Universidad de las Ciencias Informáticas

Facultad 5

Reconstrucción Tridimensional de Superficies de Modelos Anatómicos.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Osniel Junco Arias
David Delgado Ballester

Tutor: Ing. Osvaldo Pereira Barzaga

Ciudad de La Habana

Marzo, 2010

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Firma del Autor: Osniel Junco Arias. Firma del Autor: David Delgado Ballester.

_____

Firma del Tutor: Ing. Osvaldo Pereira Barzaga

Datos de Contacto

Tutor: Ing. Osvaldo Pereira Barzaga.

Edad: 25 años.

Ciudadanía: cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría Docente: Profesor Instructor.

E-mail: opereira@uci.cu

Graduado de la UCI, con seis años de experiencia en el tema de la Gráfica Computacional y líder de un proyecto de Visualización Médica en la Universidad de las Ciencias Informáticas.

Dedicatoria

A mi querida y adorada madre

Esa persona que me trajo al mundo y que siempre ha estado a mi lado y que me ha apoyado en todo momento, que ha luchado tanto por mi futuro y que daría la vida, si fuera necesario, por verme triunfar. Por ser siempre comprensible y por haber sacrificado muchas cosas para que hoy yo fuera la persona que soy.

A mi querido y adorado padre

Por siempre estar a mi lado y ser un gran ejemplo de padre para mí, por enseñarme desde muy pequeño muchas cosas buenas que han hecho posible mi formación y por estar siempre pendiente de mis cosas y que, igual que mi mamá, se que estaría dispuesto a dar lo que no tiene por verme feliz.

A mis dos hermanas

Esas dos bellas personas de las cuales siempre he recibido un gran apoyo y mucho amor. Que siempre han estado muy cerca de mí para guiarme y ayudarme en las cosas más importantes de mi vida. Por haber sido un gran ejemplo para mí.

Osniel

A la memoria de Lucas y Armando:

Por todo el cariño que me brindaron y porque aunque no están físicamente siempre fueron una razón para seguir adelante.

A mis padres:

Por inculcarme siempre la necesidad del estudio, por ser mi todo en la vida y por estar siempre dispuestos a darlo todo por mí.

A mi hermanito:

Por brindarme su ternura e incondicionalidad, por ser ese ser tan maravilloso que es.

David

Agradecimientos

Por haber llegado hasta aquí y ser lo que hoy soy, quisiera agradecer a varias personas que han influido en mi desarrollo y formación como profesional.

En especial a mis padres

Por haber sido siempre mis mayores apoyos y ejemplos, porque siempre supieron inculcarme los buenos sentimientos y los mejores valores humanos con mucho amor y mucha dedicación. Porque han estado siempre muy unidos y se han sacrificado y luchado mucho para que yo estudie y me convirtiera en una persona de bien. Les agradezco mucho porque siempre he estado en sus corazones y por llenar el mío de amor.

A mis hermanas

Porque siempre nos hemos querido y he recibido de ellas mucho cariño, comprensión y ayuda sin esperar nada a cambio. Porque gracias a sus ejemplos y consejos he logrado ser lo que hoy soy.

A mi querida novia Ismelys

Por haber estado a mi lado en estos últimos años, muy importantes de mi vida, brindándome mucho amor, cariño y dedicación. Por ser la persona a quien le he entregado mi corazón y ha sabido cuidarlo como si fuera lo más importante de su vida. Por estar presente en los momentos más difíciles dándome el mismo amor y cariño que me dieran mis padres si estarían junto a mí. Porque ha sido, es y seguirá siendo muy especial para mí.

A mis sobrinitas hermosas

Porque desde que llegaron al mundo me han dado mucha felicidad y porque las quiero mucho.

A mis tíos

Que en estos últimos años en los que he estado alejado de mis padres han sabido comportarse como lo harían ellos y me han acogido con mucho cariño y amor. Por los consejos y el apoyo que siempre he recibido de ellos.

A mis suegros

Porque desde que comparto mi amor con la persona más linda de este mundo, su hija, me han acogido como un hijo y me han brindado mucha confianza y cariño.

A todos mis maestros y profesores que desde el primer grado, de una forma o de otra, han sido fuertes colaboradores de mi formación como profesional.

A mi compañero de tesis y de lucha, David, que siempre hemos compartido una gran amistad y un gran apoyo.

A mis grandes amigos Juan Carlos, Javier (el Jávico), Talancón (el mitra), Leandro por haber compartido muchos años de amistad y por ser ejemplos de buenos amigos.

A mi tutor y amigo Osvaldo que siempre ha sido para mí un ejemplo de abnegación y de buen trabajador, por el apoyo y por todas las cosas que he aprendido de él.

Y no por último, deja de ser menos importante, quiero agradecer a nuestro comandante en jefe Fidel castro Ruz porque si no fuera por él, hoy no me estaría convirtiendo en un profesional eternamente comprometido con la revolución cubana.

En fin, quiero agradecer a todas las personas que de una forma u otra me han apoyado y han hecho posible mi formación, tanto en la vida cotidiana como en la profesional. A las que menciono aquí y a las demás personas quiero darles las gracias y eternamente agradecido les estaré.

Osniel

A mi madre:

Por ser mi más fiel protectora, por reír cuando yo reía y calmar mi sufrimiento cuando la tristeza estaba presente en mí. Por indicarme siempre el camino del bien, por ser la mejor madre del mundo.

A mi padre:

Por ser siempre mi ejemplo a seguir, por ser la persona que nunca ha dejado de encontrar palabras exactas para hacerme entrar en razón, por enseñarme con su ejemplo valores como la modestia y la sencillez e inculcarme que toda la gloria del mundo cabe en un granito de maíz.

A mi hermano:

Por ser esa personita que contagia a los demás de esa dulzura e ingenuidad que tanto lo caracteriza, por estar a mi lado todos estos años de mi vida, por hacer que me supere cada día más para darle el ejemplo que siempre ha buscado en mí.

A mis abuelas:

Por esa dulzura que siempre me han brindado, por malcriarme tanto y por ser más que abuelas, madres.

A mis abuelos:

Por guiarme en el buen camino, por incitarme siempre a estudiar y a superarme, por estar ahí siempre para mí.

A mis tías y tíos:

Por quererme cada uno a su manera pero de forma sincera, por transmitirme sus experiencias y acogerme como un hijo más.

***A Mile**:*

Por estar a mi lado en este año tan duro sin importar la dificultad del momento dándome todo su amor y cariño, por soportar con paciencia todas mis malacrianzas.

A mi familia de Ciego:

Por quererme como un miembro más de la familia y contagiarme de esa armonía, alegría y amor que siempre tienen.

A mis amigos Dariel, Yadira y Ubalquis:

Por compartir mis alegrías y tristezas, por estar siempre ahí cuando más lo necesité.

A Osniel:

Por ser mi compañero de batallas, por contar con su amistad incondicional, por haberme sacado de tantos apuros.

A Mayra, profesores y personal no docente:

A Mayra por guiarnos al frente de la facultad de forma magistral y darme la oportunidad de conocerla un poco mejor y darme cuenta del gran corazón que tiene y a los profes y demás personal porque de todos aprendí muchísimo.

A Fidel:

Por crear la Universidad de las Ciencias Informáticas que me ha dado la posibilidad de ser ingeniero y crecer intelectualmente, y por ser el líder de una Revolución que ha traído tanto bien.

Quisiera agradecer a todas aquellas personas que de una forma u otra han tenido que ver con mi desarrollo como persona; a mi grupo de La Lenin, a mis compañeros del 5108, a mis compañeros de aula, a todas las amistades que hice en esta universidad, a la brigada médica en La Grita, Venezuela; sepan que lo que soy hoy como ser humano se debe en gran medida al aporte que hizo cada uno de ustedes en mi vida.

David

Resumen

La reconstrucción de imágenes médicas se ha convertido actualmente en un campo muy importante y de amplias investigaciones en la medicina. Actualmente existen muchas técnicas de reconstrucción que hacen posible una mejor visualización de las estructuras anatómicas de cada paciente.

En este trabajo se hace referencia a la reconstrucción de superficies, y se caracterizan algunos algoritmos que sirven para representar las superficies del modelo de datos a visualizar, los cuales generan una malla triangular en tres dimensiones (3D) de dicho objeto.

Se propone el Marching Cubes como algoritmo de reconstrucción de superficies basándose en ventajas como la fácil implementación, buena calidad en la malla generada y un bajo coste computacional.

Además, se describen dos modificaciones a la implementación original del algoritmo: la primera es una propuesta de estructura de datos denominada MCTable que elimina el problema de generar vértices repetidos durante el proceso de reconstrucción de la superficie; y la segunda es una variante de implementación del algoritmo en la Unidad de Procesamiento Gráfico (GPU), con el objetivo de disminuir los tiempos de respuesta de dicho algoritmo.

Palabras Claves

Reconstrucción, Tridimensional, Vóxel, Estructuras Anatómicas, Malla, Modelo, Imágenes Médicas, DICOM, 3D, Realidad Virtual.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Con el surgimiento en el año 1895 de la imagenología como ciencia, la calidad de vida de las personas ha mejorado en gran medida; ya que los médicos pueden realizar diagnósticos de determinadas patologías presentes en las estructuras anatómicas de una persona de forma no invasiva y muy precisa.

Desde entonces se han desarrollado diferentes técnicas y equipos para la adquisición de imágenes médicas digitales, tales como: Tomografía Axial Computarizada (TAC) y la Resonancia Magnética (RM). Estas técnicas modernas de adquisición generan grandes volúmenes de imágenes que representan en dos dimensiones una sección de un fino corte del cuerpo del paciente.

A los médicos en ocasiones le es difícil identificar dentro de tanto volumen de información las regiones anatómicas, por lo que se hace necesario realzar las estructuras que le son de interés y representar las relaciones estructurales de las mismas en un entorno 3D de forma que puedan visualizarlas e interactuar con ellas.

Muchas han sido las propuestas de soluciones a la problemática anterior, pero sin dudas, la más novedosa y difundida ha sido la de aplicar técnicas de visualización y realidad virtual. Las aplicaciones informáticas que integran los conocimientos médicos con técnicas de realidad virtual se conocen en el ámbito científico como software de visualización médica.

En nuestro país una de las esferas sociales priorizadas es la salud pública, es por ello que se llevan a cabo muchos proyectos con el fin de aumentar la calidad de la misma. Ejemplo de ello es el proyecto VISMEDIC, el cual radica en la Universidad de las Ciencias Informáticas, en La Habana, Cuba. El mismo tiene como objetivo desarrollar soluciones para el entrenamiento y diagnóstico quirúrgico de los especialistas en cirugías de mínimo acceso. En los inicios del mismo, los modelos 3D que se representaban en las escenas virtuales se realizaban con una herramienta de diseño, elemento que disminuía el nivel de realismo de las escenas quirúrgicas. Con el objetivo de eliminar esta deficiencia se desarrollaron investigaciones previas que

generaban un modelo 3D a partir de las imágenes obtenidas de un Tomógrafo o una Resonancia Magnética. Aunque las mismas arrojaron resultados prometedores, estas no cuentan con la calidad visual y estructural que requieren los médicos especialistas.

Luego la **situación problémica** es la siguiente: en el proyecto Vismedic existe un módulo de reconstrucción de superficies que reconstruye un modelo que no cuenta con la calidad geométrica requerida por los especialistas médicos.

Es por ello que se define como **problema científico**: ¿Cómo obtener una representación 3D de los modelos anatómicos de mayor calidad geométrica que la que se obtiene con el módulo de reconstrucción de superficies que actualmente se utiliza en el proyecto VISMEDIC?

Por tanto se toma como **objeto de investigación** la reconstrucción de superficies de modelos 3D a partir de Imágenes.

Luego el **objetivo** de esta investigación es desarrollar un módulo para la reconstrucción de superficies de Modelos Anatómicos a partir de Imágenes DICOM. Y dentro de este amplísimo campo del conocimiento se propone como **campo de acción** la reconstrucción de Superficies de Modelos Anatómicos 3D a partir de Imágenes DICOM.

En este trabajo se defiende la idea de que con el empleo del Marching Cube, como algoritmo de reconstrucción de superficies de modelos anatómicos a partir de imágenes DICOM, se obtendrá mayor calidad geométrica en los modelos que los obtenidos con el algoritmo con que cuenta el módulo de reconstrucción actual del proyecto VISMEDIC.

Para dar cumplimiento al objetivo de la investigación se proponen las siguientes **tareas investigativas**:

- Análisis del proceso de reconstrucción de superficies a partir de imágenes DICOM para conocer sus características fundamentales.
- Estudio de la estructura y especificación del formato de las imágenes DICOM para su manejo y reconstrucción.

- Identificación y caracterización de las técnicas y algoritmos de reconstrucción de superficies, a partir de imágenes DICOM, de estructuras anatómicas tridimensionales para conocer sus ventajas y desventajas.

- Selección del algoritmo de reconstrucción a emplear a partir del marco teórico estudiado para su posterior implementación.
- Desarrollo de un módulo de reconstrucción de superficies de estructuras anatómicas a partir de imágenes DICOM para obtener mejores resultados geométricos en los modelos reconstruidos.

Para todo el proceso de investigación y elaboración de este trabajo se tomará en cuenta la utilización de varios **métodos científicos de investigación** como:

En el nivel teórico:

Analítico – Sintético: Este método se utilizará en la investigación para extraer y analizar la información acerca de las técnicas de visualización y reconstrucción de superficies de modelos anatómicos.

Histórico – Lógico: Este método teórico se utilizará en la investigación para el análisis de la evolución y tendencias actuales del proceso de reconstrucción de superficies.

Modelación: Se utilizará para realizar un diseño simplificado de la realidad a través de diagramas de clases, diagramas de flujo y diagramas de componentes.

En el nivel empírico:

Análisis de fuentes de información: Se utilizará para analizar las distintas fuentes de información como internet, libros, revistas, entre otras.

Consulta de Especialistas: Se consultará a distintos profesores de la facultad que tienen conocimientos del tema a tratar en la investigación, tales como Osvaldo Pereira Barzaga. Profesor Instructor.

Observación: Se utilizará para observar como los distintos algoritmos de reconstrucción influyen en el rendimiento de la computadora y en la calidad de la visualización.

Experimento: Una vez implementado el algoritmo de reconstrucción de superficies se realizarán distintas pruebas con el fin de ver si se comporta según lo esperado para distintas entradas.

En el nivel estadístico: Se empleará la estadística descriptiva, como cálculo porcentual y tablas.

Capítulo 1: Fundamentación Teórica.

En este capítulo se abordarán de manera sintetizada los elementos teóricos de las principales técnicas y algoritmos propuestos a nivel mundial para la reconstrucción 3D de superficies de modelos a partir de imágenes digitales. Además se explicará el proceso que se sigue para la realización de dicha reconstrucción, independientemente del algoritmo o técnica empleada en la misma.

1.1. Reconstrucción 3D de superficies.

La reconstrucción 3D de superficies fue la primera técnica de representación tridimensional que existió aplicada al diagnóstico médico; se caracteriza por visualizar solamente la superficie de los objetos presentes en las imágenes a procesar. Esta técnica tiene una serie de aplicaciones tanto en la medicina como en el campo de la realidad virtual. Muchos de los sistemas informáticos aplicados en la salud con fines de entrenamiento quirúrgico, diagnósticos y prácticas preparatorias requieren que, partiendo de las imágenes adquiridas por modalidades de adquisición de imágenes como las TAC y la RM, se realice una representación 3D de los órganos del paciente; ya que a los médicos especialistas les es más práctico realizar un análisis de los órganos en 3D. El ejemplo más reciente de estos sistemas son los simuladores quirúrgicos de cuarta generación, en los cuales se representan anotomías reales de pacientes obtenidas de un proceso de reconstrucción 3D a partir de imágenes DICOM. Además, la reconstrucción de superficies, como las demás técnicas de reconstrucción 3D, les permite a los médicos hacer un estudio detallado de la anatomía interna de un paciente determinado y así poder tomar decisiones importantes sobre dicho paciente.

Las técnicas de reconstrucción de superficies involucran solamente la generación de la superficie 3D del modelo, esta particularidad o esta característica trae como consecuencia que la imagen final solo muestra el exterior del objeto reconstruido, no pudiéndose analizar las estructuras

internas del mismo, elemento este que puede ser visto tanto como una ventaja o como una desventaja dependiendo de los requerimientos de la aplicación.

Sin embargo las técnicas de reconstrucción de superficies 3D, por lo general, utilizan solamente el 10% del volumen de datos a reconstruir, ya que se centran en la reconstrucción de una estructura específica y no en el volumen completo, por lo que constituye una técnica muy rápida y eficaz. **[23]**

1.2. Etapas del flujo de datos para la reconstrucción 3D de superficies.

Las aplicaciones médicas de reconstrucción 3D de superficies tienen cuatro pasos fundamentales, ver **Figura 1.1**, aunque en un mismo algoritmo se pueden combinar los últimos tres pasos. El proceso está compuesto por las siguientes etapas:

Figura 1.1: Etapas del proceso de reconstrucción 3D de superficies.

1.2.1. Adquisición de los datos.

La adquisición de los datos consiste en la toma de muestras del mundo real para generar datos que puedan ser manipulados por una computadora. Consiste en tomar un conjunto de variables

físicas y convertirlas en tensiones eléctricas y digitalizarlas de manera tal que una computadora pueda manejarlas.

Este término de adquisición de datos es muy utilizado en diferentes campos de la ciencia y en todos tiene una gran importancia, ya que mediante la reconstrucción 3D de estos datos los objetos reconstruidos pueden ser analizados en una computadora y utilizados en diferentes campos de la computación. En cada uno de los campos donde se aplica la adquisición de datos existen diversas técnicas que convierten, de una forma o de otra, los datos del mundo real a datos digitales; en la medicina, por ejemplo, los datos pueden ser adquiridos de diferentes formas y por diversos equipos que extraen una muestra de las estructuras internas de un paciente determinado y producen múltiples capas (imágenes) 2D de información, ver **Figura 1.2**. Entre las técnicas de adquisición de datos en la medicina se encuentran la Tomografía Axial Computarizada (TAC), la Resonancia Magnética (RM), entre otras. **[29]**

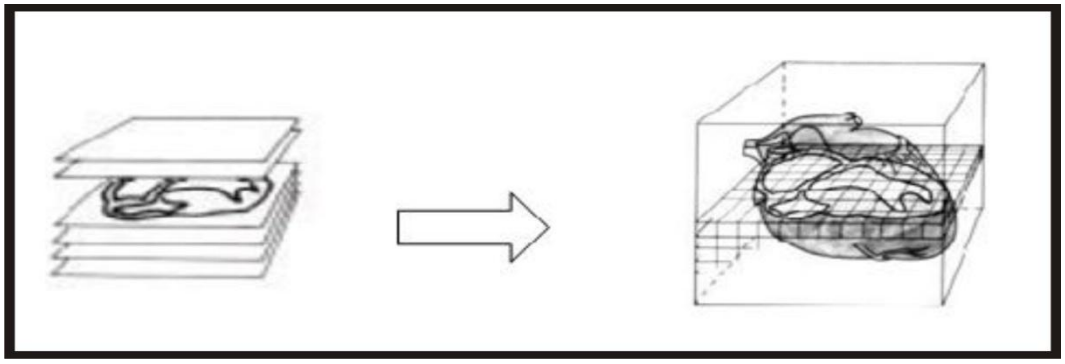

Figura 1.2: Estructuras de las imágenes médicas digitales.

Para la reconstrucción 3D de superficies no importa mediante que técnica se adquieran los datos, aunque sí existen unas mejores que otra; simplemente estos datos tienen que cumplir una serie de características que hacen aplicable el proceso de reconstrucción 3D sobre ellos, como por ejemplo, tienen que ser representados por una nube de puntos donde cada uno de ellos posee una valor escalar que representa el color del objeto en esa parte.

En el campo de la medicina, donde se centra este trabajo, las técnicas de adquisición exportan sus ficheros en el formato **DICOM** (**D**igital **I**maging and **Co**mmunications in **M**edicine) un estándar desarrollado en 1983, donde el Colegio Estadounidense de Radiología (ACR) y la Asociación Nacional de Fabricantes Eléctricos (NEMA) formaron un comité cuya misión era hallar o desarrollar una interfaz entre el equipamiento y cualquier otro dispositivo que el usuario quiera conectar. Además de las especificaciones para la conexión de hardware, el estándar se desarrollaría para incluir un diccionario de los elementos de datos necesarios para la interpretación y exhibición de imágenes.

El estándar describe el formato de archivos y la especificación de los datos primordiales de un paciente en la imagen, así como el encabezado requerido, describiendo un lenguaje común a distintos sistemas médicos. De esta forma las imágenes vienen acompañadas de mediciones, cálculos e información descriptiva relevante para diagnósticos. Utiliza archivos con extensión .dcm.

1.2.1.1. Formato de un archivo DICOM.

Un solo archivo DICOM contiene una cabecera que almacena la información sobre el nombre del paciente, el tipo de exploración, dimensiones de los datos, así como todos los datos de la imagen que pueden contener la información en tres dimensiones.

El formato genérico del archivo DICOM consiste en dos partes: **Header** seguido inmediatamente por un **Data Set** de DICOM.

El **Data Set** de DICOM contiene la imagen o las imágenes especificadas. El **Header** contiene la sintaxis de transferencia **UID** (identificador único) que especifica la codificación y la compresión del **Data Set**. **[25]**

1.2.2. Procesamiento de la imagen.

El procesamiento de las imágenes médicas se ha convertido en unos de los campos más importantes en el proceso de reconstrucción 3D, ya que posibilita que los datos adquiridos mediante alguna técnica de adquisición de datos sean mejorados y se acondicionen con los objetivos de la reconstrucción.

Las imágenes médicas adquiridas en el paso anterior se caracterizan fundamentalmente por la dificultad que existe a la hora de generar información válida para ser procesada. Estas imágenes poseen gran cantidad de ruido y una enorme variabilidad en sus propiedades. A la hora de realizar la reconstrucción 3D de las estructuras anatómicas presentes en dichas imágenes, algunos algoritmos usan técnicas de procesamiento de imágenes para encontrar datos de interés dentro de la estructura 3D o filtrar datos de la imagen original.

Unas de las técnicas más usadas en el procesamiento de imágenes son los filtros, que atenúan, de una forma o de otra, el ruido y mejoran visualmente la imagen, modificándole ciertas propiedades como el brillo, el contraste, entre otras. Entre los filtros más usados para este tipo de reconstrucción se encuentran los **lineales suavizadores**, **laplociono**, **gradiente** y **promediadores**. **[27]**

Otra de las alternativas para el procesamiento de la imagen es aplicar alguna técnica de segmentación. Entre las técnicas más utilizadas para la reconstrucción 3D están:

- **Los métodos de umbralización:** es una técnica efectiva para obtener la segmentación de imágenes, donde estructuras diferentes tienen intensidades contrastantes u otras características diferenciables. La partición usualmente es generada interactivamente, pero también existen métodos automáticos. **[28]**

- **Los métodos clasificadores:** son técnicas de reconocimiento de patrones que buscan particionar un espacio característico, derivado de la imagen, usando datos con etiquetas conocidas. Estos son conocidos como métodos supervisados debido a que requieren datos de

entrenamiento que son segmentados manualmente, para luego ser utilizados en la segmentación automática de nuevos datos. **[28]**

- **Crecimiento de regiones:** extrae las regiones de la imagen que estén conectadas según cierto criterio predefinido. Este criterio puede estar basado en información de intensidades y/o bordes de la imagen. En su forma más simple, este método requiere un punto semilla que es seleccionado manualmente por el usuario, y extrae todos los pixeles conectados a la semilla, que tengan el mismo valor de intensidad. **[28]**

- **Los métodos de agrupamiento (clustering):** son métodos no supervisados porque no utilizan datos de entrenamiento como los métodos clasificadores. Estos métodos iteran entre segmentar la imagen y caracterizar las propiedades de cada clase. En este sentido, los métodos de agrupamiento se entrenan usando los datos disponibles. Un algoritmo de agrupamiento común es el algoritmo de las **K-medias** o algoritmo **ISODATA**. **[28]**

1.2.3. Reconstrucción de la Superficie.

Una vez que se separen las estructuras de interés en la imagen y esta esté lista para la reconstrucción, se procede a la reconstrucción 3D de la superficie de los objetos especificados en la misma mediante alguna de las siguientes técnicas de reconstrucción:

1.2.3.1. Representación por iconos.

Los iconos (glyphs) son objetos gráficos usados para representar información de puntos dentro del modelo. Estos objetos pueden ser coloreados, escalados y orientados en dependencia del valor del campo seleccionado y son usualmente utilizados para mostrar la ubicación de los nodos en un cubo. Esta es una técnica que coloca uno de estos objetos, generalmente, en forma de esfera, diamante o flecha en cada uno de los puntos del modelo de datos, ver **Figura 1.3.** Una de

las características que posee esta técnica es que no es necesario el uso de interpolación por lo que consume poco tiempo del CPU. **[22]**

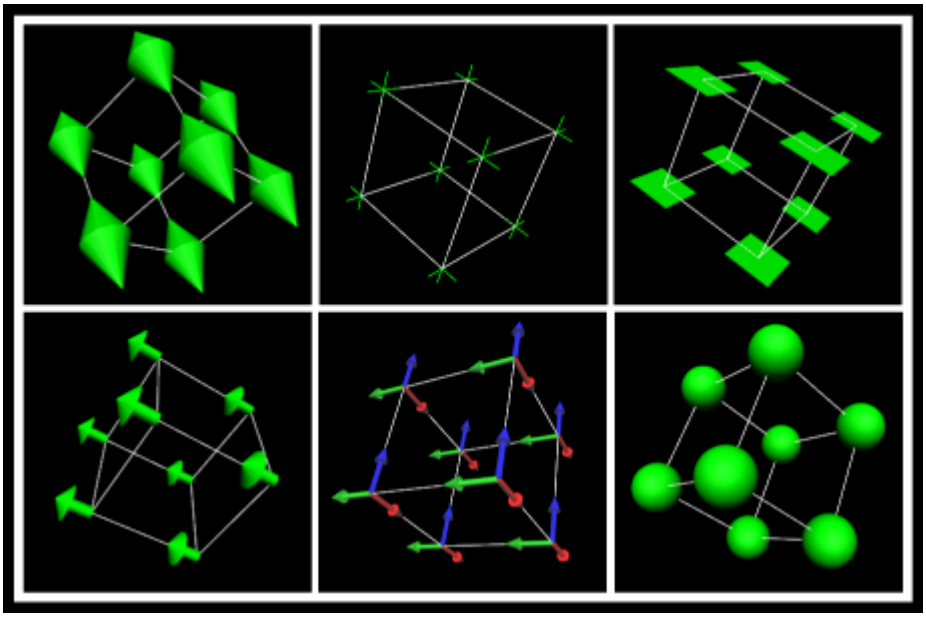

Figura 1.3: Ubicación de los iconos en los nodos del cubo.

1.2.3.2. Isosuperficie (Isosurface).

Esta técnica produce superficies en el ámbito de un valor escalar, el cual es llamado isovalue (isovalor), donde dichas superficies pueden ser coloreadas acorde al mismo valor escalar o a otro valor. Este último caso en que las superficies son coloreadas por un valor escalar cualquiera permite la búsqueda de la correlación entre dos magnitudes escalares.

Los diferentes algoritmos que han sido desarrollados bajo esta técnica utilizan interpolación lineal para construir una función continua, donde la exactitud de las superficies generadas, en correspondencia con la topología del objeto real, depende de lo bien que coincida la función continua creada con la función continua subyacente que representa el volumen de datos. **[24]**

1.2.3.3. Técnica basada en planos.

La técnica basada en planos se basa fundamentalmente en la proyección de la superficie del objeto mediantes planos de cortes, estos planos pueden estar alineados con respecto a los ejes de coordenadas o de manera trasversal, ver **Figura 1.4**, de forma tal que la superficie del objeto pueda ser vista de diferentes ángulos. La idea general de esta técnica es proyectar en dichos planos las partes del volumen de dato que son cortadas por ellos. **[30]**

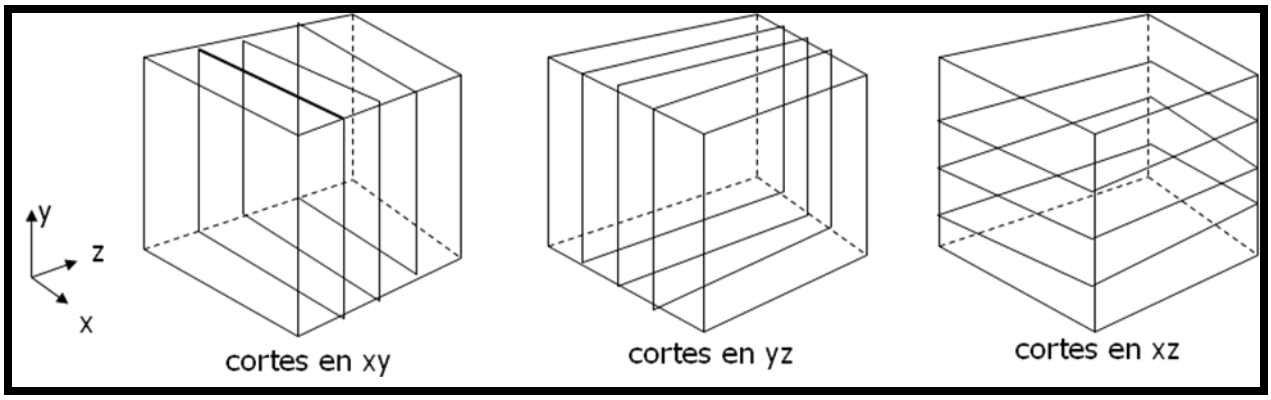

Figura 1.4: Conjunto de cortes alineados a los ejes de coordenadas.

1.2.4. Visualización.

Una vez creado el modelo que representa la superficie de los objetos presentes en el volumen de datos, en el proceso de reconstrucción, el paso final consiste en la visualización de dicho modelo. Generalmente este modelo es representado mediante una malla poligonal, aunque pudiera representarse también mediante una nube de puntos.

1.3. Isosuperficie.

Esta es una técnica de reconstrucción 3D que posibilita la extracción y visualización de la superficie de los objetos presentes en un volumen de datos. Constituye una de las técnicas más

usadas a nivel mundial donde los usuarios seleccionan la superficie deseada especificando un valor de densidad y esta es representada, generalmente, mediante una malla poligonal.

Esta técnica no solamente es usada en el campo de la medicina si no que también puede ser usada para el análisis de un conjunto de datos generados por simulaciones medioambientales. Dentro de esta técnica se han desarrollado diversos y populares algoritmos que posibilitan generar y visualizar la superficie deseada.

1.3.1. Algoritmos Basados en Marching.

Estos algoritmos constituyen métodos muy simples que extraen superficies de funciones implícitas o información tridimensional discreta, ver **Figura 1.5**. Estos algoritmos se basan en subdividir una superficie o volumen en formas más elementales y detectar uno o más contornos; ubicando el color, valor o intensidad en los vértices de la figura elegida. De esta manera, por medio de interpolación, es posible detectar cuáles aristas de la figura son interceptadas por la superficie y reconstruir, a partir de los puntos de intersección, la superficie con figuras más elementales como los triángulos.

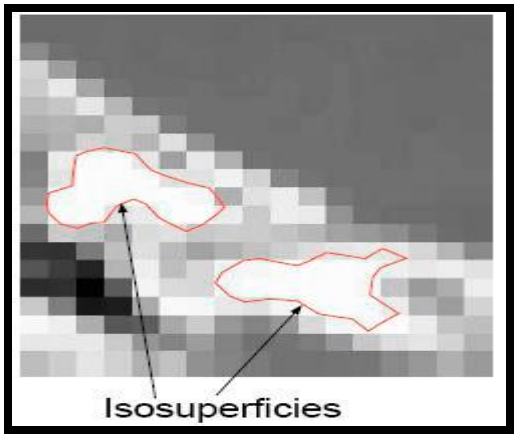

Figura 1.5: Vista 2D de una isosuperficie.

1.3.1.1. Marching Cubes.

El algoritmo propuesto por William E. Lorensen y Harvey E. Cline en 1987 nombrado Marching Cubes, es uno de los algoritmos de reconstrucción 3D de superficies más populares y más robustos que se han conocido hasta la actualidad, gracias a que brinda una manera muy sencilla y eficiente de convertir los datos volumétricos en una malla poligonal.

El algoritmo Marching Cubes basa el estudio del volumen mediante la subdivisión en pequeños cubos, donde analiza cada uno de estos pequeños elementos y calcula, de forma independiente, los triángulos que genera la intersección de cada cubo con la superficie que se quiere calcular. Para determinar qué cubos serán parte de la superficie final, el algoritmo cuenta con una tabla de búsqueda donde se encuentran los 15 casos posibles en que un cubo puede ser cortado por la superficie, ver **Figura 1.6**.

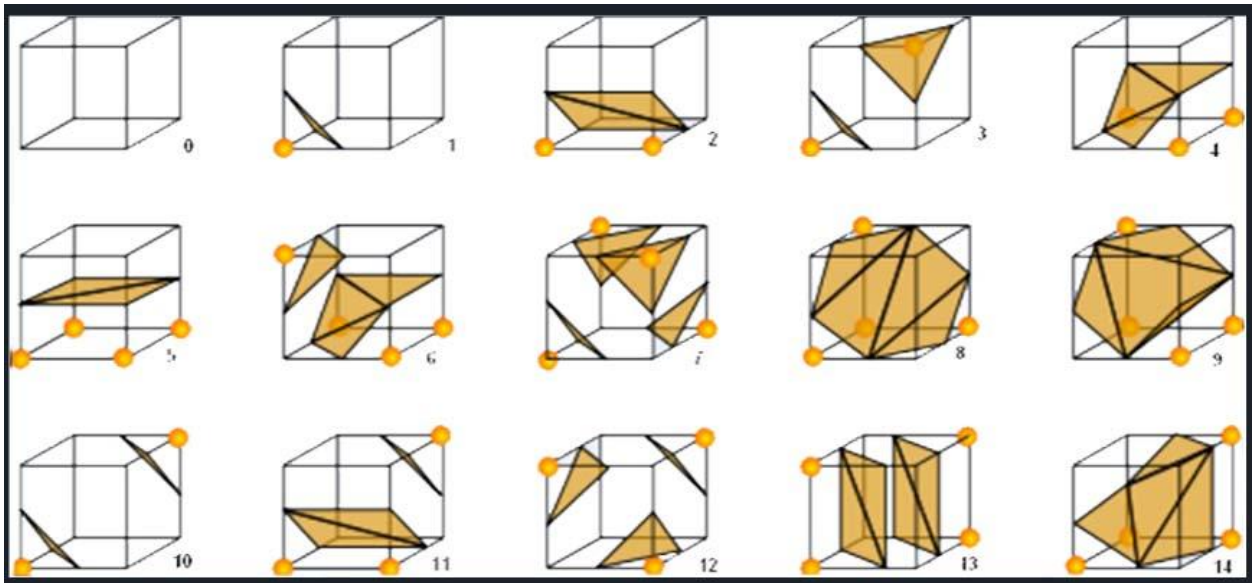

Figura 1.6: Tabla de configuración del Marching Cubes.

Para determinar en qué caso de la tabla de configuración se encuentra un cubo, se analizan sus vértices y se comparan con el valor escalar introducido por el usuario, que especifica la densidad de la superficie deseada; si todos estos vértices están por encima o por debajo del valor

introducido, quiere decir que el cubo no forma parte de la superficie y por lo tanto no es procesado, en caso contrario, el cubo es cortado en varias de sus aristas por lo que formará parte de la superficie final. **[1]**

Este algoritmo, como anteriormente se explicaba, brinda una manera muy sencilla para convertir los datos volumétricos en una malla poligonal, pero posee algunas desventajas como es el caso de las ambigüedades en su tabla de configuración y el tiempo de creación de la malla para grandes volúmenes de datos.

1.3.1.1.1. Ambigüedades del Marching Cubes.

El Marching Cubes posee ciertos casos en la tabla de configuración que, cuando se combinan, generan huecos en la malla generada, por lo que la imagen final del objeto reconstruido no se asemeja a las características reales que este posee. En la tabla de configuración del algoritmo existen dos tipos de ambigüedades: las ambigüedades en las caras y las ambigüedades internas. Las primeras ocurren cuando una cara tiene dos vértices diagonales opuestos positivos y otros dos negativos. Nielson y Hamann **[2]** muestran cómo esto puede ocurrir entre dos celdas vecinas y puede dar lugar a agujeros y a inconsistencias en la topología de la malla resultante. Las segundas ocurren en el interior de las celdas y son identificadas, independientemente, por Natarajan **[3]** y Chernyaev **[4]**, donde cada uno brinda una solución para resolverla. Otra de las formas en que este tipo de ambigüedades puede ser resuelta es adicionando un punto dentro de cada celda. Para determinar estos puntos extras, muchos métodos asumen que la función implícita del volumen de datos es lineal a lo largo de una arista, bilineal en una cara y trilineal dentro de una celda o cubo. Bajo esta suposición otros métodos han sido propuestos para resolver los casos ambiguos en las caras **[3]** y dentro de las celdas **[5]**, para así poder determinar una topología consistente.

1.3.1.2. Marching Tetrahedra.

Este algoritmo es muy semejante al Marching Cubes, la diferencia fundamental está en que basa el estudio del volumen en pequeños tetraedros y no en cubos. En este algoritmo cada celda es dividida en seis tetraedros y las aristas de estos se alinean con las celdas adyacentes, luego cada uno de estos tetraedros es procesado por separado y calculada la superficie final. El algoritmo cuenta también con una tabla de búsqueda donde se encuentran los posibles casos en que un tetraedro puede ser cortado por la superficie, ver **Figura 1.7. [6]**

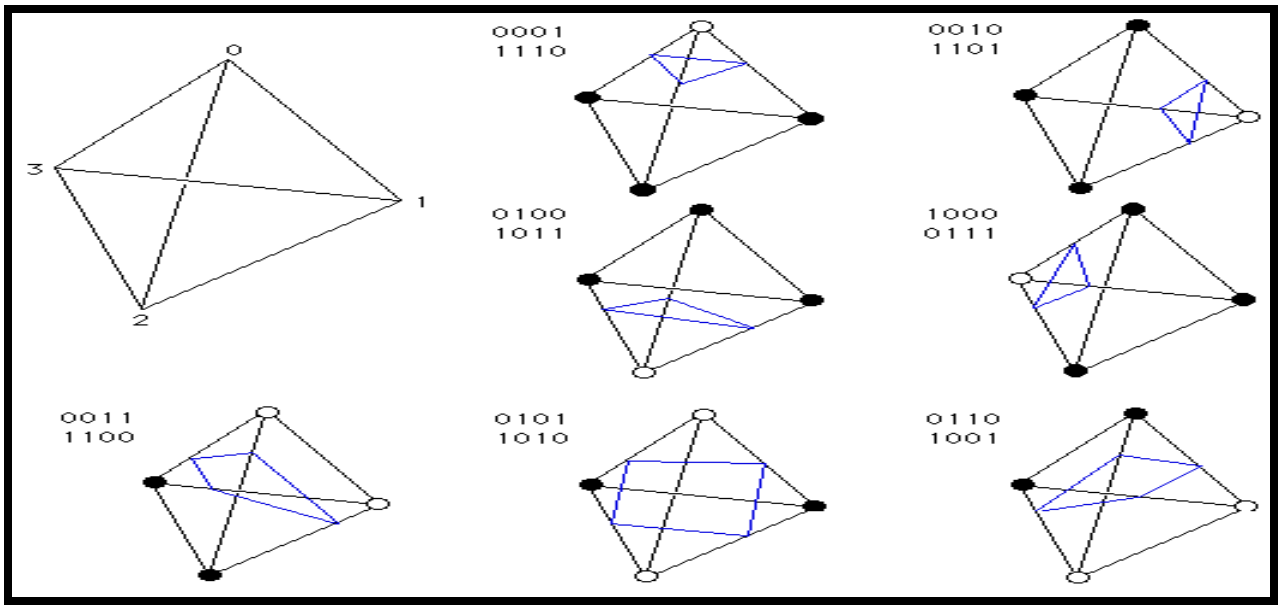

Figura 1.7: Tabla de configuración del Marching Tetrahedra.

1.3.1.3. Cubical Marching Square.

Este algoritmo fue propuesto en el 2005 por Chien-Chang Ho, Fu-Che Wu, Bing-Yu Chen, Yung-Yu Chuang y Ming Ouhyoung, y se basa en la idea de que el Marching Cubes puede ser tratado como el algoritmo Marching Square, un algoritmo que genera líneas del contorno para un valor escalar en 2D y muy similar al Marching Cubes, además de que la dependencia entra las celdas puede ser eliminada adicionando rasgos específicos en las caras y que las normales no sólo se

pueden utilizar para lograr una mejor visualización, sino que también para resolver las ambigüedades y las inconsistencia en la malla poligonal generada. Como se muestra en la **Figura 1.8** un cubo es expandido en 6 caras, para cada una de ellas se genera la isocurva usando el algoritmo Marching Square, donde esta isocurva resultante consiste en varios segmentos. Luego si se doblan estas caras nuevamente a la forma del cubo original y se conectan entre sí, obtenemos exactamente los mismos componentes como lo hace el Marching Cubes. Finalmente estos componentes son triangulados para generar la superficie. **[7]**

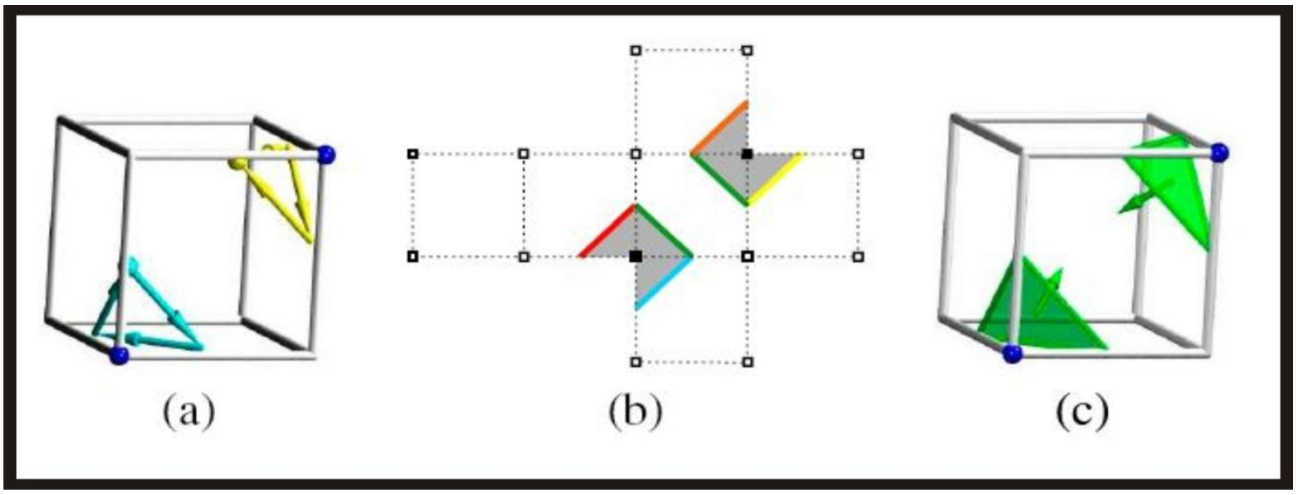

Figura 1.8: Algoritmo Cubical Marching Square.

1.3.1.4. Marching Voxel.

Este algoritmo constituye una versión mejorada del Marching Cubes original. Aquí en vez de generar la superficie a través de la configuración de los cubos, genera primero los triángulos para cada vóxel interno en correspondencia con los ejes de coordenadas, ver **Figura 1.9**. Entonces se procede a combinar los triángulos de cada vóxel para producir la superficie del objeto. Finalmente esta superficie es proyectada en un plano para formar la imagen final. Este algoritmo no posee casos ambiguos y es menos lento a la hora de generar la superficie, pero posee inconvenientes con el resultado final**. [8]**

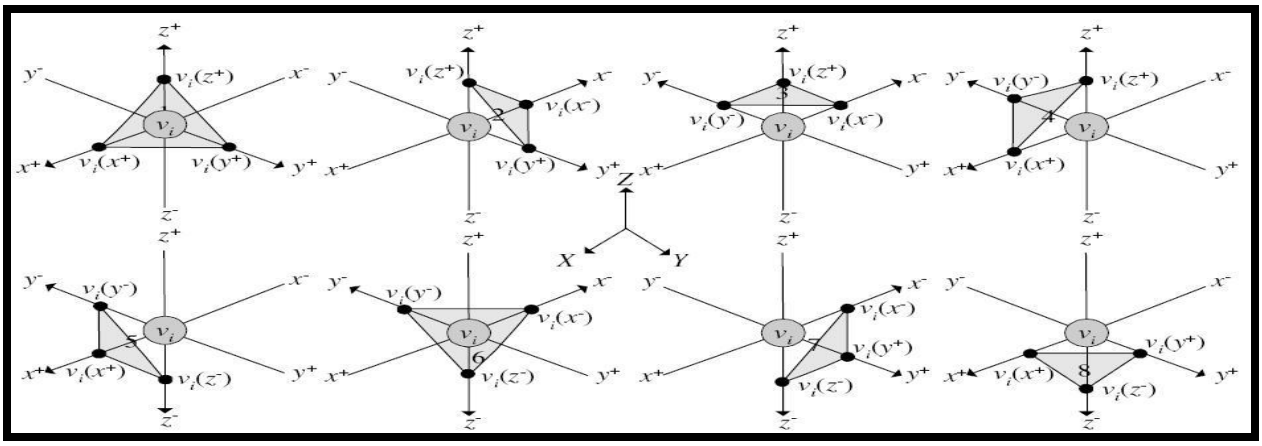

Figura 1.9: Ejemplo de los triángulos generados para un vóxel interno.

1.3.1.5. Dividing Cubes.

A diferencia de Marching Cubes, que genera triángulos como primitivas, Dividing Cubes genera puntos (nube de puntos). En este algoritmo dado un valor de densidad, se selecciona una celda (cubo) por donde la superficie pase. Luego se subdivide en una rejilla regular $n_1 X n_2 X n_3$, donde los n_i se determinan mediante la división entre el ancho del vóxel original sobre la resolución de la pantalla. Una vez realizado esto se calculan los valores para los nuevos puntos con interpolación. Luego se chequea si la superficie pasa por cada subvóxel; si pasa, se crea un punto en el medio de este, y se calcula la normal usando interpolación estándar. Estos puntos son los que van a formar parte de la nube de puntos que formarán el objeto.

Este algoritmo es muy eficiente a la hora de hacer el rendering, ya que hacer render de puntos es mucho más eficiente que el render de polígonos. Además existen otras operaciones que son más eficientes en puntos como por ejemplo, clipping o merging. Una desventaja importante es que como la superficie es una nube de puntos sin conexión, hacer zoom revela agujeros dentro del objeto. **[9]**

1.3.1.6. Otros algoritmos.

En el proyecto **VISMEDIC** de la facultad 5 anteriormente se implementó un algoritmo para la reconstrucción 3D de superficies, en este algoritmo el proceso de segmentación es muy importante, puesto que garantiza que en las regiones se almacenen solamente aquellos vóxeles que son fronteras (pertenecientes al contorno), donde cada uno posee una referencia a los vóxeles vecinos, evitando así búsquedas innecesarias de estos vecinos en cada vóxel. El primer paso de este algoritmo consiste en detectar en cada uno de estos vóxeles las caras fronteras. Luego para cada una de estas caras se procede a la triangulación inicial, que consiste en dividirla, a partir de un trazado diagonal, en dos triángulos.

Una de las ventajas fundamentales de este algoritmo es que, gracias a la triangulación que realiza en las caras fronteras de los vóxeles, todos los triángulos generados son isósceles haciendo que la malla final de la superficie del objeto sea muy buena en cuanto a la estructuras de sus triángulos. La principal desventaja de dicho algoritmo consiste en que necesita de un post-procesamiento de suavizado para disminuir las irregularidades y lograr una apariencia más semejante a la forma originales de los materiales, ver **Figura 1.10. [31]**

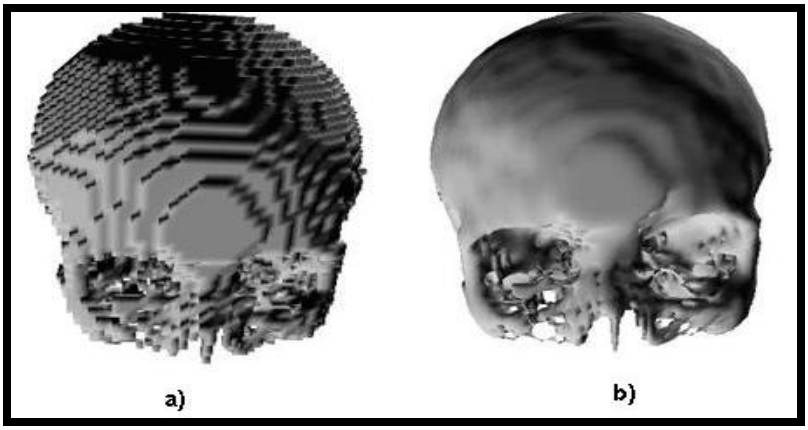

Figura 1.10: Descripción poligonal de la superficie. (a) Triangulación inicial. (b) Superficie suavizada.

1.4. Programación en el hardware gráfico

La **unidad de procesamiento gráfico** o **GPU** es un procesador diseñado para los cómputos implicados en la generación de gráficos 3D interactivos. Inicialmente fueron utilizados en la industria de los videos juegos, pero en la actualidad no solo tienen gran impacto en las aplicaciones de gráficos por computadoras, sino también, en el ámbito científico y de simulación. Entre las características que poseen estos procesadores gráficos se encuentran el bajo costo a la hora de realizar cálculos, el gran paralelismo y la optimización para cálculos en coma flotante.

Una **GPU** está altamente segmentada, lo que indica que posee gran cantidad de unidades funcionales. Estas unidades funcionales se pueden dividir principalmente en dos: aquellas que procesan vértices (Vertex Shader), y aquellas que procesan píxeles (Fragment Shader). Por tanto, se establecen el vértice y el píxel como las principales unidades que maneja la GPU. Adicionalmente, y no con menos importancia, se encuentra la memoria que se destaca por su rapidez, y va a jugar un papel relevante a la hora de almacenar los resultados intermedios de las operaciones y las texturas que se utilicen.

El **Vertex Shader** (VS) es una subrutina que se encarga del procesamiento de los vértices. Llevan a cabo tareas como la transformación del vértice, la normalización, entre otras. Los VS se ejecutan una vez por cada vértice que se manda a la tarjeta y puede ocurrir que se procesen varios vértices a la vez, en paralelo, dependiendo del hardware **[10, 11]**.

El **Fragment Shader** (FS) es la subrutina que se encarga, entre otras cosas, de decidir el color con el que se va a pintar un píxel en pantalla. Cada vez que se envía una primitiva a dibujarse, el VS se ejecutará una vez por cada vértice de la primitiva, pero el FS se ejecutará una vez por cada píxel dentro de la primitiva **[10, 11]**.

En el 2006 la NVidia incorpora al mercado una nueva tarjeta gráfica, la GeForce 8800, que soportaba un nuevo tipo de shader, el **Geometry Shader**. Este tipo de shader opera entre el Vertex y el Fragment Shader y comienza con una simple primitiva (punto, línea o triangulo), y puede leer los atributos de alguno de los vértices en la primitiva y usarlo para emitir nuevas primitivas **[12]**.

1.4.1. Reconstrucción de superficies en el GPU.

Los equipos de adquisición de imágenes médicas actuales generan imágenes con alta precisión, por lo que poseen una enorme cantidad de información que dificulta el procesamiento de dichas imágenes en tiempo real. La visualización en tiempo real de las estructuras anatómicas presentes en estas imágenes ha sido un desafío importante durante muchos años. Con la llegada del nuevo hardware gráfico se ha hecho posible generar, de forma rápida, la malla de la superficie de los modelos presentes en el volumen de datos.

Existen diferentes vías para generar la superficie a través del GPU: en el Fragment Shader con múltiples pasadas de render para generar la geometría 3D **[14, 15]**, en el Vertex Shader con una sola pasada de render **[16, 17, 18, 19]** y a través del Geometry Shader **[20, 21].**

1.5. Metodologías y herramientas de desarrollo.

1.5.1. Metodologías de Ingeniería de Software.

En el desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. No existe una metodología de software universal. Las características de cada proyecto exigen que el proceso sea configurable.

Atendiendo a este criterio para la realización del análisis y diseño de la aplicación se hizo uso de la metodología RUP (Proceso Unificado de Desarrollo), ya que es una metodología muy utilizada mundialmente y probada en muchos proyectos de gran impacto a nivel mundial. Es muy robusta puesto que unifica los mejores elementos de metodologías anteriores y además es orientada a objetos, un paradigma de la programación muy usado actualmente.

Las características fundamentales de esta metodología que influyeron en su selección se brindan a continuación:

- **Dirigido por casos de usos:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí, los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

- **Iterativo e incremental:** RUP divide el proyecto en fases de desarrollo, propone además que cada una de ellas se desarrolle en iteraciones, las cuales aportan un incremento en el proceso de desarrollo y terminan con el cumplimiento del punto de control trazado en la fase.

1.5.2. Herramientas de desarrollo.

- **Rational Rose Enterprise Edition** para el modelado de la aplicación, puesto que brinda una manera muy sencilla de modelado gracias a su interfaz, además, constituye una potente herramienta para el apoyo a la ingeniería de software como soporte al Proceso Unificado de Desarrollo (RUP).

- **Microsoft Visual Studio.NET 2005** como IDE (entorno integrado de desarrollo), gracias a las facilidades que brinda para la codificación y completamiento de código.

1.5.3. Lenguajes.

1.5.3.1. Lenguajes de programación.

El lenguaje de programación escogido fue el C++ ya que constituye uno de los leguajes más robustos y más usados a nivel mundial para aplicaciones de realidad virtual, además de ser un lenguaje multiplataforma y orientado a objetos. Permite además un gran control de la memoria de la computadora por lo que para estas aplicaciones es de vital importancia ya que se procesan grandes volúmenes de datos.

1.5.3.2. Lenguaje de modelado.

Se escogió como lenguaje de modelado para la realización del análisis y diseño el UML (Lenguaje Unificado de Modelado), ya que constituye un estándar mundial para el modelado de aplicaciones, además de que se usa para modelar sistemas que usan tecnología orientada a objetos y visualiza, especifica, construye y documenta los artefactos que se crean durante el proceso de desarrollo.

Capítulo 2: Solución Propuesta.

Después del estudio teórico realizado en el capítulo anterior, en este capítulo se describe en detalle el algoritmo seleccionado que conformará el proceso de reconstrucción tridimensional de los modelos; así como las modificaciones que fueron realizadas al mismo, con el objetivo de adaptarlo a las particularidades de la solución.

2.1. Reconstrucción 3D de superficies mediante isosuperficie.

La reconstrucción 3D de superficie, como se explicaba en el capítulo anterior, es una técnica de reconstrucción tridimensional que se caracteriza por representar solamente la superficie de las estructuras anatómicas de interés presentes en las imágenes médicas a procesar, ver **1.1**.

Para generar la superficie de un objeto se puede hacer por diferentes formas; una de ellas es utilizando la técnica de isosuperficie, que genera la superficie a partir de un valor escalar que especifica el valor de densidad de la superficie deseada, ver **1.3**. Dentro de esta técnica han surgido una serie de algoritmos que generan de una forma o de otra dicha superficie, tal es el caso del algoritmo Marching Cubes.

En este trabajo los esfuerzos han sido centrados en dicho algoritmo puesto que, aparte de ser muy usado, brinda una manera muy sencilla de convertir los datos volumétricos en una malla poligonal, además produce superficies de muy buena calidad con bajo coste computacional y alta resolución, y proporciona un fácil proceso de render y manipulación de los datos.

De este algoritmo han surgido una serie de mejoras para resolver algunos problemas que posee, como es el caso de las ambigüedades en la tabla de configuración, ver **Figura 1.3.** Estas mejoras de una forma u otra resuelven dicho problema pero descuidan otros de gran importancia, como es el caso del consumo de recursos de la computadora y la originalidad de la malla generada.

De manera general el algoritmo consta de dos etapas fundamentales: La creación del grid de los datos a procesar y la generación de la superficie de dichos datos.

2.2. Adquisición de Imágenes Médicas.

Los datos obtenidos a partir de imágenes médicas digitales (DICOM) usualmente se obtienen como un conjunto de imágenes individuales, ver **Figura 2.1**. Donde cada imagen representa un fino corte de la parte del cuerpo del paciente escaneada y está compuesta por píxeles (elementos de la imagen). Estos píxeles están organizados en una rejilla bidimensional (grid 2D), ver **Figura 2.1**, donde la distancia entre dos píxeles por lo general es constante en cada dirección.

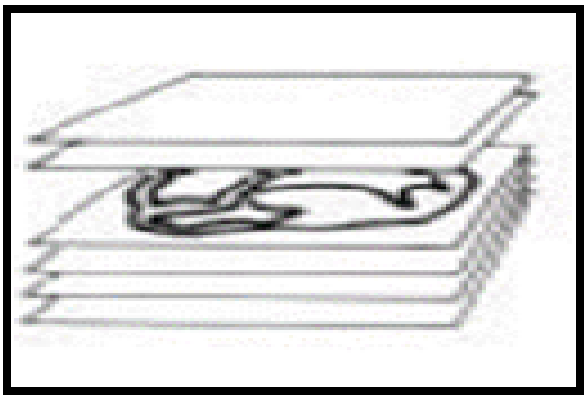

Figura 2.1: Datos de imágenes médicas digitales.

Para muchas técnicas de adquisición de imágenes médicas, como TAC y RM, las direcciones horizontales (eje x) y verticales (eje y) poseen igual distancia entre los píxeles, la cual desde este momento se nombrará *"distancia entre píxeles"*. Una distancia constante entre los píxeles en ambas direcciones permite el cálculo de la posición original del píxel ***P(x, y)*** con sólo multiplicar la distancia entre píxeles ***d*** por los índices ***I (i, j)*** correspondientes al píxel en el grid 2D. Ver **ecuación 2.1** y **Figura 2.2. [29]**

$$P(x, y) = I\,(i, j) * d \qquad (2.1)$$

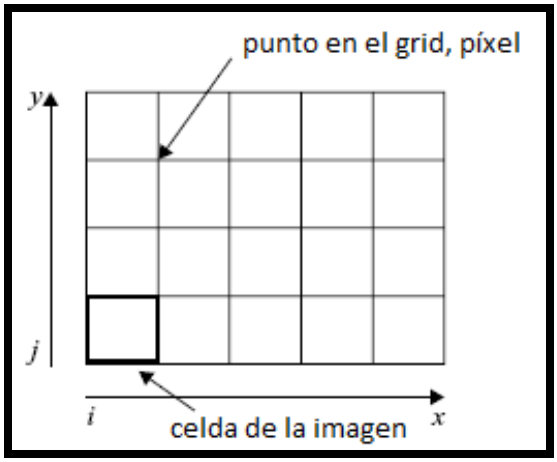

Figura 2.2: Rejilla bidimensional (Grid 2D).

2.2.1. Construcción de Imágenes Médicas Tridimensionales.

Las imágenes volumétricas (imágenes tridimensionales o imágenes 3D) combinan cada imagen individual, de la pila de imágenes obtenidas, en un espacio tridimensional a partir de la construcción de una estructura de datos conocida como rejilla tridimensional (grid 3D), ver **Figura 2.3**.

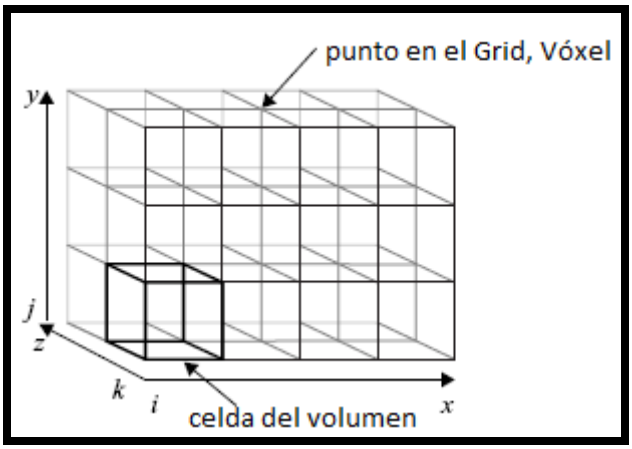

Figura 2.3: Rejilla Tridimensional (Grid 3D).

Una imagen tridimensional estará compuesta por elementos de datos denominados vóxeles (elementos volumétricos), estos elementos están ubicados en los puntos del grid como mismo lo estaba un píxel en una imagen bidimensional, sólo que en adición a las direcciones horizontal (eje x) y vertical (eje y), ahora se tendrá otra dirección correspondiente a la profundidad (eje z), ver **Figura 2.3**. La separación entre vóxeles en la dirección de la profundidad está definida como la distancia entre dos imágenes consecutivas en la pila de imágenes y estará dada por un atributo que se denomina "*distancia entre cortes"*.

Similar a los píxeles, la posición de un vóxel **V*(x, y, z)*** en el espacio tridimensional está dada por los valores de distancia en las direcciones ***x, y, z*** y los índices ***I (i, j, k)*** correspondiente al vóxel en el grid 3D, ver **Figura 2.4. [29]**

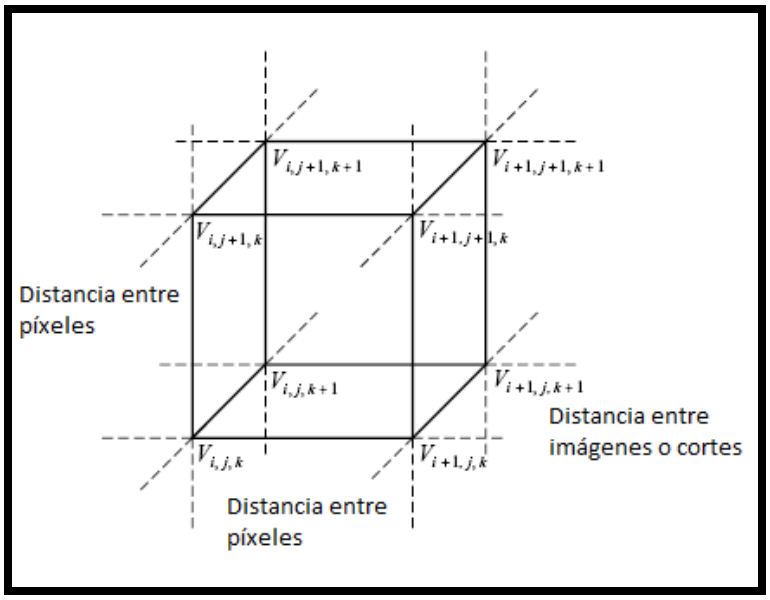

Figura 2.4: Celda 3D.

2.3. Marching Cubes.

Una vez que los datos son almacenados en una rejilla tridimensional se procede a generar la superficie deseada. Este grid 3D junto con el valor de densidad que especifica la superficie a reconstruir constituye la entrada fundamental del algoritmo.

De manera general el algoritmo Marching Cubes determina cómo la superficie intercepta cada una de las celdas que forman el grid 3D, analiza cada una de forma independiente y luego mezcla el resultado obtenido de la triangulación individual en cada celda para obtener la superficie final.

2.3.1. Creación de los cubos.

Para determinar qué partes del volumen formarán la superficie final, se crea un cubo lógico a partir de dos capas contiguas, donde para cada uno de sus vértices (vóxeles) se almacenará su valor de densidad, ver **Figura 2.5**.

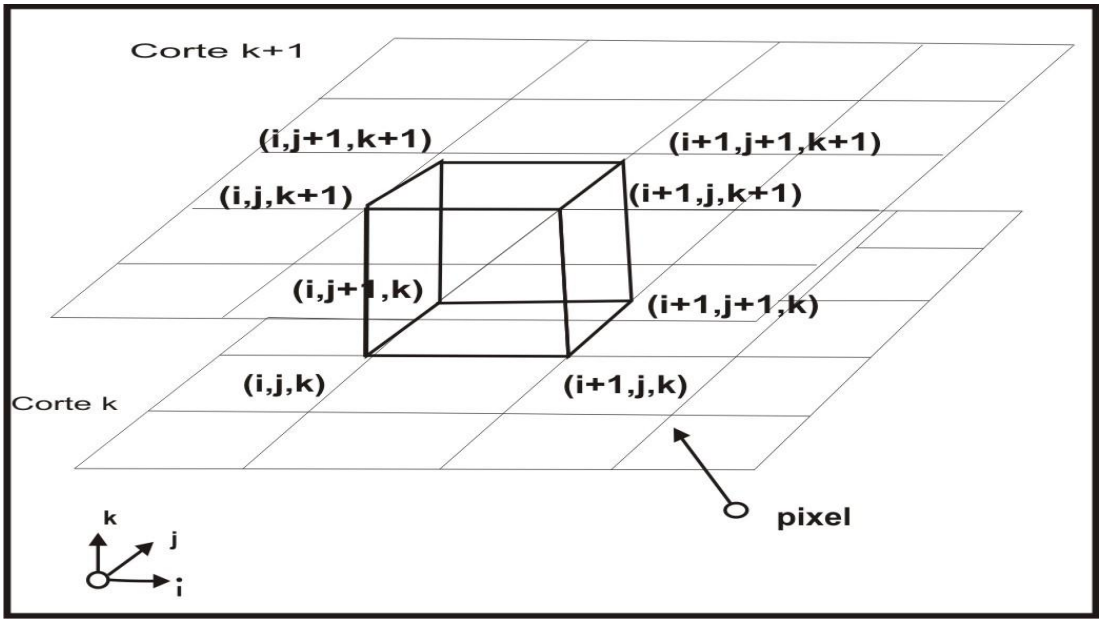

Figura 2.5: Creación de un cubo lógico para el Marching Cubes.

Para cada cubo se almacena, además del valor de densidad de los ocho vértices que lo forman, la posición real de cada vértice, los índices ***i, j, k*****,** el vector normal de cada uno de ellos y el estado, externo (0) o interno (1), en que se encuentra con respecto a al valor de densidad que especifica la superficie deseada.

El vector normal se halla mediante el cálculo del vector gradiente a través del uso de diferencias centrales a lo largo de los tres ejes de coordenadas como se muestra a continuación.

$$G_x(x,y,z) = \frac{D(i+1,\ j,\ k) - D(i-1,\ j,\ k)}{\Delta X} \qquad (2.2)$$

$$G_y(x,y,z) = \frac{D(i,\ j+1,\ k) - D(i,\ j-1,\ k)}{\Delta Y} \qquad (2.3)$$

$$G_z(x,y,z) = \frac{D(i,\ j,\ k+1) - D(i,\ j,\ k-1)}{\Delta Z} \qquad (2.4)$$

Donde $\boldsymbol{D(i,\ j,\ k)}$ es la densidad del píxel $\boldsymbol{(i,\ j)}$ en el corte $\boldsymbol{k}$ y $\boldsymbol{\Delta X, \Delta Y, \Delta Z}$ son las longitudes de las aristas del cubo (distancia entre los píxel).

Para cada cubo formado se calculan los triángulos que se generan según el estado en que se encuentre y la tabla de configuración.

2.3.2. Creación de los triángulos.

Para determinar los triángulos que se generan en un cubo determinado, primeramente se tiene que buscar el estado en la tabla de configuración del Marching Cubes, ver **Figura 1.6**, en que se encuentra dicho cubo con respecto a la superficie buscada.

Enumerando cada vóxel y cada arista del cubo, como se muestra en la **Figura 2.6**, se pueden representar los índices de los vóxeles en un entero de 8 bit, donde cada bit corresponde a uno de los vóxeles. De esta forma, quedarían con valor 1 los bits correspondientes a los vóxeles externos y en 0 los bits de los vóxeles internos. Esta combinación conforma un índice que permite saber el estado del cubo y, además, buscar en tiempo constante en la tabla de aristas **[6]** las aristas interceptadas.

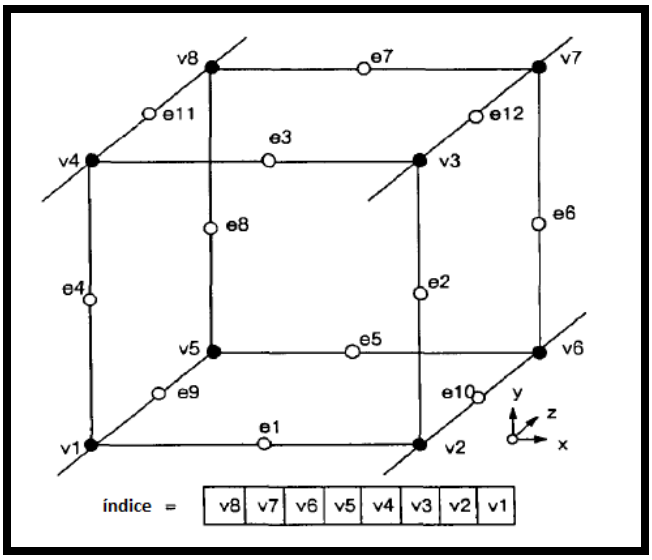

Figura 2.6: Índices del cubo.

Luego, para cada arista interceptada, se calcula el punto exacto por donde pasa la superficie haciendo uso de la interpolación lineal entre las intensidades de los colores y la posición x, y, z de los vóxeles, ver **Figura 2.7** y **ecuaciones 2.4** y **2.5. [1]**

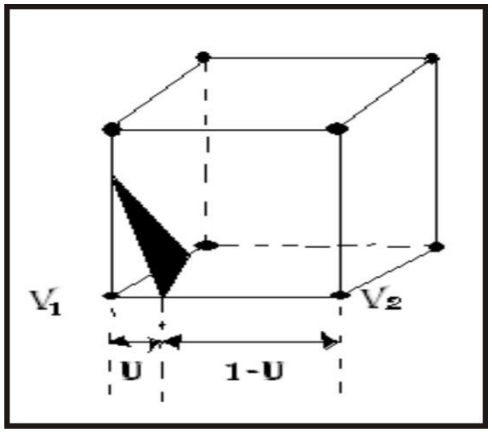

Figura 2.7: Interpolación lineal.

$$V_i = V_1 * (1 - U) + V_2 * U \qquad (2.4)$$

$$U = \frac{(\text{IsoValor} - V_1)}{(V_2 - V_1)} \qquad (2.5)$$

Donde V_i es la posición del vértice por donde la superficie corta a la arista, e **isoValor** es el valor de densidad de la superficie que el usuario quiere representar.

Para obtener una mejor calidad en la visualización de la superficie, se calcula el vector normal de cada uno de los vértices generados en las aristas del cubo, también por interpolación lineal como se muestra a continuación. **[1]**

$$\overrightarrow{N_i} = U * \overrightarrow{V_2} + (1 - U) * \overrightarrow{V_1} \qquad (2.6)$$

Donde $\overrightarrow{N_i}$ es la normal del nuevo punto calculado y $\overrightarrow{V_1}$ y $\overrightarrow{V_2}$ son las normales de los puntos del cubo V_1 y V_2 respectivamente.

Cada uno de los vértices calculados es almacenado en un arreglo que contiene 12 posiciones, donde cada posición corresponde a las 12 aristas del cubo. En el caso de las aristas que no son cortadas, en la posición del arreglo correspondiente no se almacena nada.

Luego, para formar los triángulos en el cubo, se cuenta con una tabla de triángulos **[6]** donde se guardan los índices de los vértices que formarán parte de estos triángulos. Estos índices se encuentran en forma consecutiva para poder ser accedidos correctamente a través del índice que indica el estado del cubo y, según este valor y los almacenados en la tabla, se busca en el arreglo los vértices correspondientes.

2.3.3. Optimización del algoritmo.

Una de las desventajas que posee este algoritmo es que, para grandes volúmenes de datos, se generan gran cantidad de vértices. Estos vértices en muchos casos están repetidos por lo que se hace uso de memoria innecesaria. Para darle solución a esta desventaja se hace uso de una estructura de datos nombrada MCTable.

Otra de las desventajas que posee dicho algoritmo es que, para grandes volúmenes de datos, el tiempo de generación de la superficie, así como el proceso de render de la misma es considerable. Para darle solución a esta problemática se hace uso de la tarjeta grafica.

2.3.3.1. MCTable.

Para garantizar que cada vértice generado a partir de la intercepción de la superficie con las aristas sea adicionado una sola vez en la lista de vértices, se creó una estructura de datos que está compuesta por un arreglo unidimensional donde se van a guardar la posición de los diferentes vértices en la Lista de vértices.

La idea de esta estructura de datos es asignar un índice a cada vértice a partir de sus coordenadas ***x***, ***y***, ***z*** el cual se define con la siguiente ecuación:

$$índice = x + y * ancho + z * alto * profundidad \qquad (2.7)$$

Donde **x, y, z** son las coordenadas del vértice dentro del grid y ***ancho***, ***alto*** y ***profundidad,*** las dimensiones de la rejilla tridimensional. Con esta estructura se garantiza saber, en un tiempo constante con orden de complejidad ***O (1)***, si un vértice fue o no generado con anterioridad; en caso de que el mismo ya haya sido generado lo que se devuelve es la posición del vértice dentro de la lista de vértices.

Se usa esta estructura de datos pues es muy sencilla y tiene un gran rendimiento, aunque presenta la desventaja de que en ocasiones no se usa toda la memoria, por lo que haría falta hacer un estudio más profundo de este problema para optimizar esta estructura.

2.3.3.2. GPU.

Con el objetivo de garantizar que la geometría a generar y la interacción con la misma sean en tiempo real, se hizo una implementación del algoritmo usando el GPU. Esta implementación se

realizó sobre el lenguaje **GLSL** (OpenGL Shading Language), un lenguaje de alto nivel basado en la programación del **lenguaje C**. Creado por **OpenGL ARB** (OpenGL Architecture Review Board) para darle a los desarrolladores un control más directo del pipeline gráfico sin tener que utilizar el lenguaje ensamblador **[10, 11]**. También se hace uso del Geometry Shader, el cual proporciona una vía muy sencilla y eficiente de procesar los datos y generar la superficie deseada.

La idea fundamental de esta implementación es convertir los datos volumétricos y las tablas de aristas y triángulos usadas para la creación, de forma rápida, de los triángulos en cada uno de los cubos en texturas 3D y 2D respectivamente y enviarlas a la tarjeta gráfica. Luego, en el Geometry Shader especificándole como primitiva de entradas, puntos; se pueden generar nuevas primitivas, como triángulos, a partir de los vértices enviados. Estos vértices que se envían a la tarjeta representan el conjunto de cubos a procesar en el Geometry Shader y son enviados haciendo uso del **Vertex Buffer Object** (VBO), un nuevo método usado para describir primitivas geométricas que hace posible a las aplicaciones de OpenGL obtener beneficios en cuanto al rendimiento y flexibilidad **[32].**

Capítulo 3: Análisis y Diseño del sistema.

3.1. Reglas del Negocio.

Las imágenes que se deseen visualizar deben estar en el formato *.dcm o en el *.raw pues el módulo implementado solo visualiza estos tipos de archivos.

Las imágenes con el formato *.dcm deben tener la estructura original pues hay algunas versiones de este formato que no cumplen con este requisito por lo que no podrán ser visualizados de forma correcta.

Las imágenes con el formato *.raw deben tener las dimensiones del estudio sin modificaciones ya que esto traería consigo una mala visualización.

3.2 Modelo del Dominio.

En la **Figura 3.1** se describe el negocio a partir de un modelo de dominio donde se pretende facilitar la comprensión del funcionamiento del mismo.

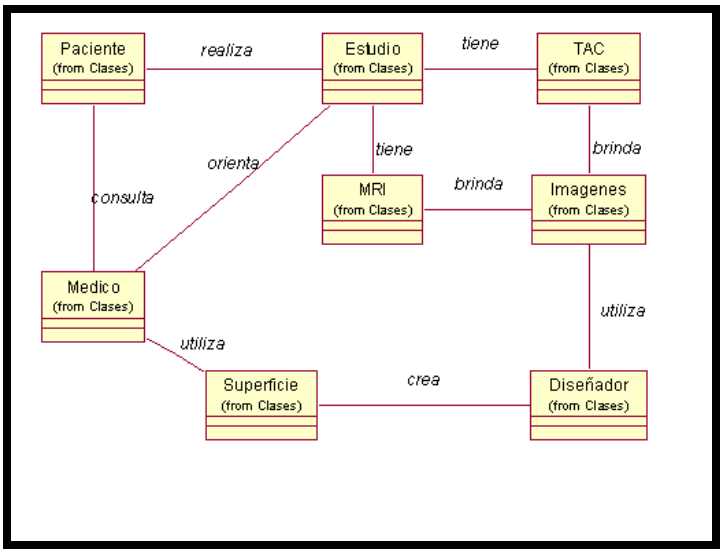

Figura 3.1: Modelo de Objetos.

3.3. Descripción del Negocio.

El médico le orienta al paciente un estudio que puede ser una Tomografía Axial Computarizada (TAC), una Imagen de Resonancia Magnética (MRI) o ambas; los cuales brindan imágenes que son aprovechadas por el diseñador para modelar una superficie que le sirva al médico para dar un diagnóstico de los padecimientos del paciente.

3.4. Captura de Requisitos.

Un requerimiento es una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, norma, especificación u otro documento formal. Estos facilitan el entendimiento entre usuarios y desarrolladores del sistema a elaborar. A continuación se exponen los requisitos funcionales por los que se regirá el sistema y los no funcionales que exponen las características de la aplicación.

3.4.1. Requisitos Funcionales.

Los requisitos funcionales son capacidades o funciones que el sistema debe cumplir. A continuación se presentan los requerimientos del módulo a desarrollar.

R1. Cargar archivos.

R1.1 Seleccionar los archivos a cargar.

R1.2 Cargar archivos DICOM (*.dcm).

R2. Visualizar Modelo.

R2.1 Seleccionar el isovalor.

R2.2 Reconstruir la superficie.

R2.3 Mostrar el modelo obtenido.

3.4.2. Requisitos No Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener y representan las características que hacen al producto atractivo, usable, rápido y confiable. Seguidamente se enumeran los requisitos no funcionales del módulo a desarrollar.

1. **De Software:**
   - Sistema Operativo Windows o cualquier distribución de Linux.
2. **De Hardware:**
   - Procesador Pentium 4 a 3.0 GHz.
   - Memoria RAM de 512 MB de capacidad.
   - Espacio en disco duro de 1 GB.
3. **De Seguridad:**
   - *Confidencialidad*: Los modelos 3D obtenidos en la visualización deben representar, lo más real posible, la anatomía humana.
   - *Integridad*: No debe haber pérdidas de información ni de calidad en las imágenes obtenidas.
   - *Usabilidad*: Los usuarios deben tener un conocimiento básico sobre la visualización tridimensional aplicada a la medicina.
4. **De Soporte:**
   - Soporte para los sistemas operativos Windows XP y versiones de Linux.
5. **De Restricciones en el diseño e implementación:**

- Se utilizará el lenguaje de programación C++ utilizando el paradigma orientado a objetos.

3.5. Modelo de Casos de Uso del Sistema.

En esta sección se mostrarán los actores del sistema, los casos de uso del sistema, así como su descripción para un mejor entendimiento de la aplicación.

3.5.1. Actores del Sistema.

Los actores de un sistema son agentes externos, roles que las personas (usuarios) o dispositivos juegan cuando interactúan con el software. En este caso particular quien hará uso del sistema será un especialista médico que, como actor del sistema, será llamado médico especialista. En la **Tabla 3.1** se justifica la selección de este actor.

Actores	Justificación
Médico Especialista	El médico especialista es quien interactúa con el sistema para ejecutar las funcionalidades de: Cargar imágenes DICOM perteneciente a un estudio realizado a un paciente y visualizar el modelo 3D obtenido para emitir un diagnóstico.

Tabla 3.1: Actor del Sistema.

3.5.2. Diagrama de Casos de Uso del Sistema.

El diagrama de casos de uso del sistema representa gráficamente los casos de uso y su interacción con los actores. En la **Figura 3.2** se muestra el diagrama de casos de uso del sistema para el módulo a implementar.

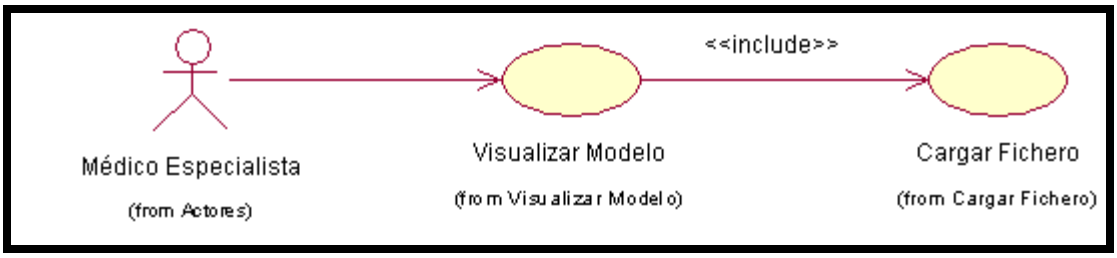

Figura 3.2: Diagrama de Casos de Uso.

3.5.3. Descripción de los Casos de Uso.

Cada caso de uso tiene una descripción de los pasos que ejecutará el sistema propuesto como respuesta a las acciones del usuario. La **Tabla 3.2** describe los pasos del caso de uso Visualizar Modelo. La descripción del caso de uso Cargar Archivos está expuesta en **[31].**

Caso de Uso:	Visualizar Modelo
Actor:	Médico Especialista
Propósito:	Generar un modelo 3D y visualizarlo en pantalla.
Resumen:	El caso de uso se inicia una vez que se haya ejecutado el CU Cargar imágenes, donde se utiliza el grid creado para generar el modelo 3D. Luego se visualiza el mismo.
Referencia:	R2.1, R2.2, R2.3, CU Cargar Archivos
CU asociados:	
Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción Visualizar.	**1.1.** Pide la dirección de las imágenes a visualizar.

2. Inserta la dirección de las imágenes.	**2.1.** Pide el valor del isovalor.
3. Entra el isovalor.	**3.1.** Selecciona del grid, los cubos que pertenecen a la superficie. **3.2.** Construye la superficie y la visualiza
4. El médico Especialista puede acercar o alejar y rotar el modelo.	**4.1.** El sistema muestra el modelo con la opción seleccionada aplicada.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones:	
Prioridad:	**Crítico**

Tabla 3.2: Descripción del CU Visualizar Modelo.

3.6. Diagrama de Clases del Análisis.

A continuación se muestra el diagrama de clases del análisis con el fin de esclarecer la comprensión global del sistema.

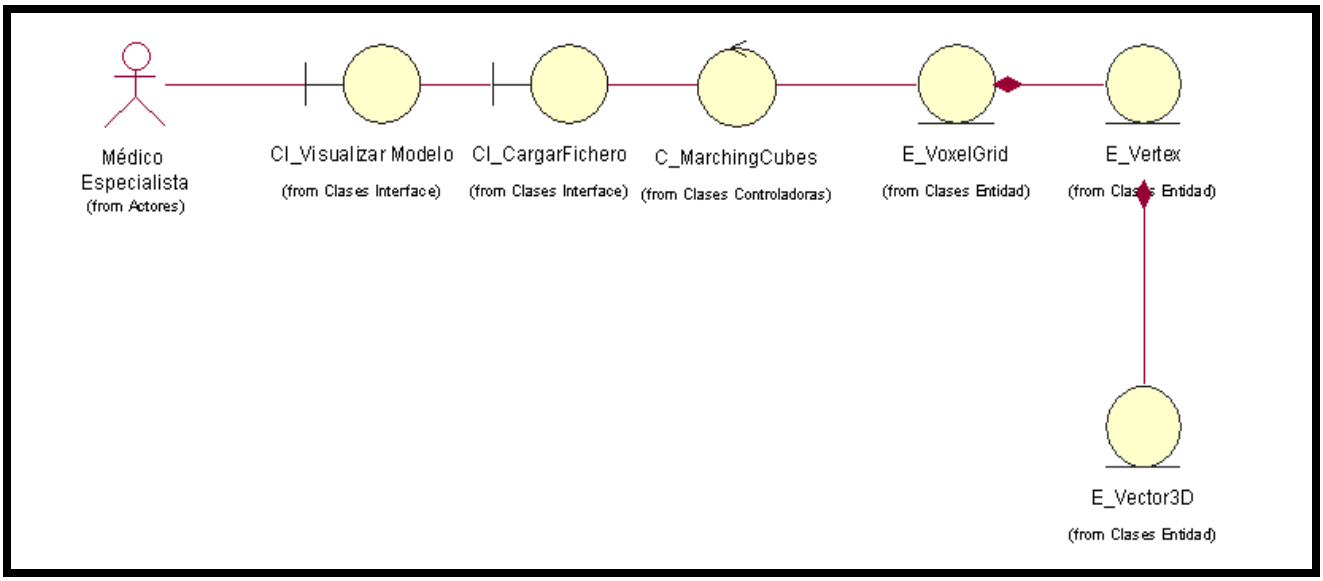

Figura 3.3: Diagrama de Clases del Análisis.

3.7. Diagrama de Secuencia.

Con el objetivo de aclarar la interacción entre los objetos de las clases del análisis se realiza el diagrama de secuencia del caso de uso Visualizar Modelo.

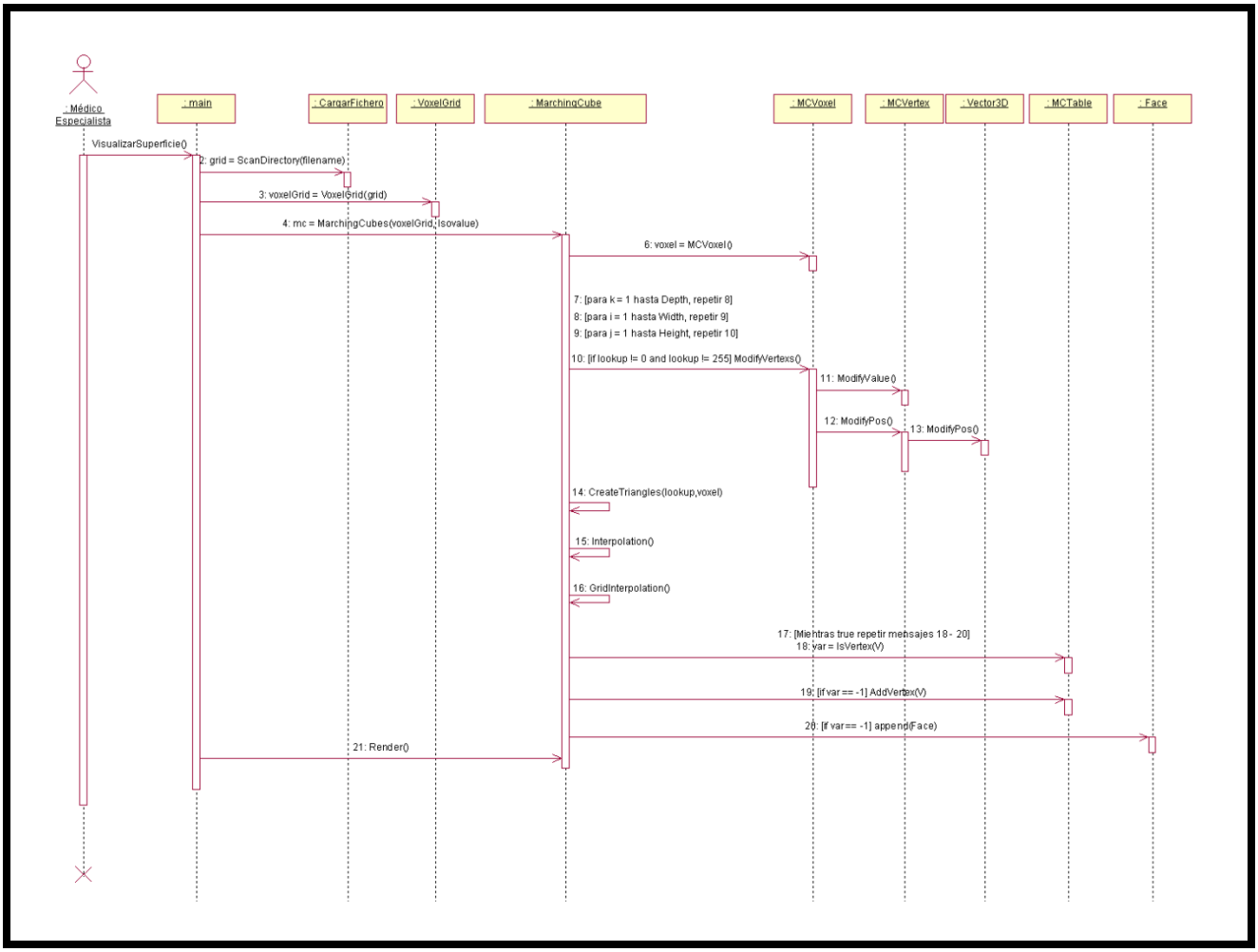

Figura 3.4: Diagrama de Secuencia del caso de uso Visualizar Modelo.

3.8. Diagrama de Clases del Diseño.

A continuación se muestra el diagrama de clases del diseño donde se pueden observar todas las clases que estarán presentes en el módulo.

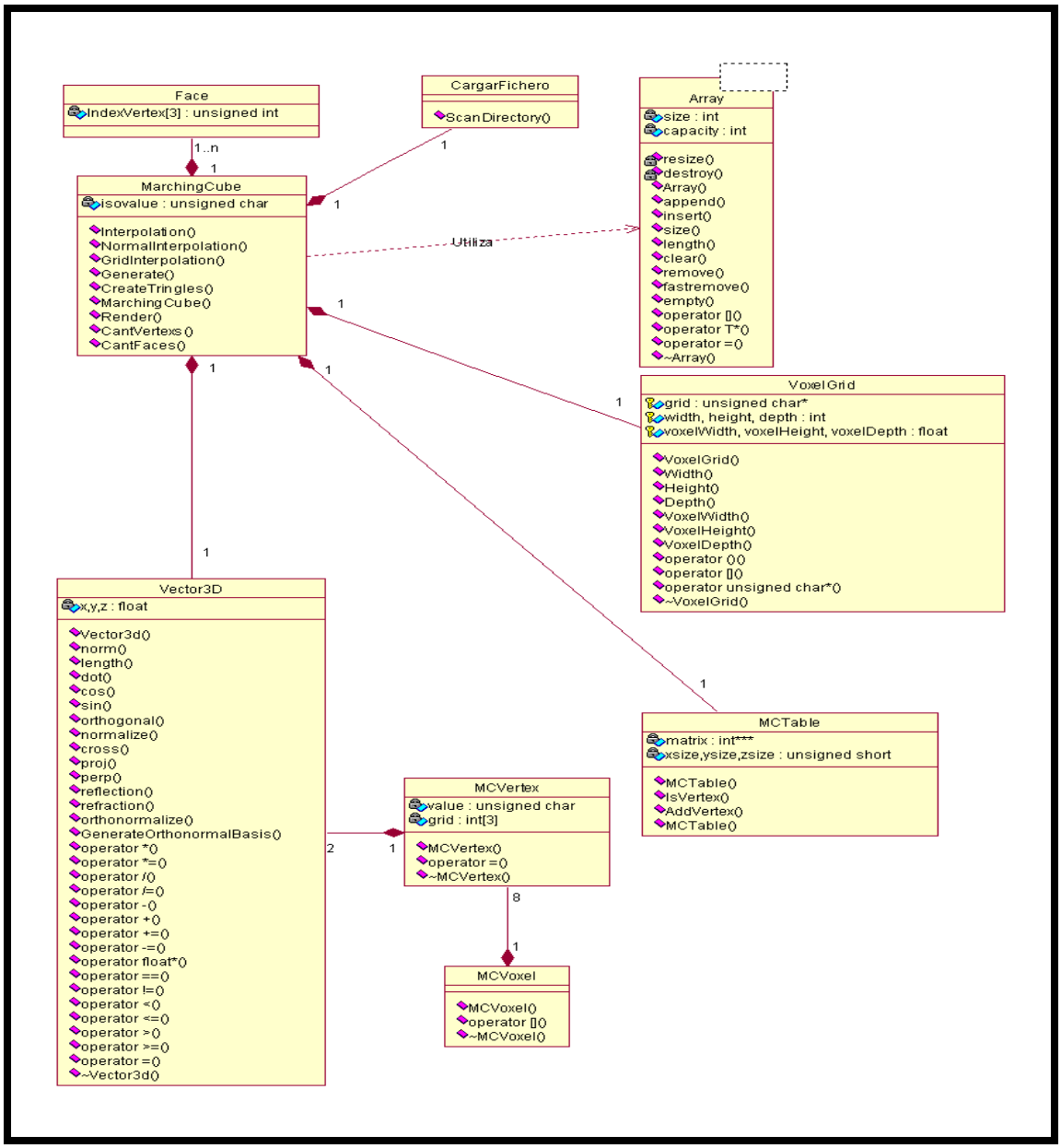

Figura 3.5: Diagrama de Clases del Diseño.

Capítulo 4: Implementación y Resultados.

En este capítulo se abordan los temas de la implementación del módulo basados en todo el trabajo acumulado a lo largo de los capítulos anteriores y, además, se hará un análisis de los resultados obtenidos en cuanto a rendimiento, cumplimiento de los objetivos propuestos, comparaciones entre los dos módulos implementados, entre otras cosas.

4.1. Diagrama de Componentes.

Las clases resultantes del análisis y el diseño se hacen físicamente mediante componentes. A continuación se muestra la relación que hay entre estos componentes.

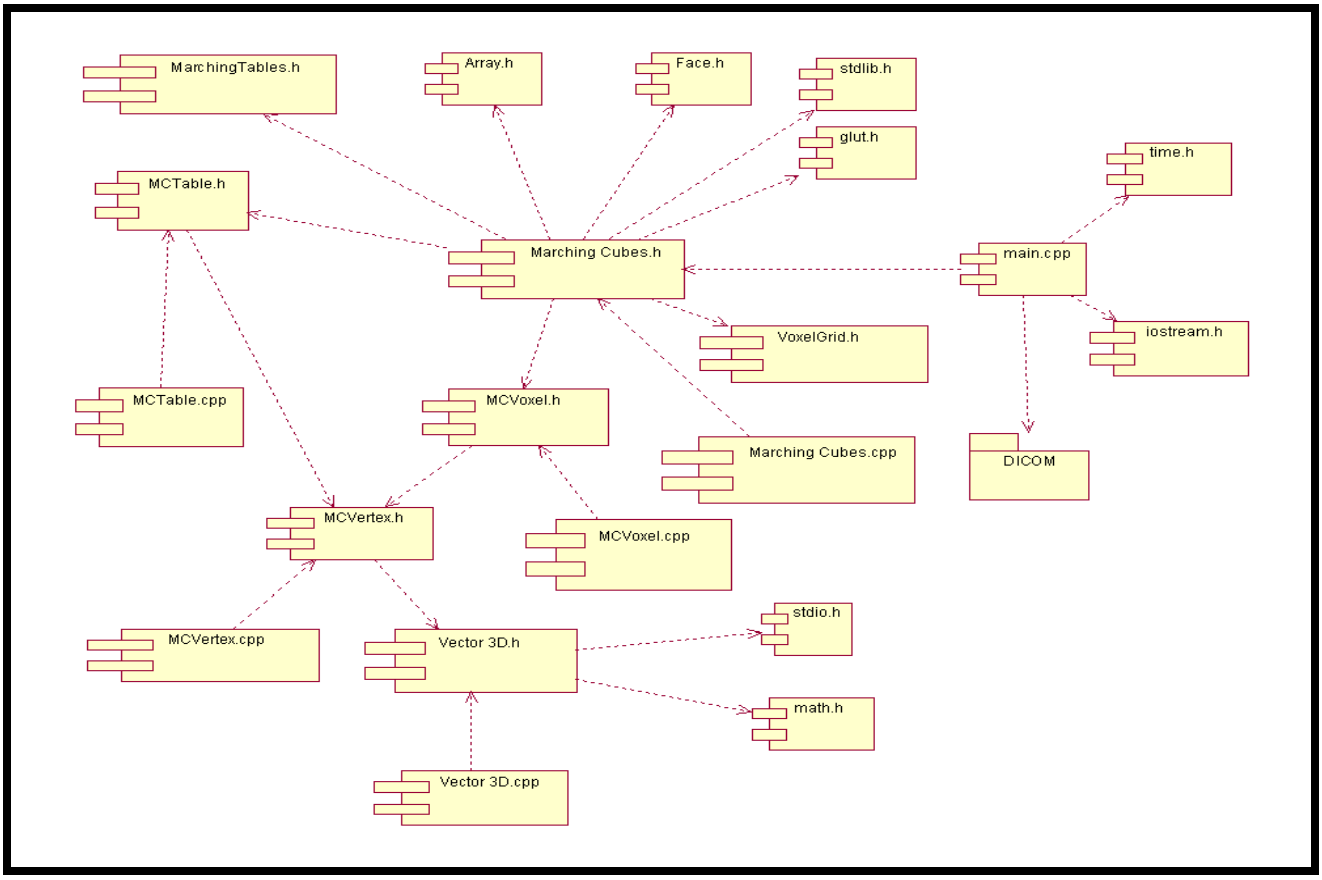

Figura 4.1: Diagrama de Componentes.

4.2. Valoración de los Resultados.

En este epígrafe se hará una valoración de los resultados obtenidos. Para esto se escogieron algunos estudios con el fin de mostrar los resultados. Se muestra una comparación entre el módulo que existe actualmente en Vismedic y el módulo de reconstrucción que se propone en este trabajo.

4.2.1. Datos de entrada.

Para realizar las pruebas, se tomaron como datos de origen las imágenes de los estudios referidos a la **Tabla 4.1**. La tabla está compuesta por tres campos de los cuales el primero representa el número o ID del Caso de Estudio, la segunda columna corresponde a la dimensión del estudio y la tercera al isovalor con el que se va a reconstruir la superficie.

ID del Caso	Dimensiones (X * Y * Z)	Isovalor
1	512 * 512 * 1024	50
2	512 * 512 * 46	50
3	256 * 256 * 107	50

Tabla 4.1: Datos de Entrada.

4.2.2. Parámetros a Medir.

Para realizar la comparación entre el módulo de reconstrucción actual de Vismedic y el propuesto en este trabajo se tendrán en cuenta dos criterios para cada uno de los casos de prueba mencionados en la **Tabla 4.1**. Estos criterios fueron escogidos teniendo en cuenta que los sistemas de reconstrucción 3D, justifican la demora en tiempo de procesamiento y

el alto consumo de recursos computacionales, con los elevados volúmenes de datos que se manipulan en el proceso. La primera columna representa el Id del caso de prueba que se va a usar, la segunda es para el tiempo de compilación del módulo actual, la tercera para el consumo de memoria del módulo actual y la cuarta y la 5ta van a representar los tiempos de compilación y el uso de memoria del módulo que se propone.

ID del Caso	Tiempo de compilación del módulo actual (s)	Consumo de memoria del módulo actual (Mb)	Tiempo de compilación del módulo propuesto (s)	Consumo de memoria del módulo propuesto (Mb)
1	1903	480	159.50	201.45
2	16	162.80	2.60	71.80
3	62	302.60	8.22	109.66

Tabla 4.2: Datos de la comparación entre los dos módulos.

La **Tabla 4.2** evidencia que el módulo propuesto para estudios relativamente iguales disminuye en todos los casos el tiempo de compilación y el consumo de memoria en más de un 50% por lo que aparte de mejorar la calidad geométrica de la malla obtenida en la reconstrucción, también disminuye considerablemente estos parámetros importantes.

Conclusiones

Con la investigación realizada se logró erradicar el problema de falta de calidad y realismo en las mallas obtenidas por el módulo que se estaba utilizando en el proyecto VISMEDIC.

De los algoritmos para reconstruir superficies existentes, se seleccionó el Marching Cubes; y en la solución propuesta se modificó la implementación original del mismo con el objetivo de disminuir el uso excesivo de memoria y la demora en la reconstrucción. También se brindan dos posibles variantes de implementación; la primera utilizando el CPU de la computadora y la segunda aprovechando las potencialidades de paralelización que brinda el GPU.

Esta investigación evidenció que el proceso de reconstrucción 3D es un proceso que involucra un amplio conocimiento en varias ramas científicas, elemento que fundamentan las futuras investigaciones que darán continuidad a este trabajo de diploma.

Recomendaciones.

Mostrar una superficie lo más real posible, utilizando la menor cantidad de recursos de la computadora y con un tiempo de reconstrucción breve es una meta bastante ambiciosa por lo que se deben tener en cuenta las siguientes recomendaciones:

1. Investigar cómo reducir la dimensión de la MCTable usada en este trabajo para disminuir la lista de vértices y así la memoria necesaria.
2. Investigar cómo optimizar el proceso de reconstrucción mediante el empleo de técnicas de programación GPGPU.
3. Exportar la geometría generada en un fichero de extensión estándar con el objetivo de que los modelos obtenidos puedan ser importados por un software de diseño asistidos por computadoras (CAD).

Bibliografía

1. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* **Lorensen, Willianm E. and Cline, Harvey E.** Nueva York, USA : s.n., 1987.

2. *The asymptotic decider: resolving the ambiguity in marching cubes.* **Nielson, Gregory M and Hamann, Bernd.** Arizona, USA : s.n., 1991.

3. *On Generating Topologically Correct Isosurfaces from Uniform Samples.* **Natarajan, B. K.** Michigan, USA : s.n., 1991.

4. *Marching Cubes: Construction of Topologically Correct Isosurface.* **Chernyaev, Evgeni V.** Russia : s.n., 1995.

5. [5]. *Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing.* **Lopes, Adriano and Brodlie, Ken.** USA : s.n., 2003.

6. **Bouke, Paul.** *Polygonising a scalar field.* [Online] Mayo 1994. http://local.wasp.uwa.edu.au/~pbourke/geometry/polygonise/.

7. *Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data.* **Chang Ho, Chien, et al.** Taiwan : s.n., 2005.

8. *A Marching Voxels Method for Surface Rendering of Volume Data.* **Feng Lin, Chin, Lin Yang, Don and Ching Chung, Yeh.** Taiwan : s.n., 2001.

9. *Two Algorithms for the Three-Dimensional Construction of Tomograms.* **Cline, H., et al.** New York. USA : s.n., 1998.

10. **Kessenich, John, Baldwin, Dave and Rost, Randi J.** *The OpenGL Shading Language v1.20.* USA : Addison Blvd., 2004. 0- 321- 33489-2.

11. **Rost, Randi J.** *The OpenGL Shading Language, Second Edition.* USA : Addison Wesley Professional, 2006. 0-321-33489-2.

12. **Yongming, Xie.** Geometry Shader Tutorials. *Geometry Shader Tutorials.* [Online] 2006. http://appsrv.cse.cuhk.edu.hk/~ymxie/Geometry_Shader/.

13. *The Visualisation Toolkit.* **Schroeder, W., Martin, K. and Lorensen, B.** Kitware : s.n., 2001.

14. *Hardware-accelerated Reconstruction of Polygonal Isosurface Representations on Unstructured Grids.* **Klein, Thomas, Stegmaier, Simon and Ertl, Thomas.** Alemania : s.n., 2004.

15. *GPU Construction and Transparent Rendering of Iso-Surfaces.* **Kipfer, Peter and Westermann, Rudiger.** Alemania : s.n., 2005.

16. *Isosurface Computation Made Simple: Hardware Acceleration, Adaptive Refinement and Tetrahedral Stripping.* **Pascucci, V.** USA : s.n., 2004.

17. *Realtime Isosurface Extraction with Graphics Hardware.* **Reck, Frank, et al.** Alemania : s.n., 2004.

18. *Real-Time Marching Cubes on the Vertex Shader.* **Goetz, Frank, Junklewitz, Theodor and Domik, Gitta.** Alemania : s.n., 2005.

19. *Accelerating Marching Cubes with Graphics Hardware.* **Johansson, Gunnar and Carr, Hamish.** Suecia : s.n., 2006.

20. **Crassin, Cyril.** OpenGL Geometry Shader Marching Cubes. *OpenGL Geometry Shader Marching Cubes.* [Online] 2007. http://www.icare3d.org/blog_techno/gpu/opengl_geometry_shader_marching_cubes.html.

21. *High-speed Marching Cubes using Histogram Pyramids.* **Dyken, Christopher, et al.** Noruega : s.n., 2008.

22. Visualizing fields at points using glyphs. *Visualizing fields at points using glyphs.* [Online] 2009. http://www.cmiss.org/cmgui/wiki/VisualizingFieldsAtPointsUsingGlyphs.

23. *Sistemas de reconstrucción tridimensional (3D).* **Escobar, Viviana and Aristizabal, Elizabeth.**

24. Surface rendering techniques. *Surface rendering techniques. .* [Online] 2005. http://www.cc.gatech.edu/scivis/tutorial/linked/surfaceren.html.

25. *Estándar y Protocolo de Imágenes Medicas DICOM. – Universidad de Deusto.* **PAS, Grupo.** España : s.n., 2003.

26. *Segmentación de imágenes médicas por crecimiento de regiones con conocimiento adicional.* **del Fresno, Mariana and Vénere, Marcelo J.** Argentina : s.n., 2002.

27. *Digital Image Processing.* **Gonzalez, Rafael C. and Woods, Richard E.** Brazil : s.n., 2002.

28. *Current Methods in Medical Image Segmentation.* **Pham, Dzung L., Xu, Chenyang and Prince, Jerry L.** USA : s.n., 2000.

29. **Preim, Bernhard and Bartz, Dirk.** *Visualization in Medicine.* USA : Morgan Kaufinann, 2007.

30. *Real-Time Volume Graphics, .* **Engel, Klaus, et al.** USA : s.n., 2006.

31. *Reconstrucción tridimensional de modelos anatómicos a partir de imágenes medicas digitales.* **Pereira, Osvaldo and Kindelan, Rolando.** Habana, Cuba : s.n., 2008.

32. *Vertex Buffer Object.* **Riccio, Chistrophe.** USA : s.n., 2006.

Anexos

Anexo A: A continuación se muestran una serie de imágenes obtenidas mediante el módulo desarrollado.

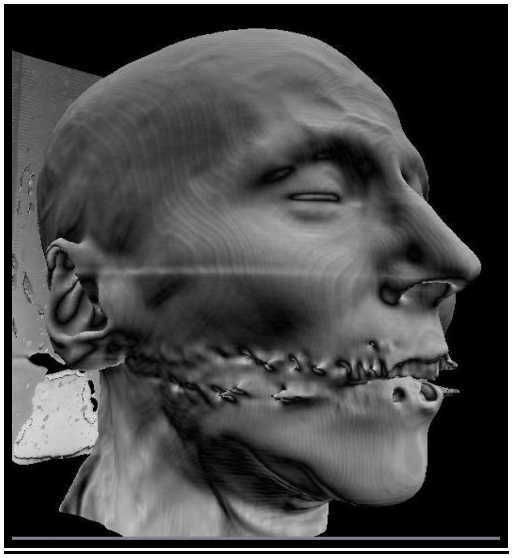

Figura A.1: Reconstrucción de una cabeza utilizando la implementación en el GPU. Con un isovalor de 50.

Figura A.2: Reconstrucción de una cabeza utilizando la implementación en el GPU. Con un isovalor de 50.

Figura A.3: Reconstrucción de una cabeza utilizando la implementación en el CPU. Con un isovalor de 100.

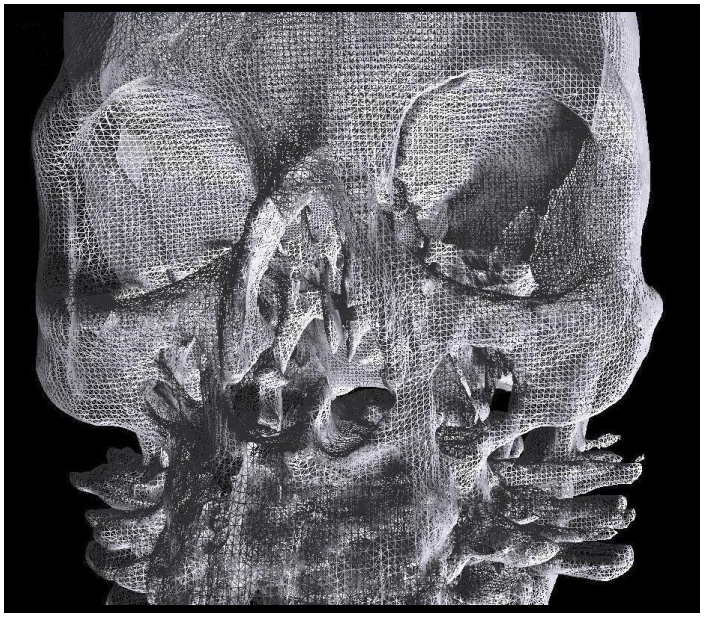

Figura A.3: Visualización de la malla generada a partir de la reconstrucción de una cabeza utilizando la implementación en el CPU. Con un isovalor de 100.

Glosario

A

Algoritmo: Es una lista que, dado un estado inicial y una entrada, propone pasos sucesivos para arribar a un estado final obteniendo una solución.

B

Bidimensional: adj. Que tiene dos dimensiones, altura y anchura.

C

Contorno: Conjunto de las líneas que limitan una figura o composición. Ver frontera.

CPU: Unidad de procesamiento central.

D

Diagnóstico: Etimológicamente el concepto diagnóstico proviene del griego, tiene dos raíces, día- que es a través de, por. Y gignoskein que es conocer, así etimológicamente diagnóstico significa conocer a través de. El concepto de este significado (imagen que representamos en la mente) es la identificación de la naturaleza o esencia de una situación o problema y de la causa posible o probable del mismo, es el análisis de la naturaleza de algo.

Digital: Quiere decir que utiliza o que contiene información convertida al código binario, el lenguaje de números (ceros y unos) que emplean los ordenadores para almacenar y manipular los datos.

E

Estándar: adj. Se dice de lo que sirve como tipo, modelo, norma, patrón o referencia por ser corriente.

F

Fragment Shader: Programa que calcula el color de los píxeles individuales. Controla cómo las texturas son aplicadas a los fragmentos.

Frontera: Sinónimo de borde, límite.

G

GPU: Unidad de procesamiento central.

Gradiente: Denota una dirección en el espacio según la cual se aprecia una variación de una determinada propiedad o magnitud física.

Grid: Estructura para almacenar la información de las imágenes médicas contenidas en los estudios.

I

Imagen: Figura, representación, semejanza y apariencia de algo.

Imagenología: Comprende la realización de todo tipo de exámenes diagnósticos y terapéuticos en los cuales se utilizan equipos que reproducen imágenes del organismo. Los siete servicios de Imagenología son Ecotomografía, Imagenología Mamaria, Medicina Nuclear, Radiología, Rayos Infantil, Resonancia Magnética y Tomografía Computada o Scanner.

Interactividad: Cualidad de interactivo. Que permite una interacción, a modo de diálogo, entre el ordenador y el usuario.

Interpolación: Algoritmo matemático que a partir de varios puntos en el espacio, describe una función que contiene a los puntos intermedios.

M

Malla: Forma de representar un modelo a partir de polígonos. Colección de vértices, aristas y polígonos conectados de forma que cada arista es compartida como máximo por dos polígonos.

Metodología: Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal.

Modelo 3D: Modelo tridimensional, real o virtual para representar un cuerpo o una parte del mismo.

Módulo: Pieza o conjunto unitario de piezas que se repiten en una construcción de cualquier tipo, para hacerla más fácil, regular y económica.

P

Paralelismo: Es una forma de computación en la cual varios cálculos pueden realizarse simultáneamente, basado en el principio de dividir los problemas grandes para obtener varios problemas pequeños, que son posteriormente solucionados en paralelo.

Pase de render: Es el conjunto de datos que pasan completamente a través del pipeline.

Pipeline: División de un proceso en etapas, asignando a cada una de ellas distintos recursos.

Píxel: Abreviatura de “picture element”. Es la mínima unidad de información dentro de una imagen bidimensional.

Primitiva geométrica: Formas geométricas consideradas primitivas por su básica constitución en las partes que la conforman, que pueden ser combinadas para crear formas geométricas más complejas.

Polígono: Porción de plano limitada por líneas rectas.

R

RAM: Es la memoria de acceso aleatorio (en inglés: *random-access memory*), desde donde el procesador recibe las instrucciones y guarda los resultados. Es el área de trabajo para la mayor parte del software de un computador.

Realidad Virtual: Simulación generada por computadora de imágenes o ambientes tridimensionales interactivos con cierto grado de realismo físico o visual.

Reconstrucción 3D: Es el proceso mediante el cual objetos reales son reproducidos en la memoria de un computador, manteniendo sus características físicas (dimensiones, volumen y forma).

Render: Es el proceso de convertir modelos geométricos 3D en escenas 2D para presentarlas a los usuarios.

S

Segmentación: Se utiliza en el Procesamiento de Imágenes para el reconocimiento de objetos o estructuras de interés en la imagen.

Shader: conjunto de instrucciones capaces de ser ejecutadas por un procesador gráfico.

Superficie: Parte externa de un cuerpo que sirve de delimitación con el exterior.

T

Tetraedro: Sólido determinado por cuatro planos o caras.

Textura: son imágenes que pueden ser mapeadas en cualquiera de las primitivas gráficas para adicionar detalles a la escena.

Tomografía Axial Computarizada (TAC): Es un examen médico no invasivo ni doloroso que ayuda al médico a diagnosticar y tratar enfermedades. Las imágenes por TAC utilizan un equipo de rayos X especial para producir múltiples imágenes o visualizaciones del interior del cuerpo, a la vez que utiliza conjuntamente una computadora que permite obtener imágenes transversales del área en estudio. Luego, las imágenes pueden imprimirse o examinarse en un monitor de computadora.

Topología: Es el método matemático – lógico usado para definir las relaciones espaciales entre los objetos espaciales. Hace referencia a las propiedades de vecindad o adyacencia, inclusión, conectividad y orden, es decir, propiedades no métricas.

Triángulo: Polígono de tres lados. Se representa en este trabajo por 3 vértices.

Tridimensional: adj. Se dice de lo que se desarrolla en las tres dimensiones espaciales, altura, anchura y profundidad.

U

Unidimensional: Término utilizado para describir figuras que sólo se pueden medir en una dirección, o sea, que poseen una sola dimensión, como una línea, que sólo tiene longitud.

V

Vector Normal: Vector cuyos puntos están en dirección perpendicular a una superficie.

Vertex shader: Es una función de procesado gráfico que manipula los valores de un vértice en un plano 3D mediante operaciones matemáticas sobre un objeto. Estas variaciones pueden ser

diferencias en el color, en las coordenadas de la textura, en la orientación en el espacio o en el tamaño del punto. Permite el control de las transformaciones sobre los vértices.

Vértices: son puntos en el espacio 3D que definen primitivas gráficas tales como triángulos, polígonos y rectángulos, usados para construir la geometría de la escena que será dibujada.

Viewport: Rectángulo definido en el sistema de referencia de la pantalla, cuyo objetivo es seleccionar que área del mundo se desea ver en un sub-área de la pantalla.

Vóxel: (del inglés: *volumetric pixel*) es la unidad cúbica que compone un objeto tridimensional. Constituye la unidad mínima procesable de una matriz tridimensional y es, por tanto, el equivalente del píxel en un objeto 2D.