Universidad
Rey Juan Carlos

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Curso Académico 2002/2003

Proyecto de Fin de Carrera

RECONSTRUCCIÓN DE VOLÚMENES MEDIANTE SUPERFICIES 3D

Autor: Jose Luis Castaño Cabrales
Tutores: Luis Pastor Pérez
Jose Miguel Espadero Guillermo

22 Septiembre de 2003

A mis hermanos

Agradecimientos

Antes de continuar quisiera agradecer a todas las personas que de alguna manera me han ayudado a realizar este proyecto y sin la cual no hubiese sido capaz de llegar hasta el final.

En primer lugar, agradecer a las dos personas que me han guiado en el proyecto hasta el final. El primero, Luis Pastor Pérez, quien tuvo confianza en mi para la realización del mismo al comienzo del proyecto, y en segundo lugar, en especial a José Miguel Espadero Guillermo, quién me ha guiado durante todo el desarrollo del mismo con paciencia. Gracias por su dedicación y apoyo. Sin su dirección y supervisión, este proyecto no hubiera sido posible.

En segundo lugar, agradecer el apoyo recibido de mis padres, mis compañeros, en especial Alberto, y amigos, en especial Marcelino y Raquel, que siempre han estado ahí. También, y de forma especial, me gustaría agradecer el ánimo que me ha dado en todo momento mi cuñada Leo, sobre todo en los momentos difíciles.

En último lugar, y el más especial de todos, a mis hermanos, Javier y David, a los que dedico este proyecto. Son tantas cosas las que podría decir de ellos, que no acabaría nunca, simplemente daros las gracias por todo.

Índice general

Índice de imágenes

Resumen

Este proyecto tiene como objetivo el desarrollo de una aplicación que obtenga la representación de un volúmenes a través de superficies 3D, para conseguir un mejor entendimiento del mismo, así como generar superficies que permitan igualmente construir entrenadores de cirugía con realidad virtual. De este modo la aplicación permite,

- Cargar un volumen de datos.
- Reescalar el volumen de datos.
- Visualizar el volumen o capas de los datos.
- Visualizar el histograma del volumen.
- Obtener superficie del volumen.
- Visualizar la superficie 3D generada.
- Extras de la aplicación.

Para la visualización de un conjunto de puntos, que forman una imagen volumétrica, se utilizan diferentes técnicas que crean representaciones tridimensionales de datos (superficie) a partir de datos numéricos (volumen), las cuáles son, tanto para su estudio como para su visualización, más cómodas para el ojo humano. De entre dichas técnicas las más conocidas son la de Volumen Rendering y Surface Rendering, está última aplicada mediante el algoritmo de Marching Cubes (MC). Para la realización del proyecto utilizaremos el algoritmo de MC aplicado tanto sobre volúmenes como sobre imágenes previamente segmentadas, en los que los contornos de las diferentes zonas están marcados, dando la posibilidad de obtener la superficie de una zona determinada. Este proyecto se ha realizado en paralelo con otro proyecto fin de carrera que realiza la segmentación de volúmenes.

El proyecto ha sido desarrollado en el Departamento de Arquitectura de Computadores de la ESCET de la Universidad Rey Juan Carlos.

1. Introducción

Dado su gran interés, el estudio de la representación tridimensional (3D) en medicina se ha expandido considerablemente en los últimos años. En el campo clínico, la visualización en 3D, puede ser de gran utilidad en planificaciones, entre otras, de cirugía, entrenadores quirúrgicos, ortopedia, traumas músculo-esqueléticos, análisis de tejidos y reconstrucción de vasos sanguíneos, partiendo de estudios de imágenes médicas tales como el ultrasonido, la tomografía axial computarizada, la resonancia magnética nuclear, o la medicina nuclear, métodos a través de los cuales se extrae información anatómica o funcional del órgano o zona a estudiar.

Las posibilidades de despliegue que presentan actualmente la mayoría de las estaciones de trabajo han permitido obtener resultados espectaculares en el dominio de la representación 3D. Sin embargo, en la mayoría de los sistemas existentes actualmente, se ha dado énfasis a la etapa de despliegue, en detrimento de la calidad de la representación. Está situación se agrava considerando el caso específico de la aplicación a las imágenes médicas, puesto que el médico se basa en el resultado en 3D para emitir un diagnóstico fiable.

El desarrollo de tecnologías de procesamiento digital de imágenes, está dando una mayor calidad a la representación 3D, lo cuál, está permitiendo, tanto a los médicos en el campo clínico mejorar su valor de diagnóstico, como generar superficies que permitan construir entrenadores de cirugía con realidad virtual (RV), con el consiguiente beneficio para un cirujano de poder planear una operación o practicar sobre una zona a estudiar. El procesamiento digital de imágenes, incluye un conjunto de técnicas que operan sobre la representación digital, a objeto de destacar algunos de los elementos que conforman la escena, de modo que se facilite su posterior análisis, bien sea por parte de un usuario (humano) o un sistema de realidad virtual (RV).

En general las técnicas de procesamiento digital son aplicadas cuando resulta necesario realzar o modificar una imagen para mejorar su apariencia o para destacar algún aspecto de la información, o cuando se requiera, medir, contrastar o clasificar algún elemento contenido en la misma. También se utilizan técnicas de procesamiento, cuando se requiere combinar imágenes, porciones de las mismas o reorganizar su contenido.

Con el presente proyecto, se trata de facilitar un mecanismo de reconocimiento de imágenes médicas que permita la obtener superficies a partir de una imagen volumétrica, aportando una mayor comprensión en el campo clínico. A través de la aplicación desarrollada, podemos obtener imágenes superficiales con la suficiente calidad como para ser usadas en diferentes ámbitos, no sólo para la realización de diagnósticos por parte de un médico, sino también, para facilitar casos de prueba a los entrenadores quirúrgicos de RV con las que los médicos puedan practicar nuevas técnicas quirúrgicas, entre ellas las técnicas de cirugía minimamente invasivas (MIS).

Para conseguir la representación de una imagen en 3D, partimos de una serie de imágenes de cortes del objeto que contienen toda la información necesaria acerca del mismo. Posteriormente apilamos las imágenes de forma que obtenemos un volumen. Una vez obtenida la imagen volumétrica, podemos aplicar el algoritmo de MC, que genera una superficie a partir de un valor de umbral aportado.

La aplicación esta dirigida a desarrolladores de software de aplicaciones de entrenadores quirúrgicos con RV que necesiten nuevos casos de prueba, con los que poder perfeccionar la herramienta. Aunque también podría ser de ayuda a profesionales de la medicina que desean obtener una representación en 3D de un objeto en estudio.

La interfaz de la aplicación está implementada con el lenguaje PHP, lo que da una mayor flexibilidad a la misma para poder ejecutarse desde cualquier máquina con acceso a Internet. Por otra parte, los algoritmos están implementados

mediante el lenguaje de programación C estándar, siendo compilables en entornos Linux y Windows.

Por último, decir que la aplicación se ejecuta bajo Linux, donde se ha configurado un servidor Apache para permitir el acceso a la aplicación desde cualquier máquina a través de un navegador web.

El Apéndice A constituye un breve manual de usuario con una explicación paso a paso del funcionamiento de la aplicación.

2. Objetivos

El objetivo principal del proyecto es construir una herramienta que permita la reconstrucción de volúmenes a través de superficies 3D. La utilidad de está aplicación es proporcionar, tanto a los entrenadores de artroscopia casos de prueba reales con los que poder perfeccionar el software del mismo por parte de un informático, como a aquellos médicos que lo utilizan para poner en práctica técnicas minimamente invasivas (MIS).

2.1 Descripción del problema

El principal problema al que se enfrenta un médico cuando tiene que realizar un estudio, diagnóstico o cirugía teniendo un conjunto de imágenes en 2D, es tener una percepción espacial del objeto a estudio. Las imágenes en dos dimensiones de TC (Tomografía Computerizada) o RM (Resonancia Magnética) sirven de gran ayuda a los médicos en los diagnósticos de sus pacientes, pero tienen el inconveniente de que no se ven los órganos en su forma original.

Inicialmente, no es fácil imaginarse una imagen médica en 3D para aquellos que nunca lo han hecho, incluso para aquellos que llevan años estudiando imágenes en 2D, les resulta difícil emitir un diagnóstico o el alcance de una lesión o enfermedad. Otro inconveniente, es que cada imagen médica, procedente de un paciente a estudio, es totalmente diferente, por lo que la evaluación o previa experiencia que se tenga sobre una lesión o enfermedad de un mismo órgano o zona puede no servir de mucho, llevando incluso a la confusión a la hora de emitir un diagnóstico por parte del médico. Por ejemplo, la mala definición de las imágenes obtenidas con el aparato de RX (Rayos-X), puede inducir al médico a emitir un diagnóstico erróneo, con el consiguiente perjuicio que puede sufrir el paciente a la hora de recibir cirugía sobre la zona afectada.

Los profesionales de la medicina, desearían muchas veces, poder aislar una determinada zona de la imagen 2D a estudio y tener una representación 3D de la misma. De esta manera, tanto los diagnósticos emitidos, como la cirugía realizada sobre esta zona serían más fiables, con la consiguiente ventaja tanto para el médico que no tendría que realizar un esfuerzo para imaginarse un objeto en el espacio 3D, como para el paciente que vería como realizan operaciones quirúrgicas con total seguridad sobre la zona afectada.

Por todo ello, se utiliza la representación de volúmenes en superficies 3D, que permite observar los órganos de forma no invasiva. Las herramientas tridimensionales se utilizan para el diagnóstico, planificación y simulación quirúrgica, herramientas que día a día adquieren una mayor relevancia en el ámbito hospitalario y gracias a los esfuerzos realizados en los últimos tiempos están adquiriendo cada vez mayores prestaciones.

2.2 Reconstrucción de superficies en medicina

Con el fin de obtener unos resultados satisfactorios al finalizar la aplicación, se determinan las características que debería tener y el modo en que se iba a llevar a cabo, se realizó un estudio del estado del arte de la representación de superficies en el campo de la medicina.

La reconstrucción de superficies se está aplicando a una amplia gama de áreas médicas, planificación de la cirugía, la educación y el entrenamiento médico, el tratamiento de dolencias, el entrenamiento de habilidades médicas en cirugía con la consiguiente reducción del dolor en el paciente.

Con la tecnología que existía antes de la reconstrucción de superficies, los médicos veían la información de 3D en rodajas de 2D (ver imagen 2.1). Una de las habilidades que debían desarrollar los médicos era la de ser capaces de imaginar mentalmente, partiendo de las imágenes en 2D, la imagen en 3D. La representación volúmenes mediante superficies permite construir imágenes en 3D

que antes sólo estaban en la imaginación del médico. De esta manera se puede examinar en detalle, compartir y discutir con otros médicos la imagen obtenida.

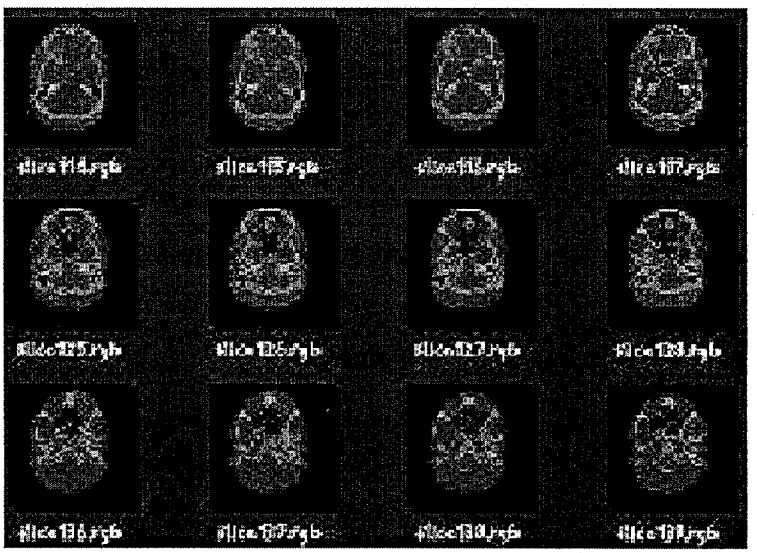


Imagen 2.1: Imágenes en 2D obtenidas mediante CT

2.2.1. Obtención y representación de datos

La obtención de imágenes médicas para la generación de nuevos casos de prueba para un entrenador quirúrgico depende totalmente de los datos del paciente. Durante el entrenamiento, los cirujanos operan sobre un modelo construido a partir de los datos del paciente. Lógicamente, esto requiere que el modelo se ajuste tanto como sea posible a los datos disponibles. Se recopilan datos anatómicos o fisiológicos del paciente para las imágenes. Se usan técnicas gráficas por ordenador (representación y modelado) para presentar los datos como (una parte de) un cuerpo virtual, de tal forma que puedan ser examinados y manipulados.

En el siguiente apartado, se mencionan algunas de las distintas fuentes de adquisición de imágenes médicas.

2.2.1.1 Tipos de imágenes médicas

El área de imagen médica, considera un conjunto de modalidades de adquisición, las cuales se diferencian en cuanto a la naturaleza de los principios físicos involucrados en el proceso de adquisición. Adicionalmente existen también diferencias en cuanto a la aplicación médica. Las modalidades más comunes de imágenes médicas son los rayos X, la tomografía computada, la resonancia magnética nuclear y las imágenes por ultrasonidos. Aunque estás no son las únicas si son las más utilizadas, por lo que nos centraremos en ellas.

Las fuentes de la imagen médica se diferencian fundamentalmente por el tipo de información que es detectada. Cada modalidad da una representación diferente de los órganos o de su funcionamiento. En la siguiente sección se dará una breve descripción de las diferentes modalidades de la imagen médica.

2.2.1.1.1. La ecografía

Es una técnica de exploración por ultrasonidos. El dispositivo transductor (Barra piezoeléctrica que funciona como emisor y receptor) emite una onda ultrasónica de una frecuencia de varios MHz, en una orientación dada. La señal detectada corresponde a la superposición de ecos o reflexiones que se producen debido a los cambios de impedancia acústica en las diferentes fronteras de los órganos (Quistgaard, 1997). La imagen 2D se obtiene por el barrido de un haz según un plano cualquiera. Una de las dimensiones está dada por el barrido del haz, la otra dimensión corresponde al tiempo de retorno del eco.

Si bien la resolución geométrica es inferior en relación con otras modalidades (Tomografía computada e imagen por resonancia magnética nuclear), sus ventajas se fundamentan en el hecho que la velocidad de adquisición es elevada (15 á 30 imágenes por segundo) lo cual permite la exploración de órganos en movimiento. La naturaleza no ionizante del proceso de adquisición y el precio razonable de los equipos. Una ecografía que muestra la vena aorta abdominal en color se presenta en la imagen 2.2, y otra que muestra una ecografía del cerebro de un neonato (ver imagen 2.3).

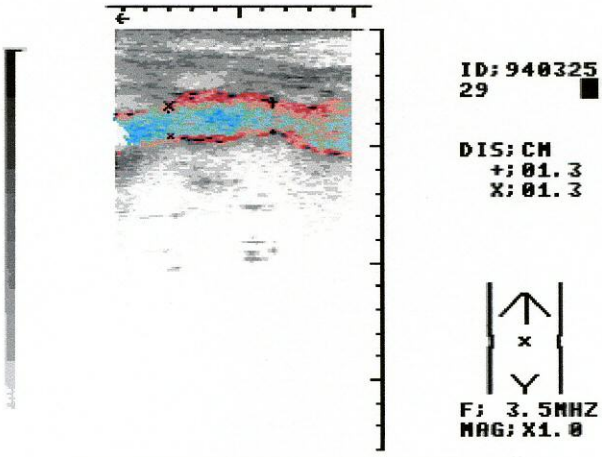


Imagen 2.2: Vena aorta mediante ecografía

Imagen 2.3: Cerebro de un neonato mediante ecografía

Las aplicaciones de ultrasonido incluyen cardiología, neurocirugía, ginecología, e imágenes abdominales y vasculares. Al ser esta técnica inofensiva e instantánea, se usa a menudo como una herramienta de guía en tiempo real en las intervenciones quirúrgicas.

2.2.1.1.2 La radiología

Los rayos X son una clase de radiación electromagnética similar a la luz para la cual la longitud de onda es más pequeña. Sus propiedades físicas más importantes son su capacidad para atravesar la materia, producir fosforescencia e imprimir películas con emulsiones fotográficas. Los rayos X pueden también producir cambios en los tejidos biológicos y son capaces de producir la ionización de los materiales gaseosos.

Los rayos X son a menudo utilizados para producir imágenes médicas y para desarrollar efectos terapéuticos en algunos pacientes. En la actualidad, las modalidades de imagenología médica basadas en la utilización de rayos X incluyen la radiografía convencional (ver imagen 2.4), la video-angiografía (ver imagen 2.5) o fluoroscopia y la tomografía. La imagenología por rayos X es en la actualidad una de las modalidades más utilizadas en el dominio médico en general. En el contexto de la angiografía, la imagenología por rayos X permite obtener la resolución espacial indispensable para la definición y cuantificación de la red arterial y venosa.

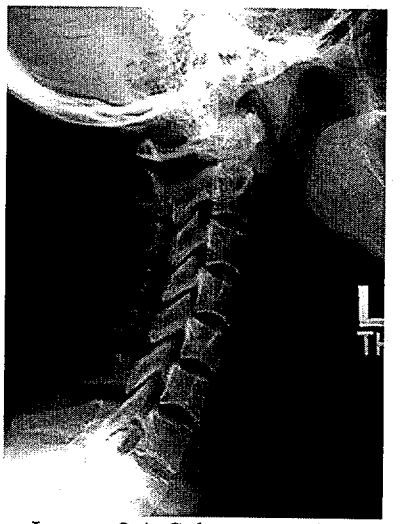


Imagen 2.4: Columna cervical

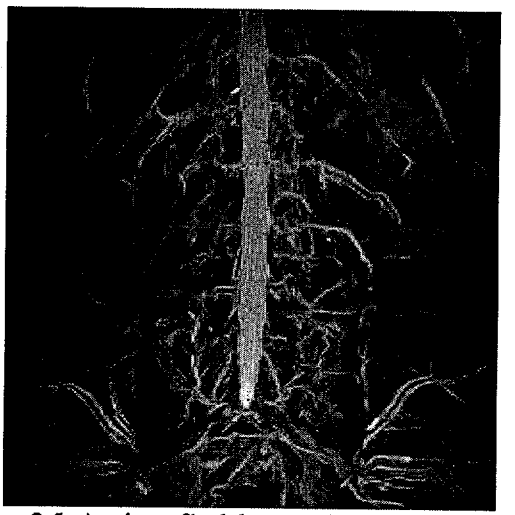


Imagen 2.5: Angiografía del tórax obtenida mediante RX

2.2.1.1.3 La tomografía computada ó tomografía por rayos X

Una importante modalidad en la obtención de imágenes médicas es la tomografía asistida por computadora (TC) o tomografía computada en donde las imágenes del interior del cuerpo se reconstruyen a partir de un conjunto de medidas de proyección (Herman, 1980). Tales proyecciones son obtenidas mediante la exposición de un objeto a un tipo de radiación, según diversos ángulos y midiendo, para cada posición angular, la intensidad del haz de radiación que atraviesa el objeto. Algunas variaciones de las CT son tomografías por ordenador en espiral (Spiral CT) y tomografías por ordenador abiertas (Open CT).

Las técnicas de TC crean imágenes de cortes transversales de un objeto juntando numerosas imágenes de proyección sobre el objeto en el plano de interés. Una imagen de proyección es una imagen unidimensional en la que el brillo de cada píxel es igual a la absorción de rayos X en la sección del objeto. Combinando las múltiples vistas de proyección, se sintetiza la imagen del corte transversal. Según el plano de orientación existen tres tipos de cortes: axial, coronal y sagital (ver imagen 2.6). En TC si el corte es axial dicha imagen se conoce como TAC (ver imagen 2.7). En la imagen 2.8 podemos observar un dispositivo de captura de imágenes TAC.

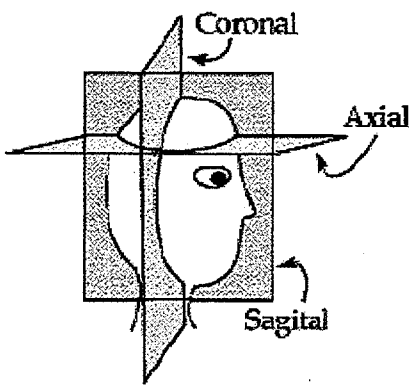


Imagen 2.6: Planos de orientación

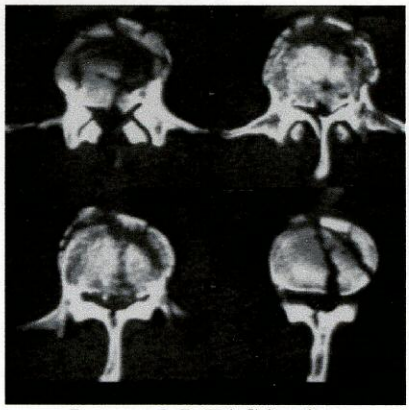


Imagen 2.7: TAC lumbar

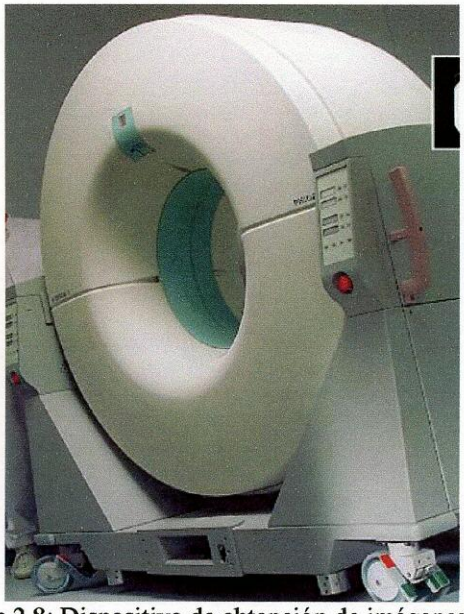


Imagen 2.8: Dispositivo de obtención de imágenes TAC.

2.2.1.1.4 Resonancia magnética

La resonancia magnética (MRI), consiste en medir la concentración y el tiempo de relajación de ciertos núcleos atómicos (generalmente núcleos de Hidrógeno) excitados bajo la acción de un campo magnético fijo y de un pulso de radio-frecuencia (Siedband, 1992). El interés de la imagen por resonancia magnética MRI se debe al hecho de que el equipo utiliza una radiación no ionizante que permite una buena discriminación de los tejidos y la adquisición tridimensional de una zona del cuerpo. El campo de aplicación de esta modalidad es muy amplio y la limitación principal son los costos de los equipos. Las

imágenes que vemos con la MRI se realizan mediante cortes según los 3 planos en que dividimos el cuerpo humano, coronal, axial y sagital (ver imagen 2.6).

Respecto a otras técnicas como RX y la TAC, se usa cada vez más tanto por sus ventajas, como por permitir cortes más finos, y en varios planos, ser más sensible para demostrar accidentes vasculares cerebrales, tumores y otras patologías, y no utilizar radiaciones ionizantes. La siguiente imagen muestra una MRI de la rodilla derecha con rotura del ligamento cruzado anterior (ver imagen 2.9).

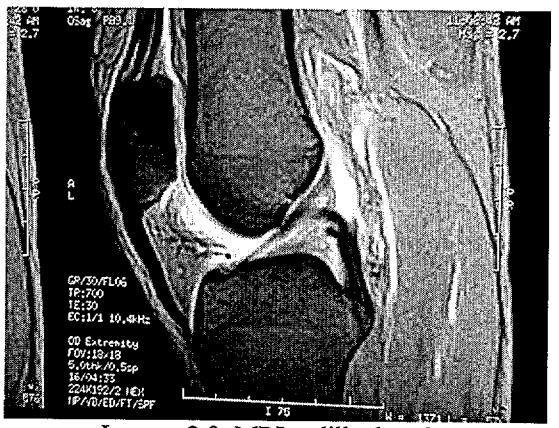


Imagen 2.9: MRI rodilla derecha

2.2.1.1.5 Resonancia magnética nuclear

Las imágenes por resonancia magnética nuclear (MRN) se basan en la detección de la radiación emitida por cada punto de un órgano después de administrar al paciente una sustancia que incluye trazas de un radioelemento, el cual tiene un periodo de vida muy corto y se degrada hasta convertirse en una sustancia inerte. Esta modalidad generalmente proporciona imágenes cuya intensidad cuantifica el funcionamiento del órgano, proporcionando información acerca de la capacidad de tal órgano de asimilar o transformar la sustancia que se ha inyectado.

Generalmente se usan tres esquemas de adquisición, uno de ellos es la gammagrafía, en donde la emisión de radioelemento se detecta en un arreglo de sensores fijo produciendo una imagen 2D. Los otros dos esquemas son de tipo

tomográficos y corresponden a la emisión de fotones gamma únicos (Single Photon Emission Tomography: SPECT), donde se administra un rayo gama que emite la sustancia química del radioisótopo, y se registran los fotones emitidos por el isótopo en decaimiento (ver imagen 2.10), e imágenes por emisión de positrones (PET: Positron Emission Tomography), las cuáles son más exactas pero también más costosas, puesto que se necesita un ciclotrón estático para proporcionar los isótopos de emisión de positrones. Los escáneres para PET son también más costosos que las cámaras de fotón único. La tomografía por emisión de positrón, permite examinar el corazón, el cerebro y otros órganos. Las imágenes de PET muestran el funcionamiento químico de un órgano o de un tejido, (ver imagen 2.11) en contraposición a los rayos X, a las CT o a las MRI, que sólo muestran la estructura del cuerpo.

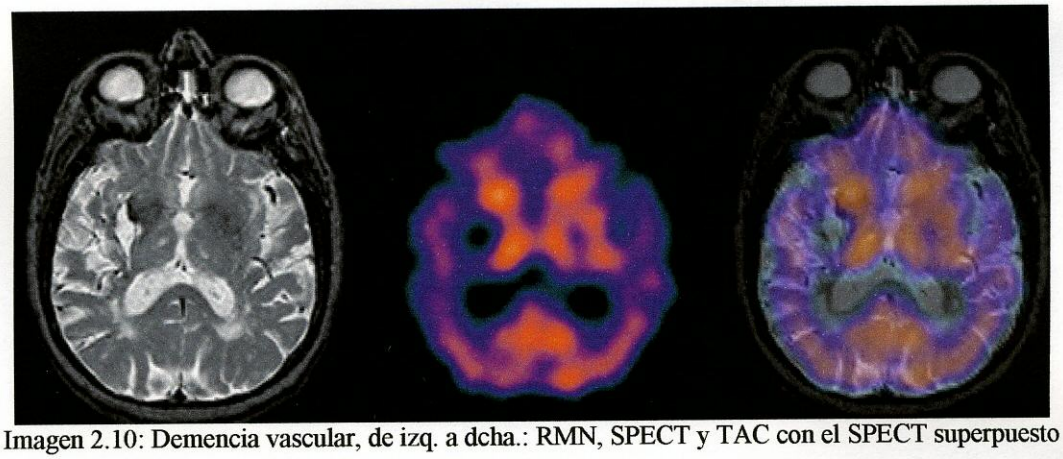


Imagen 2.10: Demencia vascular, de izq. a dcha.: RMN, SPECT y TAC con el SPECT superpuesto

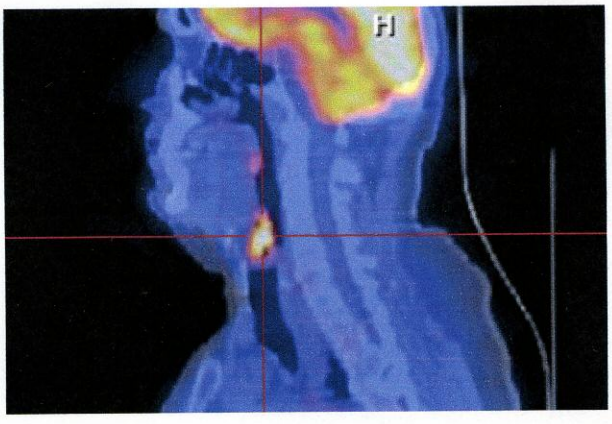


Imagen 2.11: PET que identifica un cáncer en la base de la lengua.

2.2.2 Representación de volúmenes

Partiendo de imágenes en 2D, obtenidas mediante alguna de las técnicas comentadas anteriormente, es posible obtener una representación en 3D de las mismas. Hay dos técnicas que permiten la representación 3D a partir de cortes transversales 2D. La primera realiza una visualización volumétrica directa de las imágenes (Volume Rendering) y la otra realiza una visualización de las superficies de los datos (Surface Rendering).

Cada corte transversal que forma parte del volumen está formado por un vector de 2D de píxeles, donde cada uno viene dado por una posición (x, y). Al unir cada corte transversal cada píxel vendrá determinado por una posición (x, y, z), donde la coordenada z indica el número de corte en el volumen. Además estas tres coordenadas de un píxel representan el brillo que tiene un volumen, es decir, podemos decir que un píxel tiene profundidad, dado por la coordenada z, con lo que posibles huecos que pudiera haber en la imagen no se muestran.

En las siguientes subsecciones se realiza una breve descripción de ambas técnicas.

2.2.2.1 Surface Rendering

Se basa en los contornos del objeto para obtener una malla de polígonos (generalmente triángulos). Antes de realizar está operación de reconocimiento de contornos, se realiza una segmentación del volumen de datos en la que a través de un umbral dado (la umbralización es una de las muchas técnicas usadas para realizar la segmentación de un volumen) se obtienen los contornos del objeto (ver imagen 2.12). En el campo médico esta técnica es usada cuando se desea obtener una visión global del objeto, o realizar manipulaciones, por ejemplo mediante entrenadores quirúrgicos.

El algoritmo de Marching Cubes, el cuál hemos usado para realizar este proyecto, genera de manera eficaz una malla triangular 3D teniendo en cuenta la

pertenencia de los voxels al contorno del objeto. A continuación se citan las ventajas y desventajas que tiene la técnica de Surface Rendering.

- Permite obtener una mejor aproximación de la superficie del objeto.
- El procesamiento de la imagen 3D obtenida es más rápido, ya que son todos datos geométricos, aunque la generación de la superficie es mucho más lento debido al tamaño de los datos volumétricos y a la necesidad de acceder a ellos en orden aleatorio.
- Requiere un preproceso para la identificación de contornos.
- Cuando el objeto tiene mucha información, es decir, hay demasiadas intensidades, la identificación de los contornos del objeto puede llegar a ser difícil.

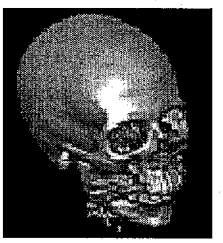


Imagen 2.12: Cráneo

2.2.2.2 Volume Rendering

Está técnica no requiere segmentación previa del objeto, aquí cada píxel de la imagen forma parte del resultado final del objeto obtenido. Básicamente, se puede decir que se realiza una proyección de los voxels sobre un dispositivo de salida a través del cual se visualiza. El proceso de obtención del volumen tiene dos fases, una de sombreado, donde a cada voxel de la imagen se le asigna un color y una opacidad que representan el color y la transparencia con la que se visualizará el objeto, y una fase de proyección, en la que los voxels de la imagen son proyectados sobre la pantalla del ordenador para obtener una imagen 3D (ver imagen 2.13). En el campo médico esta técnica es usada cuando se desea obtener una representación de los datos desde dentro, ya que se visualizan toda la

información tal y como está almacenada. A continuación se citan las ventajas y desventajas que tiene esta técnica.

- En teoría no requiere preproceso, aunque es necesario calcular las opacidades existentes en el objeto.
- Es muy cara computacionalmente, ya que en cada corte de la imagen se examina cada voxel para calcular la opacidad.
- El proceso de mostrar la imagen es muy caro, ya que ocupa muchos Mb, aunque sólo sea una pequeña parte del volumen.

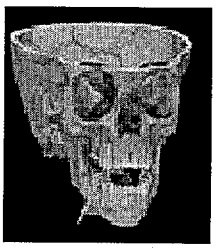


Imagen 2.13: Cráneo.

En ambas técnicas para poder trabajar directamente sobre la imagen generada de manera correcta, sería conveniente hacerlo en tiempo real mediante hardware específico. Un ejemplo son las tarjetas gráficas existentes en los equipos de Silicon Graphics [1], para superficies 3D, y las tarjetas VolumePro de TeraRecon Inc [2]. para volúmenes.

2.3 Estudio de las alternativas

En las siguientes subsecciones se describen las decisiones que se han tomado para el desarrollo de la aplicación.

2.3.1 Representación de superficies 3D

La representación de un volumen a través de una superficie 3D se puede obtener aplicando diferentes técnicas de reconstrucción de volúmenes.

Para poder visualizar un volumen podemos aplicar tanto la técnica de Volume Rendering como la de Surface Rendering. Teniendo en cuenta la finalidad de este proyecto, donde el objetivo es la generación de casos de prueba para un entrenador quirúrgico, se ha elegido la técnica de Surface Rendering. Esta técnica nos permite construir una superficie 3D del objeto en estudio, dando una visión global del mismo, que es lo que busca el cirujano para poder tener nuevos casos de prueba para el entrenador quirúrgico. La malla de polígonos con la que se representa la superficie 3D, permite que se pueda cargar, visualizar y manipular en un entrenador quirúrgico.

Dentro de los posibles algoritmos que construyen superficies 3D a partir de un volumen, hablaremos de tres técnicas, la descrita en el algoritmo de RayCasting [3], la descrita en el algoritmo del escultor [4] y por último la descrita en el algoritmo de Marching Cubes [5], que será la que finalmente usemos para la realización de este proyecto.

RayCasting Este algoritmo, dentro del Rendering de volúmenes se clasifica dentro del Rendering Directo de Volúmenes (DVR), técnica que visualiza directamente la información sin pasar por representaciones previas, por lo que son apropiados para datos no estructurados, o cuya forma es geométricamente compleja. El RayCasting consiste en lanzar rayos desde el observador hacia la escena y cuando el rayo intersecciona con un objeto se calcula la intensidad de la luz en ese punto. Cuantos más rayos se lancen hacia la escena la superficie obtenida será de mayor calidad, aunque de esta manera el coste de computación aumenta considerablemente (ver imagen 2.14). Los pasos básicos que realiza este algoritmo para la obtención de la superficie son tres; primeramente se asigna un color y una opacidad a cada muestra de datos (función de transferencia), en segundo lugar se proyectan las muestras sobre el plano de la imagen y por último se mezclan las proyecciones basándose en un modelo físico de propagación de la luz sobre un material semitransparente. Hay dos tipos de RayCasting; el primero basado en voxels (ver imagen 2.15), con interpolación constante (ver imagen 2.16), por lo que tiene la ventaja de mayor simplicidad, velocidad, y la desventaja de mayor aliasing por discretización de rayos (las líneas rectas inclinadas se ven como escalones), y el segundo basado en celdas (ver

imagen 2.17), con interpolación trilineal (ver imagen 2.18), por lo que estos algoritmos son más costosos y complejos, pero con mayor calidad.

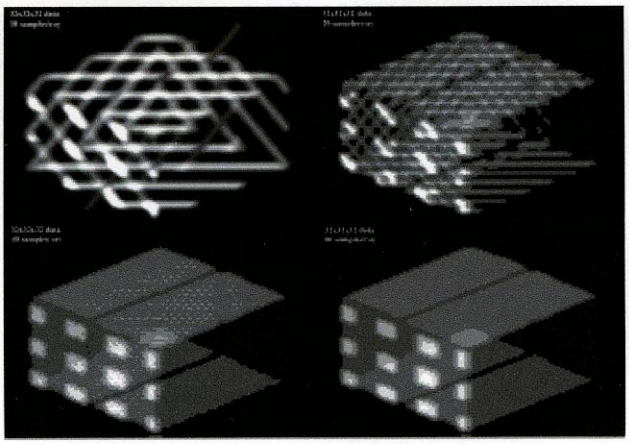


Imagen 2.14: Plano obtenido por incidencia de rayos, de izq. a dcha. (10, 20, 40, 80)

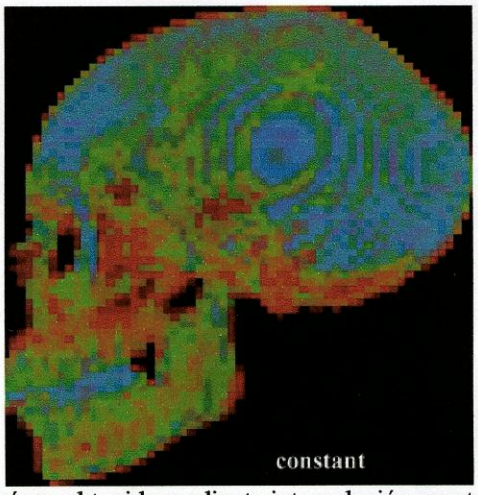


Imagen 2.15: Cráneo obtenido mediante interpolación constante (500 rayos)

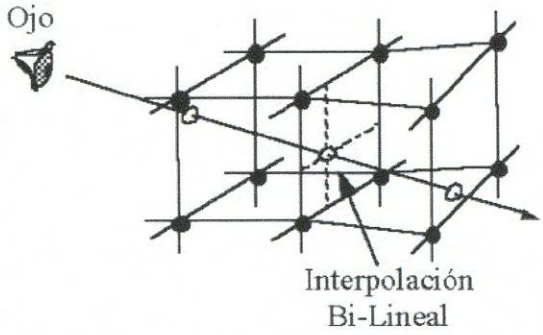


Imagen 2.16: RayCasting basado en voxels

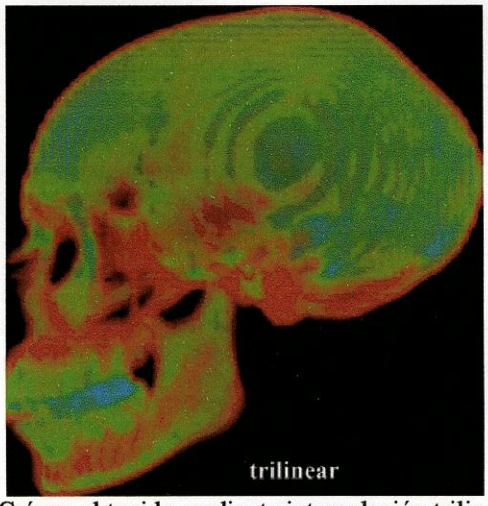


Imagen 2.17: Cráneo obtenido mediante interpolación trilinear (500 rayos)

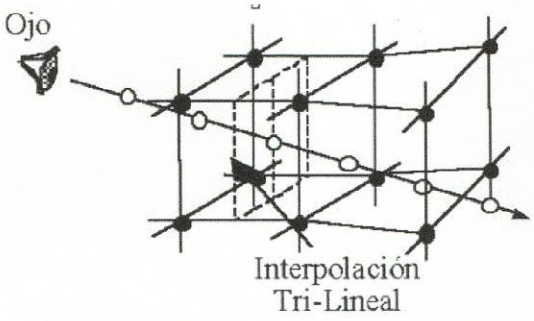


Imagen 2.18: RayCasting basado en celdas

Escultor Es un algoritmo de reconstrucción interpolativo, en el que partiendo de una nube de puntos se obtiene una superficie 3D. El algoritmo se divide en tres pasos básicos (ver imagen 2.19).

- Generar la tetraedralización de Delaunay de la nube de muestras, es decir, realizando todas las combinaciones posibles se unen todos los puntos mediante tetraedros.
- Discernir qué tetraedros deben pertenecer al modelo. Para ello se realiza lo siguiente.
    - Se genera una lista de tetraedros con alguna cara externa.
    - Se calcula el perímetro medio de las caras externas (P).
    - Se eliminan de la lista los tetraedros con alguna cara externa cuyo perímetro sea mayor que $k \cdot P$.

- Se introducen en la lista los tetraedros expuestos al exterior al ser eliminados.
- Se repite desde el principio hasta que no haya cambios.

- Extraer la superficie externa de los tetraedros seleccionados

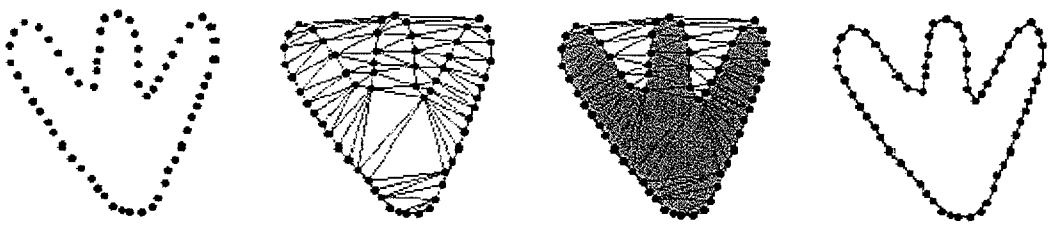
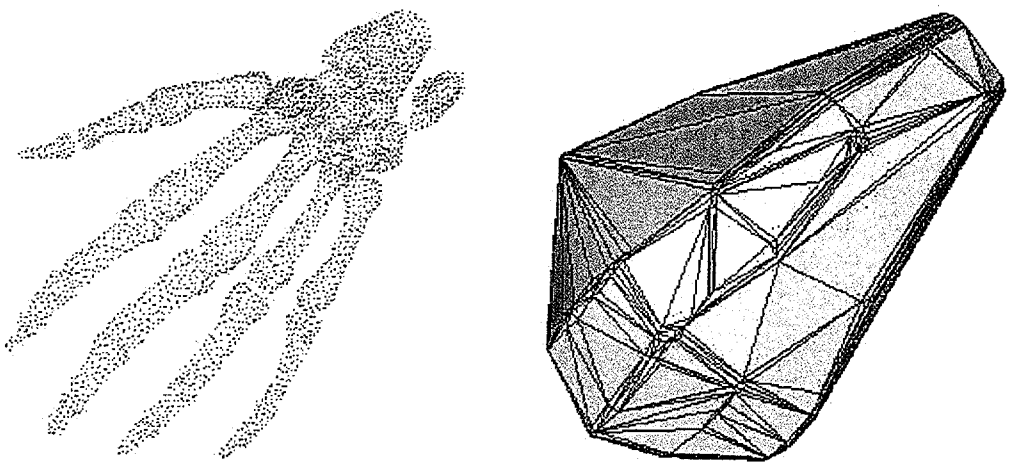


Imagen 2.19: Imagen explicativa del algoritmo del Escultor

En las siguientes imágenes (ver imagen 2.20) se muestra el proceso de reconocimiento del esqueleto de una mano.



nube de entrada

tetraedralización inicial

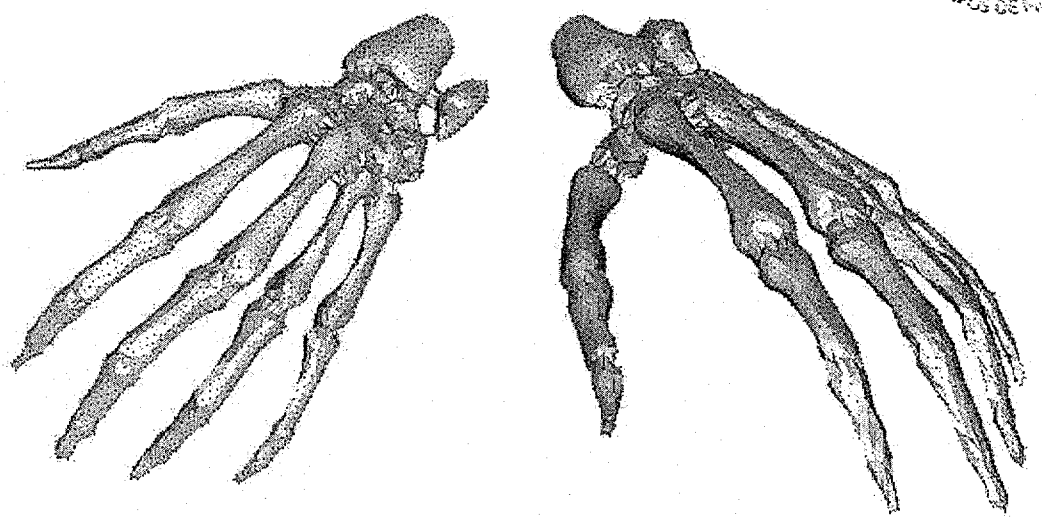


Imagen 2.20: Esqueleto de la mano obtenida mediante el algoritmo del Escultor

Entre las limitaciones de este algoritmo podemos mencionar; primero, que la tetraedralización de Delaunay es un proceso costoso para cantidades elevadas de muestras; segundo, que las muestras deben tomarse a una distancia similar en toda la superficie y por último, que el resultado depende de un parámetro (k), que mide el perímetro de la mayor cara externa aceptada respecto del promedio.

Marching Cubes Este algoritmo, dentro del Rendering de volúmenes se clasifica dentro del Rendering de superficies, técnica que primero convierte los datos a una representación intermedia (malla de polígonos), que en general es una superficie umbral del volumen. Está técnica clasifica cada celda según los valores de sus vértices respecto al valor umbral. Una celda pertenece a la superficie si tiene al menos un vértice con valor inferior al umbral y otro con valor superior a éste. Hay 2^8 combinaciones de celdas de superficie, que se reducen a 128 por complemento (se invierten los puntos marcados y luego las normales de los triángulos) y a 15 por rotaciones (ver imagen 2.21). El Marching Cubes produce superficies de muy buena calidad con bajo coste computacional, por lo que es una técnica apropiada para la generación de casos de prueba de entrenadores quirúrgicos. En la imagen 2.22 se observa en un cráneo visto desde dos posiciones.

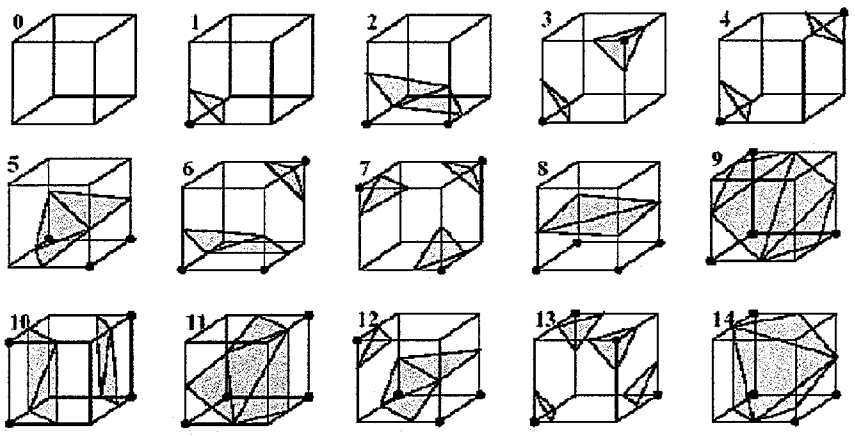


Imagen 2.21: Configuraciones usadas en Marching Cubes


Imagen 2.22:Vistas de un cráneo obtenido mediante Marching Cubes

En cuanto al lenguaje informático utilizado para desarrollar el algoritmo de Marching Cubes, se ha tenido en cuenta que este fuese portable, por lo que el lenguaje utilizado ha sido C estándar. Las aplicaciones utilizadas para compilar la aplicación han sido Borland C++ 5.02 para entornos Windows y gcc el compilador de GNU. Para representar la malla de polígonos generada por el algoritmo, se ha utilizado Open Inventor, que también es portable para cualquier entorno.

2.3.2 Interfaz de la aplicación

Para la realización de la interfaz, se ha buscado la portabilidad para entornos Windows y Linux. Se podía elegir entre una interfaz web o una interfaz programada en C.

Se ha elegido la interfaz web programada en PHP, que se ejecuta en un equipo Linux, lo que permite ejecutar la aplicación desde un explorador web.

2.4 Metodología empleada

La metodología empleada ha sido Programación Extrema (XP: Extreme Programming) [6]. La Programación Extrema es una metodología de desarrollo de software que se basa en la simplicidad, la comunicación y la retroalimentación. En la programación extrema se crea un plan de pequeñas pero frecuentes entregas.

La planificación del presente proyecto incluye la división en tareas pequeñas e independientes, así como un calendario de entregas con plazos breves. Cada semana se convoca una reunión de seguimiento. Esto permite controlar la velocidad del proyecto y detectar retrasos lo antes posible. Se va corrigiendo la propia metodología XP cuando falla. El diseño busca la simplicidad. Crea soluciones puntuales para reducir riesgos e intenta reaprovechar cuando sea posible. Durante el desarrollo, el cliente debe estar siempre disponible. Para este proyecto, al no existir un cliente concreto, se ha tenido en cuenta la realimentación proporcionada por mi tutor. Se han ido desarrollando unidades de prueba. La integración del código se ha realizado frecuentemente, dejándose las optimizaciones para el final.

A continuación se muestra una tabla con la duración de las tareas de que ha constado el proyecto de acuerdo con la metodología de programación extrema.

Nº	Tarea	Inicio	Fin	Descripción
1	Algoritmo de Marching Cubes	03/03/03	30/04/03	Algoritmo de extracción de superficies 3D a partir de un volumen
2	Algoritmo de reescalado	04/09/03/	10/09/03	Algoritmo que permite reescalar un volumen en sus tres dimensiones.
3	Algoritmo de selección de capa	03/09/03	03/09/03	Algoritmo que permite seleccionar una capa del volumen.
4	Interfaz de usuario	02/09/03	20/09/03	Implementación de la interfaz de usuario en PHP.
5	Documentación	09/09/03	21/09/03	Documentación de cada una de las tareas y de la memoria final.

3. Descripción informática

El lenguaje de programación elegido, por motivos de portabilidad, es C estándar para el desarrollo de algoritmos, y de PHP para la realización de la interfaz de usuario. El código entregado compila tanto en plataformas Windows como Linux. Para el desarrollo del código se ha utilizado el compilador de Windows Borland 5.02, y para la interfaz el entorno de programación PHP Dreamweaver.

En las siguientes subsecciones se describen los algoritmos utilizados, así como las decisiones tomadas para desarrollar la aplicación. Principalmente hablaré acerca del algoritmo de generación de superficies 3D Marching Cubes y del desarrollo de la interfaz de usuario, aunque también comentare brevemente algunos conceptos como el de interpolación y segmentación.

3.1 Algoritmo de Marching Cubes

El algoritmo desarrollado esta basado en la descripción realizada por William E. Lorensen y Harvey E. Cline [1], el cuál permite de manera sencilla y rápida crear una malla 3D a partir de un volumen dado. Para la construcción de la superficie se tiene en cuenta un valor de umbralización, que determina si los píxeles forman parte de la superficie final o no. Para tener una idea acerca de los valores de umbral que tiene el volumen se puede realizar un histograma.

Histograma de una imagen El histograma de una imagen es ampliamente utilizado como herramienta tanto cualitativa como cuantitativa. A continuación se presenta un ejemplo, a través del cual se ve la utilidad de los mismos. Este corresponde a un gráfico de la distribución de valores de intensidad de los píxeles de una imagen (niveles de gris) o de una porción de la misma. Denotamos como h(i), el número de píxeles que dentro de la región de interés tiene el valor de intensidad i, donde i = 0, 1, 2, ..., L-1 es el número posible de

niveles de gris para la imagen. Los valores h(i), corresponderán entonces a los valores del histograma. El gráfico del histograma es bidimensional y en él se grafica h(i) en función de i. Tal gráfico, puede proporcionar importante información acerca del brillo y contraste de una imagen así como de su rango dinámico. En la imagen 3.1 se muestra el dibujo de un histograma típico.

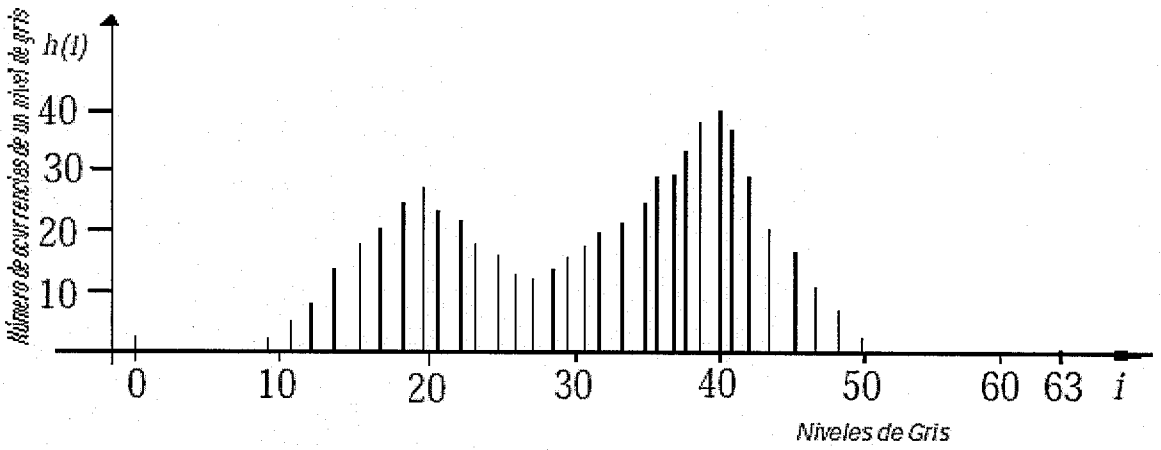


Imagen 3.1: Histograma para los niveles de intensidad de una imagen, con L = 64

Si bien el histograma anterior no representa a una imagen real, resulta posible a partir del mismo deducir alguna información hipotética acerca de lo que sería la imagen. Por ejemplo, la imagen tiene 64 niveles de gris, sin embargo, tal rango no es utilizado de manera completa, pues no se tienen píxeles con valores superiores a 50.

El Marching Cubes (MC) obtiene superficies 3D a partir de volúmenes obtenidos mediante TC, RMN o cualquier otra técnica. La superficie generada es de buena calidad, pero tiene el problema de que cuando hay demasiada información en el objeto a estudio, por ejemplo en la RMN de una rodilla se observan los huesos, meniscos, ligamentos, la superficie obtenida muestra una visión global. Si el algoritmo trabajase sobre un volumen previamente segmentado, se podría obtener la superficie 3D de la zona deseada.

Segmentación de imágenes La segmentación consiste en extraer objetos de interés a partir de las imágenes en tonos de grises. La segmentación proporciona información superficial al generar una imagen binaria de la estructura

de interés o proporciona una imagen con información hacia adentro del volumen. Este tema ha sido ampliamente investigado, pero no se ha llegado a obtener un método universal que clasifique automáticamente (aproximadamente) cualquier estructura anatómica de interés, a partir de cualquier modalidad de imagen, de ahí que se continúe esta investigación según la aplicación. En la imagen 3.2 se muestra un corazón al que se le ha aplicado segmentación. Se aprecia que después de la segmentación se puede observar con mayor claridad el contorno de las venas que rodean el corazón.

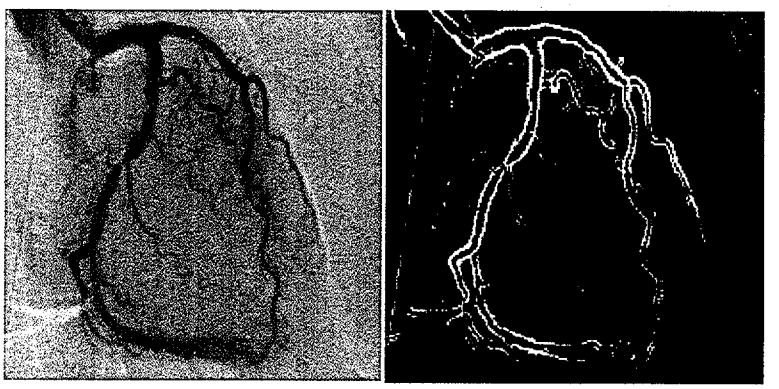


Imagen 3.2: Corazón, imagen original a la izq., imagen segmentada a la dcha.

3.1.1 Descripción

La finalidad del algoritmo es crear una superficie de densidad constante a partir de un vector de datos 3D, donde las dimensiones vienen dadas por el alto, ancho y número de capas del volumen. La idea es crear una malla poligonal (generalmente triángulos) que se aproxime a la imagen volumétrica.

Para determinar que partes del volumen formarán la isosuperficie, se crea un cubo lógico a partir de dos capas contiguas. Posteriormente se comparan los ocho vértices del cubo lógico formado con el valor umbral introducido. Un cubo pertenece a la isosuperficie si al menos hay un vértice que esta por debajo del umbral y otro que esta por encima, si cumple esto indica que el cubo formará parte de la isosuperficie final. Cuando todos los vértices del cubo están por debajo o por encima del umbral no se procesan ya que no formarán parte de la isosuperficie final. La imagen 3.3 muestra como se forma el cubo lógico a partir de dos capas del volumen.

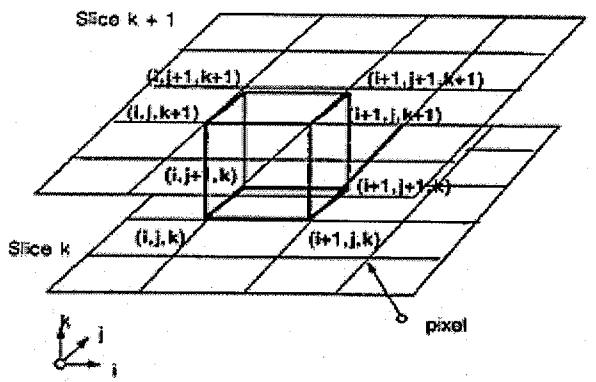


Imagen 3.3: Formación de un cubo lógico a partir de dos capas (slice)

Al marcar un cubo, hay $2^8 = 256$ posibilidades diferentes en las que la superficie podría interseccionar. Cuantos más vértices estén por debajo del umbral, más posibilidades habrá y por tanto habrá que realizar más triangulaciones. Mediante rotaciones y complementos se reducen las 2^8 a 15 posibilidades diferentes (ver imagen 3.4). Creando un índice para cada caso posible (ver imagen 3.5), basado en el estado del vértice, podemos determinar que arista del cubo intersecciona con un vértice y así hallar todas las posibles combinaciones. A continuación se interpola (mediante interpolación linear) la intersección con la superficie a lo largo de cada arista teniendo en cuenta el valor de umbral (ver imagen 3.6);

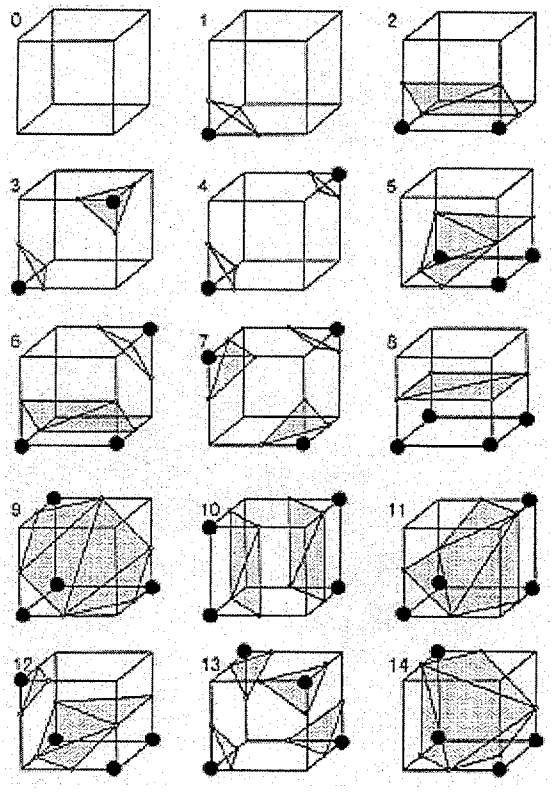


Imagen 3.4: Casos de triangulación posibles en un cubo

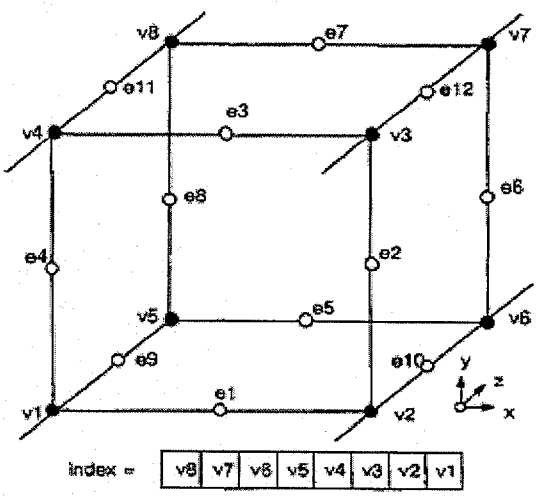
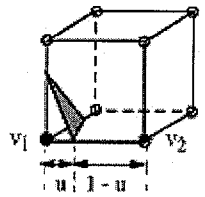


Imagen 3.5: Numeración de los vértices y aristas de un cubo



$$v_i = v_1 * (1-u) + v_2 * u \qquad u = \frac{v_1 - v_i}{v_1 - v_2}$$

Imagen 3.6: Interpolación lineal

Por último, para el cubo marcado, se calculan las normales en cada vértice del cubo, de esta manera se consigue una imagen con un efecto Gouraud-shaded, y se interpolan las normales en los vértices de los triángulos (ver imagen 3.7).

$$G_x(i,j,k) = \frac{D(i+1,j,k) - D(i-1,j,k)}{\Delta x}$$

$$G_y(i,j,k) = \frac{D(i,j+1,k) - D(i,j-1,k)}{\Delta y}$$

$$G_z(i,j,k) = \frac{D(i,j,k+1) - D(i,j,k-1)}{\Delta z}$$

$$\overrightarrow{n_1} = u\overrightarrow{g_2} + (1-u)\overrightarrow{g_1}$$

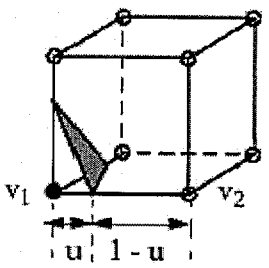


Imagen 3.7: Calculo de las normales (arriba), interpolación de las normales (abajo)

En resumen, los pasos que ejecuta el algoritmo de MC para generar una superficie de 3D son los siguientes.

1. Leer el volumen de datos y cargarlo en memoria.
2. Coger dos capas contiguas del volumen y formar cubos con cuatro píxeles de una capa y sus vecinos de la capa superior.
3. Calcular un índice del cubo a partir de la pertenencia o no de los vértices a la superficie.
4. Con el índice mirar en la tabla de aristas para determinar el numero de aristas que interseccionan.

5. Usando las densidades (valor de cada píxel) en cada vértice de la arista, encontrar una superficie que interseccione con la arista a través de la interpolación lineal.
6. Calcular una normal en cada vértice del cubo usando diferencias centrales e interpolar la normal en cada vértice del triángulo.

A continuación se muestran varias imágenes obtenidas al aplicar MC. La imagen 3.8 muestra la superficie de un cráneo que contiene 550.000 triángulos, la imagen 3.9 muestra la superficie de una cara desde dos perspectivas que contienen 375.000 triángulos.

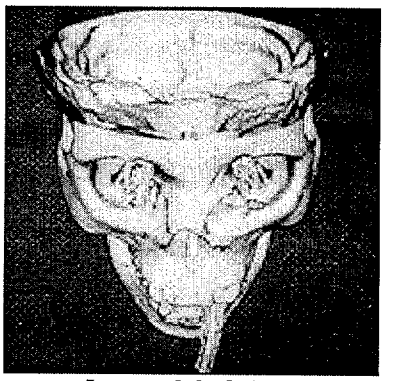


Imagen 3.8: Cráneo

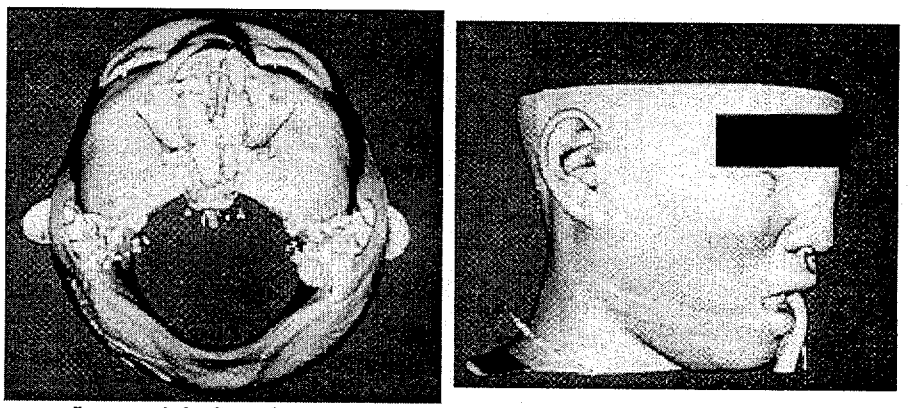


Imagen 3.9: Cara de una persona vista desde dos perspectivas.

Ventajas del MC; fácil renderizado y manipulación, y alta resolución. Inconvenientes del MC; posibles agujeros en la superficie generada debido a caras ambiguas (ver imagen 3.10) y complejidad en el modelo.

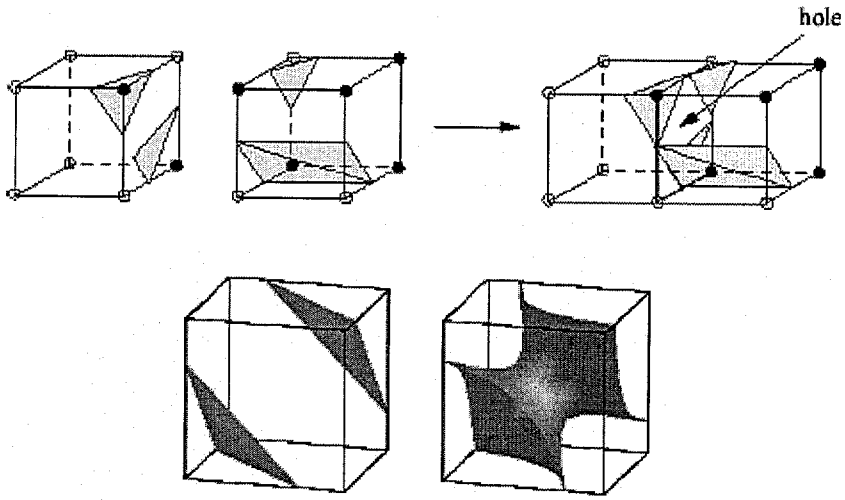


Imagen 3.10: Agujeros en el algoritmo de Marching Cubes

3.1.2 Diseño e implementación

En el apartado B.1 se realiza una descripción de las funciones utilizadas en el algoritmo. Por lo tanto, en este punto se comentaran algunos aspectos a tener en cuenta en la ejecución del MC, así como las modificaciones realizadas en el algoritmo y demás consideraciones que se estiman oportunas para el correcto funcionamiento del mismo.

Modificaciones realizadas al algoritmo original Para determinar la superficie que formara parte de la isosuperficie final, introducen dos valores de umbral, uno alto y otro bajo. De esta manera, además de ser sólo procesados aquellos píxeles que pertenezcan al intervalo dado por los umbrales *u1* y *u2*, el tiempo de ejecución del algoritmo será menor ya que se procesan menos cubos en el volumen dado. Un ejemplo de la utilidad de esta modificación es poder reconstruir ciertas partes del volumen. Por ejemplo, dado un hueso de una pierna, se puede obtener únicamente la superficie formada por los bordes del hueso, dejando el interior del mismo hueco. Para que se obtengan unos resultados óptimos, sería deseable que la imagen estuviese previamente segmentada, ya que los bordes o otras áreas estarían mejor delimitadas.

Datos de entrada que recibe el MC El MC únicamente procesa ficheros de 8 y 16 bits por píxel en formato RAW. Los ficheros de datos RAW, además de contener para cada píxel sólo valores en gris, no contienen ningún tipo de

información que no sea la referente a la imagen, como por ejemplo tamaño, densidad u otras características que podría tener cualquier otro tipo de fichero con extensión no RAW. Por este motivo, es necesario que se conozca información acerca del fichero, como son, el ancho, el alto, el número de capas y los bits por píxel que tiene el volumen.. El fichero de salida generado, tiene formato Open Inventor para su posterior visualización. En el apartado 3.1.3 de la memoria se habla brevemente del uso de Open Inventor.

Parámetros de ejecución del MC

mc dataset iDim jDim kDim b_pixel u_1 u_2 [o_file] [a b c]

dataset	Volumen de datos procesado
iDim	Ancho de la imagen (número de píxeles)
jDim	Largo de la imagen (número de píxeles)
kDim	Alto de la imagen (número de imágenes)
b_pixel	Bits por píxel que contiene el dataset
u_1	Umbral 1 usado para procesar el dataset
u_2	Umbral 2 usado para procesar el dataset
[o_file]	Fichero que almacena la superficie
[a b c]	Reescalado de la superficie en x,y,z (1:1:1 por defecto)

Por ejemplo, para un volumen de datos que tiene 8 bits por píxel, y de tamaño 128 de ancho, 128 de alto y 64 capas, la entrada por la línea de comandos sería, "`mc vertebra.raw 128 128 64 8 0 20`", donde los valores de umbral que hemos introducido son 0 para el inferior y 20 para el umbral superior.

3.1.3 Uso de Open Inventor

Es una librería orientada a objetos para gráficos en 3D, basada en OpenGL y desarrollada por Silicon Graphics [1]. El formato de archivos de Inventor fue la base para la especificación del estándar VRML 1.0. ya que está bien documentado, contiene descripciones de objetos tridimensionales bastante completas, posee muchos modelos disponibles, convertidores fáciles de escribir,

modeladores y permite que la funcionalidad del sistema se vaya construyendo poco a poco.

Además de las características de visualización, Inventor incluye tanto soporte para el manejo de eventos, manipulación directa de gráficos en 3D, lectura y escritura de archivos, búsqueda por atributos, animación y representación de datos, como también un tipo de sistema jerárquico de tiempo de ejecución. En este proyecto Open Inventor se ha utilizado para escribir un archivo que permita representar la superficie generada por el algoritmo de MC.

Los nodos son "comandos" utilizados para definir formas y propiedades en Open Inventor. Están estructurados jerárquicamente, y en función de su tipo y secuencia, permiten configurar una escena. A continuación se describen los nodos usados para representar la superficie 3D en el algoritmo de MC.

Nodo Separator Nodo separador que delimita y contiene dentro del mismo los aspectos correspondientes a aquellos nodos de propiedad que aloja. Dicho de otra forma, los nodos de propiedad solamente pueden afectar a aquellos otros nodos que se alojan dentro del nodo Separador. El nodo separador es un nodo de grupo que aísla o separa los nodos ubicados dentro del separador del resto de los objetos en escena.

Nodo Transform Nodo de transformación que sirve para mover, dar escala y rotar objetos. El campo scaleFactor nos permite redimensionar objetos sin tener que recurrir, a variables algebraicas. Los factores menores que 1 reducen el tamaño original del objeto y los mayores lo incrementan.

Nodo ShapeHints Nodo de apariencia que proporciona información acerca de la forma que tiene la figura geométrica.

Nodo Material Nodo de apariencia que proporciona información acerca del material usado para representar la figura geométrica. Los campos en el nodo Material determinan la forma en que la luz se refleja sobre un objeto para crear color. El campo diffuseColor refleja todas las fuentes de luz dependiendo del

ángulo en que incide la luz sobre la superficie del objeto. Mientras más directa sea la incidencia de luz sobre la superficie del objeto, mayor será la reflexión de luz difusa. Los campos specularColor y shininess determinan los resaltes especulares. Cada uno de los campos del nodo Material posee en teoría tres columnas de números que se identifican, cada una de ellas, de izquierda a derecha con uno de los colores RGB.

Nodo IndexedFaceSet Nodo de forma que contiene la información necesaria para representar la figura geométrica, es decir contiene las posiciones de los polígonos representados en la malla.

3.2 Algoritmo de reescalado

Este algoritmo se ha integrado en la aplicación para permitir reescalar un volumen de datos. Para realizar el reescalado se ha aplicado una técnica de interpolación en todos las dimensiones del volumen (x, y, z) de datos. Está técnica nos permiten obtener un nuevo volumen de nuevas con la mínima perdida de información.

La principal causa de integración de este algoritmo en la aplicación, es poder generar mallas de menor tamaño a partir del algoritmo de MC Esto permitirá visualizar de manera correcta la superficie 3D en máquinas con pocos recursos hardware.

De entre las posibles técnicas de interpolación que hay, se ha decidido utilizar la técnica de Interpolación Cúbica. Está técnica obtiene resultados de muy buena calidad, aunque tiene el inconveniente de que tiene una alta complejidad debido a que hace uso de muchos puntos para su cálculo, en concreto 4 en 1D, 16 en 2D y 64 en 3D.

Antes de realizar una breve descripción del algoritmo, cabe señalar que únicamente puede ejecutarse en máquinas Linux. Esto se debe a que para realizar la interpolación en los ejes x e y se ha utilizado un comando de la librería

ImageMagick[7], en concreto el comando *convert* que permite a través del parámetro *–geometry* permite reescalar geométricamente el volumen en las dimensiones *x* e *y*.

3.2.1 Descripción

Para desarrollar este algoritmo, me he basado en las explicaciones aportadas por mi tutor, Jose Miguel Espadero, y las encontradas en la referencia [8] y teoría de la asignatura de cálculo que curse en primero de carrera.

La finalidad del algoritmo es obtener un volumen con dimensiones (x, y, z) nuevas a las originales. Para realizar la interpolación en la dimensión *z* del volumen, la cual indica el número de imágenes que se tiene del mismo, se realizan los siguientes pasos.

1. Se calcula un factor de reescalado para el eje *z*, $z = zOrigen / zDestino$.
2. Se calculan cuatro puntos *v0, v1, v2, v3*, donde cada uno se corresponde con una capa del volumen origen, a partir del factor de escalado y el punto *X*. La imagen 3.11 muestra una idea de la posición de los puntos *v* para ser hallados.

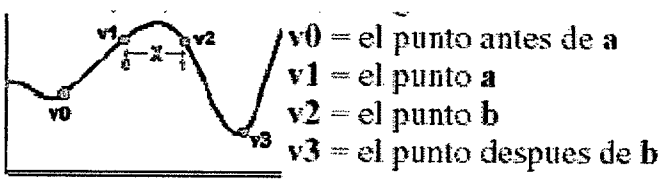


Imagen 3.11: Disposición de los puntos *v* y *X*.

3. Se halla el nivel de gris que hay en cada punto hallado.
4. Se calcula la distancia que hay en el eje *z* entre el nuevo punto destino *X* y los cuatro puntos del origen *v*.
5. Se realiza la interpolación cúbica entre los niveles de gris de los cuatro puntos *v* y la distancia al punto *X* (ver imagen 3.12).

```
function interpolacionCubica(v0, v1, v2, v3,x)
    P = (v3 - v2) - (v0 - v1)
    Q = (v0 - v1) - P
    R = v2 - v0
    S = v1

    return Px^3 + Qx^2 + Rx + S
end
```

Imagen 3.12: Ecuación usada para calcular la interpolación cúbica en 1D

3.2.2 Diseño e implementación

En el apartado B.2 se realiza una descripción de las funciones utilizadas en el algoritmo. Por lo tanto, en este punto se comentaran algunos aspectos a tener en cuenta en la ejecución del algoritmo para el correcto funcionamiento del mismo.

Datos de entrada que recibe el algoritmo El algoritmo procesa únicamente ficheros de 8 y 16 bits por píxel en cualquier tipo de formato. En caso de introducir un fichero que no tenga cabecera, es decir, con formato RAW, debe tener la extensión ".raw". Para los demás no importa la extensión. En ambos casos es necesario conocer el tamaño y bits por píxel. El fichero de salida generado tiene que contiene el volumen tiene formato RAW.

Parámetros de ejecución del algoritmo

interpolate dataset iDimO jDimO kDimO b_pixel iDimD jDimD kDimD o_file

dataset	Fichero de entrada que contiene el volumen
iDimO	Ancho de la imagen (número de píxeles)
jDimO	Largo de la imagen (número de píxeles)
kDimO	Alto de la imagen (número de imágenes)
b_pixel	Bits por pixel que contiene el volumen
iDimD	Ancho de la imagen destino
jDimD	Largo de la imagen destino
kDimD	Alto de la imagen destino
o_file	Fichero que contiene el volumen reescalado

Por ejemplo, para un volumen de datos que tiene 8 bits por píxel, y de tamaño 128 de ancho, 128 de alto y 64 capas, y que deseamos reescalar a un tamaño de 64x64x64 la entrada por la línea de comandos sería, "`interpolate vertebra.raw 128 128 64 8 64 64 64 vertebra_64_64_64.raw`".

3.3 Algoritmo de selección de una capa en un volumen

Este algoritmo se ha integrado en la aplicación para permitir visualizar una capa de un volumen. De esta manera se podrá visualizar un corte concreto del volumen de datos.

Este algoritmo se puede ejecutar tanto en entornos Linux como Windows. Para ello se ha utilizado el lenguaje de programación C estándar. En el apartado B.3 se realiza una descripción de las funciones utilizadas en el algoritmo. Por lo tanto, en este punto se comentaran algunos aspectos a tener en cuenta en la ejecución del algoritmo para el correcto funcionamiento del mismo.

Datos de entrada que recibe el algoritmo El algoritmo procesa únicamente ficheros de 8 y 16 bits por píxel con formato RAW. Es necesario conocer el tamaño y bits por píxel que contiene el volumen.

Parámetros de ejecución del algoritmo

choose dataset iDim jDim kDim b_pixel imagen o_file

dataset	Volumen de datos del que se obtiene la imagen
iDim	Ancho de la imagen (número de píxeles)
jDim	Largo de la imagen (número de píxeles)
kDim	Alto de la imagen (número de imágenes)
b_pixel	Bits por pixel que contiene el volumen
imagen	Número de capa del volumen a obtener
o_file	Fichero que contiene la capa seleccionada

Por ejemplo, para un volumen de datos que tiene 8 bits por píxel, y de tamaño 128 de ancho, 128 de alto y 64 capas, del que deseamos visualizar la copa 60 la entrada por la línea de comandos sería, "**choose vertebra.raw 128 128 64 8 60 vertebra_60.raw**".

3.4 Interfaz de la aplicación

La interfaz se ha desarrollado con el lenguaje de programación PHP. Este lenguaje permite que podamos ejecutar la aplicación desde cualquier máquina que disponga de un navegador web, por ejemplo, Mozilla para Linux y Explorer para Windows.

La aplicación PHP se ejecuta en una máquina Linux, por lo que habrá tener configurado un servidor Apache que permita el acceso a la misma desde cualquier máquina con acceso a la web. Para el correcto funcionamiento de la aplicación, se debe comprobar y modificar el fichero PHP.INI (normalmente se encuentra en /etc/php.ini en Linux). Este fichero tiene dos parámetros; file_uploads que determina si se pueden subir o no ficheros a través de HTTP a la máquina Linux; y upload_max_filesize que determina el tamaño máximo de un fichero. A continuación se muestran las líneas modificadas y los valores puestos para la aplicación.

```
.............
,,,,,,,,,,,,
File Uploads
.............
,,,,,,,,,,,,

; Wheter to allow HTTP file uploads
file_uploads = On

; Maximum allowed size for uploaded files
upload_max_file_size = 12M
```

Uno de los motivos de ejecutar la aplicación PHP en una máquina Linux, es poder utilizar tanto los comandos propios de Linux como los proporcionados por la librería ImageMagick [7]. Según vaya describiendo el funcionamiento de la aplicación se hará un comentario de los comandos utilizados.

Antes de empezar, se comentaran las decisiones que se han tomado para implementar la aplicación.

Almacenamiento de ficheros de la aplicación Para almacenar los ficheros que forman parte de la aplicación, se han creado dos carpetas, html_scripts y public_html. La primera contiene los ejecutables de los algoritmos implementados, y la segunda contiene los ficheros PHP asociados a los menús de cada aplicación.

Almacenamiento de imágenes Las imágenes que se utilicen en la aplicación, se almacenan en una carpeta, llamada tmpfiles, que está bajo el directorio public_html. El principal problema que se plantea, es poder almacenar las imágenes de un usuario y que no se confundan con las imágenes de otro. Para resolverlo, cada vez que se carga la pagina principal, se crea una carpeta con la IP asociada al usuario, y que colgara bajo el directorio tmpfiles, a su vez, y para diferenciar los imágenes que se puedan generar en cada menú de la aplicación, se crean las siguiente carpetas, que estarán bajo el directorio de la IP de cada usuario, imag (almacena el volumen cargado en formato RAW), view (almacena el volumen o una capa de un volumen en formato GIF), scale (almacena los volúmenes reescalados en formato RAW), hist (almacena el histograma del volumen cargado en formato GIF), mc (almacena las superficies 3D con formato IV), ident (almacena la imagen no RAW de la que se ha solicitado las características). El contenido de todas las carpetas pertenecientes a un usuario se borraran cada vez que se conecte a la aplicación, o cada vez que introduzca un nuevo volumen, de esta manera, sólo habrá información referente a un volumen de datos.

Nombre de los ficheros Para no tener que pedir continuamente las características de los volúmenes a un usuario, cada vez que su cargue una imagen y se obtenga el volumen en formato RAW, se le cambiara el nombre, añadiéndole las características de este, como son el ancho, alto, número de imágenes y bits por pixel. En el caso de los ficheros generados por el algoritmo Marching Cubes, se le añadirá información acerca del umbral usado.

Ejecución de comandos Cada vez que se ejecuta un algoritmo o comando de Linux, por pantalla se muestran dos líneas, una con el resultado de la acción y otra con el comando ejecutado. Opcionalmente, otras dos que muestran la salida estándar y la salida de errores en caso de que haya algo que mostrar. A continuación se describe el contenido de cada línea mostrada,

- Resultado, muestra el resultado de ejecutar una acción en el menú de la aplicación.
- Línea de comando, que contiene la línea con el comando o comandos ejecutados para este menú de la aplicación.
- Salida estándar, que contiene la salida producida por el comando ejecutado en caso de que produzca alguna.
- Salida de errores, que contiene, en caso de que los halla los errores producidos al ejecutar un comando, de esta manera, podemos determinar el posible fallo. El más común es introducir las características de un volumen de manera incorrecta.

Formularios de datos En los menús de la aplicación donde haya formularios para introducir datos, antes de proceder a la ejecución del comando con el contenido del formulario, se comprueba que se hayan introducido todos aquellos que sean necesarios, y además que estos sean coherentes con el comando ejecutado. En caso de no ser correcto, se mostrará un mensaje de error.

Comandos Linux Para el desarrollo de la aplicación se han ejecutado algunos comandos de Linux, algunos de ellos son, `mkdir`, `rm`, `cat`, `mv`, `identify` [7], `convert` [7]. El comando `identify`, permite obtener las características de una imagen en formato no RAW, y el comando `convert`, permite convertir ficheros de un formato a otro. Si se desea más información acerca de cada de uno ellos en [10].

En las siguientes subsecciones se hace una descripción de los menús de la interfaz de usuario implementada desde el punto de vista informático. En el apartado A se realiza una descripción del funcionamiento de la aplicación desde el punto de vista del usuario.

3.4.1 Cargar un volumen

A través de este menú, podemos cargar cualquier tipo de volumen de datos, para su posterior procesamiento en la aplicación.

Con el comando `convert`, convertiremos los volúmenes cargados a formato RAW, que será el tipo de fichero que procesan los algoritmos implementados. La sintaxis para convertir a formato RAW es la siguiente,"**`convert ficheroOrigen -depth bitsPixel GRAY:ficheroDestino.raw`**".

Para realizar la carga de los ficheros de datos, es necesario conocer las características de este, como son las dimensiones y el número de bits por píxel que tiene la imagen. Esta petición se realizará en todos los submenús de carga de volúmenes, a través de un formulario de datos con los campos necesarios para que el usuario los introduzca. Una vez se ha cargado el volumen se comprueba el tamaño del fichero en disco con las características dadas, en caso de no ser correctas se mostrará un mensaje de error indicándolo.

Dependiendo del tipo de fichero que se desee cargar, la aplicación realizara unas operaciones u otras. En las siguientes subsecciones se explica lo que se hace dependiendo del tipo de fichero que el usuario desee subir.

3.4.1.1 Volumen RAW

Este submenú permite subir un fichero que contiene un volumen de datos RAW.

Tras pulsar enviar, el volumen se guarda en la carpeta imag del usuario, realizándose a continuación, y aunque no sea necesario, la ejecución del comando `convert`, así nos aseguramos que las dimensiones introducidas son correctas. La sintaxis del comando ejecutado sería, "**`convert -size AnchoxAlto -depht bitsPixel GRAY:ficheroOrigen -depth bitsPixel GRAY:ficheroDestino`**". Es necesario especificar el tamaño y bits por pixel de la imagen RAW ya que estos ficheros no tienen cabecera que indique el tamaño.

3.4.1.2 Cargar imágenes RAW

Este submenú permite subir ficheros que forman un volumen de datos RAW.

Una vez pulsado el botón de seleccionar, se muestra otro formulario, a través del cuál se introducen las rutas de los ficheros que forman el volumen, con tantos campos como número de imágenes tenga el volumen

Tras pulsar enviar, las imágenes se guardan en la carpeta imag del usuario, y se concatenan todos los ficheros para formar un volumen Esto lo hacemos con el comando `cat`, cuya sintaxis sería "**`cat * > volumen.raw`**". A continuación, y aunque no sea necesario, se ejecuta el comando `convert`, así nos aseguramos que las dimensiones introducidas son correctas. La sintaxis del comando ejecutado sería, "**`convert -size AnchoxAlto -depht bitsPixel GRAY:ficheroOrigen -depth bitsPixel GRAY:ficheroDestino`**". Es necesario especificar el tamaño y bits por pixel de la imagen RAW ya que estos ficheros no tienen cabecera que indique el tamaño.

3.4.1.3 Volumen no RAW

Este submenú permite subir un fichero que contiene un volumen de datos no RAW.

La diferencia con respecto al funcionamiento del submenú "Cargar volumen RAW", es que este permite identificar las características de un volumen no RAW cuando el usuario no las conoce. Para ello, una vez el usuario ha elegido el volumen y ha pulsado sobre el botón Identify, se carga el volumen en la carpeta ident del usuario y se ejecuta el comando *`identify`*, cuya sintaxis sería "**`identify ficheroOrigen`**". En este caso la sintaxis del comando *`convert`* sería "**`convert ficheroOrigen -depth bitsPixel GRAY:ficheroDestino.raw`**".

3.4.1.4 Imágenes no RAW

Este submenú permite subir ficheros que forman un volumen de datos no RAW.

La diferencia con respecto al funcionamiento del submenú "Cargar imágenes RAW", es que este permite identificar las características de un fichero del volumen no RAW cuando el usuario no las conoce. Para ello, una vez el usuario ha elegido el volumen y ha pulsado sobre el botón Identify, se carga el volumen en la carpeta ident del usuario y se ejecuta el comando *identify*, cuya sintaxis sería "**identify ficheroOrigen**". En este caso la sintaxis del comando *convert* sería "**convert * ficheroOrigen -depth bitsPixel GRAY:ficheroDestino.raw**".

3.4.2 Visualizar volumen

Este menú, permite obtener una visión del volumen de datos o de una capa (slice) del mismo, de esta manera podemos tener una idea de la superficie 3D que obtendremos ejecutando el algoritmo de MC.

Para poder visualizar el volumen entero o una capa del mismo, es necesario que se haya cargado previamente un volumen en la aplicación, si no hubiese ninguno se mostrara un mensaje de error advirtiéndolo.

Los volúmenes de datos que serán procesados en este menú, se obtienen de las carpetas imag y scale, esta última contiene volúmenes reescalados a partir del original. Los nombres de los ficheros se listan en la pantalla con las dimensiones de cada uno, para que el usuario seleccione uno. Una vez se ha hecho clic sobre el enlace del fichero, se procesan las características de este para ejecutar el comando que se procese en cada submenú.

Los ficheros generados en este menú, que estarán en formato GIF, se almacenan en la carpeta view del usuario. El formato de imágenes GIF, además de permitir mostrar volúmenes, es bueno para mostrarlo en exploradores web. A

través del comando convert, cuya sentencia sería "**convert -size AnchoxAlto -depth bitsPixel GRAY:origen.raw GIF:destino.gif**", convertimos los volúmenes RAW en formato GIF,

Dependiendo de la visualización que se desee obtener del volumen, se elegirá un submenú u otro, que a continuación se describen.

3.4.2.1 Volumen entero

Este submenú, permite visualizar un volumen de datos RAW, mediante un fichero GIF animado en la pantalla de la aplicación.

Una vez el usuario ha seleccionado el volumen, de entre los ficheros disponibles, se convierte a formato GIF mediante el comando convert, cuya sintaxis sería "**convert -size AnchoxAlto -depth bitsPixel GRAY:ficheroOrigen.raw GIF:ficheroDestino.gif**". Las propiedades del fichero, como son el ancho, alto y bits por pixel, se obtienen del nombre del fichero, de esta manera el usuario no tiene que volver a introducir los datos del mismo. El fichero de salida generado, se visualiza en la pantalla del explorador.

3.4.2.2 Capa del volumen

Este submenú, permite visualizar una capa volumen de datos RAW, mediante un fichero GIF en la pantalla del explorador web.

Una vez el usuario ha seleccionado el volumen, de entre los ficheros disponibles, saldrá un formulario para que el usuario seleccione el número de capa a visualizar. Una vez ha pulsado enviar, se procesa el algoritmo de selección de capa choose, cuya sintaxis sería "**choose ficheroOrigen.raw Ancho Alto Capas bitsPixel capa ficheroDestino.raw**". Las características del fichero, como son el ancho, alto y bits por pixel, se obtienen del nombre del fichero que el usuario ha seleccionado. Por último, se convierte la imagen RAW generada a formato GIF, para su visualización el la pantalla de la aplicación, mediante el

comando convert, cuya sintaxis sería "**convert -size AnchoxAlto -depth bitsPixel GRAY:ficheroOrigen.raw GIF:ficheroDestino.gif**"

3.4.2.3 Histograma del volumen

Este menú permite visualizar un histograma del volumen de datos RAW en la pantalla de la aplicación.

Para poder realizar el histograma de un volumen, es necesario que se haya cargado previamente un volumen en la aplicación, si no hubiese ninguno se mostrara un mensaje de error advirtiéndolo.

Para realizar el histograma, siempre se cogerá el volumen de datos de la carpeta imag, que contiene el volumen original cargado por el usuario. Una vez se obtiene la imagen del histograma en formato GIF, se almacena en la carpeta hist. No se muestran todos los posibles volúmenes que haya también en la carpeta scale, ya que el histograma de la imagen será el mismo.

Para realizar el histograma, hemos usado el comando de Linux `pnmhistmap`, que devuelve un fichero de dimensiones 256x256 con los niveles de gris de la imagen. Este comando trabaja con ficheros PGM y PNM, por lo que previamente, debemos convertir el volumen de datos RAW a formato PGM, para ello usamos el comando convert, cuya sintaxis es "**convert -size AnchoxAlto -depth bitsPixel GRAY:ficheroOrigen.raw PGM:ficheroDestino.pgm**". Una vez tenemos el fichero PGM, ejecutamos el comando `pnmhistmap`, cuya sintaxis es "**pnmhistmap ficheroDestino.pgm > ficheroDestinoHistograma.pgm**", y cuya salida la hemos redirigido a otro fichero PGM que posteriormente convertiremos a formato GIF para su visualización en la pantalla de la aplicación. Mediante el operador '>' creamos un fichero con el resultado del histograma. La sintaxis del comando convert sería "**convert PGM: ficheroDestinoHistograma.pgm GIF:ficheroDestinoHistograma.gif**".

3.4.3 Reescalar volumen

Este menú permite reescalar el volumen de datos cargado por el usuario en sus tres dimensiones.

Para poder realizar el reescalado de un volumen, es necesario que se haya cargado previamente un volumen en la aplicación, si no hubiese ninguno se mostrara un mensaje de error advirtiéndolo.

Para realizar el reescalado, siempre se cogerá el volumen de datos de la carpeta imag, que contiene el volumen original cargado por el usuario. Una vez realizado el reescalado, el volumen generado se almacena en la carpeta scale. Hay que tener en cuenta que el usuario puede realizar tantos reescalados como desee, por lo que este directorio puede contener más de un fichero de datos RAW.

Las características del volumen se obtienen del nombre del volumen, por lo que el usuario sólo debe introducir las dimensiones de reescalado, es decir, los valores del ancho, alto y número de capa. Una introducidas, se ejecuta el algoritmo de reescalado, cuya sintaxis sería "**`interpolate ficheroOrigen.raw AnchoO AltoO CapasO bitsPixel AnchoD AltoD CapasD ficheroDestino.raw`**".

La ejecución no produce ningún resultado visual, si se desea ver, se deberá ir al menú "Visualizar volumen" de la aplicación.

3.4.5 Extraer superficie

Este menú permite ejecutar el algoritmo Marching Cubes para obtener una superficie 3D a partir del volumen. Se puede decir, que con este menú se cumple con los objetivos del proyecto, que es generar casos de prueba para un entrenador quirúrgico.

Para poder ejecutar el MC sobre un volumen , es necesario que se haya cargado previamente un volumen en la aplicación, si no hubiese ninguno se mostrara un mensaje de error advirtiéndolo.

Los volúmenes de datos que serán procesados en este menú, se obtienen de las carpetas imag y scale. Los nombres de los ficheros se listan en la pantalla con las dimensiones de cada uno, para que el usuario seleccione uno. Una vez se ha hecho clic sobre el enlace del fichero, se procesan las características de este y se muestra un formulario para que el usuario introduzca otros datos necesarios para la ejecución del mismo, como son los valores de umbral. Opcionalmente, también puede introducir tres valores, que permiten reescalar la superficie cuando se visualice con un visor de ficheros Open Inventor. Los ficheros que se obtengan como resultado en este menú, se almacenan en la carpeta mc.

Una vez se pulsa sobre el botón ejecutar, se ejecuta el MC, cuya sintaxis es "**`mc fichero.raw Alto Ancho Capas bitsPixel umbral1 umbral2 fichero.iv`**", o "**`mc fichero.raw Alto Ancho Capas bitsPixel umbral1 umbral2 ficheroSalida.iv a b c`**" en caso de introducir los parámetros opcionales.

3.4.6 Visualizar superficie 3D

Este menú permite descargar la superficie 3D generada a partir de MC para ser visualizada en el equipo del usuario.

Para poder descargar alguna superficie es necesario haber ejecutado previamente el MC sobre un volumen. En caso de no haber ningún fichero Inventor, se mostrará un mensaje de error.

Cuando se pulsa sobre este menú, aparece una lista con las superficies 3D generadas, para que el usuario se las descargue. Adicionalmente, se facilitan dos link, para que el usuario se descargue dos visores, uno para Windows y otro para Linux. Estos visores, se encuentran en formato ZIP en la carpeta visores.

3.4.7 Extras

Este menú permite acceder a algunos extras que aportan a la aplicación más funcionalidad.

3.4.7.1 Volúmenes de ejemplo

Este submenú permite descargarse volúmenes de datos para ser ejecutados en la aplicación. Estos volúmenes se almacenan en la carpeta extras que cuelga bajo el directorio public_html.

Los volúmenes facilitados, son una vértebra con dimensiones 128x128x128 y 8 bits por pixel y un humeral con dimensiones 128x128x128 y 16 bits por pixel.

3.4.7.2 Memoria PDF

Este submenú permite visualizar la memoria del proyecto en formato PDF, la cuál está almacenada en la carpeta extras del directorio public_html.

3.4.7.3 Limpiar ficheros temporales

Este submenú permite limpiar la carpeta tmpfiles, donde se almacenan las imágenes utilizadas en la aplicación.

A través de esta opción evitamos tener gran cantidad de imágenes, que están ocupando espacio en el servidor, y evitamos tenerlo que hacerlo manualmente en el servidor Linux. Se pueden realizar dos tipos de borrado; el primero, permite borrar todas los ficheros que cuelgan bajo el directorio del usuario, y la segunda, que permite borrar todo los ficheros de todos los usuarios que cuelgan bajo el directorio tmpfiles.

4. Conclusiones

Considero que los objetivos iniciales, que se propusieron al iniciarse el proyecto se han cumplido en su totalidad.

4.1 Resultados obtenidos

La aplicación obtenida permite, gracias a su fácil manejo, reconstruir volúmenes en superficies 3D.

Para controlar el correcto desarrollo del proyecto, se realizó una planificación de casos de prueba y entrega de resultados siguiendo la metodología de programación extrema [6]. La planificación incluye la división en tareas, que incluye la generación de casos de prueba. El volumen utilizado para la generación de los casos de prueba han sido una vértebra de dimensiones 128x128x128 con 8 bits por pixel. A continuación se hace una descripción del desarrollo de cada tarea.

4.1.1 Primera tarea: Algoritmo de Marching Cubes

Esta primera tarea consistió en la implementación del algoritmo de MC y la generación de un caso de prueba para comprobar el correcto funcionamiento. Como caso de prueba, tenemos el volumen de una vértebra, que mediante la ejecución en Linux del siguiente comando "`mc vertebra.raw 128 128 128 8 0 45`", obtenemos una malla de 36276 triángulos.

Las siguientes imágenes muestran la superficie 3D vista desde diferentes perspectivas.

Imagen 4.1: Vértebra – Figura 1

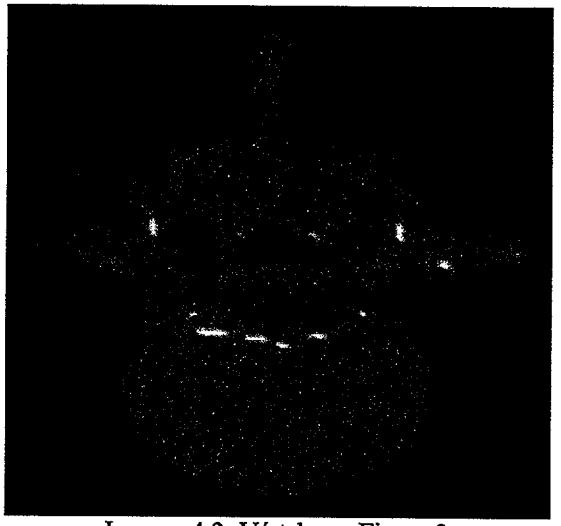


Imagen 4.2: Vértebra – Figura 2

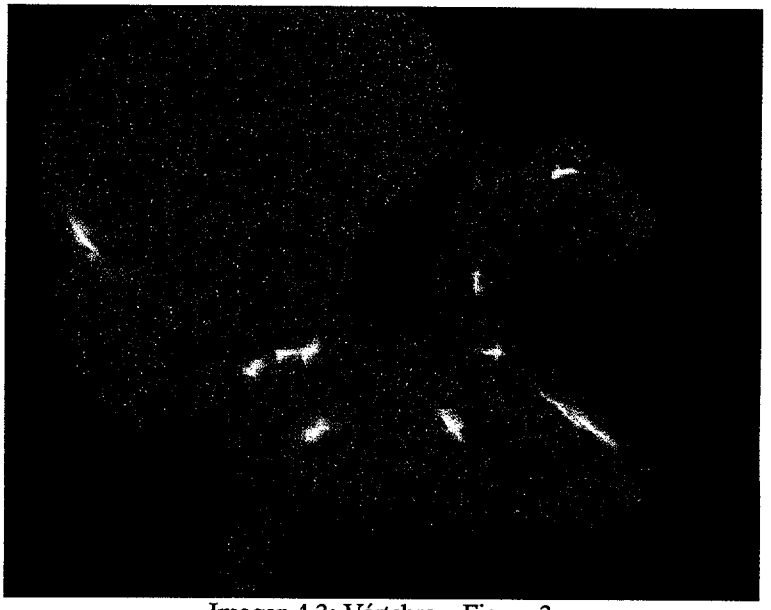


Imagen 4.3: Vértebra – Figura 3

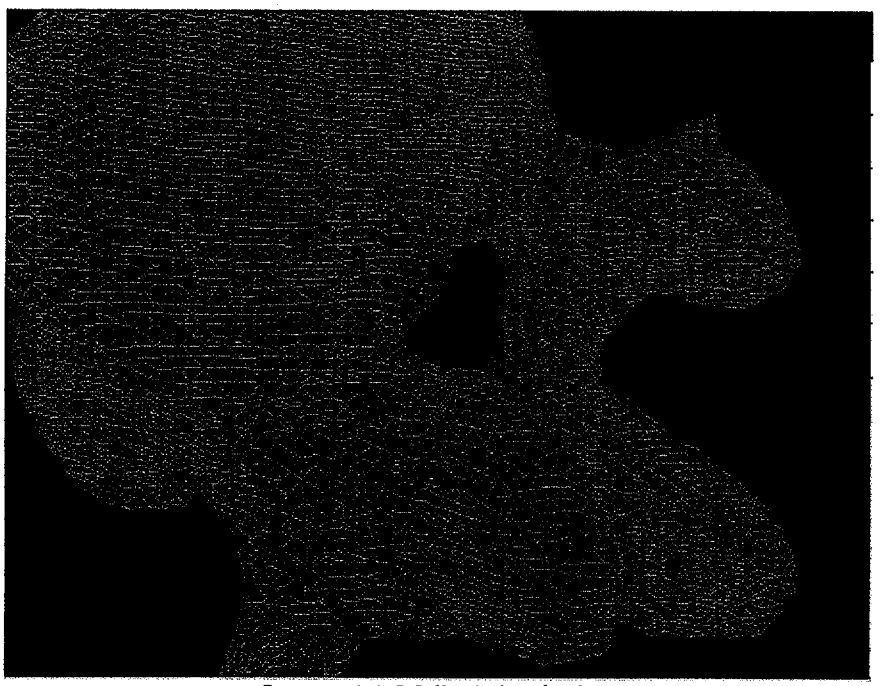


Imagen 4.4: Malla de la vértebra

4.1.2 Segunda tarea: Algoritmo de reescalado

Esta segunda tarea consistió en la implementación del algoritmo de reescalado y la generación de un caso de prueba para comprobar el correcto funcionamiento. Como caso de prueba, tenemos el volumen de una vértebra, que mediante la ejecución en Linux del siguiente comando "`interpolate vertebra.raw 128 128 128 8 64 64 64 vertebraReescalda.raw`", un volumen de dimensiones 64x64x64. Aplicando a continuación el algoritmo de selección de capa, obtenemos la siguiente imagen.

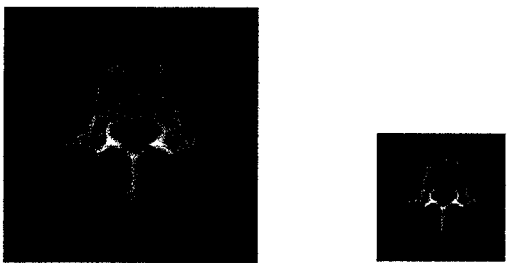


Imagen 4.5: Vértebra original a la izq (capa 64). y reescalada a la dcha. (capa 32)

4.1.3 Tercera tarea: Algoritmo de selección de una capa

Esta tercera tarea consistió en la implementación del algoritmo de selección de capa y la generación de un caso de prueba para comprobar el correcto funcionamiento. Como caso de prueba, tenemos el volumen de una vértebra de dimensiones 128x128x128 y 8 bits por pixel, que mediante la ejecución en Linux del siguiente comando "`choose vertebra.raw 128 128 128 8 50 vertebraCapa.raw`", obtenemos la capa 50 del volumen.

Imagen 4.6: Corte de la vértebra (capa 50)

4.1.4 Cuarta tarea: Aplicación PHP

La cuarta tarea consistió en el desarrollo de una aplicación PHP que integrase, además de las funcionalidades anteriores, la carga e histograma de un volumen, y la visualización de las superficies 3D generadas, facilitando los ficheros para su visualización. En el apéndice A, se muestra un caso de prueba mostrando el funcionamiento, de la aplicación.

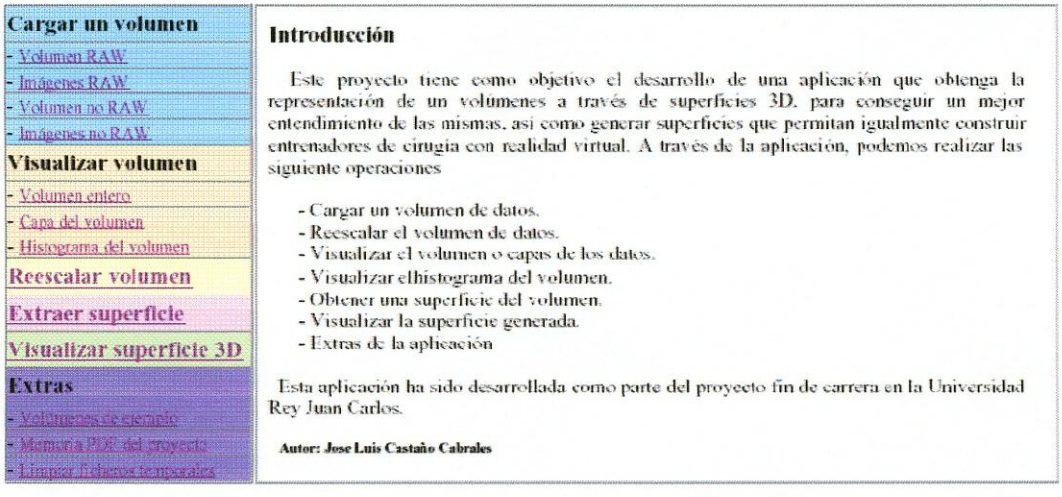


Imagen 4.7: Pantalla inicial de la aplicación

4.5 Quinta tarea: Documentación

La memoria se ha generado con Microsoft Word 2000, aunque en ocasiones he utilizado la herramienta Adobe Acrobat 5.0. Para la documentación del código he utilizado la herramienta Doxygen de Linux, que crea varios documentos en HTML que permiten una fácil navegación a través del mismo.

4.2 Conocimientos adquiridos

Al comienzo el proyecto, mis conocimientos en el campo de la reconstrucción de volúmenes y tratamiento de imágenes eran escasos. A través de la realización del mismo, no sólo he aprendido a saber reconstruir volúmenes en superficies 3D, sino que también ha despertado mi curiosidad en el procesamiento de imágenes médicas y su utilidad en el campo clínico.

Con el desarrollo del proyecto, en cuanto a lenguajes de programación, me ha servido, tanto para ampliar mi conocimiento sobre C estándar, como para aprender Open Inventor y PHP.

4.2 Posibles trabajos futuros

En cuanto al algoritmo de MC, se podría resolver el problema de los agujeros debido a caras ambiguas, ampliando el número de casos de básicos de 15 a 30.

En cuanto a la aplicación, se podría hacer que los ficheros PHP fuesen servidos desde un servidor con plataforma Windows, para ello sería necesario implementar algunos algoritmos que funcionasen igual que los comando Linux utilizados en la aplicación, como por ejemplo el convert o el identify.

Se podría añadir a la aplicación algoritmos de segmentación y tratamiento de imágenes volumétricas que permitiesen obtener mejores resultados en la extracción de una superficie a partir de un volumen.

A. Manual de usuario

A continuación se incluye un pequeño manual de usuario, con el fin de facilitar un rápido aprendizaje y manejo de la aplicación. Se irán mostrando las diferentes pantallas obtenidas en la ejecución de cada menú utilizando como caso de prueba el volumen de la vértebra con dimensiones 128x128x128 y 8 bits por píxel. Para cada ejecución de alguna opción del menú se muestra el resultado, si es visual, o un mensaje indicando si se ha realizado correctamente o no.

A.1 Carga del volumen de datos

Los datos introducidos en la aplicación pueden estar en cualquier formato, en caso de ser formato RAW se debe conocer las dimensiones y los bits por pixel de los ficheros. Hay cuatro menús diferentes que facilitan la carga de datos para su posterior procesamiento. A continuación se muestra la pantalla que muestra como cargar el volumen de la vertebra.

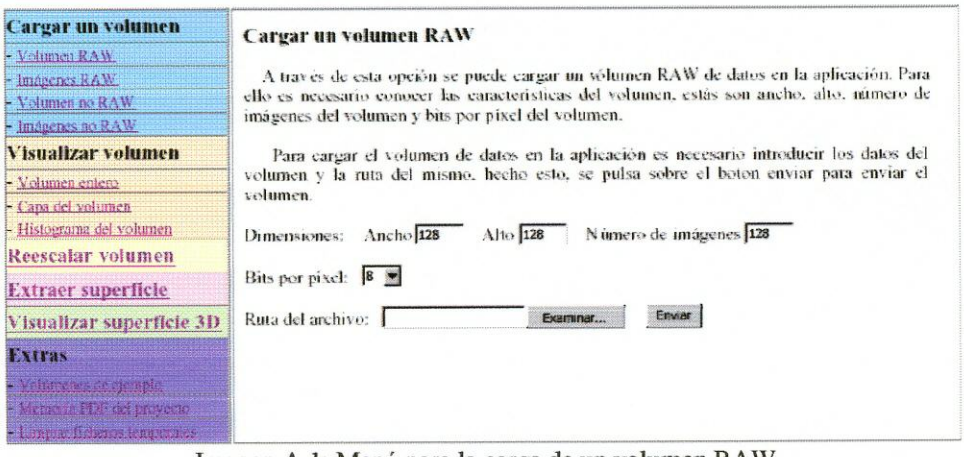


Imagen A.1: Menú para la carga de un volumen RAW

Una vez se ha pulsado enviar, aparece una pantalla indicando el resultado de la carga del volumen en la aplicación.

A.2 Visualización del volumen de datos

A través de esta opción se visualizar el volumen de datos, una capa del mismo o ver el histograma del volumen. En caso de seleccionar una capa, el usuario debe introducir el número de capa que desea visualizar. Para poder utilizar es necesario que se haya cargado un volumen. Las siguientes imágenes muestran el resultado de ejecutar cada uno de los menús.

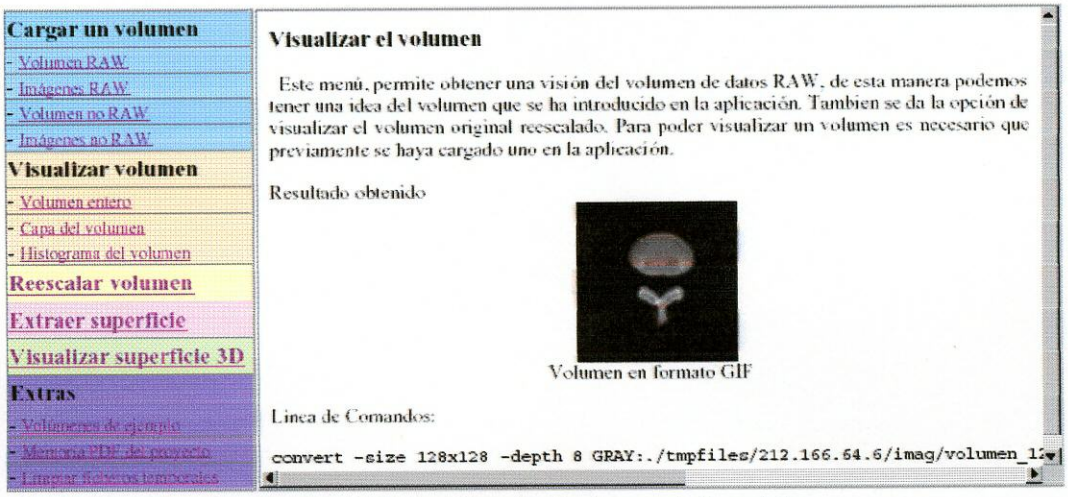


Imagen A.2: Visualizando el volumen de la vertebra

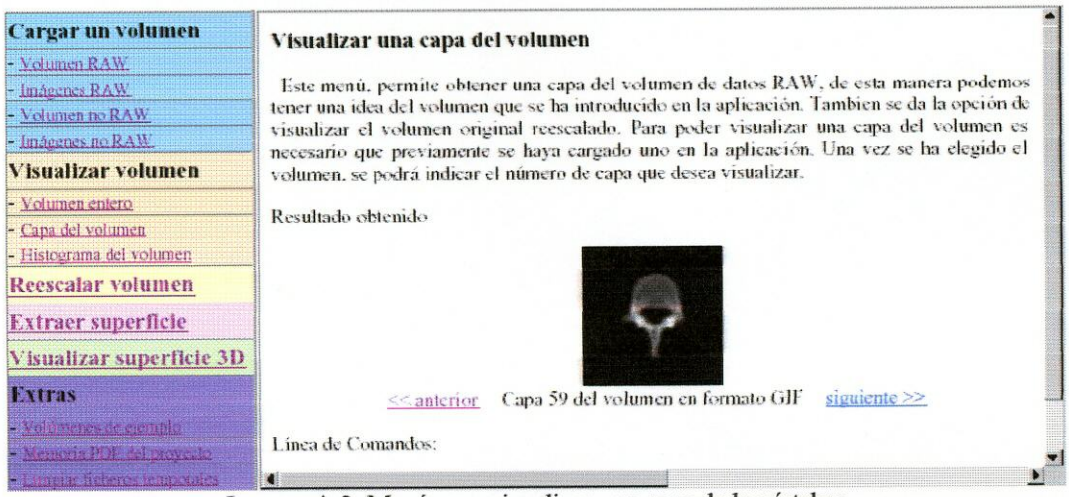


Imagen A.3: Menú para visualizar una capa de la vértebra

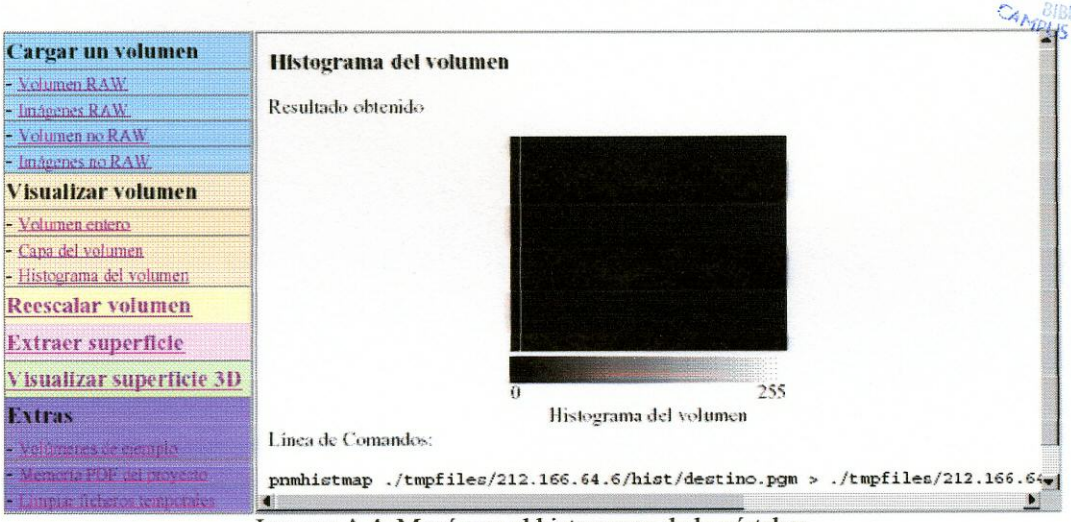


Imagen A.4: Menú con el histograma de la vértebra

A.3 Reescalado del volumen de datos

Con esta opción el usuario puede reescalar el volumen de datos de RAW. Se recomienda su uso cuando el volumen tiene dimensiones muy grandes y la máquina donde posteriormente se visualizara el la superficie 3D no tiene muchos recursos hardware. El usuario deberá introducir las dimensiones de destino que desee para realizar el reescalado del volumen. Para poder utilizar es necesario que se haya cargado un volumen.

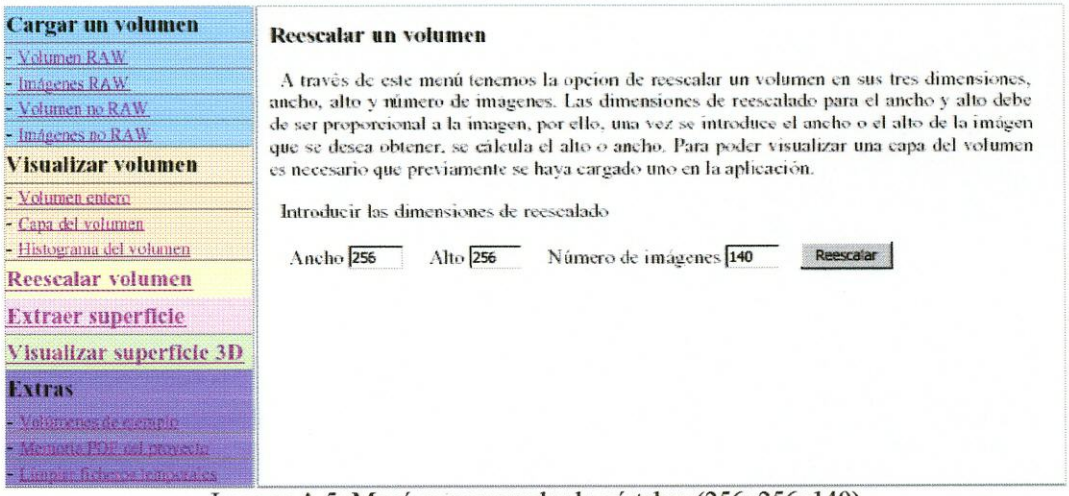


Imagen A.5: Menú para reescalar la vértebra (256x256x140)

A.4 Extraer superficie 3D

Permite ejecutar el algoritmo de representación de volúmenes en superficies 3D. El usuario debe introducir los valores de umbral obligatoriamente, y opcionalmente puede introducir el nombre del fichero de salida y tres valores que permiten reescalar la superficie 3D en los ejes x, y, z.

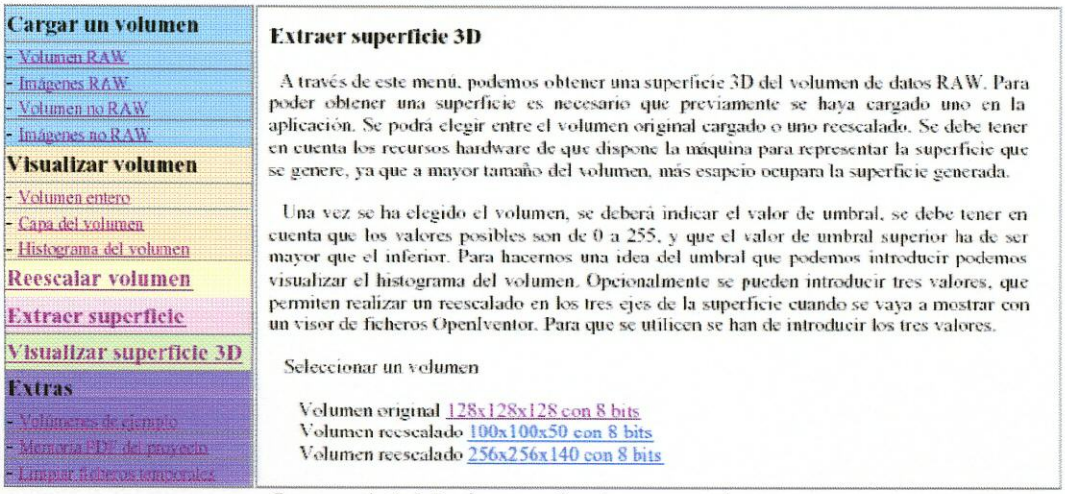


Imagen A.6: Menú para seleccionar un volumen

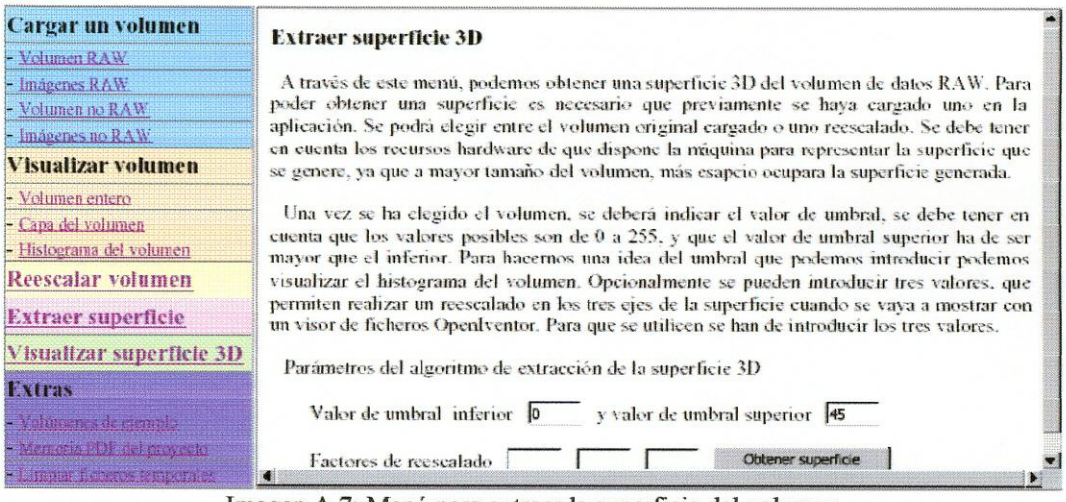


Imagen A.7: Menú para extraer la superficie del volumen

A.5 Visualización de la superficie generada

Esta menú permite descargar al usuario la superficie para su visualización, y en caso de que no tenga visores, dos enlaces para descargar visores para Linux y Windows.

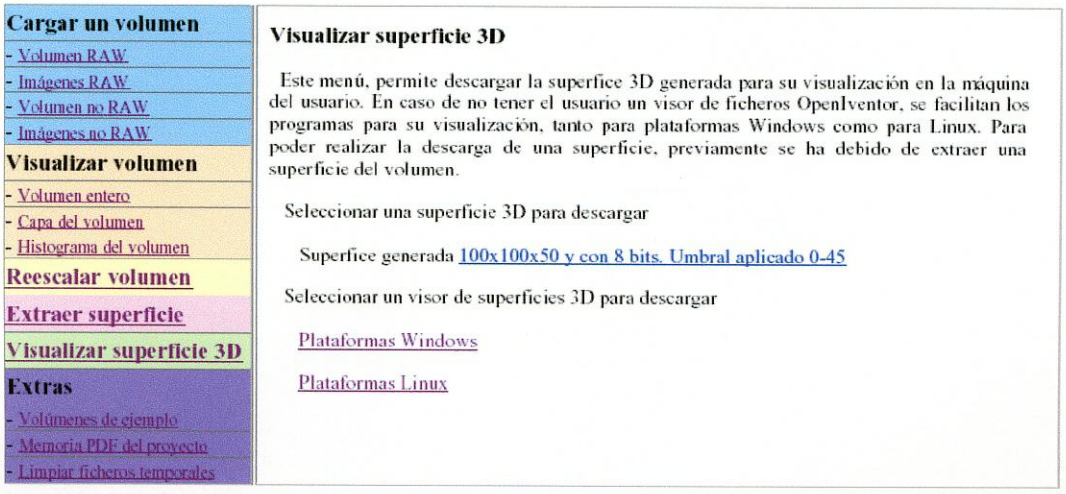


Imagen A.8: Menú para descargar la superficie 3D y los visores

Imagen A.9: Vista de la vértebra con el visor de ficheros Open Inventor

A.7 Extras

En este menú se pueden descargar dos volúmenes de ejemplo, una vértebra y un humeral, la memoria del proyecto en formato PDF, y se permite limpiar los archivos temporales para no llenar el disco de espacio.

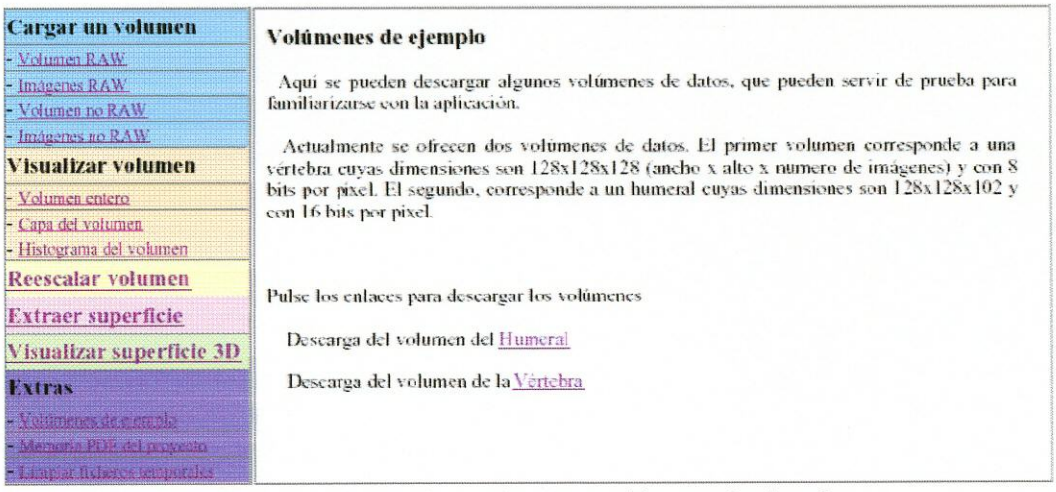


Imagen A.10: Menú para descargar volúmenes de ejemplo

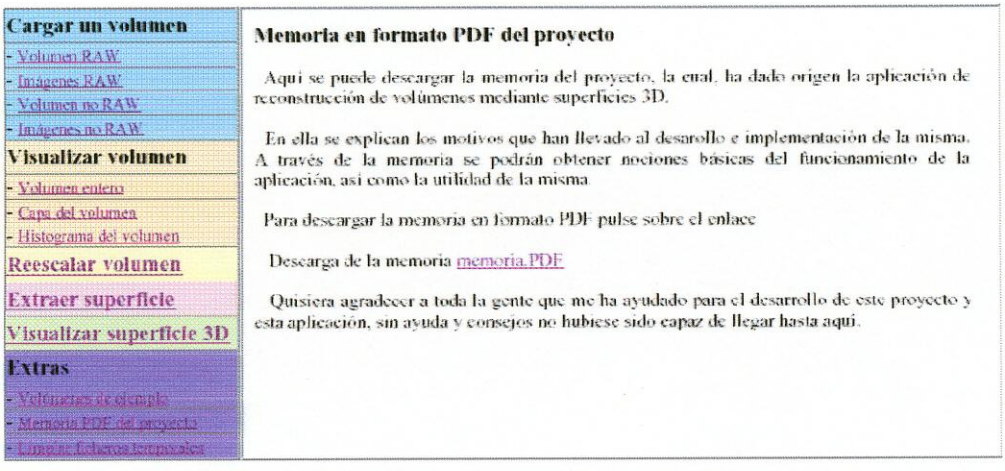


Imagen A.11: Menú para descargar la memoria del proyecto

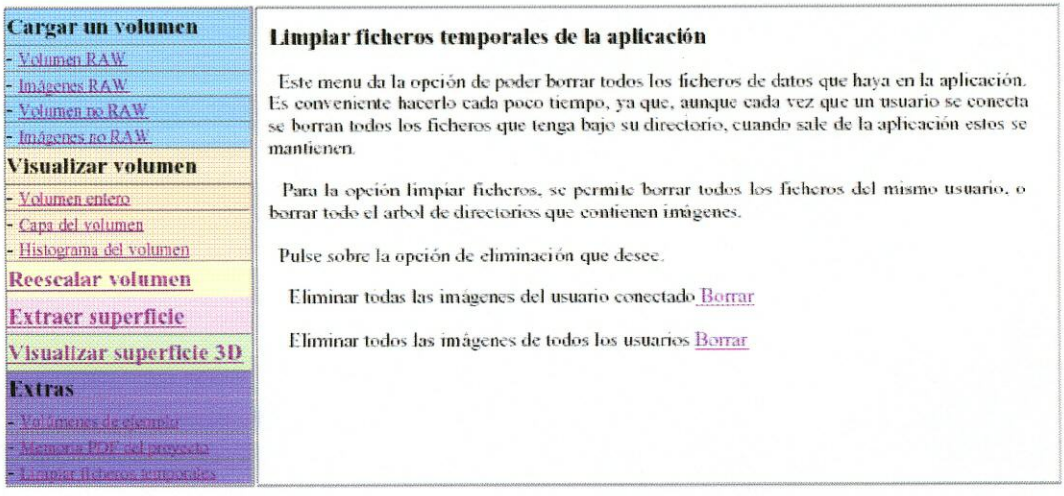


Imagen A.11: Menú para limpiar los ficheros temporales

B. Documentación del código

A continuación se hace una breve descripción de las funciones utilizadas en los algoritmos implementados.

B.1 Descripción de las funciones del algoritmo de Marching Cubes

- **void march**

Marca el volumen para generar la superficie. Parámetros usados,

  - dataset Datos del volumen en memoria
  - capa Capa del volumen que se esta procesando
  - k Numero de capa que se esta procesando
  - umbral_1 Valor de umbral inferior aplicado
  - umbral_2 Valor de umbral superior aplicado
  - iso Isosuperficie generada en la capa k

- **long makeVertex**

Crea los vértices que interseccionan en una arista. Parámetros usados,

  - cualArista Arista donde se calcula el vértice
  - i, j, k Posición del pixel procesado
  - umbral_1 Umbral inferior aplicado
  - umbral_2 Umbral superior aplicado
  - dataset Datos del volumen en memoria
  - capaIso Isosuperficie generada para la capa k

- **void writeInventorHeader**

Escribe la cabecera del fichero Open Inventor. Parámetros usados,

  - f Puntero al fichero generado
  - argc Numero de argumentos introducidos
  - argv Valor de los argumentos introducidos

- **void writeVertices**
- **void writeNormals**
- **void writeTriangles**

Escribe en el fichero los vértices, normales y triángulos de la superficie. Parámetros usados,

  - iso Isosuperfice con la superficie generada
  - f Fichero donde se escribe

- **FILE* createFile**

Crea el fichero donde se almacena la superficie. Parámetros usados,

  - argc Numero de argumentos introducidos
  - argv Valor de los argumentos introducidos

- **datum* readDataset**

Carga en memoria el fichero de datos. Parámetros usados,

  - argv Valor de los argumentos introducidos
  - bits Numero de bits por pixel de la imagen

B.2 Descripción de las funciones del algoritmo de reescalado

- **char* buildCommand**

Construye el comando que ejecuta el reescalado en x e y a través del comando convert de Linux. Parámetros usados,

  - argv Valor de los argumentos introducidos

- **float cubicInterpolation**

Realiza la interpolación cúbica en el eje z. Parámetros usados,

  - v0, v1, v2, v3 Niveles de gris en cuatro puntos
  - z Distancia en el eje z

- **datum* interpolateDataset**

Interpola el volumen de datos en el eje z. Parámetros usados,

  - dataset Puntero al volumen de datos en memoria.

- **void createFile**

  Crea un fichero con el volumen reescalado. Parámetros usados,

  - argv Valor de los argumentos introducidos
  - bits Numero de bits por pixel en la imagen
  - dataset Puntero al volumen

- **datum* readDataset**

  Carga en memoria el fichero de datos. Parámetros usados,

  - argv Valor de los argumentos introducidos
  - bits Numero de bits por pixel que tiene la imagen

B.3 Descripción de las funciones del algoritmo selección de capa

- **void chooseLayer**

  Crea una capa del volumen. Parámetros usados,

  - argv Valor de los argumentos introducidos
  - n_imagen Numero de imagen elegida del volumen
  - bits Numero de bits por pixel que tiene la imagen

C. Relación de ficheros

Para Linux se adjunta un pequeño script que compila los fuentes escribiendo simplemente `make` en la línea de comandos. Para cada algoritmo hay uno, llamado Makefile, y debe estar en el mismo directorio que los ficheros fuentes.

Ficheros relacionados con los algoritmos implementados					
Algoritmo	**Definición**	**Implementación**	**Windows**	**Linux**	**Compilación**
MC	mc.h	mc.c	mc.exe	mc	Makefile
Reescalado	interpolate.h	interpolate.c	-	interpolate	Makefile
Seleccion	choose.h	choose.c	choose.exe	choose	Makefile

Ficheros relacionados con la aplicación PHP	
Menú de la aplicación	**Fichero PHP asociado al menú**
Inicial	index.php, main.php
Cargar volumen	volumen.php, imagenes.php, uploadFiles.php
Visualizar volumen	visualizar.php, capa.php, histograma.php
Reescalar volumen	rescale.php
Extraer superficie	mc.php
Visualizar superficie 3D	visualizar3d.php
Extras	Extras.php
Otros ficheros PHP	**Descripción**
global.php	Contiene las funciones usadas en cada menú
imagen.php	Permite visualizar imágenes GIF
download.php	Obliga a realizar la descarga de ficheros

Bibliografía

[1] Silicon Graphics http://www.sgi.com

[2] TeraRecon http://www.rtviz.com

[3] "I/O Conscious Volume Rendering", Chuan-kai Yang and Tzi-cker Chiueh, VisSym '01, Joint Eurographics - IEEE TCVG Symposium on Visualization, Ascona, Switzerland, May, 2001.

[4]"Delaunay Surface Reconstruction from Scattered Points", A. Rodríguez, J. Espadero, D. López, L. Pastor, DGCI 2000, 13 - 15 Diciembre 2000, Uppsala. Páginas 272 – 283.

[5] W.E Lorensen and A.V. Cline. Marching cubes: a high resolution 3d surface construction algorithm. ACM Computer Graphics (Siggraph Conference Proceedings), 21(4):163-196, 1987.

[6] Extreme Programming http://www.extremeprogramming.org/ y http://www.xprogramming.com/

[7] ImageMagick.org http://www.imagemagick.com

[8] "A note on Cubic Interpolation", Erik Meijering and Michael Unser, IEEE TRANSACTIONS ON IMAGE PROCESING, VOL. 12, NO. 4, APRIL 2003

[9] Coin3D project. A free implementation of OpenInventor. http://www.coin3d.org

[10] GNU http://www.gnu.org/

[11] Manual de referencia de C, Tercera Edición, Herbert Schildt