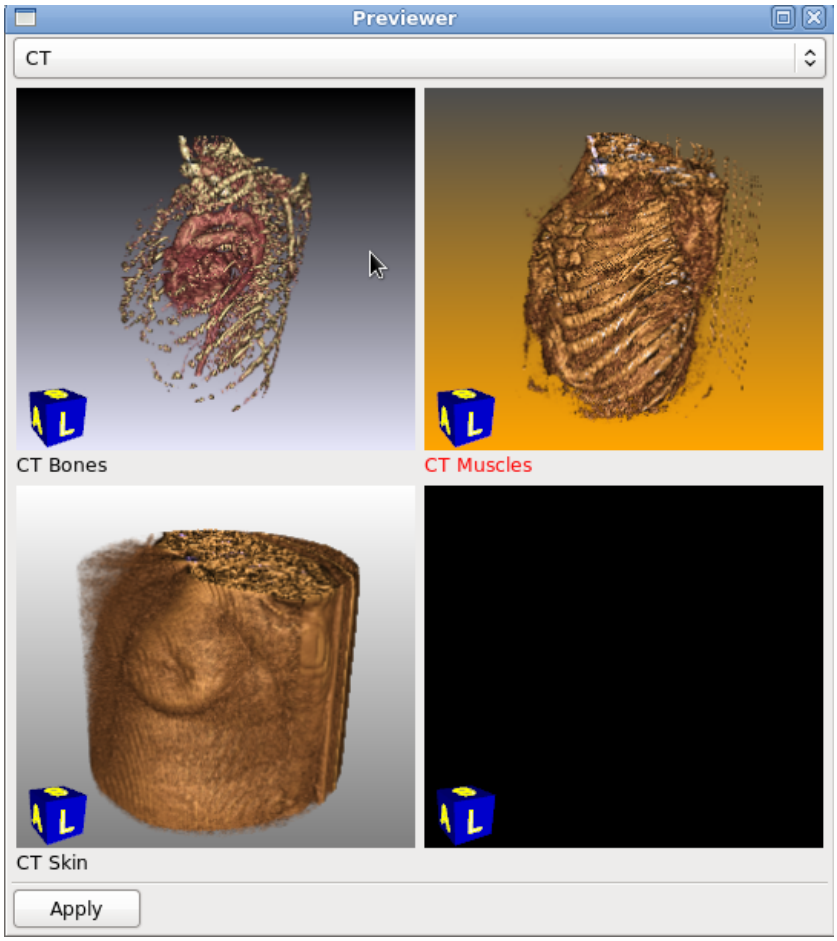


Figura 3.15: Ventana de cónfiguraciones rápidas

Capítulo 4

Resultados

En este capítulo se ilustrarán los resultados obtenidos a lo largo del desarrollo de este TFM. AMILab posee una *dokuwiki* [7] bastante completa donde se detallan las distintas funcionalidades del software, así como manuales de instalación y de desarrollo. Dentro de la *dokuwiki* se encuentra un apartado con una breve descripción del renderizado volumétrico y sus funcionalidades, a modo de manual de usuario. Además, recientemente se ha creado un canal en YouTube, AMILabCast [4], donde se presentan vídeos de demostración con el objetivo de mostrar de manera más cercana las funcionalidades del software.

Como ya se a comentado AMILab es un software en desarrollo orientado a la visualización y tratamiento de imágenes médicas, que surgió como fruto de la tésis doctoral de Karl Krissian. A lo largo de su desarrollo ha ido variando de aspecto y añadiendo nuevas funcionalidades, que hacen que sea una herramienta de tratamiento de imágenes alternativa a las comerciales, aunque su funcionalidad está orientada básicamente al uso académico y la investigación . Su aspecto en la última versión en desarrollo, versión 3.1.1, lo podemos ver en la Figura 4.1. Esta cuenta con la incorporación del renderizado volumétrico que le ha añadido atractivo a sus funcionalidades. Con esta funcionalidad se busca que los usuarios, principalmente investigadores y médicos puedan visualizar tanto los datos de las pruebas como los resultado de sus trabajos de investigación sobre cualquier tipo de volúmenes.

AMILab cuenta con la implementación de una de las técnicas con más detalles y a la vez complejas de renderizado volumétrico, *Ray Casting*. En la Figura 4.2 se muestra un ejemplo de esta empleando la opción de mapeado en GPU. En dicha figura se puede apreciar la calidad y el nivel de detalle de los

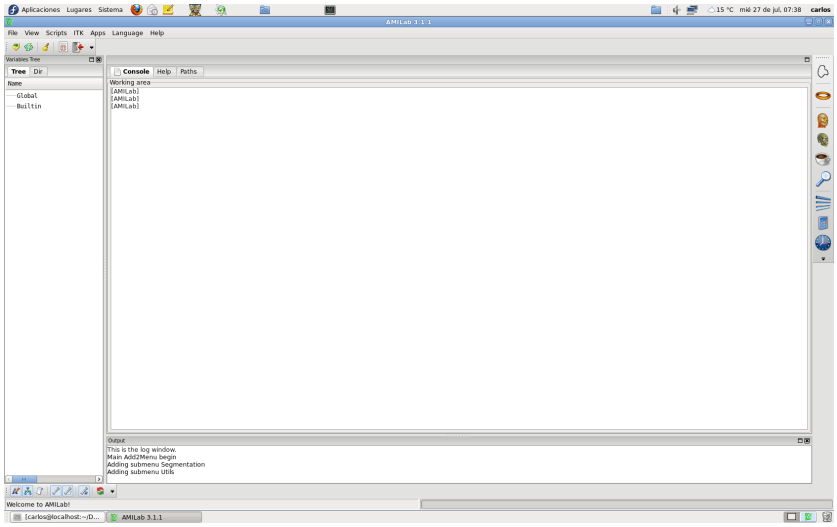


Figura 4.1: Interfaz de AMILab

resultados que se llegan a alcanzar con esta representación. En la secuencia de la Figura 4.3, se muestra otro ejemplo. En esta secuencia se realiza un *zoom* hacia el interior del volumen pudiéndose observar en cada imagen de la secuencia un mayor rango de detalles de la cavidad del volumen. Aunque no se puede apreciar en las figuras, la interactividad con este tipo de volúmenes es total. Esta condición se mantendrá siempre y cuando se trabajen con datos que puedan ser mapeados. En caso de que el tamaño del volumen supere al de la memoria disponible en la tarjeta es recomendable aplicar el operador de reducción sobre el volumen para trabajar con este.

Recapitulando el esquema general del algoritmo de visualización volumétrica, este quedaría formado por:

1. *Ray Casting*. Para cada píxel de la imagen final se traza un rayo visual a través del volumen.

2. Muestreado. A lo largo del rayo visual dentro del dominio volumétrico se muestrean a intervalos regulares los valores de este. Por lo general las muestras que se necesitan no están alineadas con el volumen, es necesario aplicar interpolación usando diferentes filtros de muestreo.

3. *Shading*. Para cada punto muestreado, se calcula el gradiente local para

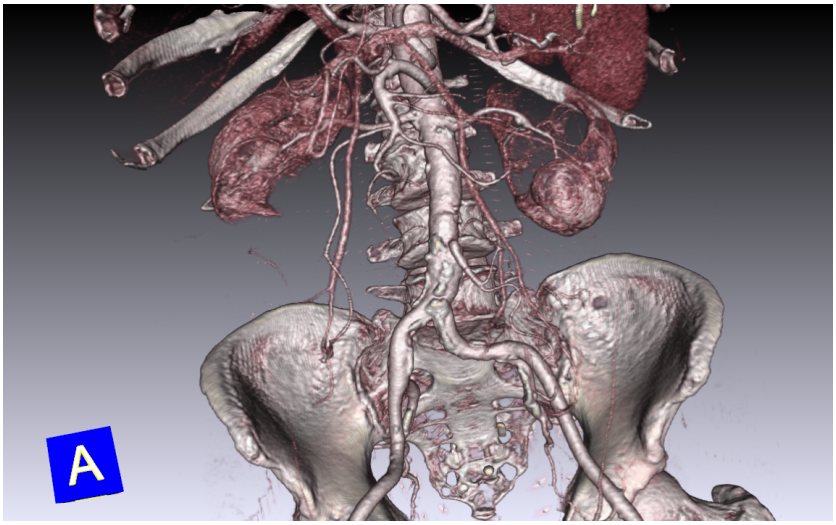


Figura 4.2: Ejemplo de renderizado volumétrico de un paciente con problemas renales

obtener estimaciones de las normales locales del punto, estas muestras son entonces sombreadas (*shaded*) , coloreadas e iluminadas de acuerdo con su orientación con respecto a las fuentes de luz, y a la emisión y absorción del volumen entre estas y la muestra en cuestión.

4. Composición. Después que todas las muestras han sido sombreadas se procede entonces a la integración o composición de la contribución final al píxel de la imagen, este proceso parte de las ecuaciones anteriormente derivadas.

5. Ajuste de configuración.

Una vez recordado el el esquema de funcionamiento se irán comentando y exponiendo los resultados de las distintas funcionalidades del software que de una manera o otra influirán en alguno de estos puntos. Como se comentó en el apartado de desarrollo, es posible configurar el tipo de mapeado a aplicar desde la interfaz, aunque siempre siguiendo la mismas técnica *Ray Casting*. Se ha elegido incluir 3 tipos de mapeado, aunque la posibilidad de añadir una nueva configuración es trivial. Actualmente se cuenta con las técnbicas GPU, CPU y Fixed Point. La aplicación está pensada para que use por defecto la

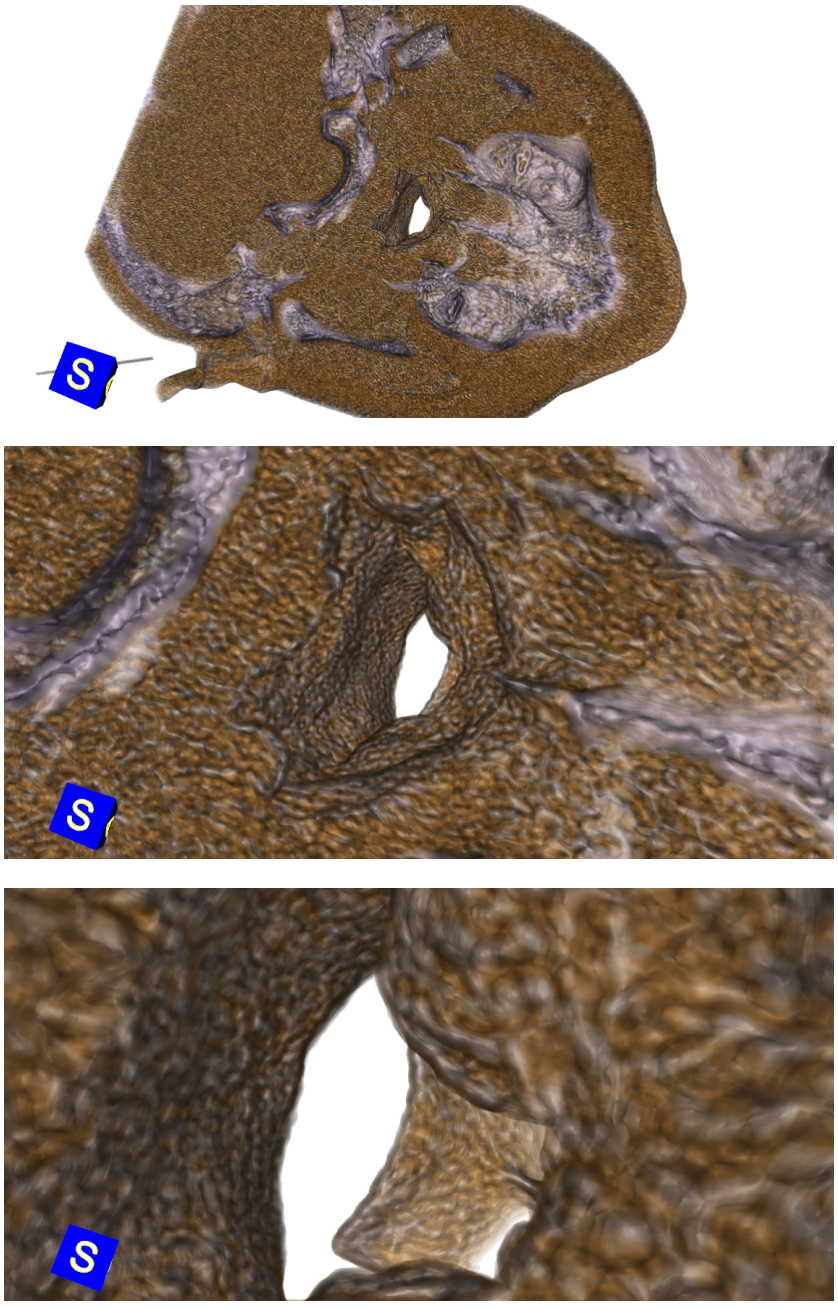
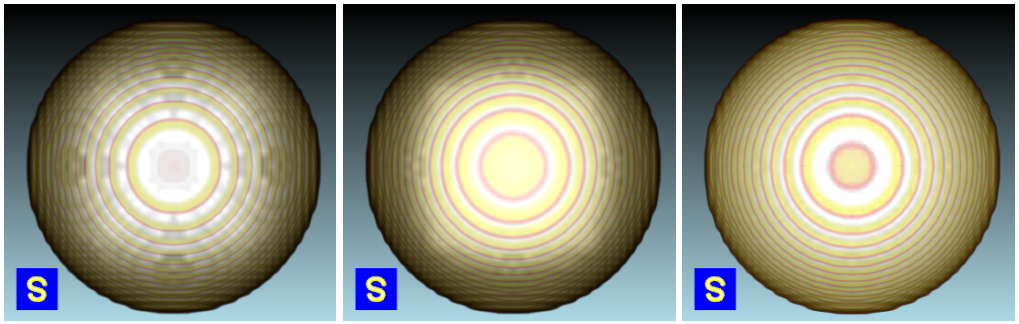


Figura 4.3: *Ray Casting* con GPU



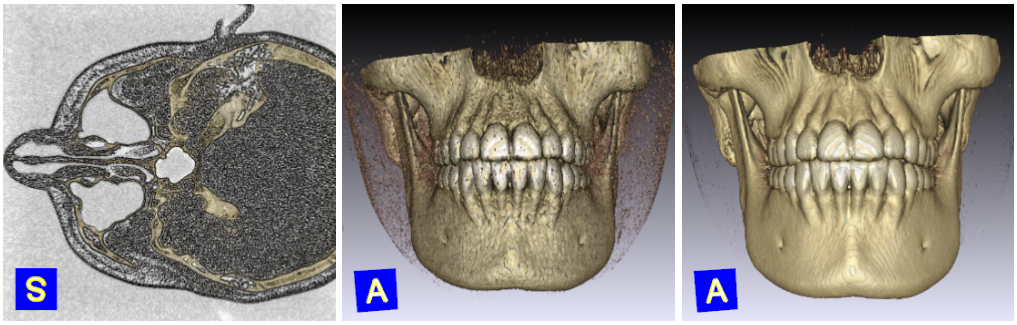
(a) CPU (b) Fixed Point (c) GPU

Figura 4.4: Muestra de las diferentes técnicas de trazado del *Ray Casting*

implementación GPU y en caso de no disponer una tarjeta gráfica que soporte dicha técnica usar la implementación CPU. Mientras que la opción de *Fixed Point* se incluyó como parte del estudio de combinación de varios volúmenes sobre una misma escena. Fixed Point es una técnica con menor calidad que las anteriores y aún se encuentra bajo desarrollo. En las Figuras 4.4 y 4.5 se muestran ejemplos de renderizado con cada una de las técnicas, en la primera secuencia se muestran los resultados obtenidos con un volumen sintético y en la segunda secuencia se muestran sobre un volumen real. En la Figura 4.4 se puede ver como el volumen es visible con todas las técnicas apreciándose únicamente diferencias visuales en el renderizado. Sin embargo, en la Figura 4.5 se aprecia que al emplear la versión CPU el volumen mostrado no es correcto. Esto se debe a la incompatibilidad de la estructura de los datos del volumen y el tipo de mapeado.

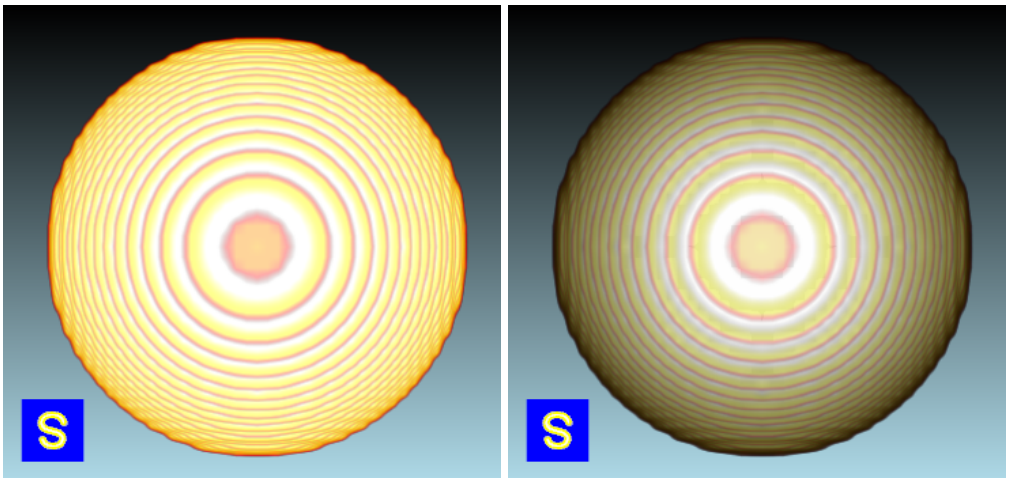
Otra de las funcionalidades que se incorpora es la configuración de la iluminación de los volúmenes, esta paremetrización es aplicable en tiempo real. El uso de iluminación viene a aportar realismo y detalles a los volúmenes como ya se comentó. El efecto que se consigue con esta funcionalidad es la apariencia de profundidad y realismo en la imagen 2D. Como se puede ver en las Figuras 4.6 y 4.7 el uso de las sombras en la escena juega un papel importante en el resultado final, siendo de mayor importancia cuando se representan volúmenes reales.

Siguiendo con las funcionalidades de configuración, la interfaz de renderizado en su opción GPU cuenta con la posibilidad de ajustar la calidad del volumen en un cierto rango. Esto es posible al poder configurar la distancia mínima entre rayos. En la Figura 4.8 se muestra un ejemplo, con el ajuste de



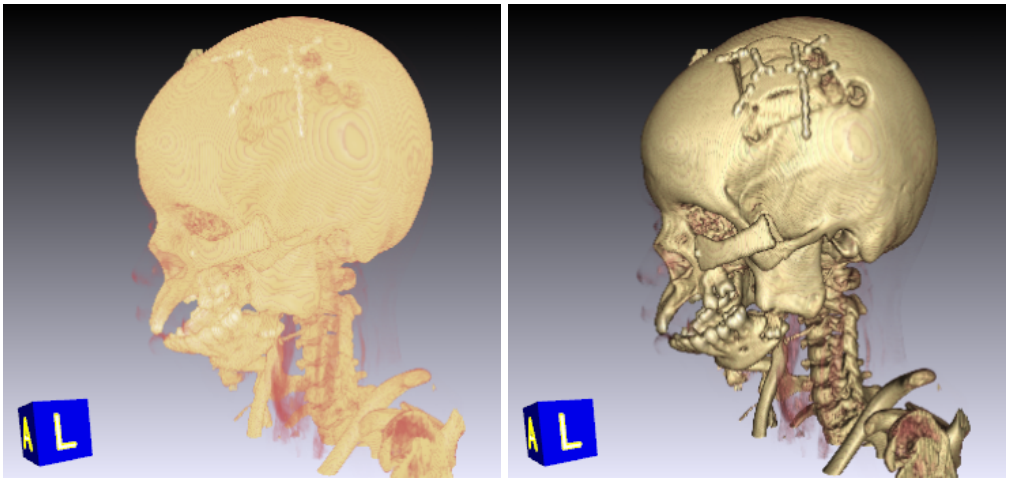
(a) CPU (b) Fixed Point (c) GPU

Figura 4.5: Muestra de las diferentes técnicas de trazado del *Ray Casting* sobre un volumen real



(a) Sin usar sombras (b) Usando sombras

Figura 4.6: Usos de sombras sobre volumen sintética



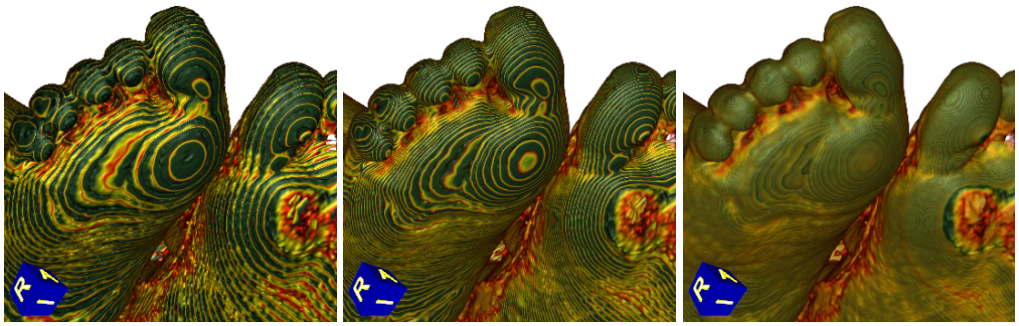
(a) Sin usar sombras (b) Usando sombras

Figura 4.7: Usos de sombras sobre volumen real

dichos parámetros se logra mayor definición en los detalles del volumen. En dicha secuencia se muestran los valores mínimos y máximo, y el que aproximadamente sería el intermedio.

La interfaz posee un sistema de pre-sets con varias configuraciones que nos permiten definir funciones de transferencia en pocos pasos. En la Figura 4.9 se muestra una vista conjunta de la interfaz de renderizado y la interfaz de previsualización. En la bibliografía existen varias técnicas de aplicación de las ecuaciones de composición, que a su vez presentan distinta precisión y complejidad computacional. AMILab cuenta con las técnicas descritas en el capítulo anterior y aportadas por VTK. El módulo cuenta con varias configuraciones para el modelado de los volúmenes, entre ellas se encuentran definidas varios ejemplos de CT, Figura 4.10, y algunas configuraciones sintéticas. También es importante destacar que existen otros métodos de renderizado volumétrico que no modelan ni la absorción ni la dispersión de la luz, como lo son las proyecciones de rayos X y las MIP (Maximun Intensity Proyection), Figura 4.11, que suelen ser utilizados sobre todo en el campo médico.

Una vez hayamos seleccionado la configuración deseada para nuestro volumen podremos realizar una serie de refinados sobre ellas. El primero que se describe, es la configuración y manejo de la curva. En la secuencia de imágenes de la Figura 4.12 se presentan los resultados obtenidos por el renderizado



(a) Resolución baja (b) Resolución media (c) Resolución alta

Figura 4.8: Resultados al variar la calidad del rayo del *Ray Casting*

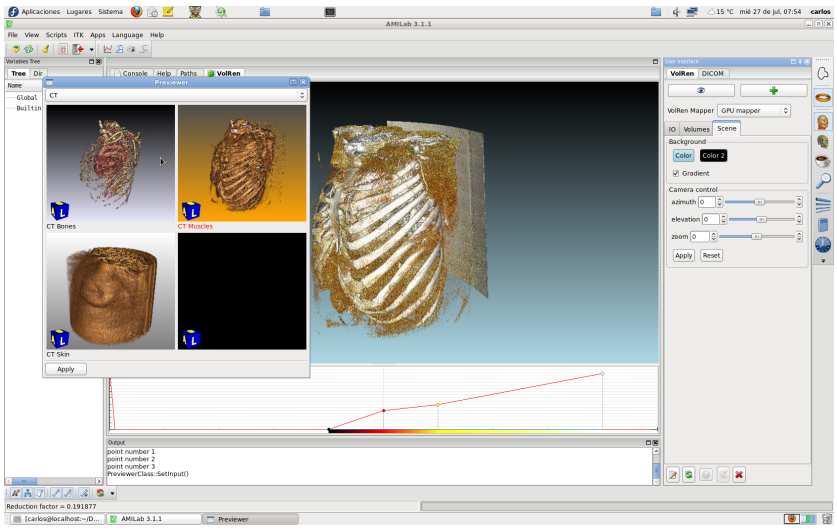


Figura 4.9: Muestra de un volumen con la ventana de configuraciones rápidas desplegada

(a) Curva de piel (b) Curva de músculo (c) Curva de huesos

Figura 4.10: Renderizado volumétrico sobre un volumen con diferentes curvas

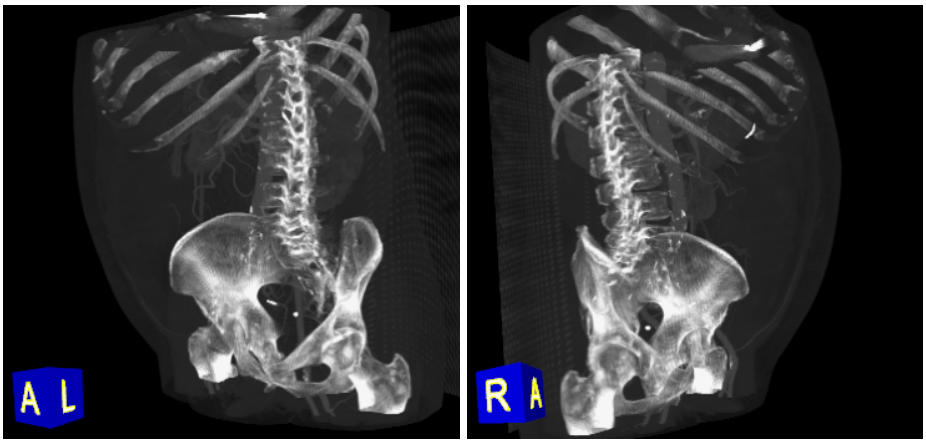


Figura 4.11: Renderizado volumétrico sobre un volumen aplicando un MIP

volumétrico al realizar varios desplazamientos en la curva de configuración a lo largo del eje de abscisas del histograma, Figura 4.13. La configuración que se puede apreciar en dicha secuencia corresponde a un ajuste de parámetros de una CT con una configuración para el realce de los músculos. En la secuencia se aprecia como al mover la curva entran en la escena visible los distintos componentes del volumen que son afectados por la función de transferencia.

Otra de las funcionalidades que tenemos en la interfaz, es la opción de recortes sobre los volúmenes. Esto se conoce comúnmente como *clipping*, la técnica más empleada es la del algoritmo Sutherland-Hodgeman [26], VTK trae implementada dicha técnica aplicada directamente a volúmenes. La interfaz incorpora un *widget* de recorte mediante planos, que permite al usuario realizar un acotamiento del volumen encerrado por el cubo del *widget*. Un ejemplo de dicha funcionalidad la encontramos en la Figura 4.14.

Para terminar este capítulo, se muestrará el manejo de varios volúmenes con AMILab, se trata de una funcionalidad compleja y poco madura dentro del renderizado volumétrico de AMILab, ya que las funcionalidades de la librería de VTK para esta temática son bastante reciente y no están muy evolucionada. En la Figura 4.15 se muestran los dos volúmenes a combinar, el primero de ellos presenta una configuración de CT de huesos y el segundo una CT de músculo. A la hora de realizar el renderizado debemos de seleccionar el modelo de mapeado *Fixed Point*, ya que es el modelo que mejor se adapta a la fusión de volúmenes. Como se comento en el anterior capítulo la carga de los volúmenes debe de realizarse de tal manera que el primero esté contenido en los sucesores, en caso contrario no se obtendrá el resultado natural. En la Figura 4.16 se muestra la combinación de los dos volúmenes anteriormente mostrados. Para este ejemplo, como se puede ver en la figura se ha realizado un recorte en ambos volúmenes (mitad-izquierda para uno y mitad derecha para el otro) pudiéndose observar de manera conjunta ambos volúmenes.

Con esto cerramos el capítulo de resultados, no sin antes recomendar una visita a nuestro canal de YouTube [4], donde se encuentran los vídeo-tutoriales del renderizado volumétrico. En los que se pueden apreciar visualmente los resultados y funcionalidades descritas.

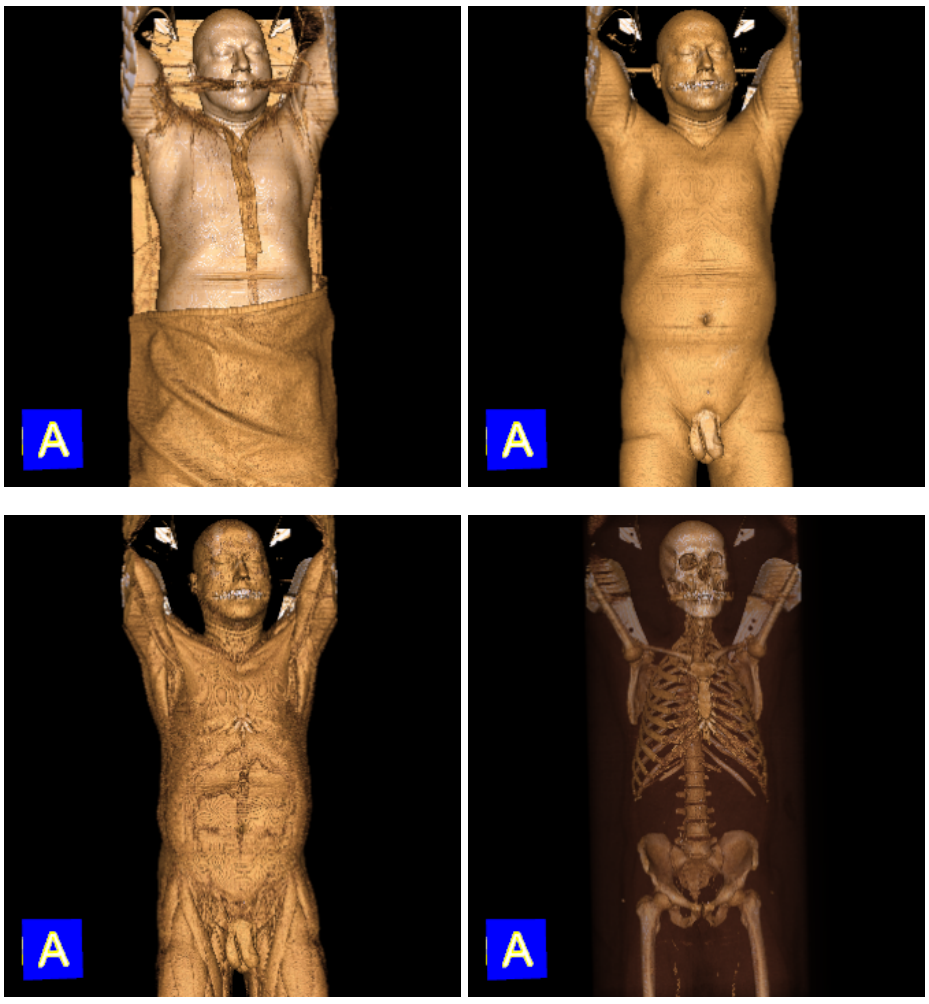


Figura 4.12: Secuencia obtenida a partir de mover la curva

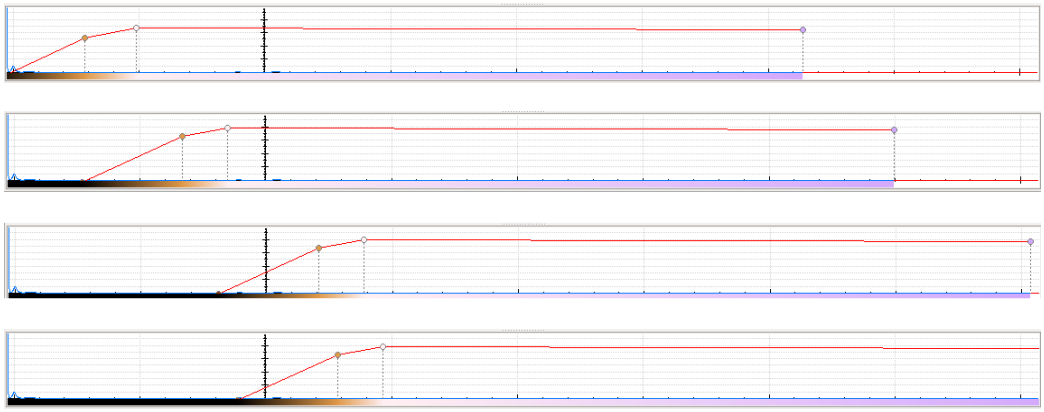


Figura 4.13: Histogramas+Curvas de la secuencia

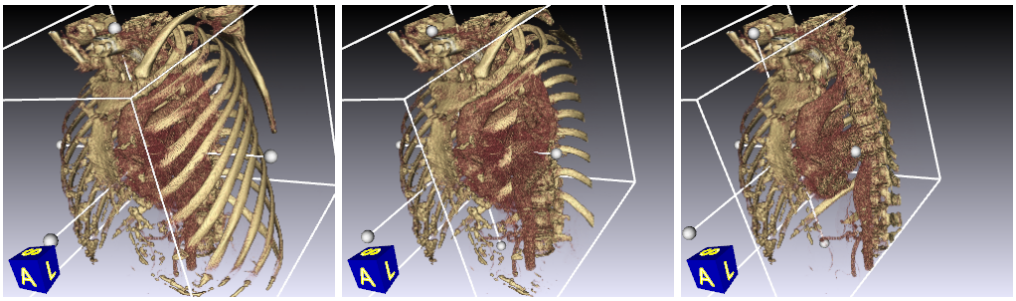


Figura 4.14: Secuencia del recorte de un volumen

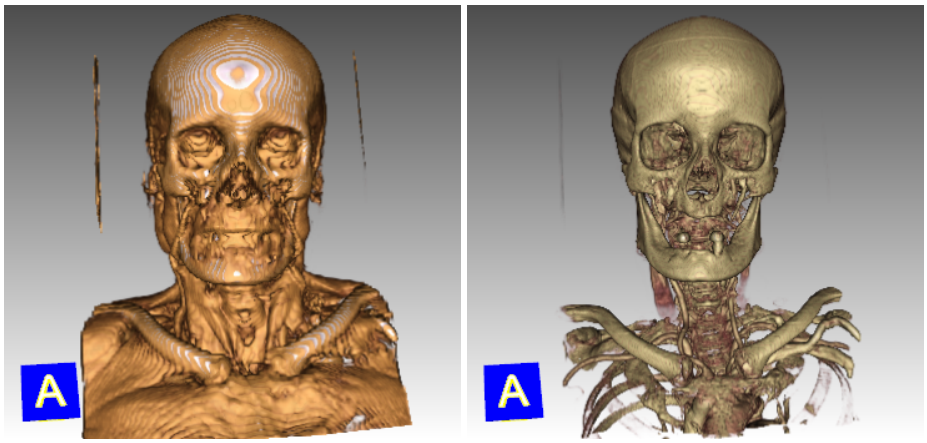


Figura 4.15: Fusión de dos volúmenes vista desde las 4 vistas

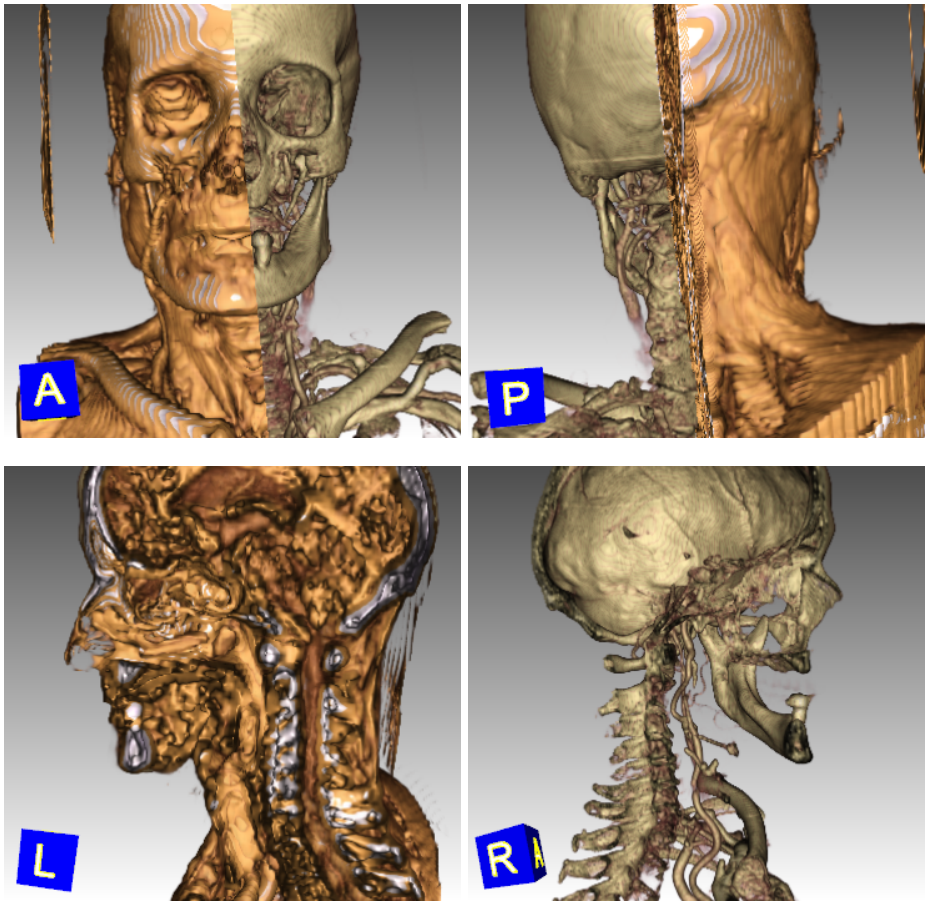


Figura 4.16: Fusión de dos volúmenes vista desde las 4 vistas

Capítulo 5

Conclusiones y Líneas Futuras

5.1. Conclusiones

A lo largo del desarrollo de este TFM se han ido alcanzado todos los objetivos marcados para este, aunque cada uno de ellos con diferentes grados de madurez. Por un lado se realizó un estudio sobre las diferentes técnicas de renderizado volúmetrico, con el objetivo de identificar las posibilidades y configuraciones de los algoritmos de renderizado. Por otro lado se evaluaron las herramientas alternativas a AMILab, observándose sus principales características, y destacándose sus puntos fuertes y débiles.

Actualmente AMILab cuenta con una interfaz para la representación de volúmenes con las características establecidas en la propuesta:

- Posee un visor que soporta y combina varios volúmenes.
- Cuenta con un módulo de configuraciones rápidas.
- Y un conjunto de parámetros configurables en tiempo real.

La colaboración en las tareas de investigacin del grupo GIMET ha resultado una experiencia satisfactoria para ambos, por un lado se ha llevado a cabo el desarrollo de la herramienta deseada, y por el otro yo he adquirido nuevos conocimientos y experiencia dentro de un proyecto nacional. Pero no todo el camino ha sido sencillo, el aprendizaje de las herramientas ha requerido un gran esfuerzo por mi parte, ya que se tratan de herramientas que se encuentran en pleno desarrollo que no nos garantizaban su buen funcionamiento, y por si no fuese poco estas presentaban documentaciones incompletas.

En términos generales estamos satisfechos con el trabajo obtenido. En el que se ha desarrollado una herramienta de calidad, que aporta una funcionalidad atractiva a AMILab. No obstante, hay que recordar que el diseño de renderizado volumétrico que se ha realizado es un mero prototipo y por lo tanto es susceptible a sufrir modificaciones tanto en sus funcionalidades, como en su diseño. Por tanto este módulo quedará a juicio por parte de la comunidad de usuarios que dictaminará sus puntos fuertes y débiles.

5.2. Líneas Futuras

AMILab es un software en continuo desarrollo, que evoluciona según las necesidades y las líneas de investigación que hayan abiertas. Pese que se ha realizado un gran esfuerzo ha quedado pendiente de estudio un conjunto de mejoras que podrían ser incorporadas con posterioridad. Entre ellas destacan:

1. La creación de un sistema de configuración basado en XML.
2. La realización de una función de ajuste de varias curvas sobre el histograma.
3. Realizar un estudio más exhaustivo de la fusión de varios volúmenes con la herramienta VTK.

Bibliografía

[1] Fábio F. Bernadon, Christian A. Pagot, Joo L. D. Comba, and Cláudio T. Silva. Gpu-based tiled ray casting using depth peeling. *journal of graphics, gpu, and game tools*, 11(4):1–16, 2006.

[2] Dick Buttlar, Jacqueline Farrell, and Bradford Nichols. *Pthreads Programming: A POSIX Standard for Better Multiprocessing.* O'Reilly Nutshell, 1996.

[3] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 symposium on Volume visualization*, VVS '94, pages 91–98, New York, NY, USA, 1994. ACM.

[4] Youtube Channel. www.youtube.com/user/amilabcast, 2011.

[5] David A. Clunie. *DICOM Structured Reporting.* PixelMed Publishing., 2000.

[6] OsiriX's DICOM collection. pubimage.hcuge.ch:8080/, 2011.

[7] AMILab dokuwiki. www.ctm.ulpgc.es/amilab/doku.php?id=help, 2011.

[8] Andrew S. Glassner, editor. *An introduction to ray tracing.* Academic Press Ltd., London, UK, UK, 1989.

[9] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18:165–174, January 1984.

[10] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8:270–285, 2002.

[11] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. *Visualization Conference, IEEE*, 0:38, 2003.

[12] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 451–458, New York, NY, USA, 1994. ACM.

[13] Marc Levoy. Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9:245–261, July 1990.

[14] Marc Levoy. Volume rendering by adaptive refinement. *The Visual Computer*, 6(1):2–7, 1990.

[15] GNU Lesser General Public License. www.gnu.org/licenses/lgpl.html, 2007.

[16] Tonya Limberg. *Osirix as a resource*. University of Illinois at Chicago, 2008.

[17] Will Schroder Luis Ibáñez and Josh Cates. The itk software guide, 2nd edition., 2005.

[18] Slicer 3.6 manual. www.slicer.org/slicerwiki/index.php/documentation-3.6, 2011.

[19] Nelson Max. Optical models for direct volume rendering, 1995.

[20] Masato Ogata, TakaHide Ohkami, Hugh C. Lauer, and Hanspeter Pfister. A real-time volume rendering architecture using an adaptive resampling scheme for parallel and perspective projections. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, VVS '98, pages 31–38, New York, NY, USA, 1998. ACM.

[21] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume on standard pc graphics hardware using multi-textures and multi-stage rasterization. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, HWWS '00, pages 109–118, New York, NY, USA, 2000. ACM.

[22] Antoine Rosset, Luca Spadola, and Osman Ratib. Osirix: An open-source software for navigating in multidimensional dicom images. *JOURNAL OF DIGITAL IMAGING*, 17:205–216, 2004.

[23] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization toolkit : an object-oriented approach to 3D Graphics*. United States of America : Kitware, cop., 2006.

[24] Julian Smart, Kevin Hock, and Stefab Csomor. *Cross-Platform GUI Programing with wxWidgets*. Prentice Hall, 2005.

[25] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A Simple and Flexible Volume Rendering Framework for Graphics-Hardware–based Raycasting. In *Proceedings of the International Workshop on Volume Graphics '05*, pages 187–195, 2005.

[26] Ivan E. Sutherland and Gary W. Hodgman. Reentrant polygon clipping. *Commun. ACM*, 17:32–42, January 1974.

[27] AMILab website. amilab.org, 2011.

[28] Bison website. www.gnu.org/software/bison/, 2011.

[29] Boots C++ website. www.boost.org/, 2011.

[30] Cmake website. www.cmake.org/, 2011.

[31] CTIM website. www.ctim.es/, 2011.

[32] Flex website. flex.sourceforge.net/, 2008.

[33] GIMET website. www.ctm.ulpgc.es/, 2010.

[34] Manfred Weiler, Martin Kraus, Markus Merz, and Thomas Ertl. Hardware-based ray casting for tetrahedral meshes. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 44–, Washington, DC, USA, 2003. IEEE Computer Society.

[35] Lee Westover. Footprint evaluation for volume rendering. *SIGGRAPH Comput. Graph.*, 24:367–376, September 1990.

[36] Orion Wilson, Allen Van Gelder, and Jane Wilhelms. Direct volume rendering via 3d textures. Technical report, 1994.

[37] Jianlong Zhou, , Jianlong Zhou, and Klaus D. Tnnies. State of the art for volume rendering.