UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE COMPUTACIÓN GRÁFICA

TETRAEDRIZACIÓN DE INTERVALOS DE VOLUMEN MEDIANTE MODIFICACIÓN DE CUBOS MARCHANTES

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela por el bachiller
Jorge Luis Bernadas Saragoza
para optar al título de Licenciado en Computación

Tutor
Prof. Rhadamés Carmona

Caracas, Abril 2009

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE COMPUTACIÓN GRÁFICA

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designados por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado, presentado por el Bachiller Jorge Bernadas, portador de la cédula de identidad 17.429.153, con el título "**Tetraedrización de Intervalos de Volumen mediante modificación de Cubos Marchantes**", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue dicho trabajo por cada uno de los Miembros del Jurado, se fijó el día 30 de abril de 2009, a las 9:00 AM, para que su autor lo defienda en forma pública, en el Centro de Computación Gráfica, mediante la exposición oral de su contenido, y luego de la cual respondieron satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas a los 30 días del mes de abril del año dos mil nueve, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Rhadamés Carmona.

_____
Prof. Rhadamés Carmona
(Tutor)

_____
Prof. Ernesto Coto
(Jurado Principal)

_____
Prof. Brígida Molina
(Jurado Principal)

AGRADECIMIENTOS

Le agradezco a mi mamá Gloria que me haya cuidado desde el cielo todo este tiempo, y que me haya guiado para tomar siempre las decisiones correctas.

Le agradezco a mi papá Jorge que desde pequeño me inculcó el valor de hacer las cosas bien hechas, sin mediocridades, así como los conocimientos básicos necesarios para ser exitoso el día de hoy. Papá, tú me enseñaste que el cero es el primer entero con el que se cuenta, y no el uno como todos creen. Además, se siente bien llegar a primer grado sabiendo leer, escribir, sumar, restar y multiplicar cuando los demás sólo sabían jugar con la plastilina.

Le agradezco a mi hermano Manuel su compañía y su cariño (pelaje) desde que nació. Yo no me imagino cómo hubiera sido mi vida como hijo único, no tendría a quién enseñarle mis logros en los juegos de computadora, en TopCoder®, y otras cosas que por más simples que parezcan son las que más importan.

Le agradezco a mi tía Rosario por cuidarnos a mi hermano y a mí desde que nacimos, y por siempre estar pendiente de todo lo que necesitamos, especialmente de mi salud, ya que yo generalmente no reparo en eso. Tía, tú me enseñaste a utilizar la computadora a los 3 años (todavía me acuerdo del Donkey Kong y me da nostalgia), así como el Nintendo y el Intellivision, eso hizo que la computadora y los juegos de video siempre me llamaran la atención.

Les agradezco a mi abuela Rosario y a mi abuelo Jorge el haberme consentido desde siempre, y que hayan tenido la dedicación de tramitar mi nacionalidad española y el pasaporte respectivo, lo cual me ha permitido conocer Estados Unidos, Canadá, Japón y Brasil y participar en maratones de programación a nivel mundial. También les agradezco que me hayan regalado mi primera computadora portátil hace cuatro años, gracias a ella logré hacer mis labores de la universidad y practicar constantemente para los maratones de programación.

Le agradezco a Ana habernos cuidado desde pequeños, y que haya sabido darnos a mi hermano y a mí la atención que hemos necesitado durante todos estos años. Ana, recuerda que está pendiente que me enseñes a preparar pasticho, asado negro, pernil, pavo, hallacas, entre otras delicateses que hemos podido degustar estando a tú lado.

Le agradezco infinitamente (`DBL_INF`) a mi novia Fiorella por brindarme su constante apoyo, cariño y amor desde que la conozco, especialmente durante el desarrollo de este trabajo de grado, desde su inicio hasta su culminación. Sin ella no hubiera podido terminar este trabajo, gracias por todos los sandwichitos y yogurts con cereal, seguro te van a extrañar en el centro. Sin ella no iría a doctores y pasaría todo el día haciendo problemas en TopCoder®. Sin ti, mi amor, no tuviera ni la mitad de la felicidad que tengo ahorita.

Gracias a los profesores Rhadamés Carmona y Ernesto Coto por guiarme durante el desarrollo de este trabajo, y por hacer todas las correcciones durante el mismo. Aprendí muchas cosas que no sabía sobre la realización de un trabajo de grado y la presentación gracias a ustedes.

Gracias al Centro de Computación Gráfica y a todos sus integrantes por darme su apoyo durante los últimos tres años de mi carrera. Sin ese apoyo, probablemente hubiera desarrollado una página web estática o una simple aplicación empresarial como tesis de grado, en lugar de un trabajo más profesional y científico.

Gracias a Carlos Guía por ser buen amigo y por decirme hace cuatro años que debía entrenar seriamente para los maratones. En verdad, con los maratones de programación he aprendido muchos más tópicos interesantes que a lo largo de la carrera, y esto es lo que realmente me diferencia de todos los demás graduandos.

Los agradecimientos a los amigos es una de las partes más difíciles de escribir, sencillamente porque podría duplicar el tamaño del documento si escribiera algo para cada uno. Además, algo que he aprendido de pensar matemáticamente es que definir los conjuntos por compresión es más fácil y menos propenso a errores que definirlos por extensión. Así que, para todo x_i en mi conjunto A de amigos, le agradezco a x_i que siempre me haya apoyado durante toda la carrera y que siempre esté ahí, tanto para las buenas como para las malas.

<sarcasm>
Muchas gracias a control de estudios por mover la fecha tope de presentación a un mes antes de lo previsto. De verdad que eso motiva saludablemente a los futuros graduandos.
</sarcasm>

Disclaimer: The only feelings I had while the development and writing of this thesis were frustration, regret and disappointment. I should have taken something simpler and get my degree `long long time ago;`

Outro: I still do not understand the fuss about finishing the thesis, the supposed happiness feeling that I should be feeling now. I think that winning the SRM 407 is going to be a lot more exciting than finally presenting this work. I just want to finish this and dedicate my time to more important stuff, like my family, my beautiful and lovable girlfriend and programming contests. Here is a short list of the things I need to do:

- Get married with Fiore and raise a lot of children with her.
- Get in contact with my friends again (this thesis made me a loner).
- Go to doctors to see if the thesis left some permanent physical damage.
- Lower my weight by at least 20Kg. and do some not computer-related exercises.
- Learn how to drive, how to cook and first aids.
- Learn more algorithms for programming contests.
- Raise my TopCoder® Algorithm Rating to **2800** again.
- Other things I cannot remember right now.

Sorry for the bad puns.
– Because I can.
jbernadas

RESUMEN

TÍTULO:
Tetraedrización de Intervalos de Volumen mediante modificación de Cubos Marchantes.

AUTOR:
Jorge Bernadas.

TUTOR:
Prof. Rhadamés Carmona.

Los dos métodos principales para la visualización de datos volumétricos consisten en la visualización directa de volúmenes y la extracción de isosuperficies. El primer método ofrece la ventaja de mostrar todo el volumen en contexto, mientras que el segundo método sólo muestra una parte del volumen fuera de contexto. Sin embargo, la extracción de isosuperficies requiere menos poder de cómputo en comparación con la visualización directa de volumen.

Otro método para la visualización de datos volumétricos consiste en un híbrido entre ambas técnicas, el cual reconstruye y muestra subvolúmenes del volumen mediante el uso de mallados tetraédricos y técnicas de visualización directa de volúmenes para el despliegue. Adicionalmente, se puede almacenar el isovalor en cada vértice del mallado obtenido, lo que permite el uso de dichos mallados para realizar simulaciones mediante el uso de elementos finitos.

En la actualidad, los algoritmos existentes para la extracción de intervalos de volumen son complicados o generan una alta cantidad de primitivas. En este trabajo se propone un algoritmo basado en la adaptación de cubos marchantes para extraer intervalos de volumen en lugar de isosuperficies, el cual tiene un tiempo de respuesta inferior a los algoritmos existentes y genera un mallado final con una menor cantidad de primitivas sin necesidad de recurrir a algoritmos complejos durante la extracción del intervalo de cada celda del volumen.

PALABRAS CLAVE:
Datos Volumétricos, Isosuperficies, Intervalos de Volumen

TABLA DE CONTENIDOS

CAPÍTULO 1. INTRODUCCIÓN

En la actualidad, los científicos utilizan las herramientas computacionales para visualizar los datos con los que trabajan, para así poder realizar el análisis correspondiente. Una de las ramas de la visualización que ha tenido el mayor auge en los últimos años es la visualización de volúmenes, que consiste en el despliegue de uno o varios conjuntos de datos tridimensionales en la pantalla, de forma que el usuario pueda entenderlos e interpretarlos satisfactoriamente.

Todos los algoritmos existentes para la visualización de volúmenes están basados en una de dos técnicas generales: visualización directa de volumen o extracción de isosuperficies. La primera técnica considera todo el volumen a la vez, generando imágenes semitransparentes de alta calidad, lo cual requiere gran poder de procesamiento para un despliegue en tiempo real. La segunda técnica sólo reconstruye y muestra isosuperficies del volumen, lo cual permite desplegar los datos a una velocidad superior, pero éstos se muestran descontextualizados, lo cual podría ser un problema para el análisis por parte del usuario.

Otro enfoque consiste en un híbrido entre ambas técnicas, el cual reconstruye y muestra subvolúmenes del volumen mediante el uso de mallados tetraédricos y técnicas de visualización directa de volúmenes para el despliegue, lo cual permite visualizar la información requerida en contexto con un mejor tiempo de respuesta al compararlo con el despliegue del volumen completo. Adicionalmente, se puede almacenar el isovalor en cada vértice del mallado obtenido, lo que permite el uso de dichos mallados para realizar simulaciones mediante el uso de elementos finitos.

En la actualidad, los algoritmos existentes para la extracción de intervalos de volumen poseen una alta complejidad o generan una alta cantidad de primitivas. Por ejemplo, Guo propone un método basado en tetraedrizaciones de Delaunay [GUO95], el cual consta de dos pasos principales: primero, la obtención de una nube de puntos a partir del volumen de datos, la cual estará compuesta por las muestras que se encuentran dentro del intervalo $[\alpha, \beta]$ a extraer y aquellos puntos donde las isosuperficies S_α y S_β intersectan al volumen, donde:

$$S_\alpha = \{(x, y, z) | F(x, y, z) = \alpha\}$$
$$S_\beta = \{(x, y, z) | F(x, y, z) = \beta\}$$

El segundo paso del algoritmo consiste en la obtención de la triangulación de Delaunay de la nube de puntos extraída del volumen. Este algoritmo posee una alta complejidad, debido a que el procedimiento para calcular la triangulación de Delaunay en tres dimensiones es complicado de programar y es muy lento para la extracción de intervalos en volúmenes grandes. Además, el uso de tetraedrizaciones de Delaunay no proporciona ventaja alguna al momento de la extracción de los tetraedros del intervalo, debido a que éstos tienden a degenerarse en las celdas intersectadas por las isosuperficies S_α y S_β.

Otro algoritmo para la extracción de intervalos de volumen fue propuesto por Fuji, Maeda y Sato [FUJ95], que consiste en extraer el intervalo de cada celda mediante la intersección de dos intervalos infinitos $[\alpha, \infty)$ y $(\infty, \beta]$, los cuales se obtienen mediante el uso de una modificación de Cubos Marchantes que calcula poliedros en lugar de triángulos, lo cual implica la necesidad de intersectar poliedros durante el procesamiento de cada celda. Para la resolución de los casos ambiguos, se pueden utilizar los métodos propuestos por Nielson y Hamann [NIE91] y por Chernyaev [CHE95] al momento de generar la estructura de la tabla de casos.

Sin embargo, la complicación del algoritmo se encuentra al momento de realizar la intersección de los poliedros, debido a que éste debe ser lo suficientemente general como para intersectar conjuntos de poliedros no convexos, lo cual, al igual que el algoritmo anterior, incrementa el tiempo de ejecución considerablemente. Además, el algoritmo está diseñado para extraer un mallado triangular que encierre aquella parte del volumen dentro del intervalo $[\alpha, \beta]$, lo cual no permite una representación adecuada para las celdas internas del intervalo.

Una alternativa a procesar cada celda directamente, sea para la extracción de isosuperficies o intervalos de volumen, consiste en subdividir cada una de éstas en tetraedros, los cuales serán procesados independientemente [CAR95]. Este algoritmo, llamado Tetracubos Marchantes, permite simplificar el análisis de casos posibles de intersección, ya que un tetraedro consiste únicamente de cuatro (4) vértices y no presenta ambigüedades durante el procesamiento.

En 1997, Nielson propone otro algoritmo para la extracción de intervalos de volumen [NIE97b], el cual consiste en extender el algoritmo de los Tetracubos Marchantes para extraer el intervalo del volumen de cada uno de los tetraedros de las celdas mediante el uso de una tabla de $3^4 = 81$ casos, la cual indica los tetraedros a generar para cada uno de los casos posibles de

intersección entre el intervalo y un tetraedro del volumen. Las ventajas principales de este algoritmo propuesto por Nielson radican en la sencillez del mismo, la pequeña cantidad de casos en la tabla y la ausencia de posibles ambigüedades en el mismo, lo cual acelera la extracción del intervalo del volumen. Sin embargo, este algoritmo genera una salida compleja difícil de visualizar por la alta cantidad de primitivas generadas. Esto se debe a que cada celda del volumen debe ser dividida en cinco o más tetraedros y cada uno de éstos puede generar hasta seis primitivas de salida, por lo que en el peor caso se pueden generar hasta treinta tetraedros por cada celda del volumen.

Por todo lo mencionado anteriormente, se propone un algoritmo para la extracción de intervalos de volumen basado en cubos marchantes, el cual procese cada celda directamente mediante una tabla de conectividad, sin necesidad de dividir cada celda en tetraedros ni de recurrir a algoritmos complejos durante el procesamiento de cada celda. Debido a esto, se presume que el algoritmo propuesto generará menos tetraedros que el algoritmo propuesto por Nielson [NIE97b] y será más eficiente que los métodos propuestos por Guo [GUO95] y Fuji et al. [FUJ95].

Con el objetivo de analizar el desempeño del algoritmo propuesto, se implementará el algoritmo propuesto por Nielson [NIE97b] y se compararán los mallados obtenidos mediante las métricas desarrolladas por el grupo Verdict [STI07], las cuales permiten realizar análisis cuantitativos sobre mallados triangulares y tetraédricos. Finalmente, se implementarán las versiones correspondientes para la extracción de isosuperficies, lo cual permitirá confirmar los resultados obtenidos ya que se espera que ambos métodos tengan desempeños similares independientemente del tipo de mallado a extraer.

1.1. OBJETIVO GENERAL

Diseñar e implementar un algoritmo para la extracción de intervalos de volumen mediante la adaptación de Cubos Marchantes, utilizando una tabla de conectividad que indique los tetraedros a generar en cada caso posible.

1.2. OBJETIVOS ESPECÍFICOS

- Implementar Cubos Marchantes [LOR87] con resolución de ambigüedades [NIE91] [CHE95] para la extracción de isosuperficies.

- Implementar Tetracubos Marchantes [CAR95] para la extracción de isosuperficies.

- Implementar un algoritmo para tetraedrizar un poliedro convexo, para cada caso posible de intersección entre una celda y el intervalo, el cual será utilizado para generar la tabla de casos.

- Diseñar e implementar un algoritmo para la extracción del intervalo de volumen en base a la tabla de casos generada anteriormente.

- Diseñar e implementar una aplicación de prueba que permita la carga de un volumen, así como la extracción de isosuperficies e intervalos de volumen a partir del mismo.

- Evaluar resultados obtenidos en la extracción de isosuperficies y en la extracción de intervalos de volumen mediante el uso de las métricas diseñadas por el grupo Verdict [STI07].

CAPÍTULO 2. MARCO TEÓRICO

A continuación se presenta una visión general del proceso de visualización de datos volumétricos, así como una breve descripción de algunos algoritmos existentes para el extracción de isosuperficies y la extracción de intervalos de volumen, para terminar con la descripción de algunas métricas utilizadas para evaluar la calidad de mallados triangulares y tetraédricos.

2.1. VISUALIZACIÓN DE VOLÚMENES DE DATOS

Un volumen de datos se define como un conjunto de datos ubicados en $\Re^3$, los cuales generalmente están definidos sobre una malla cartesiana uniforme regular con uno o más atributos escalares y, posiblemente, uno o más atributos vectoriales en cada vértice de la malla. El proceso de visualización de volúmenes de datos es el conjunto de pasos llevados a cabo para proyectar un volumen de datos hacia un plano imagen bidimensional, con el propósito de entender la estructura del mismo [ELV92].

Los científicos utilizan las imágenes obtenidas para adquirir información y conocimiento de los datos producidos por experimentos, así como para compartir sus experiencias con otros científicos e instituciones. Para poder alcanzar este objetivo, las técnicas de visualización deben ofrecer una representación entendible de los datos, así como una rápida manipulación y despliegue de los mismos que permita una interacción satisfactoria con el usuario.

En la actualidad, la visualización de volúmenes se utiliza ampliamente en la medicina, astrofísica, química, microscopía, ingeniería mecánica, pruebas no destructivas y otras áreas de la ciencia y la ingeniería. Entre los datos que los científicos e ingenieros almacenan como volúmenes se encuentran densidad, presión, temperatura, carga electrostática, calor, velocidad, entre otros. Como se puede observar, los datos almacenados tienen características muy diferentes, por lo cual algunos métodos para la visualización de volúmenes proporcionan buenos resultados para ciertos tipos de datos pero no para otros.

A continuación se explicarán las características de los distintos volúmenes de datos existentes, la clasificación de las distintas técnicas para visualizarlos, así como los pasos más comunes en los algoritmos para llevar a cabo este proceso.

2.1.1. CARACTERÍSTICAS DE LOS VOLÚMENES DE DATOS

Los volúmenes de datos pueden ser clasificados en base a diferentes aspectos, entre los que se consideran: la fuente de donde han sido obtenidos, el tipo de dato que almacenan y si tratan el volumen como un conjunto de vóxeles o como un arreglo de celdas. A continuación se explican los aspectos que son tomados en cuenta en cada tipo de clasificación.

2.1.1.1. FUENTES DE DATOS VOLUMÉTRICOS

El primer paso necesario para estudiar determinados fenómenos basados en el análisis de volúmenes de datos consiste en la obtención de los mismos. Estos datos generalmente son obtenidos mediante el escaneo (*scanning*) del material de interés utilizando Imagenología por Resonancia Magnética (MRI), Tomografía Computarizada (CT), Tomografía por Emisión de Positrones (PET) y Ultrasonidos.

Otra fuente de datos frecuentemente usada por los científicos consiste en los resultados de simulaciones de experimentos reales[1], con el fin de contrastar sus resultados con aquellos obtenidos en el experimento realizado. Sin embargo, hay ocasiones en las cuales no se puede disponer de los datos del experimento real, debido a que éste puede ser muy grande, muy pequeño, muy rápido o muy lento como para ser observado, en cuyo caso sólo se puede estudiar el fenómeno mediante los datos obtenidos de la simulación.

Los datos volumétricos también pueden ser generados mediante la voxelización de objetos geométricos, el uso de herramientas de edición de volúmenes o de programas para la generación de volúmenes mediante métodos estocásticos. No obstante, los datos volumétricos pueden ser tratados de forma similar sin importar la fuente de la que hayan sido obtenidos [ELV92].

Generalmente, los datos a visualizar provienen de una única fuente, pero en ciertos casos es preferible obtenerlos de varias fuentes diferentes, debido a que la calidad e información aportada por estos depende de la forma en la que han sido capturados. Por ejemplo, para facilitar la detección del cáncer se utiliza la técnica *PET-CT Fusion*, la cual consiste en correlacionar los

[1] Estos experimentos generalmente se realizan mediante el uso de elementos finitos y dinámica de fluidos.

datos obtenidos de una tomografía con los datos obtenidos de un PET[2], el cual se especializa en detectar áreas con alta actividad metabólica. De esta forma, en la tomografía se pueden observar resaltadas aquellas partes donde el cáncer se ha esparcido, ya que la mayoría de los tejidos cancerosos presentan una alta actividad metabólica con respecto a los tejidos normales.

2.1.1.2. TIPOS DE DATOS VOLUMÉTRICOS

Dependiendo del área en que se esté trabajando y de la fuente de donde provienen los datos, los valores almacenados en el volumen pueden diferir en varios aspectos, entre los cuales se encuentran el rango válido, el tipo base –enteros, flotantes, números complejos– y su cardinalidad. Cuando la cardinalidad de los datos es uno, se dice que los datos son escalares simples, en los demás casos se dice que los datos son vectoriales. Es importante realizar esta distinción porque los métodos utilizados para el despliegue de datos vectoriales deben ser capaces de mostrar toda la información disponible, sin dificultar el entendimiento por parte del usuario.

Una forma de desplegar datos vectoriales tridimensionales es visualizando un corte a la vez con flechas en cada punto de la malla, cuya dirección indica la dirección del dato vectorial y cuyo color indica la magnitud del mismo o cualquier otro valor escalar. En vez de flechas, también se pueden utilizar otros elementos como líneas, flechas y triángulos, y se puede enlazar el brillo, el color, la dirección y la transparencia de los elementos a distintos atributos del volumen, lo que permite mostrar la mayor cantidad de datos a la vez de forma sencilla [SAW07].

2.1.1.3. VÓXELES Y CELDAS

Los volúmenes de datos son usualmente tratados como un conjunto de vóxeles o como un arreglo de celdas. El enfoque mediante vóxeles establece que los valores de los atributos del volumen se calculan en base a la muestra más cercana. Por lo tanto, un vóxel es la región cuyos puntos son más cercanos a una muestra en específico. En algunos algoritmos la contribución del vóxel a la imagen disminuye a medida que la distancia del centro de la región de influencia aumenta. Este enfoque tiene la ventaja de no inferir el comportamiento de los datos entre los

[2] PET: *Positron Emission Tomography*, Tomografía por Emisión de Positrones.

puntos de la malla, es decir, solamente utiliza valores conocidos de los datos para generar las imágenes.

El enfoque por celdas interpreta el volumen como una colección de hexaedros delimitados por los puntos de la malla, cuyos valores varían dentro de ellas. Este enfoque intenta estimar los valores dentro de la celda mediante interpolación entre los valores en las esquinas de la celda, donde se puede utilizar interpolación trilineal o interpolación bicúbica [ELV92]. Las imágenes generadas mediante este enfoque son mucho más suaves que aquellas generadas mediante el enfoque por vóxeles. Sin embargo, como generalmente la función subyacente no es conocida, es imposible verificar la validez de la interpolación utilizada para evaluar el volumen entre puntos del mallado discreto, por lo que se debe asumir que la técnica de interpolación es válida para que las imágenes generadas se consideren correctas.

2.1.2. MÉTODOS PARA LA VISUALIZACIÓN DE VOLÚMENES

Los algoritmos de visualización de volúmenes son clasificados en dos categorías: algoritmos de visualización directa de volumen (*direct volume rendering* o DVR) y algoritmos de extracción de isosuperficies (*surface fitting* o SF).

Los algoritmos basados en DVR incluyen trazado de rayos (*ray-casting*), *shear-warp*, métodos de preintegración, *splatting* y despliegue de V-buffer, los cuales están caracterizados por aplicar elementos directamente en la imagen sin utilizar primitivas geométricas como representación intermedia **[**ELV92**]**. Estos métodos son apropiados para crear imágenes a partir de volúmenes de datos que contengan límites difusos como nubes, fluidos y gases. La principal desventaja de usar DVR es que el volumen de datos debe ser recorrido por completo cada vez que una imagen tenga que ser desplegada, por lo que generalmente se realiza una pasada en baja resolución de los datos de forma tal que el usuario pueda verificar los parámetros rápidamente, para después realizar el despliegue con alta resolución una vez los parámetros sean confirmados.

Los algoritmos basados en SF usualmente ajustan primitivas como polígonos o parches a superficies de contorno con valor constante en volúmenes de datos. El primer paso consiste en la elección de un umbral por parte del usuario, el cual es utilizado para ajustar primitivas geométricas a los contornos en el volumen que sean iguales al umbral. Este enfoque incluye

algoritmos como conexión de contornos (*contour connecting*) [KEP75], cubos marchantes (*marching cubes*) [LOR87], tetracubos marchantes (*marching tetracubes*) [CAR95], entre otros.

2.1.3. PASOS GENERALES PARA LA VISUALIZACIÓN DE VOLÚMENES

A pesar de la variedad de algoritmos existentes para la visualización de volúmenes, la mayoría de los pasos involucrados son comunes en ellos. Generalmente, los algoritmos difieren en la forma en que implementan cada uno de dichos pasos, los cuales se explican a continuación.

2.1.3.1. ADQUISICIÓN DE DATOS

El primer paso en cualquier procedimiento para la visualización de datos consiste en la obtención y preprocesamiento de éstos, de forma que se puedan obtener mejores resultados visuales. Este preprocesamiento consiste en modificar los valores para que cubran una buena distribución de valores, tengan altos contrastes y estén libres de ruido y valores fuera de rango.

Finalmente, en algunos algoritmos [GUO95] es necesario que los datos tengan la misma proporción espacial que el objeto de estudio, de forma que las imágenes desplegadas no aparezcan deformadas a la hora de visualizar el volumen. Cuando la proporción del objeto y de los datos no es la misma, puede ser necesario interpolar dos cortes para obtener uno nuevo, interpolar muestras para obtener muestras faltantes, o convertir un mallado irregular a un mallado cartesiano regular uniforme.

2.1.3.2. CLASIFICACIÓN DE LOS DATOS

La clasificación de los datos consiste en elegir la forma en que los datos deben ser desplegados en base a sus valores. Este paso es llevado a cabo por el usuario, y el procedimiento a realizar depende del algoritmo utilizado para visualizar los datos. Si el algoritmo está basado en SF, el usuario debe elegir el umbral a extraer, el cual consiste en un valor real que representa el valor a ajustar por la isosuperficie resultante. Cuando el algoritmo está basado en DVR, el usuario debe configurar la función de transferencia, la cual consiste en una función que relaciona un color y factor de absorción a cada valor posible que puedan tener los atributos del volumen (ver Sección 2.1.4.2 para más detalles sobre el uso de la función de transferencia en DVR).

La clasificación de los datos es uno de los pasos más difíciles que debe llevar a cabo el usuario, porque es necesario que éste tenga experiencia clasificando los datos y que el sistema proporcione una respuesta rápida, debido a que este procedimiento está basado en intento y error. Una solución para dar respuesta rápida al usuario consiste en ofrecer una vista con menor resolución mientras se realiza el proceso de clasificación, y generar una imagen refinada después que el usuario confirma que ha finalizado.

2.1.3.3. RECORRIDO DE LOS DATOS

Después de configurar la clasificación del volumen de datos, se deben generar las imágenes recorriendo los datos. Existen dos formas de recorrer el volumen: en orden de objeto (*object-order*), que consiste en calcular la contribución de cada elemento del volumen a los píxeles de la imagen; o en orden de imagen (*image-order*), que consiste en determinar el color en cada píxel de la imagen, buscando los elementos del volumen que contribuyen a cada uno de estos [ELV92].

Los recorridos en orden de objeto pueden ir de adelante hacia atrás (*front-to-back*) o de atrás hacia adelante (*back-to-front*). Recorrer el modelo de adelante hacia atrás tiene la ventaja de que los elementos en la parte de atrás no deben ser visitados si los de adelante ya han creado una imagen lo suficientemente opaca [ELV92].

Los recorridos en orden de imagen generalmente proceden de arriba hacia abajo, de izquierda a derecha. También se pueden calcular los píxeles en orden aleatorio, de forma que el usuario observe como la imagen es refinada mientras los píxeles faltantes son calculados.

2.1.3.4. VISUALIZACIÓN Y SOMBREADO

Para visualizar un volumen de datos se puede utilizar tanto proyección ortogonal como proyección perspectiva, sin importar que el algoritmo esté basado en DVR o en SF. Sin embargo, el uso de proyección ortogonal asegura que el usuario no se confunda al observar los datos deformados por la transformación perspectiva. No obstante, cuando no se utiliza perspectiva, se deben incluir otras características que le permitan al usuario percibir la profundidad de los elementos, tal como niebla por profundidad, atenuación por distancia o estereoscopía [ELV92].

Para realizar el sombreado en los algoritmos basados en DVR y en SF generalmente se utiliza sombreado por gradiente (*gradient shading*), el cual consiste en implementar un modelo de iluminación estándar, como *Phong* o *Blinn*, utilizando el gradiente normalizado de los datos como vector normal. Para calcular los gradientes dentro de una celda se utiliza interpolación de los gradientes en las ocho esquinas de la celda, donde el gradiente en un punto de la malla se calcula por diferencias finitas entre los puntos adyacentes en cada dirección [LOR87].

2.1.4. ALGORITMOS PARA LA VISUALIZACIÓN DE VOLÚMENES

A continuación se explican algunos algoritmos para la visualización de volúmenes, los cuales se encuentran clasificados en base al método utilizado para desplegar el volumen, ya sea mediante visualización directa de volumen (DVR) o extracción de isosuperficies (SF).

2.1.4.1. EXTRACCIÓN DE ISOSUPERFICIES

Los algoritmos de extracción de isosuperficies consisten en el cálculo de una representación intermedia que se ajuste a las partes del volumen que se desean desplegar. Las partes que se desean desplegar son elegidas mediante el uso de un umbral, el cual especifica a qué isovalor se deben ajustar las primitivas calculadas. Las primitivas utilizadas como representación intermedia generalmente son fáciles de desplegar por hardware gráfico estándar, como lo son los mallados triangulares y los cuadriláteros [ELV92].

A continuación se explicarán tres algoritmos para la extracción de isosuperficies, entre los cuales se encuentran la conexión de contornos, el algoritmo de los cubos opacos, el algoritmo de los cubos marchantes y el algoritmo de los tetracubos marchantes.

A. CONEXIÓN DE CONTORNOS

La conexión de contornos (*contour-connecting*) fue uno de los primeros algoritmos inventados para la visualización de volúmenes, el cual consiste en calcular un contorno cerrado en cada corte y conectar los contornos de cada par de cortes adyacentes [KEP75][FUC77]. Este algoritmo comienza calculando el contorno de cada corte en el valor especificado como umbral por el usuario. Anteriormente este procedimiento se realizaba a mano, pero las técnicas actuales de procesamiento de imágenes permiten la extracción automática del contorno.

Una vez que se tienen todos los contornos, el problema se reduce a conseguir una triangulación que conecte las curvas de los cortes adyacentes. Para determinar la triangulación a utilizar, se busca aquella que maximice –o minimice– alguna métrica de la triangulación generada, como el volumen del modelo o el mínimo ángulo interno de los triángulos [KEP75].

Existen dos clases de métodos para unir dos contornos: optimización y heurísticos. Los métodos de optimización calculan la mejor triangulación posible [FUC77], y por lo general tardan $O(NM)$ en ejecutarse, donde N es la cantidad de puntos de un corte y M es la cantidad de puntos del otro corte. Los métodos heurísticos calculan una triangulación aceptable, mediante el uso de una decisión local por medio de una heurística fácil de calcular en cada punto [KEP75]. Estos últimos generalmente son utilizados cuando el tiempo de ejecución es más importante que la calidad del modelo, ya que generalmente tardan $O(N + M)$ en ejecutarse.

B. CUBOS OPACOS

Otro procedimiento sencillo para la visualización de volúmenes es el algoritmo de los cubos opacos (*opaque-cubes*), el cual fue propuesto por [HER79], donde el umbral elegido por el usuario se utiliza para recorrer cada una de las celdas del objeto y extraer aquellas cuyos valores lo incluyan. Por cada una de estas celdas, se crean seis polígonos, uno por cada cara, los cuales son desplegados usando cualquier hardware estándar. Los polígonos se pueden desplegar opacos o semitransparentes, y si se eligen varios umbrales se puede desplegar cada conjunto de celdas con un color diferente.

Una de las principales desventajas de este algoritmo es que las imágenes desplegadas tienen apariencia de bloques cúbicos, lo cual genera dificultades al intentar observar características pequeñas en los datos. Sin embargo, la apariencia puede ser mejorada utilizando sombreado por gradiente (*gradient-shading*).

C. CUBOS MARCHANTES

Cubos Marchantes (*marching cubes*) es un algoritmo para la visualización de volúmenes que consiste en la extracción de una isosuperficie en base a una tabla de casos [LOR87], el cual procesa cada celda y extrae un mallado triangular que separa aquellos vóxeles menores que el

umbral de aquellos mayores que el umbral. Este algoritmo ha sido implementado y estudiado ampliamente en numerosas publicaciones [NIE91][CHE95].

Cubos marchantes comienza leyendo cuatro cortes, con los cuales calcula el gradiente en los puntos internos y se extraen triángulos de las celdas internas entre el segundo y tercer corte. Después, se carga un nuevo corte, se descarta el más viejo y se repite el procedimiento, hasta haber procesado todos los cortes. Finalmente, los triángulos extraídos son pasados a hardware gráfico estándar para el despliegue de superficies. Para más detalles acerca del algoritmo, ver la Sección 2.2.1.

D. TETRACUBOS MARCHANTES

Tetracubos Marchantes (*marching tetracubes*) es un algoritmo para la visualización de volúmenes que consiste en la extracción de una isosuperficie mediante la división de cada celda en tetraedros y su posterior procesamiento, el cual consiste en la extracción de un mallado triangular que separe aquellos vóxeles menores que el umbral de aquellos mayores que el umbral [CAR95]. Este algoritmo opera de manera similar a cubos marchantes, pero no presenta casos ambiguos, por lo cual es mucho más fácil de implementar. Para más detalles acerca del algoritmo, ver la Sección 2.2.2.

2.1.4.2. VISUALIZACIÓN DIRECTA DE VOLUMEN

Los métodos basados en la visualización directa de volumen consisten en desplegar el volumen directamente en el plano imagen sin utilizar una representación intermedia para ello. Para determinar el color de un píxel en el plano imagen se simula la propagación de un rayo de luz a través del volumen, lo cual se realiza mediante el uso de un sistema óptico basado únicamente en la absorción y emisión de luz [WIL92].

El modelo óptico utilizado para simular la propagación de un rayo de luz a través del volumen está definido por la siguiente ecuación [SCH03]:

$$C = \int_0^D c(\lambda)\tau(\lambda)e^{-\int_0^\lambda \tau(t)\partial\lambda}\partial\lambda$$

donde C es el color resultante, D es la distancia que recorre el rayo dentro del volumen, y $c(\lambda)$ y $\tau(\lambda)$ son el color y el factor de absorción a una distancia λ de la entrada del rayo en el volumen, respectivamente. Esta integral representa la suma de la emisión de la luz desde el punto de entrada del rayo en el volumen ($\lambda = 0$) hasta que sale del mismo ($\lambda = D$). La exponencial que multiplica el color en la posición λ representa el factor de extinción acumulado hasta ese punto, lo cual hace que los valores obtenidos vayan aportando menos al color final a medida que el rayo avanza.

Esta ecuación evalúa el rayo de forma continua; para realizar la evaluación de manera discreta ésta se aproxima mediante sumas de Riemann [ANT98]:

$$n = \left\lfloor \frac{D}{h} \right\rfloor, \quad \alpha(\lambda) = 1 - e^{-h\tau(\lambda)} \tag{1.1}$$

$$C \approx \sum_{i=0}^{n-1} \left(c(ih)\alpha(ih) \prod_{j=0}^{i-1} \big(1 - \alpha(jh)\big) \right)$$

donde n representa la cantidad de muestras a evaluar y jh representa la posición de la *j*-ésima muestra dentro del rayo. Esta ecuación puede ser evaluada de dos formas diferentes: la primera consiste en evaluar y acumular las muestras desde la más cercana hacia la más lejana (*front to back*) de la siguiente manera:

$$C_0 = 0, \quad C_{i+1} = C_i + A_i \alpha(ih) c(ih)$$
$$A_0 = 1, \quad A_{i+1} = A_i \big(1 - \alpha(ih)\big)$$

donde C_i y A_i son el color y el factor de extinción acumulado después de evaluar i muestras, respectivamente. El color asignado al píxel es aquel encontrado después de evaluar las n muestras, es decir, $C \approx C_n$.

La segunda forma de evaluar la ecuación consiste en evaluar y acumular las muestras desde la más lejana hacia la más cercana (*back to front*) de la siguiente manera:

$$C_n = 0, \quad C_i = \alpha(ih) c(ih) + \big(1 - \alpha(ih)\big) C_{i+1}$$

donde C_i es el color acumulado cuando quedan i muestras por evaluar. El color asignado al píxel es aquel encontrado cuando no quedan muestras por evaluar, es decir, $C \approx C_0$.

Existen varios métodos para el despliegue de volúmenes mediante visualización directa de volúmenes, entre las cuales se encuentra el uso de planos alineados al objeto, planos alienados a la imagen, *ray casting* por software, *shear-warp*, *ray casting* por GPU[3] y *splatting*. A continuación se explicará *ray casting* y *splatting*.

A. RAY CASTING

Otro algoritmo utilizado para la visualización de volúmenes con imágenes de alta calidad consiste en *ray casting*. Este algoritmo lleva a cabo un recorrido en orden de imagen, donde el color y opacidad de cada píxel se calcula disparando un rayo desde el píxel hacia el volumen de datos, acumulando las opacidades y colores encontrados durante la trayectoria del mismo [LEV88][LEV90a][LEV90b].

El primer paso consiste en la configuración de la función de transferencia, el punto de vista del usuario y la iluminación a utilizar. Entonces, se dispara un rayo por cada píxel en la imagen a generar, se determina el punto de entrada y de salida de éste del volumen, con el fin de calcular la ecuación 1.1. Para ello, el rayo es evaluado a pasos constantes de tamaño h desde el punto más cercano al ojo hasta el más lejano.

B. SPLATTING

Este algoritmo consiste en realizar un recorrido de adelante hacia atrás de los vóxeles del volumen, calculando y componiendo la contribución de cada uno de ellos en la imagen mediante el uso de una tabla. Al procedimiento se le llama *splatting* [WES89] porque se asemeja a "aplastar" cada vóxel en la imagen, donde la contribución del vóxel sobre un píxel es inversamente proporcional a la distancia entre el vóxel proyectado y dicho píxel.

El primer paso, después de que el usuario configura la función de transferencia, consiste en determinar el orden en el que se va a recorrer el volumen, lo cual se realiza buscando la esquina de la malla más cercana al punto de vista después de aplicar la matriz de visualización

[3] GPU: *Graphical Processing Unit* (Unidad Gráfica de Procesamiento)

(*modelview matrix*). Una vez determinado el orden correcto, se procesan los vóxeles desde el corte más cercano hacia el más lejano. La clasificación de un vóxel se realiza de manera similar como en trazado de rayos: primero se clasifica su valor utilizando la función de transferencia, se sombrea utilizando el gradiente y la opacidad se multiplica por la magnitud del mismo.

El siguiente paso consiste en calcular la contribución del vóxel a la imagen proyectando éste en la imagen y utilizando una huella circular [WES89]. Esta huella es colocada en la imagen sobre el vóxel proyectado, el cual sirve para decidir cuánto va a afectar el vóxel actual cada píxel de la imagen. Sin embargo, si se utiliza proyección perspectiva o escalamientos no uniformes, la proyección de la huella en el plano imagen es una elipse, por lo que se debe aplicar una transformación para convertir las coordenadas elípticas a coordenadas circulares, y así poder utilizar la misma huella para todos los vóxeles [WES90]. Luego, el color y la opacidad son mezclados con los valores acumulados en la imagen en cada píxel dentro del área cubierta por la huella, atenuando el color y la opacidad en base al valor de la huella en cada píxel a modificar. Esto ocasiona que los píxeles más cercanos al vóxel proyectado sean más afectados que aquellos más distantes. Una vez todos los vóxeles han sido procesados, la imagen está lista para ser desplegada.

2.2. EXTRACCIÓN DE ISOSUPERFICIES

Para la extracción de isosuperficies, se considera el volumen de datos como una función $F(x, y, z)$, la cual relaciona una posición dentro del volumen con un valor escalar, es decir:

$$F(x, y, z) = \Re^3 \longrightarrow \Re$$

La extracción de una isosuperficie de un volumen de datos consiste en el cálculo de una superficie que aproxime el conjunto S_α, el cual se define como:

$$S_\alpha = \{(x, y, z) | F(x, y, z) = \alpha\}$$

donde α representa el valor de la isosuperficie a extraer. La mayoría de los algoritmos representan la superficie extraída mediante un mallado triangular, el cual está compuesto por un conjunto de vértices y un conjunto de triángulos que unen dichos vértices. Esto permite visualizar la superficie fácilmente mediante el uso del hardware gráfico existente.

En los algoritmos expuestos a continuación, se asume que sólo se conocen los valores de $F(x, y, z)$ en los vértices de las celdas de una malla cartesiana escalar como la que se muestra en la Figura 2.1.

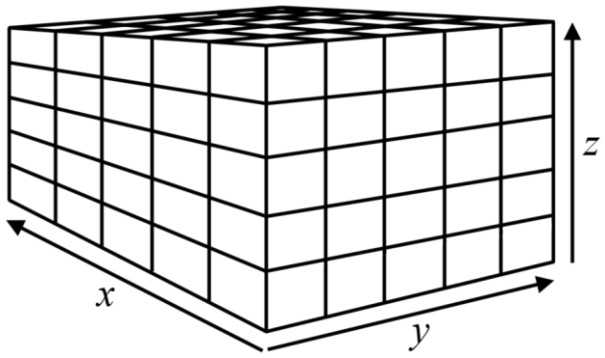

Figura 2.1: Malla cúbica utilizada para la extracción de isosuperficies.

Para evaluar $F(x, y, z)$ en un punto que no corresponda a uno de los vóxeles, se realiza interpolación sobre la celda donde se encuentra el punto a evaluar. Se pueden utilizar varios esquemas de interpolación, pero el más utilizado es interpolación trilineal [ELV92]. Para realizar interpolación trilineal, primero se debe obtener la posición relativa del punto respecto a la celda donde se encuentra de la siguiente manera:

$$q = \frac{x - X_L}{X_H - X_L}, \quad s = \frac{y - Y_L}{Y_H - Y_L}, \quad t = \frac{z - Z_L}{Z_H - Z_L}$$

donde (X_L, Y_L, Z_L) es la posición de la esquina inferior-izquierda-trasera de la celda y (X_H, Y_H, Z_H) es la posición de la esquina superior-derecha-delantera de la misma. Finalmente se interpolan los ocho isovalores de la celda de la siguiente manera:

$$\begin{aligned} F(v_{000}, \cdots, v_{111}, q, s, t) = {} & v_{000} \cdot (1-q)(1-s)(1-t) + v_{001} \cdot q(1-s)(1-t) \\ & + v_{010} \cdot (1-q)s(1-t) + v_{011} \cdot qs(1-t) \\ & + v_{100}.(1-q)(1-s)t + v_{101} \cdot q(1-s)t \\ & + v_{110} \cdot (1-q)st + v_{111} \cdot qst \end{aligned}$$

donde $v_{000}, \cdots, v_{111}$ representan los isovalores de los ocho vóxeles que delimitan la celda. A continuación, se explicarán dos algoritmos importantes para la extracción de isosuperficies a partir de mallas cartesianas, cubos marchantes y tetracubos marchantes.

2.2.1. CUBOS MARCHANTES

Uno de los métodos más conocidos y estudiados para la extracción de isosuperficies a partir de una malla cartesiana escalar es el algoritmo de los Cubos Marchantes (*Marching Cubes*) [LOR87], el cual consiste en reconstruir cada celda de la malla independientemente mediante el uso de una tabla de conectividad. Cada celda se encuentra delimitada por ocho vóxeles y doce aristas, las cuales se enumeran como se muestra en la Figura 2.2.

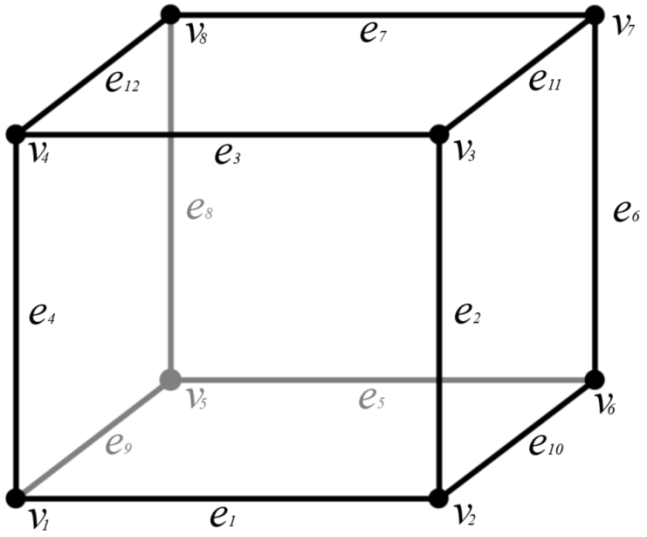

Figura 2.2: Enumeración de los vértices y aristas que delimitan la celda.

Para triangular cada celda primero se debe clasificar cada uno de los vóxeles que la delimitan. Para simplificar la clasificación de las celdas, Chernyaev [CHE95] resta el valor α a extraer del isovalor de cada vóxel, de forma tal que extraer la isosuperficie S_x antes de la resta equivale a extraer la isosuperficie S_0 después de realizar la resta, es decir:

$$S_\alpha = \{(x, y, z) | F(x, y, z) = \alpha\} = \{(x, y, z) | F(x, y, z) - \alpha = 0\}$$

De esta manera, se pueden clasificar los vóxeles en dos grupos, positivos y negativos, de acuerdo al signo del isovalor después de sustraer α. Luego, con la clasificación de los vóxeles se construye una máscara de ocho bits, donde el *k*-ésimo bit se encuentra encendido si y sólo si el *k*-ésimo vóxel es positivo (ver Figura 2.3).

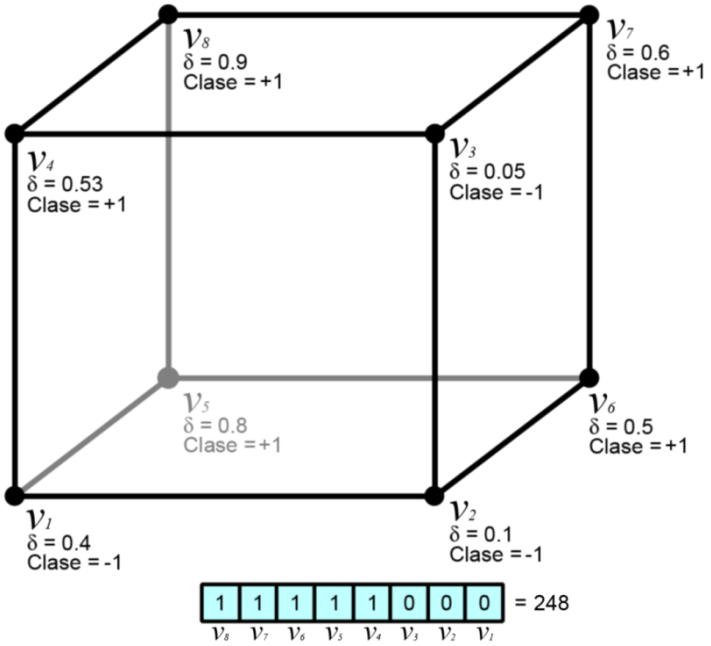

Figura 2.3: Construcción de la máscara a partir de los valores en los vértices con $\alpha = 0.5$, donde cada δ representa el isovalor en cada vértice que delimita la celda.

Debido a que cada uno de los vóxeles tiene dos estados posibles, existe un total de $2^8 = 256$ formas diferentes en que la superficie a extraer puede intersectar la celda. Sin embargo, la topología de la superficie no cambia si todos los vóxeles cambian de signo, por lo cual se pueden reducir los 128 casos superiores a los 128 casos inferiores invirtiendo el sentido de los triángulos generados (casos complementarios). Luego, por medio de reflexiones y rotaciones se pueden reducir los 128 casos a 15 casos topológicamente diferentes, los cuales se muestran en la Figura 2.4.

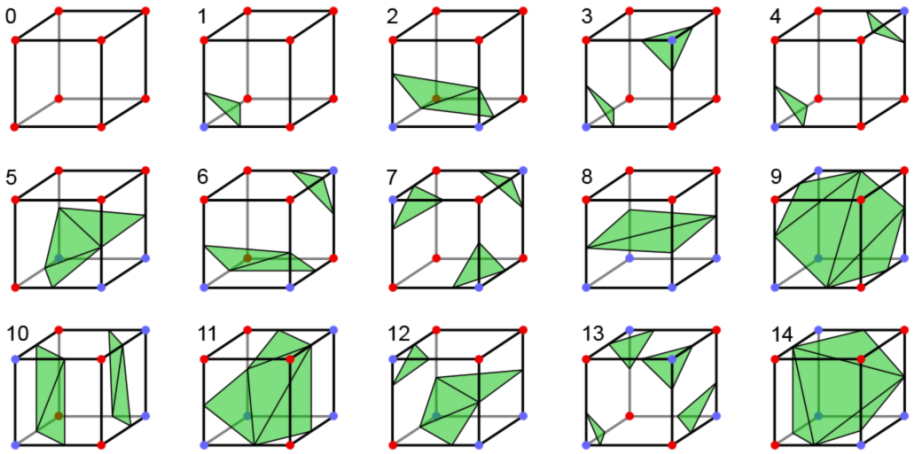

Figura 2.4: Las 15 configuraciones diferentes usadas por Cubos Marchantes. Los vóxeles rojos son negativos y los vóxeles azules son positivos.

El siguiente paso es determinar cuáles aristas de la celda son intersectadas por la superficie y dónde se encuentra el punto de intersección. Una arista es intersectada por la superficie si y sólo si los vóxeles en sus extremos tienen signos diferentes. Para determinar el punto P_x donde la arista es intersectada por la superficie se puede utilizar interpolación lineal sobre la misma de la siguiente manera:

$$P_x = P_0 + (P_1 - P_0)\left(\frac{\alpha - v_0}{v_1 - v_0}\right)$$

donde, v_0 y v_1 son los isovalores de los vóxeles que delimitan la arista, α es el umbral de la superficie que se desea extraer, P_0 y P_1 son las posiciones de los extremos de la arista y P_x es la posición del punto de intersección en ésta. Después de calcular todos los puntos de intersección de la superficie con la celda se generan los triángulos que la componen utilizando una tabla de 256 casos, la cual se indexa utilizando la máscara obtenida de la clasificación de los vóxeles.

Mediante cubos marchantes también se pueden calcular otros atributos de los vértices del mallado final, mediante interpolación lineal de los atributos en los extremos de cada arista intersectada por el modelo. Por ejemplo, se puede calcular el gradiente por vértice G_x interpolando los gradientes en los extremos de cada arista intersectada de la siguiente manera:

$$G_x = G_0 + (G_1 - G_0)\left(\frac{\alpha - v_0}{v_1 - v_0}\right)$$

donde G_0 y G_1 son los gradientes en los extremos de la arista y G_x es el gradiente en P_x.

2.2.1.1. VENTAJAS Y DESVENTAJAS DE CUBOS MARCHANTES

Cubos marchantes es uno de los algoritmos más utilizados para la extracción de isosuperficies debido a las diversas ventajas que posee, entre las cuales se encuentran:

- **Sencillez:** El algoritmo es fácil de implementar, ya que sólo se necesita una tabla de 2^8 casos para procesar cada celda en base a la clasificación de sus vóxeles.

- **Eficiencia:** El algoritmo sólo calcula las intersecciones en aquellas aristas donde es necesario, reutilizando las intersecciones de las celdas anteriores de ser posible. Además,

el algoritmo sólo necesita la construcción de una máscara y búsquedas de orden constante en tablas precalculadas.

- **Paralelizable:** El algoritmo es fácil de paralelizar, debido a que la superficie extraída de una celda es independiente de las superficies extraídas de las otras celdas.

Sin embargo, el algoritmo posee varias desventajas, para las cuales se han desarrollado técnicas para corregirlas, entre las cuales se encuentran:

- **Errores topológicos:** En ciertos conjuntos de datos, el algoritmo genera huecos incorrectos en la superficie. En las siguientes secciones se explicarán los trabajos dedicados a corregir dichos errores.

- **Triángulos degenerados:** En ciertos casos, el algoritmo genera triángulos de baja calidad, los cuáles se visualizan con artefactos dependiendo de la técnica utilizada para su despliegue.

- **Complejidad de la salida:** El algoritmo genera una cantidad excesiva de triángulos redundantes cuando la resolución de los modelos de entrada es muy alta, lo que ocasiona que la superficie final sea más difícil de visualizar. Una de las soluciones es expuesta en [SCH92], donde se explica un método que permite aproximar una superficie mediante otra con menos triángulos, la cual se podría aplicar como un post-procesamiento a la superficie generada por cubos marchantes.

2.2.1.2. MODELOS TOPOLÓGICAMENTE CORRECTOS

Para analizar el mallado final generado por el algoritmo original de Cubos Marchantes, primero se debe definir qué es una superficie topológicamente correcta. En [CHE95] se define que una superficie es topológicamente correcta si y sólo si la topología de los triángulos generados coincide con la topología de la función $F(x, y, z)$ dentro de cada celda.

Para obtener una superficie topológicamente correcta, dos vóxeles del mismo signo deben estar unidos – o no separados – dentro de una celda si y sólo si existe un camino dentro de ésta que conecte ambos vóxeles y no cambie de signo. El caso más simple para determinar si dos

vóxeles están conectados es cuando existe un camino a través de las aristas de la celda que sólo contiene vóxeles del mismo signo. Sin embargo, hay dos casos donde la regla anterior no es suficiente. Por ejemplo, cuando se tiene una cara con dos vóxeles positivos y dos vóxeles negativos en esquinas opuestas que no están conectados a través de las demás aristas de la celda, no se puede decir directamente que ambos vóxeles están separados, debido a que éstos podrían estar unidos mediante un camino que pase por dentro de la cara. A este tipo de caras se les denomina caras ambiguas, como la que se puede observar en la Figura 2.5.

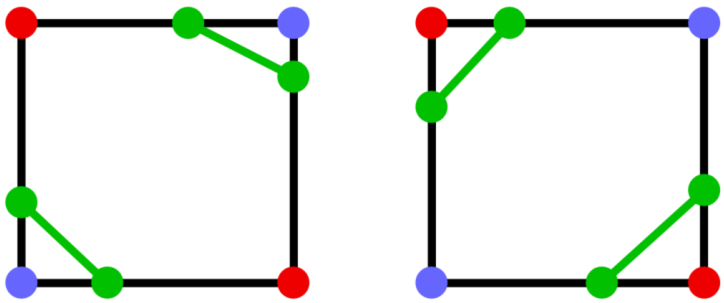

Figura 2.5: Ejemplo de una cara ambigua. A la izquierda se muestra el caso donde los vóxeles positivos se encuentran separados, a la derecha se muestra el caso donde éstos no son separados. Los vóxeles positivos se encuentran marcados en azul, y los vóxeles negativos se encuentran marcados en rojo.

El otro caso donde las reglas anteriores no son suficientes es cuando se tiene una ambigüedad interna, la cual ocurre cuando dos vóxeles del mismo signo se encuentran en esquinas opuestas de la celda y no están conectados por las aristas ni por las caras de la misma, pero podrían estar unidos a través de un camino que pase por dentro de la celda. Un ejemplo de este tipo de ambigüedad se puede observar en la Figura 2.6.

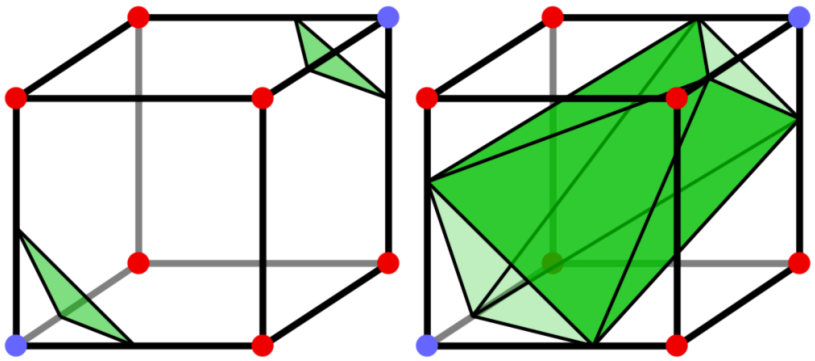

Figura 2.6: Ejemplo de ambigüedad interna. A la izquierda se muestra el caso donde los vértices positivos se encuentran separados, y a la derecha se muestra el caso donde éstos no están separados.

2.2.1.3. ERRORES TOPOLÓGICOS DE CUBOS MARCHANTES

Una de las mayores críticas realizadas al algoritmo original de Cubos Marchantes, propuesto en [LOR87], es que genera errores topológicos en ciertos volúmenes de datos, los cuales consisten en pequeños huecos en la superficie. En la Figura 2.7 se puede observar el hueco que genera el algoritmo de los cubos marchantes cuando encuentra una celda del caso 6 adyacente a una celda del complemento del caso 3.

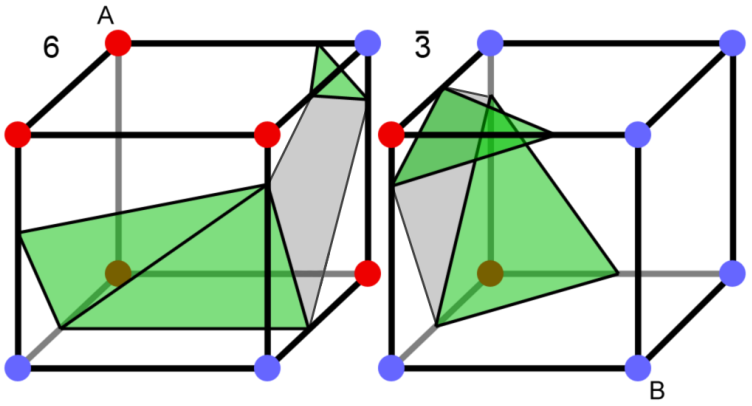

Figura 2.7: Ejemplo mostrando el hueco creado por Cubos Marchantes, el cual se encuentra representado por el polígono gris en la cara compartida por ambas celdas.

Para verificar que este hueco no es topológicamente correcto, se puede observar que el vóxel A tiene signo diferente al vóxel B, lo que implica que el segmento de línea delimitado por esos vóxeles debería intersectar la superficie generada en algún punto. Dürst [DÜR88] propone resolver este error agregando el cuadrilátero formado por los cuatro vértices en la cara compartida por ambas celdas, correspondiente al polígono gris en la Figura 2.7. Sin embargo, esta solución no es la más adecuada, debido a que ocasionaría que más de dos triángulos compartan una arista en el mallado final.

La razón por la cual se genera el hueco en la superficie es que la cara compartida por ambas celdas es una cara ambigua, la cual se conecta de una forma en la celda izquierda y de otra forma en la celda derecha. Existen dos formas de triangular una cara ambigua, una donde los triángulos generados separan ambos vóxeles positivos en la cara y otra donde éstos no son separados por la triangulación. En el primer caso se dice que la cara es una cara separada y en el segundo caso se dice que es una cara no separada. Ambas triangulaciones válidas para el caso expuesto de la

Figura 2.7 se pueden observar en la Figura 2.8, donde la triangulación superior corresponde a utilizar una cara no separada y la triangulación inferior corresponde a utilizar una cara separada.

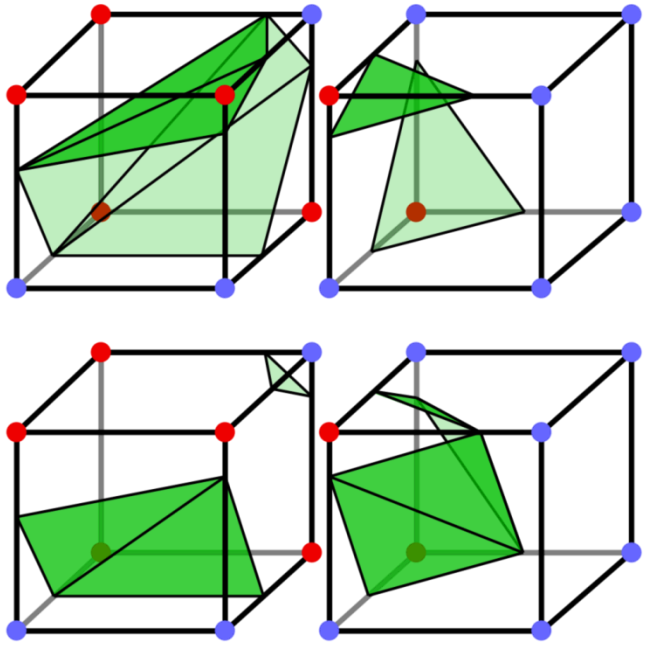

Figura 2.8: Posibles triangulaciones válidas para el caso mostrado en la **Figura 2.7**.

Cualquier método de extracción correcto debe conectar los vértices de una cara ambigua de la misma forma en ambas celdas a las que pertenece, ya sea separando los vóxeles positivos en la triangulación o no. En el algoritmo original propuesto en [LOR87] no se cumple esta condición, debido a que al utilizar los complementos de los casos los vóxeles positivos quedan unidos dentro de las caras ambiguas, por lo cual se genera un hueco al unir un caso normal con uno complementario (ver Figura 2.7).

Montani et al. [MON94] proponen un método para evitar la aparición de huecos en la superficie, el cual consiste en modificar la tabla de conectividad del algoritmo original de forma que los vóxeles positivos siempre sean separados en las caras ambiguas. Para esto, no se pueden reducir los casos con caras ambiguas mediante complementariedad, sino únicamente por medio de rotaciones y reflexiones. Los casos se pueden clasificar en tres conjuntos diferentes, los cuales se procesan como se indica a continuación:

- No hay más de cuatro vóxeles positivos: Se pueden generar mediante rotaciones y reflexiones de los casos existentes en el algoritmo original [LOR87], debido a que los vértices positivos en las caras ambiguas ya se encuentran separados (ver Figura 2.4).

- Hay más de cuatro vóxeles positivos y no hay caras ambiguas: Se pueden generar mediante rotaciones, reflexiones y complementariedad de los casos existentes en el algoritmo original, porque al no existir caras ambiguas se puede aplicar complementariedad libremente.

- Hay más de cuatro vóxeles positivos y hay caras ambiguas: Se deben crear nuevas triangulaciones, debido a que no se puede recurrir a la complementariedad por la existencia de caras ambiguas. Estas triangulaciones nuevas corresponden a los complementos de los casos 3, 6 y 7, los cuales se pueden observar en la Figura 2.9.

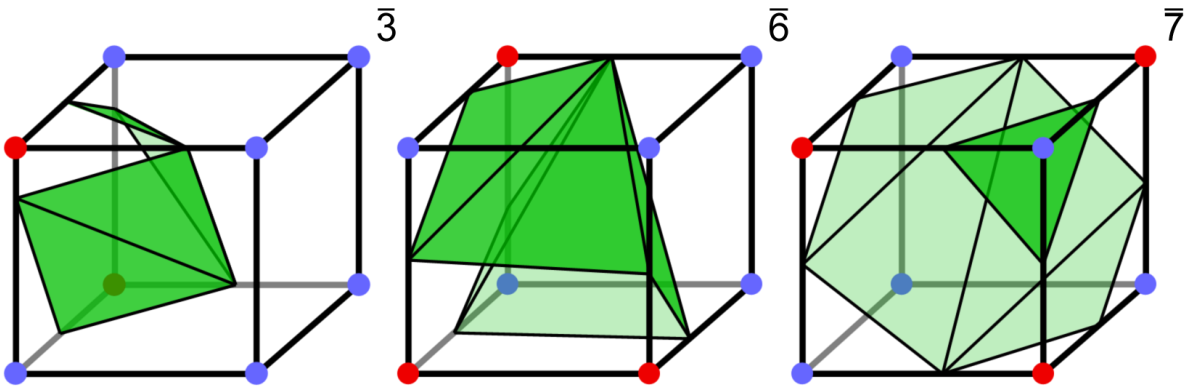

Figura 2.9: Casos adicionales para prevenir huecos en la superficie.

Este método posee todas las ventajas del algoritmo original, y no conlleva ninguna desventaja adicional, debido a que la tabla de conectividad se conserva del mismo tamaño ($2^8 = 256$) y no se requiere de ningún procesamiento adicional para procesar cada celda.

2.2.1.4. MÉTODO DE DECISIÓN ASINTÓTICA PARA CUBOS MARCHANTES

Como se mencionó anteriormente, la elección entre las dos formas válidas de conectar los vértices en una cara ambigua puede ser totalmente arbitraria siempre y cuando sea consistente a lo largo de todas las celdas. Sin embargo, Nielson y Hamman presentaron el Método de la Decisión Asintótica (*Asymptotic Decider*) [NIE91], el cual consiste en elegir entre separar o no separar los vóxeles positivos de una cara ambigua en base a la evaluación de un punto medio de

ésta por medio de interpolación bilineal, de forma tal que la superficie generada sea topológicamente correcta (ver Sección 2.2.1.2).

La interpolación bilineal a través de una cara es la extensión natural a realizar interpolación lineal a través de un segmento en dos dimensiones. Después de realizar un cambio de variables, se puede asumir que el dominio de la cara es un cuadrado unitario $\{(s,t)|0 \leq s,t \leq 1\}$, lo cual lleva a la siguiente fórmula para realizar interpolación bilineal:

$$B(s,t) = (1-s \quad s)\begin{pmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \end{pmatrix}\begin{pmatrix} 1-t \\ t \end{pmatrix}$$

donde $B_{0,0}$, $B_{0,1}$, $B_{1,0}$ y $B_{1,1}$ son los isovalores de los vóxeles que delimitan la cara ambigua. En la Figura 2.10 se puede observar una gráfica que muestra el isovalor de la cara dentro del dominio, donde los ejes s y t representan el dominio de la cara y el eje vertical representa el valor de la función $F(x,y,z)$ en la cara.

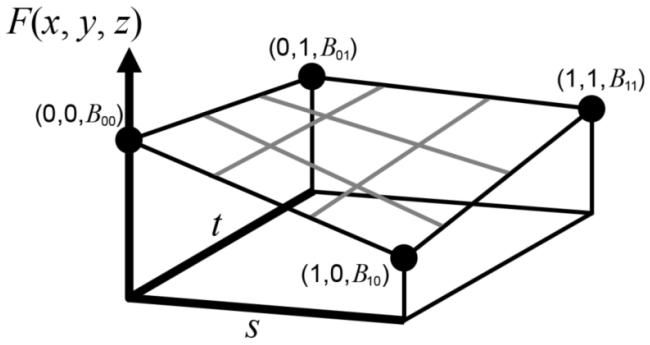

Figura 2.10: Interpolación bilineal a través de la cara.

Se puede verificar fácilmente que la curva $\{(s,t)|B(s,t) = \alpha\}$ es una hipérbola. Hay varias formas en que el dominio de la cara puede intersectar (o no intersectar) la hipérbola, como se puede observar en la Figura 2.11:

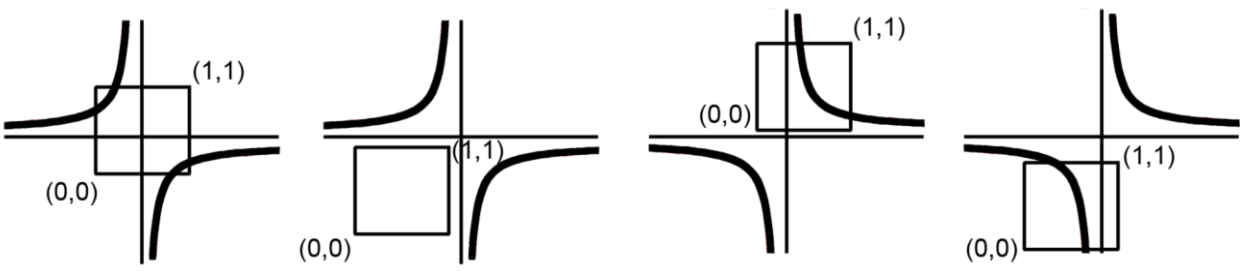

Figura 2.11: Distintos resultados posibles de intersección con la hipérbola.

Los casos ambiguos se dan cuando el dominio intersecta ambas componentes de la hipérbola, en cuyo caso no se puede determinar a priori si los vóxeles positivos están dentro de la misma región de las hipérbolas. Por ejemplo, en la Figura 2.12 se pueden observar dos formas posibles en que el dominio intersecte la hipérbola en una cara ambigua.

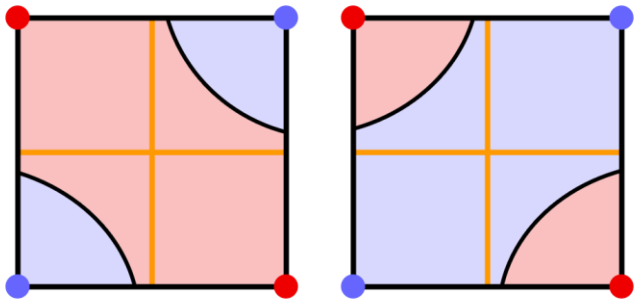

Figura 2.12: Ambas hipérbolas posibles de la cara ambigua.

Una forma sencilla pero incorrecta de determinar qué caso se debe aplicar consiste en evaluar exactamente el punto medio de la cara, y en base a esto decidir si los vóxeles positivos deben ser separados o no. Sin embargo, este método falla en los casos donde el centro del dominio se encuentra dentro de alguna de las hipérbolas. En la Figura 2.13 se puede observar claramente que ambos vóxeles positivos no deben ser separados dentro del modelo, pero la evaluación del punto medio arroja como resultado que éstos deben ser separados.

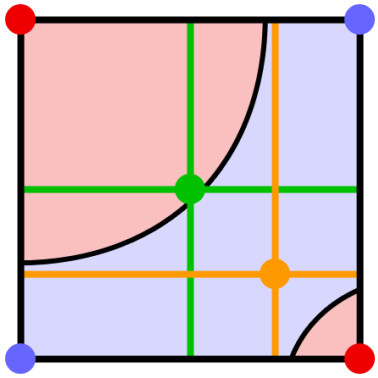

Figura 2.13: Caso donde evaluar en el centro de la cara (punto verde) da resultados erróneos, mientras que evaluar en la intersección de las asíntotas (punto anaranjado) da el resultado correcto.

La solución propuesta por Nielson [NIE91] consiste en determinar si los vóxeles positivos están separados o no mediante la evaluación de $F(x, y, z)$ en el punto de la cara donde las

asíntotas de las hipérbolas se intersectan, lo cual siempre arrojará el resultado correcto porque éste se encuentra fuera de las mismas.

El primer paso consiste en buscar las asíntotas $\{(s,t)|s = S_h\}$ y $\{(s,t)|t = T_h\}$, donde:

$$S_h = \frac{B_{0,0} - B_{0,1}}{B_{0,0} + B_{1,1} - B_{0,1} - B_{1,0}}, \quad T_h = \frac{B_{0,0} - B_{1,0}}{B_{0,0} + B_{1,1} - B_{0,1} - B_{1,0}}$$

por lo cual se puede calcular el valor en $B(S_\alpha, T_\alpha)$ mediante interpolación bilineal:

$$B_{S,T} = B(S_h, T_h) = \frac{B_{0,0}B_{1,1} - B_{1,0}B_{0,1}}{B_{0,0} + B_{1,1} - B_{0,1} - B_{1,0}}$$

Finalmente, para determinar a qué caso de la Figura 2.12 corresponde la cara ambigua se compara $B_{S,T}$ con el valor de corte α (o con 0 si se realiza la transformación de Chernyaev [CHE95]): Si $B_{S,T}$ se encuentra fuera de la superficie ($B_{S,T} < \alpha$) entonces los vóxeles positivos deben ser separados, en caso contrario ($B_{S,T} \geq \alpha$) no deben ser separados. Ambos casos se pueden observar en la Figura 2.14.

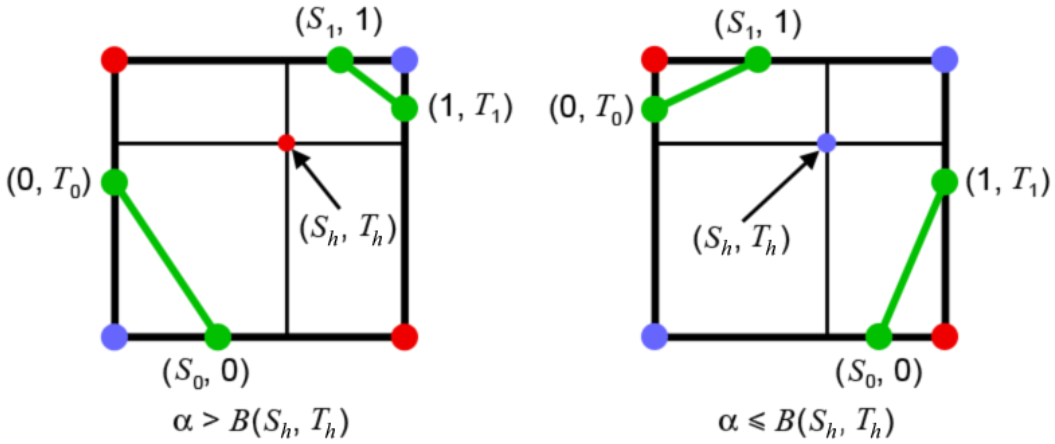

Figura 2.14: Triangulación final después de evaluar la intersección de las asíntotas.

2.2.1.5. RESOLUCIÓN DE LA AMBIGÜEDAD INTERNA

En 1995, Chernyaev [CHE95] propone un método para la resolución de la ambigüedad interna basado en la variación bilineal de $F(x,y,z)$ sobre cualquier plano paralelo a una cara de la celda. Si existen dos áreas positivas separadas en las caras pero unidas dentro de la celda,

entonces existe un plano paralelo a una cara de la celda, donde la cara ambigua formada por la intersección de este plano con la celda es una cara no separada (ver Figura 2.15).

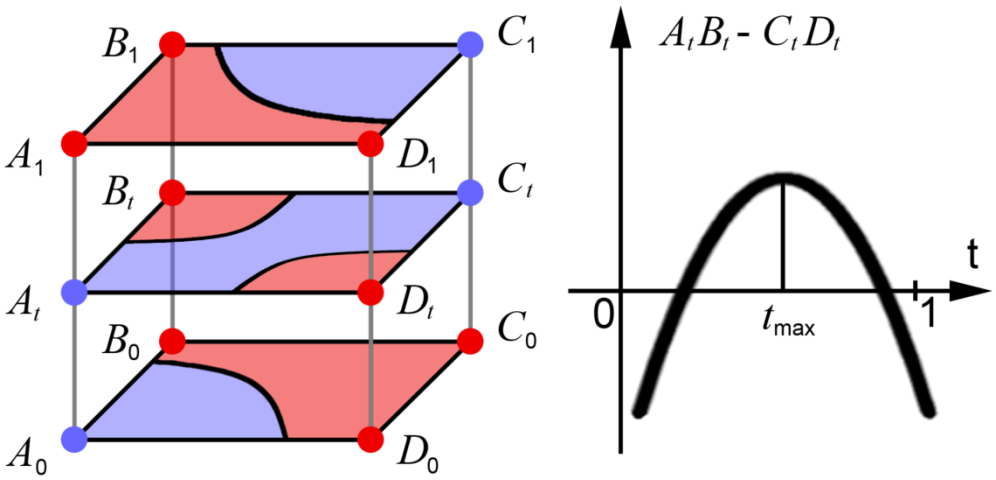

Figura 2.15: Resolución de ambigüedad mediante cara paralela.

Dados A_0, B_0, C_0 y D_0, que representan los valores de los vóxeles de la celda cuando $t = 0$, y A_1, B_1, C_1 y D_1, que representan los valores de los vóxeles de la celda cuando $t = 1$, se puede verificar fácilmente que si las áreas a resolver son positivas y unen los vóxeles A_0 y C_1 entonces:

$$A_0C_0 - B_0D_0 < 0 \qquad (2.1)$$
$$A_1C_1 - B_1D_1 < 0$$

Si las áreas se encuentran unidas dentro de la celda, entonces existe un t tal que:

$$A_t > 0, \quad C_t > 0, \quad A_tC_t - B_tD_t > 0 \qquad (2.2)$$

Debido a que función $F(x, y, z)$ varía linealmente a través de las aristas de la celda, se tiene que la cara media está definida por:

$$A_t = A_0 + (A_1 - A_0)t \qquad (2.3)$$
$$B_t = B_0 + (B_1 - B_0)t$$
$$C_t = C_0 + (C_1 - C_0)t$$
$$D_t = D_0 + (D_1 - D_0)t$$

Sustituyendo (2.3) en (2.2) se tiene que:

$$at^2 + bt + c > 0 \tag{2.4}$$

donde:

$$a = (A_1 - A_0)(C_1 - C_0) - (B_1 - B_0)(D_1 - D_0)$$
$$b = A_0(C_1 - C_0) + C_0(A_1 - A_0) - B_0(D_1 - D_0) - D_0(B_1 - B_0)$$
$$c = A_0C_0 - B_0D_0$$

Las áreas positivas están unidas únicamente en el caso donde la parábola (2.4) es como la que se muestra en la Figura 2.15: la parábola se extiende hacia abajo, el máximo es positivo y se encuentra entre 0 y 1. Estas condiciones se pueden verificar mediante los siguientes pasos:

- Verificar que a sea negativo, para que la parábola esté dirigida hacia abajo.
- Calcular el punto donde se alcanza el máximo t_{max} y verificar que se encuentre en el intervalo $[0, 1]$, donde $t_{max} = \frac{-b}{2a}$.
- Verificar que la cara ambigua formada por A_t, B_t, C_t y D_t cuando $t = t_{max}$ cumple las condiciones establecidas en (2.2).

2.2.2. TETRACUBOS MARCHANTES

Una alternativa a resolver las ambigüedades inherentes a cubos marchantes consiste en descomponer cada una de las celdas de la malla cartesiana (ver Figura 2.1) en varios tetraedros y triangular cada uno de éstos independientemente. Este algoritmo es conocido como Tetracubos Marchantes (*Marching Tetracubes*), el cual fue propuesto por Carneiro et al. [CAR95]. Sin embargo, este método proporciona un resultado diferente al obtenido mediante el uso de Cubos Marchantes, debido a que se asume interpolación lineal a través de los tetraedros en vez de interpolación trilineal con los vóxeles originales de la celda. Si se utiliza interpolación trilineal en las aristas de los tetraedros vuelven a aparecer las ambigüedades [YON95].

2.2.2.1. DIVISIÓN DE LA CELDA EN TETRAEDROS

El primer paso del algoritmo consiste en dividir cada celda de la malla en tetraedros. Para realizar esto se utiliza una tabla de división, la cual indica los cuatro vértices que delimitan cada

uno de los tetraedros a generar a partir de la celda. Es importante que la tabla de división a utilizar cumpla con las siguientes propiedades:

- La unión de las regiones de todos los tetraedros generados debe ser exactamente igual a la región cúbica inicial de la celda.

- Para cada par de tetraedros T_1 y T_2, la intersección de T_1 y T_2 debe ser vacía o alguna cara, arista o vértice común en T_1 y T_2.

- Para todo par de celdas adyacentes, la triangulación de la cara común de ambas celdas debe coincidir.

Carneiro et. al. [CAR95] proponen el uso de una división de la celda en cinco tetraedros, debido a que se generaría a lo sumo el doble de triángulos de salida que en cubos marchantes. No obstante, si se aplica la misma tabla de división a todas las celdas se generarán conexiones erróneas entre celdas adyacentes, como se puede observar en la Figura 2.16.

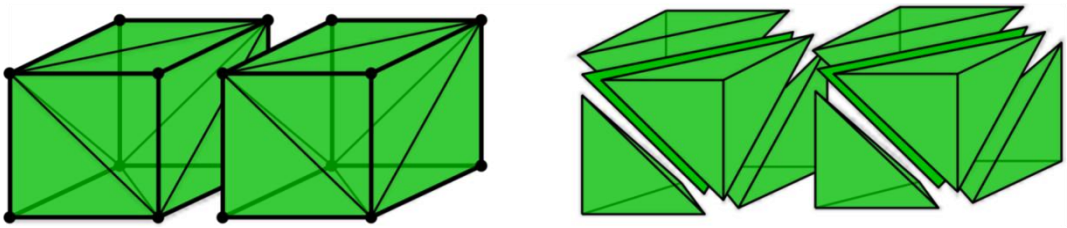

Figura 2.16: Conexión errónea en cubos adyacentes.

Para evitar que este tipo de conexiones ocurran, se debe utilizar una tabla de división para las celdas pares y otra tabla simétrica para las celdas impares, como se puede observar en la Figura 2.17.

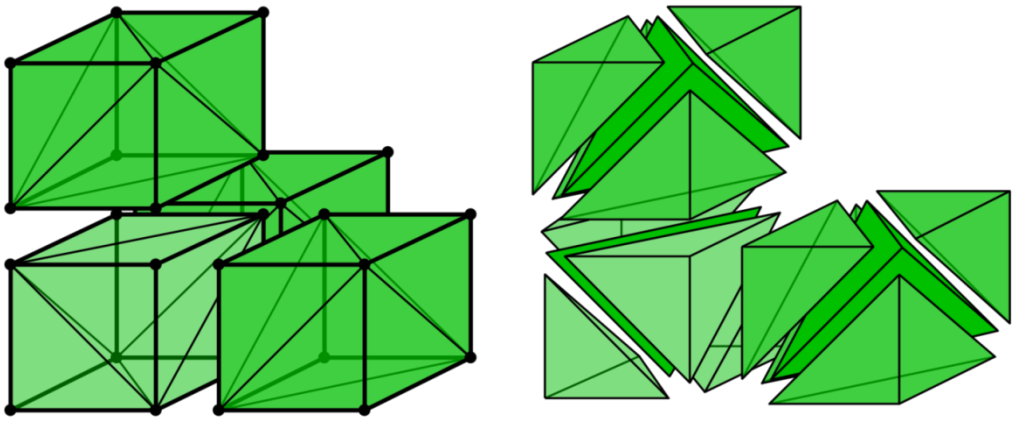

Figura 2.17: Conexión correcta usando dos tablas de divisiones, una para las celdas oscuras (pares) y otra simétrica para las celdas claras (impares).

2.2.2.2. TRIANGULACIÓN DE UN TETRAEDRO

Después de descomponer cada celda en tetraedros, se procede a triangular cada uno de éstos por separado para reconstruir la superficie S_α. Sin embargo, es necesario redefinir $F(x, y, z)$ para realizar interpolación lineal en cada tetraedro de la siguiente forma:

$$F(x, y, z) = Ax + By + Cz + D$$

Para determinar los valores de A, B, C y D, se puede observar que las siguientes ecuaciones se deben cumplir de forma tal que $F(x, y, z)$ aproxime de forma correcta los vóxeles que delimitan al tetraedro:

$$\begin{pmatrix} x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

donde (x_i, y_i, z_i) es la posición del i-ésimo vóxel del tetraedro y v_i es el isovalor del mismo (ver Figura 2.18). Como se puede observar, para determinar los coeficientes A, B, C y D se debe resolver un sistema de cuatro ecuaciones con cuatro incógnitas, el cual tendrá solución única si y sólo si el tetraedro no es degenerado.

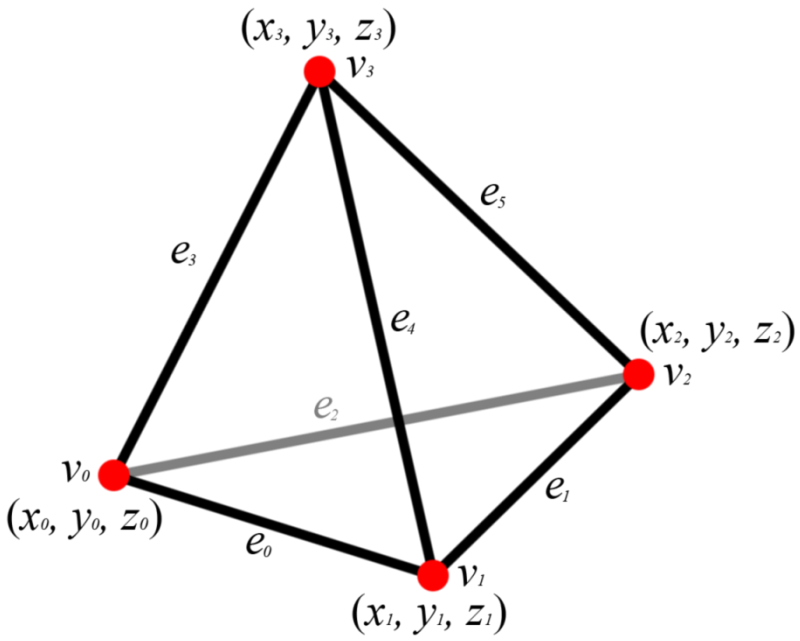

Figura 2.18: Tetraedro utilizado para la extracción de isosuperficies.

Debido a que la interpolación a lo largo del tetraedro es lineal, la interpolación a lo largo de cada arista del tetraedro también es lineal, por lo cual la triangulación de un tetraedro se puede realizar de forma similar a la triangulación de una celda mediante el uso de Cubos Marchantes. El primer paso consiste en determinar cuáles vóxeles del tetraedro se encuentran dentro o fuera de la isosuperficie, con lo cual se construye una máscara de 4 bits, donde el *i*-ésimo bit estará encendido si y solo si el isovalor del *i*-ésimo vóxel del tetraedro es mayor o igual que el isovalor α, como se muestra en la Figura 2.19.

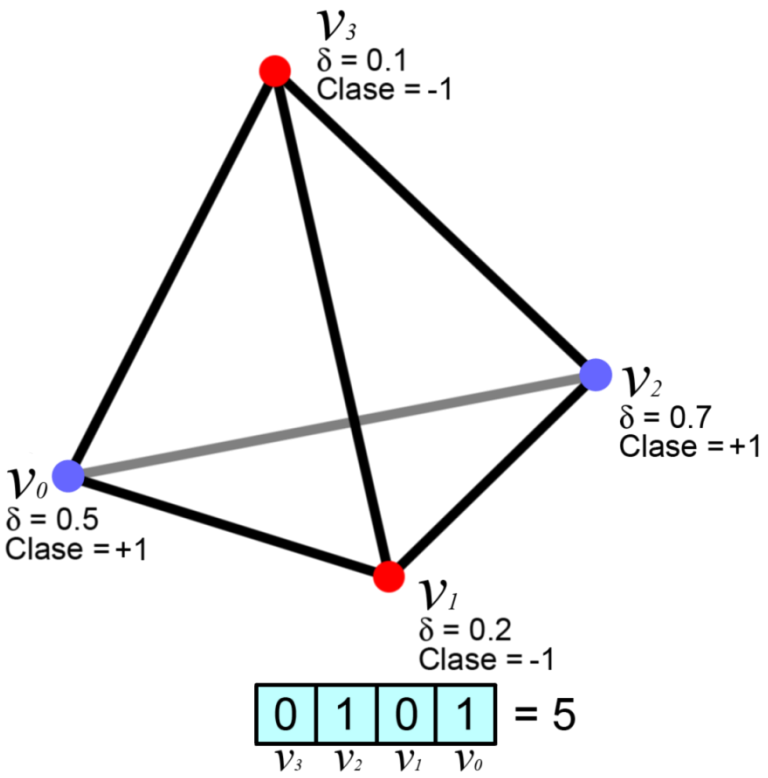

Figura 2.19: Creación de la máscara para el tetraedro utilizando $\alpha = 0.3$, donde cada δ representa el isovalor en cada vértice que delimita el tetraedro.

Debido a que cada uno de los vóxeles tiene dos estados posibles, existe un total de $2^4 = 16$ formas diferentes en que la superficie a extraer puede intersectar el tetraedro. De forma similar a Cubos Marchantes, por medio de rotaciones, simetría y complementariedad se pueden reducir los dieciséis casos a tres casos topológicamente diferentes, los cuales se muestran en la Figura 2.20.

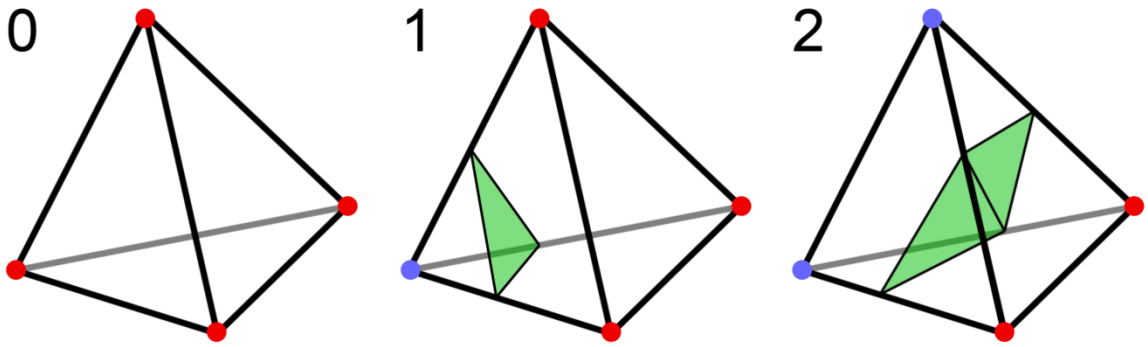

Figura 2.20: Posibles intersecciones de la isosuperficie con un tetraedro.

El siguiente paso es determinar cuáles aristas del tetraedro son intersectadas por la superficie y dónde se encuentra el punto de intersección. Una arista es intersectada por la superficie si y

sólo si uno de sus extremos se encuentra marcado y el otro no. Para determinar el punto P_x donde la arista es intersectada por la superficie se puede utilizar interpolación lineal sobre la misma de la siguiente manera:

$$P_x = P_0 + (P_1 - P_0)\left(\frac{\alpha - v_0}{v_1 - v_0}\right)$$

donde P_0 y P_1 son las posiciones de los extremos de la arista, v_0 y v_1 son los isovalores de la función en los extremos de las aristas y α es el umbral de la superficie que se desea extraer. Después de calcular todos los puntos de intersección de la superficie con el tetraedro, se generan los triángulos utilizando una tabla de conectividad de 16 casos, la cual se indexa utilizando la máscara obtenida de la clasificación de los vóxeles.

2.3. EXTRACCIÓN DE INTERVALOS DE VOLUMEN

Para la visualización de volúmenes existen dos técnicas diferentes [FUJ95]: visualización directa de volumen y extracción de isosuperficies (o DVR[4] y SF[5], por sus siglas en inglés, respectivamente). Sin embargo, la visualización directa de volumen tiene la desventaja de requerir mucho poder de procesamiento y experiencia por parte del usuario para inicializar la función de transferencia. Por otro lado, mediante extracción de isosuperficies solamente se visualiza una parte del volumen, lo que ocasiona que el usuario no observe la información dentro del contexto.

Debido a esto, se introdujo el concepto de extracción de intervalo de volumen, el cual consiste en el cálculo de la parte del volumen dentro del rango $[\alpha, \beta]$, la cual se define como:

$$\mathrm{IV}(\alpha, \beta) = \{(x, y, z) | \alpha \leq F(x, y, z) \leq \beta\},$$

el cual puede ser visualizado de diferentes maneras, dependiendo de la representación utilizada por el algoritmo de extracción.

Como se puede observar, este enfoque es una generalización de DVR y SF, debido a que permite visualizar el volumen completo utilizando $\mathrm{IV}(-\infty, \infty)$, y también permite visualizar la

[4] DVR: *Direct Volume Rendering*.
[5] SF: *Surface Fitting*.

isosuperficie S_δ utilizando $\mathrm{IV}(\delta, \delta)$. Inclusive, se puede visualizar S_δ utilizando una tolerancia de 2ε utilizando $\mathrm{IV}(\delta - \varepsilon, \delta + \varepsilon)$, lo cual es importante a la hora de trabajar con datos que puedan estar sujetos a pequeñas perturbaciones.

Otra ventaja de los intervalos de volumen sobre la extracción de isosuperficies es el manejo correcto de las celdas con valor constante α, ya que los métodos de extracción de isosuperficies no muestran nada, mientras que al visualizar $\mathrm{IV}(\alpha, \alpha)$ se observará toda la celda, y para todo $\delta \neq \alpha$, al visualizar $\mathrm{IV}(\delta, \delta)$ no se observará nada.

Otra forma de visualizar un subconjunto del volumen equivalente a $\mathrm{IV}(\alpha, \beta)$ consiste en utilizar DVR configurando la opacidad de la función de transferencia de forma que el intervalo $]\alpha, \beta[$[6] sea completamente transparente. Sin embargo, este procedimiento tiene la desventaja de tener que procesar todo el volumen cuando el punto de vista del usuario cambia, mientras que mediante el uso de intervalos de volumen se ahorra tiempo de procesamiento durante la proyección, debido a que sólo se debe recorrer la parte del volumen correspondiente al intervalo de interés.

A continuación se describirán los tres algoritmos principales para la extracción de intervalos de volumen, el primero está basado en triangulaciones de Delaunay y formas alfa, el segundo está basado en cubos marchantes y el último en tetracubos marchantes.

2.3.1. EXTRACCIÓN POR MEDIO DE FORMAS ALFA[7]

Este algoritmo fue propuesto por Guo [GUO95], el cual consiste en aproximar el intervalo $[A, B]$ del volumen mediante el uso de una forma alfa S_α. A continuación, se introducirán los complejos de *simplices*[8], las formas alfa, y el procedimiento utilizado para la construcción de éstas con el fin de aproximar el intervalo $[A, B]$ de un volumen de datos.

[6] $]A, B[= \{x \in \mathbb{R} | x \notin (A, B)\}$ **[GRA94]**
[7] "Formas alfa" viene del término en inglés *α-shapes*.
[8] *Simplices*: plural de *simplex*.

2.3.1.1. COMPLEJOS DE SIMPLICES[9]

Un n-*simplex* Δ_N se define como un conjunto de $(n+1)$ puntos en $\Re^m$, con $m \geq n$, donde ningún punto puede ser expresado como una combinación afín de los demás, es decir:

$$\bigwedge_{i \in \{0,1,\cdots,n\}} \overline{\left[\bigvee_{\alpha \in \Re^{n+1}} \left(\sum_{j=0, j \neq i}^{n} \alpha_j = 1 \wedge \sum_{j=0, j \neq i}^{n} \alpha_j \Delta_{N,j} = \alpha_i \right) \right]}$$

Por ejemplo, un 0-*simplex* es un punto, un 1-*simplex* es una línea, un 2-*simplex* es un triángulo, un 3-*simplex* es un tetraedro, y así sucesivamente.

La frontera de un n-*simplex* Δ_N se define como el conjunto de todos los k-*simplices* Δ_K, tal que $k < n$ y todos los puntos de Δ_K se encuentran en Δ_N, es decir:

$$\text{boundary}(\Delta_N) = \{\Delta_K | \Delta_K \subset \Delta_N\}$$

Por ejemplo, la frontera de un tetraedro está compuesta por 4 triángulos, 6 líneas y 4 puntos, y la frontera de un triángulo está compuesta por 3 líneas y 3 puntos.

Sea C un conjunto de *simplices* de distintas dimensiones, C es un complejo de *simplices* si y sólo si se cumplen las siguientes condiciones:

- Para cada *simplex* $\Delta_T \in C$, la frontera de Δ_T se encuentra en C.
- Para todo par de *simplices* $(\Delta_{T1}, \Delta_{T2})$ donde $\Delta_{T1}, \Delta_{T2} \in C$, su intersección se encuentra en la frontera de Δ_{T1} y en la frontera de Δ_{T2}.

El espacio subyacente $|C|$ de un complejo de *simplices* C es la unión de todos los *simplices* contenidos en C, y un complejo de *simplices* C' es un subcomplejo de C si y solo si $C' \subset C$.

2.3.1.2. FORMAS ALFA

Sea S un conjunto de puntos en $\Re^n$, Edelsbrunner y Mucke [EDE94] definen la familia de formas alfa $\{S_\alpha | 0 < \alpha \leq \infty\}$ en base a una familia de subcomplejos de la triangulación de

[9] "Complejos de *simplices*" viene del término en inglés *simplicial complexes*.

Delaunay $\mathrm{D}(S)$, de los cuales se derivan éstas como los espacios subyacentes de dichos subcomplejos.

Para un *n-simplex* Δ_T, sea O_T la esfera cerrada más pequeña cuya frontera contenga Δ_T. Los complejos alfa $\{C_\alpha | 0 < \alpha \leq \infty\}$ están compuestos por los siguientes elementos: (a) todos los elementos $\Delta_T \in \mathrm{D}(S)$, tal que la esfera O_T de Δ_T tenga radio estrictamente menor que α y no haya punto de S dentro de la esfera abierta[10] limitada por O_T, y (b) la frontera de todos los elementos descritos en (a). Finalmente, se define para cada α $(0 \leq \alpha \leq \infty)$ la forma alfa S_α como el espacio subyacente del complejo C_α, es decir, $S_\alpha = |C_\alpha|$.

Por ejemplo, dado un conjunto de puntos S en $\Re^3$, la forma S_α consiste en la triangulación de Delaunay $\mathrm{D}(S)$, donde los tetraedros cuyas esferas tienen radio mayor o igual a α son sustituidos por las cuatro caras triangulares que los delimitan. Luego, aquellos triángulos cuyas esferas tienen radio mayor o igual a α son sustituidos por las tres líneas que los delimitan, y así sucesivamente hasta llegar a los vértices.

2.3.1.3. CONSTRUCCIÓN DE LAS FORMAS ALFA

La construcción de una forma alfa S_α para aproximar el intervalo $[A, B]$ de un volumen de datos se realiza en tres pasos: evaluar el volumen para generar un conjunto de puntos S en el intervalo $[A, B]$, construir la triangulación de Delaunay $\mathrm{D}(S)$ y generar el complejo C_α cuyo espacio subyacente $|C_\alpha|$ sea S_α.

El conjunto S de puntos a utilizar incluye todas aquellas muestras $V_{i,j,k}$ del volumen que se encuentren dentro del rango $[A, B]$. Sin embargo, este conjunto no permite la construcción de los bordes del intervalo, por lo cual se añaden los puntos de las isosuperficies S_A y S_B, donde:

$$S_A = \{(x, y, z) | F(x, y, z) = A\}$$
$$S_B = \{(x, y, z) | F(x, y, z) = B\}$$

los cuales se calculan de forma similar a como se calculan los vértices de una isosuperficie mediante el uso de Cubos Marchantes. En la Figura 2.21 se pueden observar los puntos

[10] Una esfera abierta con centro C y radio r de define como todos aquellos puntos cuya distancia a C es estrictamente menor que r.

pertenecientes a S en un mallado bidimensional, junto con la aproximación del borde del intervalo de volumen mediante líneas.

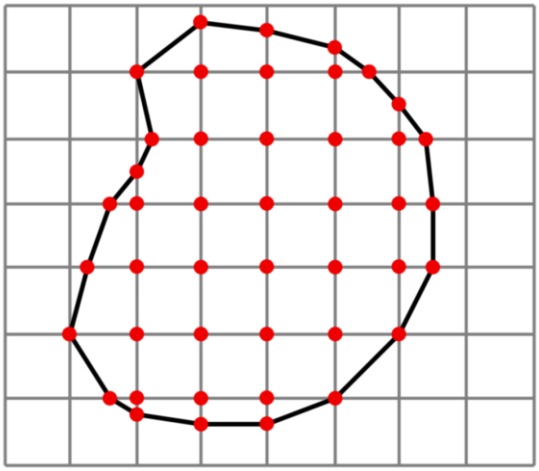

Figura 2.21: Mallado bidimensional a procesar mediante formas alfa, los puntos pertenecientes a S se muestran con puntos rojos, y la aproximación del borde del intervalo de volumen mediante el uso de *simplices* se muestra con líneas negras.

Después de calcular los puntos pertenecientes a S, se procede a calcular la triangulación de Delaunay $\mathrm{D}(S)$, lo cual se puede realizar utilizando cualquier algoritmo existente para este propósito. No obstante, se debe preprocesar la nube de puntos para que se encuentre en posición general, es decir, no deben existir cuatro o más puntos coplanares ni deben existir cinco o más puntos en la frontera de la misma esfera. Sin embargo, cuando se trabaja con mallas cartesianas uniformes regulares, la nube de puntos S generalmente contiene múltiples grupos de puntos coplanares, específicamente todos aquellos pertenecientes a un mismo corte cuyos isovalores se encuentren dentro del rango $[A, B]$. Para solucionar esto, se puede utilizar una técnica llamada Simulación de Simplicidad[11] [EDE90], la cual consiste en perturbar la nube de puntos infinitesimalmente con el objetivo de llevar ésta a posición general.

Una vez que se tiene la triangulación de Delaunay $\mathrm{D}(S)$, la tarea principal para la extracción de S_α consiste en elegir un valor α apropiado para la generación del volumen. En general, a medida que α disminuye, las características gruesas del volumen son sustituidas por características más finas. Sin embargo, después que α disminuye debajo de cierto valor $\breve{\alpha}$,

[11] También llamado SoS, por sus siglas en inglés (*Simulation of Simplicity*).

empiezan a aparecer cavidades en S_α, lo cual es indeseable debido a que el intervalo debe ser representado principalmente por tetraedros.

Para determinar el valor de $\breve{\alpha}$, se analizará el complejo alfa C_α en un cubo unitario. Los elementos incluidos en C_α dependen de las siguientes condiciones:

- Si $\alpha > 0$, existen esferas con radio positivo tal que contengan cada vértice del cubo, por lo cual los ocho vértices del cubo pertenecen a C_α.

- Si $\alpha > \frac{1}{2}$, existen esferas con radio menor que α que contienen cada arista del cubo, por lo cual las doce aristas del cubo pertenecen a C_α.

- Si $\alpha > \frac{\sqrt{2}}{2}$, existen esferas con radio menor que α que contienen cada cara del cubo, por lo cual los doce triángulos que componen las caras del cubo pertenecen a C_α.

- Si $\alpha > \frac{\sqrt{3}}{2}$, existe una esfera con radio menor que α que contiene el cubo, por lo cual los seis tetraedros que componen al cubo pertenecen a C_α.

En consecuencia, para un cubo unitario se tiene que $\breve{\alpha} = \frac{\sqrt{3}}{2} + \varepsilon$, donde ε es un valor muy pequeño, cuyo objetivo es prevenir que errores de redondeo ocasionen la aparición de cavidades dentro del modelo. Luego, para un cubo de lado d se tiene que $\breve{\alpha} = \left(\frac{\sqrt{3}}{2} + \varepsilon\right)d$, debido a que todas las primitivas analizadas anteriormente también son escaladas en la misma proporción que el cubo con relación al cubo unitario. Finalmente, se calcula y visualiza $S_{\breve{\alpha}}$, la cual representará el intervalo $[A, B]$ del volumen.

2.3.2. EXTRACCIÓN POR MEDIO DE CUBOS MARCHANTES

En 1995, Fujishiro, Maeda y Sato [FUJ95] publicaron un método para la extracción de intervalos de volumen, el cual consiste en una adaptación de cubos marchantes para extraer un mallado triangular que encierre aquellas partes del volumen que se encuentren dentro del intervalo $[\alpha, \beta]$ a extraer. De forma similar a como se realiza en cubos marchantes, la extracción del intervalo se realiza celda por celda independientemente.

A continuación, se explicarán las consideraciones locales y globales que se deben tomar en cuenta al momento de la extracción del intervalo de volumen $[\alpha, \beta]$ mediante esta técnica.

2.3.2.1. EXTRACCIÓN LOCAL DEL INTERVALO DE VOLUMEN

Para extraer los triángulos de la celda que representen el intervalo $[\alpha, \beta]$, se puede adaptar Cubos Marchantes de dos formas diferentes. La primera, consiste en clasificar cada uno de los vóxeles utilizando la comparación entre su isovalor δ_i y el intervalo $[\alpha, \beta]$ de la siguiente manera:

$$\text{clase}(\delta_i, \alpha, \beta) = \begin{cases} -1, & si\ \delta_i < \alpha \\ +1, & si\ \delta_i > \beta \\ 0, & si\ \alpha \leq \delta_i \leq \beta \end{cases}$$

Por ejemplo, en la Figura 2.22 se puede observar una celda con los isovalores de cada vóxel, así como la clasificación de cada uno de ellos.

Después de clasificar los vóxeles de la celda, se puede construir un índice mediante la interpretación de las clases de los vóxeles en base-3, el cual se utiliza para acceder a una tabla de conectividad de $3^8 = 6561$ casos, de donde se obtendrán los triángulos a generar a partir de la celda. Para la resolución de ambigüedades se pueden aplicar las mismas técnicas aplicadas en cubos marchantes [NIE91][CHE95].

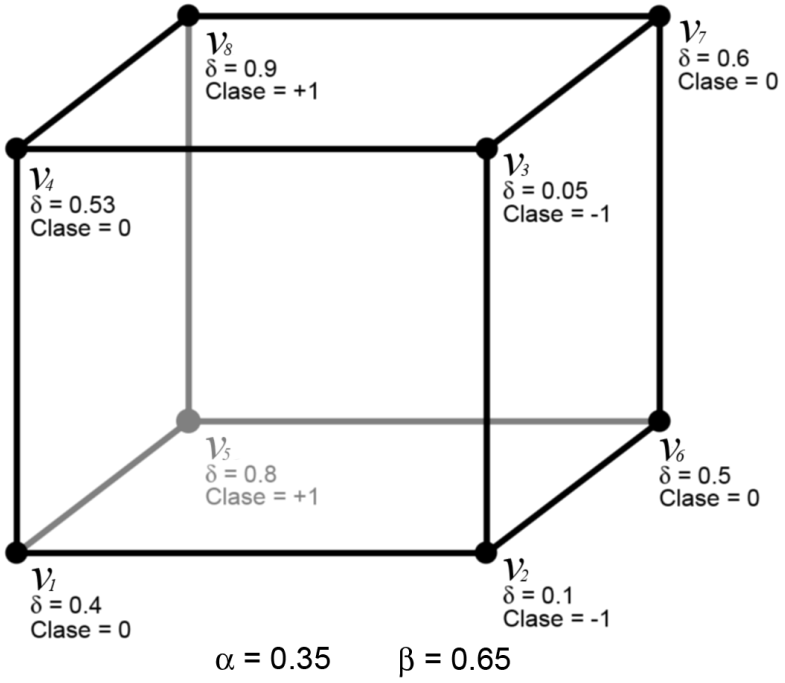

Figura 2.22: Clasificación de celda en base a los isovalores en sus vóxeles, donde cada δ representa el isovalor en cada vértice que delimita la celda.

Sin embargo, se pueden aprovechar las operaciones de conjuntos para simplificar la cantidad de casos a tomar en cuenta, mediante la expresión del intervalo $[\alpha, \beta]$ como la intersección de los intervalos $[\alpha, \infty)$ y $(\alpha, \beta]$, es decir:

$$\mathrm{IV}(\alpha, \beta) = \mathrm{IV}(-\infty, \beta) \cap \mathrm{IV}(\alpha, \infty)$$

Para representar los intervalos de volumen $\mathrm{IV}(\alpha, \infty)$ y $\mathrm{IV}(-\infty, \beta)$ se utilizan poliedros, también llamados cubos-α y cubos-β, respectivamente. Esto permite simplificar la extracción del intervalo en cada celda a únicamente $2^8 = 256$ casos, debido a que cada uno de los vóxeles sólo tendrá dos clasificaciones posibles, dentro del intervalo o fuera de éste.

Para extraer el cubo-α de una celda del volumen, se modifica la tabla de casos de Cubos Marchantes para extraer la subregión positiva de la celda cuya frontera es la isosuperficie S_α, la cual se determina en base a los isovalores de los vóxeles de la celda. Para extraer el cubo-β se realiza el mismo procedimiento, pero extrayendo la subregión negativa de la celda cuya frontera es S_β. En la Figura 2.23 se pueden observar los poliedros generados para algunas configuraciones de los isovalores de las celdas.

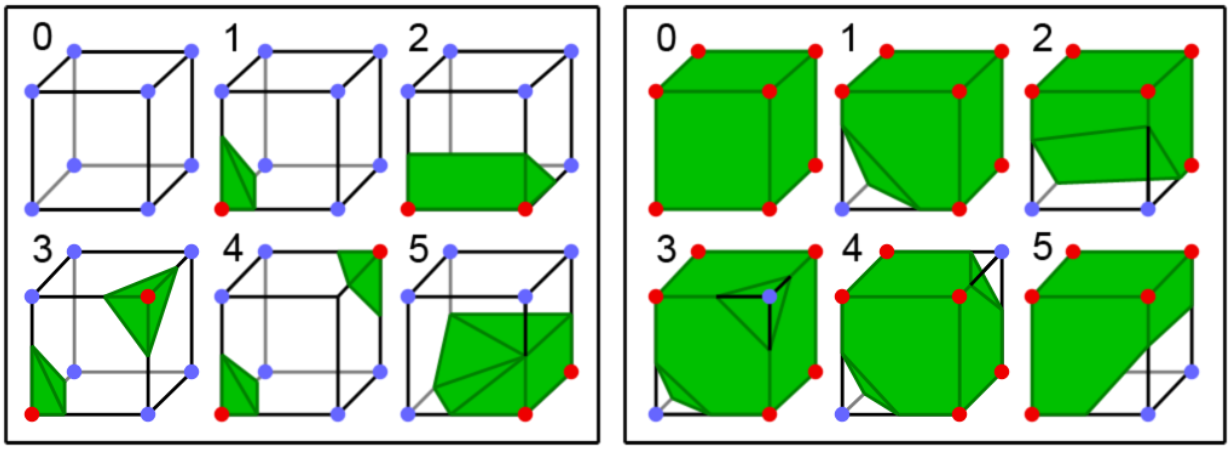

Figura 2.23: Algunas configuraciones posibles durante la extracción de los cubos-α (izquierda) y los cubos-β (derecha).

En el caso general donde $\alpha < \beta$, la intersección de los cubos-α y los cubos-β de las celdas no es vacía, siempre y cuando ésta sea intersectada por el intervalo a extraer (ver Figura 2.24). Debido al teorema del valor medio, se pueden determinar los siguientes hechos:

- Entre un vóxel negativo y uno positivo existirá un punto con valor α y otro con valor β.
- Entre un vóxel negativo y un vóxel neutro existirá un punto con valor α.
- Entre un vóxel positivo y un vóxel neutro existirá un punto con valor β.

Para calcular la posición de los puntos donde el cubo- α o el cubo- β intersectan las aristas de la celda se realiza un procedimiento similar al utilizado en cubos marchantes. El punto $P_E(\delta)$ donde una arista E tiene valor δ está dado por:

$$P_E(\delta) = P_0 + (P_1 - P_0)\frac{\delta - v_0}{v_1 - v_0}$$

donde P_0 y P_1 son las posiciones de los extremos de la arista E, y v_0 y v_1 son los isovalores de los vóxeles en los extremos de la arista E. También se pueden calcular los valores de otros atributos $X_E(\delta)$ en el punto $P_E(\delta)$ de la siguiente manera:

$$X_E(\delta) = X_0 + (X_1 - X_0)\frac{\delta - v_0}{v_1 - v_0}$$

donde X_0 y X_1 son los valores del atributo en los extremos de la arista E.

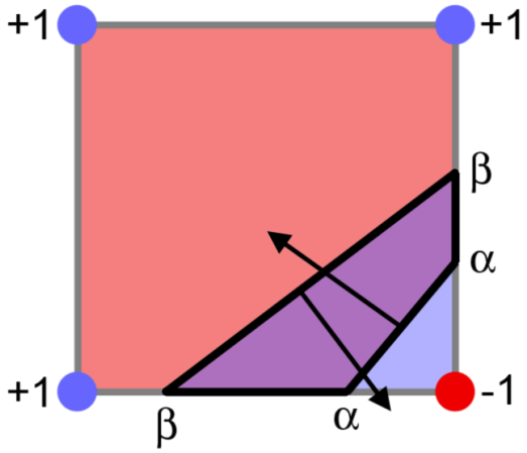

Figura 2.24: Caso general donde $\alpha < \beta$. La zona de color rojo pertenece únicamente al cubo- α, la zona de color azul pertenece únicamente al cubo- β, y la zona de color violeta pertenece a ambos cubos. Los bordes del polígono extraído de la celda se encuentran resaltados.

Para verificar que este enfoque es una generalización de la extracción de isosuperficies, se puede observar si $\alpha = \beta$, la intersección del cubo- α y del cubo- β en cada celda será igual a la isosuperficie $S_\alpha = S_\beta$, como se puede observar en la Figura 2.25.

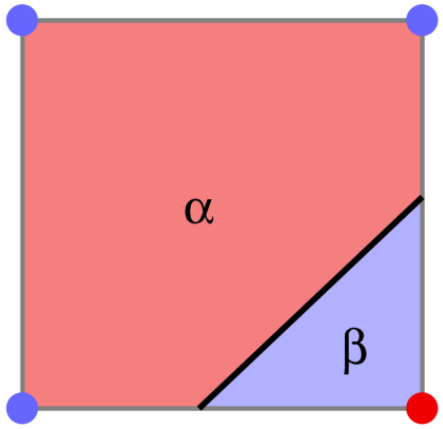

Figura 2.25: Caso especial donde $\alpha = \beta$. La zona de color rojo pertenece únicamente al cubo- α, la zona de color azul pertenece únicamente al cubo- β, y el polígono extraído se encuentra resaltado.

2.3.2.2. EXTRACCIÓN GLOBAL DEL INTERVALO DE VOLUMEN

A diferencia de cubos marchantes, es importante tener en cuenta la posición de la celda en el volumen al momento de la extracción del intervalo en ésta. Esto se debe a que los bordes de las celdas son incluidos al momento de realizar la intersección de los cubos- α y cubos- β, lo que ocasiona que en el mallado final existan aristas compartidas por más de dos triángulos.

Para solucionar esto, se deben ignorar aquellas caras de los poliedros extraídos de la intersección de los cubos-α y los cubos-β que se encuentren alineados a alguna de las caras de las celdas que no den al exterior del volumen (ver Figura 2.26). Las caras de los poliedros que dan al exterior no son ignoradas, de forma tal que el mallado generado sea cerrado.

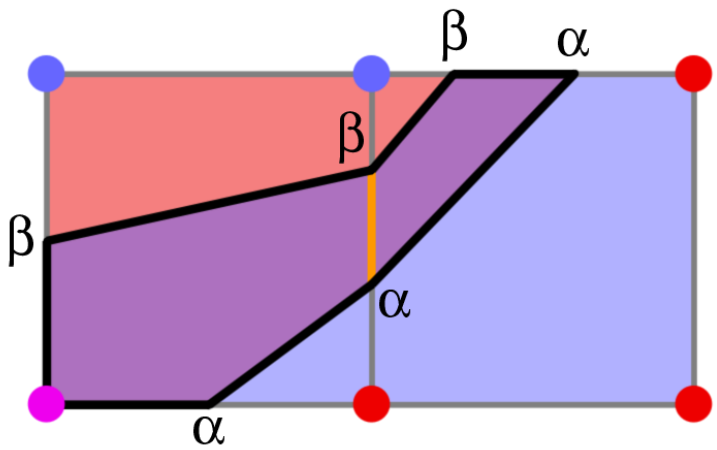

Figura 2.26: Intervalo de volumen en celdas adyacentes. Las líneas negras representan las caras del poliedro final, y la línea anaranjada entre las dos caras representa la cara interna ignorada en el mallado final.

2.3.3. EXTRACCIÓN POR MEDIO DE TETRACUBOS MARCHANTES

En 1997, Nielson y Sung [NIE97b] publicaron un método para la extracción de intervalos de volumen, el cual consiste en una adaptación de tetracubos marchantes para extraer un mallado tetraédrico que aproxime aquellas partes del volumen que se encuentren dentro del intervalo $[\alpha, \beta]$ a extraer. De forma similar a como se realiza en tetracubos marchantes, la extracción del intervalo se realiza celda por celda.

Los pasos a seguir para la extracción del mallado tetraédrico mediante esta técnica se pueden observar en la Figura 2.27. Primero, el algoritmo procesa el volumen como un conjunto de celdas independientes, las cuales son divididas en varios tetraedros, dentro de los cuales se asume que los valores varían linealmente. La división de cada celda en tetraedros se realiza de la misma manera como se realiza en tetracubos marchantes (ver Sección 2.2.2.1). Después de dividir la celda en tetraedros, para cada uno de estos se calculan aquellos puntos de las aristas donde estas se evalúan a α o β y aquellos vóxeles que se encuentran dentro del intervalo. Una vez calculados

estos puntos se obtiene un poliedro-$\alpha\beta$, el cual posteriormente es dividido en uno o más tetraedros de salida.

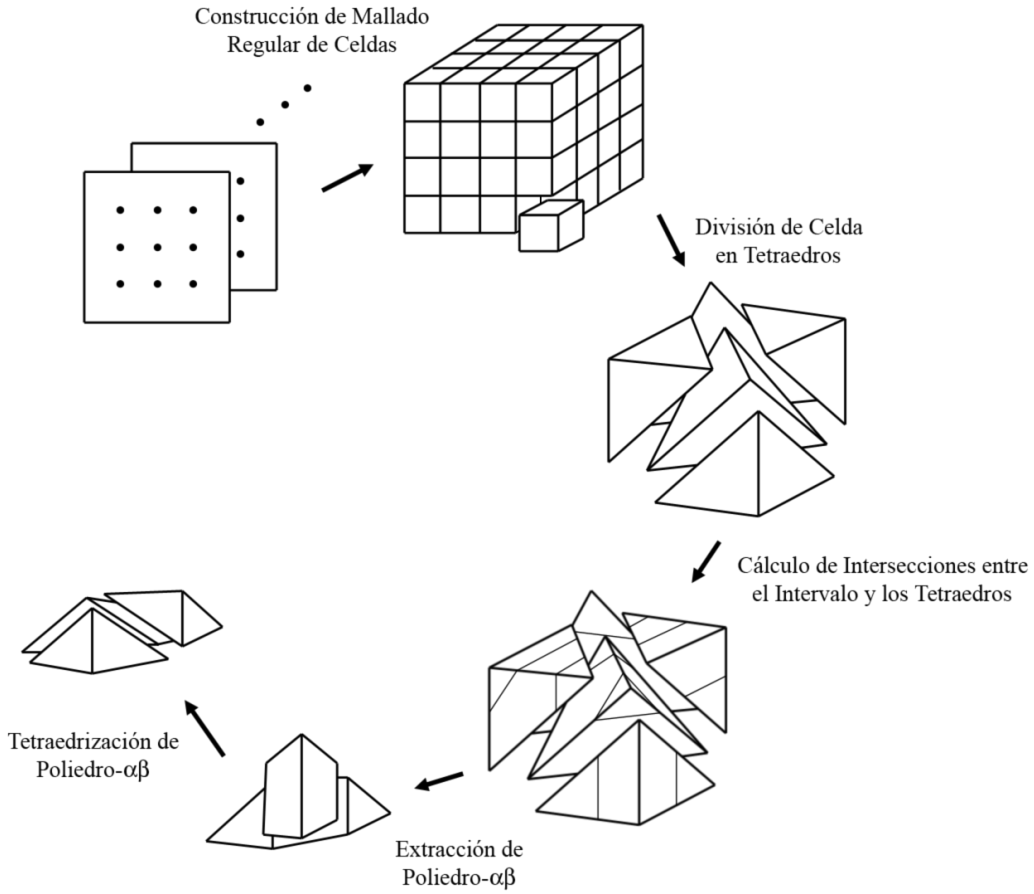

Figura 2.27: Pasos necesarios para la tetraedrización de un intervalo de volumen.

A continuación, se describirá el proceso para la extracción del poliedro-$\alpha\beta$ a partir del tetraedro de entrada, así como el proceso para dividir el poliedro-$\alpha\beta$ obtenido en tetraedros de salida.

2.3.3.1. EXTRACCIÓN DEL POLIEDRO ALFA-BETA

Esta sección explica el procedimiento que se debe llevar a cabo para la extracción del poliedro-$\alpha\beta$ a partir de un tetraedro de entrada. Como se mencionó anteriormente, se asume que se tiene el valor de la función $F(x, y, z)$ en los vértices del tetraedro y que los valores internos se

obtienen mediante el uso de interpolación lineal, de la misma forma como se realiza en tetracubos marchantes (ver Sección 2.2.2.2).

El primer paso que se debe llevar a cabo es la clasificación de cada uno de los vértices del tetraedro en base a su isovalor de la siguiente manera:

$$\text{clase}(\delta_i, \alpha, \beta) = \begin{cases} -1, & si\ \delta_i < \alpha \\ +1, & si\ \delta_i > \beta \\ 0, & si\ \alpha \leq \delta_i \leq \beta \end{cases}$$

donde δ_i es el isovalor del vértice a preprocesar.

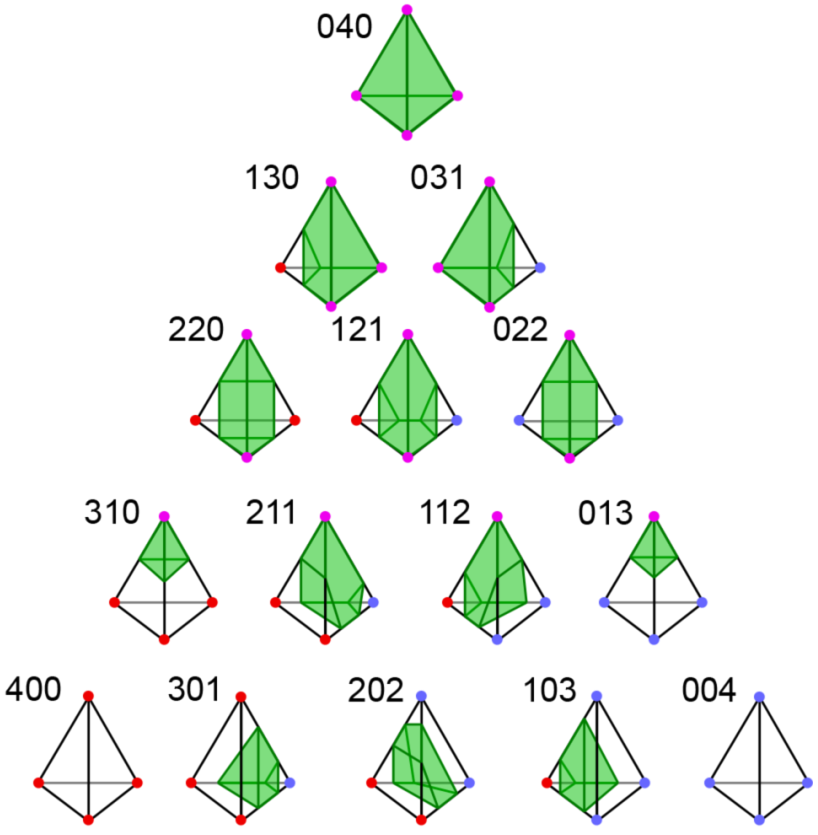

Figura 2.28: Casos posibles durante la extracción del poliedro- $\alpha\beta$ a partir del tetraedro. Los vértices rojos se encuentran por debajo de α (negativos), los vértices azules se encuentran por arriba de β (positivos) y los vértices violetas se encuentran dentro del intervalo a extraer (neutros).

Una vez clasificados los vértices del tetraedro, se construye un índice entero mediante la interpretación de las clasificaciones de los vértices como dígitos en base-3, lo cual genera un total de $3^4 = 81$ casos. Sin embargo, dos tetraedros son equivalentes por medio de rotaciones y

reflexiones si poseen la misma cantidad de vértices de cada clase, debido a que todos éstos se encuentran conectados entre sí por medio de aristas. Esto nos permite reducir el total de casos a 15, los cuales se pueden observar en la Figura 2.28, donde el primer dígito del caso indica la cantidad de vértices negativos (clase -1), el segundo dígito indica la cantidad de vértices neutros (clase 0) y el tercer dígito indica la cantidad de vértices positivos (clase $+1$).

Es importante observar que los poliedros-$\alpha\beta$ generados en cada caso son convexos y poseen caras planas, debido a que la interpolación dentro del tetraedro es lineal. Esto facilita la tetraedrización del poliedro en el siguiente paso.

2.3.3.2. TETRAEDRIZACIÓN DEL POLIEDRO ALFA-BETA

En esta sección se explicará el procedimiento que se debe llevar a cabo para descomponer el poliedro-$\alpha\beta$ de cada celda en tetraedros. Es importante destacar que los poliedros-$\alpha\beta$ son tetraedrizables, debido a que éstos son convexos [NIE97a]. Sin embargo, si no se toman medidas especiales al llevar a cabo la tetraedrización de cada poliedro, puede ocurrir que la tetraedrización de un poliedro no coincida con la de otro poliedro adyacente, lo cual sucede cuando en la cara compartida por los poliedros se elije una triangulación de un lado y otra triangulación diferente en el otro (ver Figura 2.29).

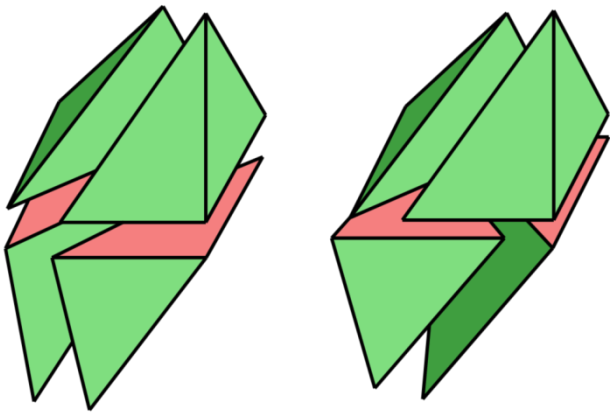

Figura 2.29: Posibles tetraedrizaciones alrededor de la cara roja, la cual es compartida por dos poliedros diferentes. A la izquierda se muestra la tetraedrización correcta, a la derecha se muestra la incorrecta.

Para solucionar este problema, se debe establecer un método para escoger de forma consistente la triangulación a utilizar en cada cara del poliedro. Nielson [NIE97b] propone utilizar la regla de conexión por índices (*index connection rule*), para lo cual es necesario

establecer una relación de orden total entre los vértices del mallado final. Sea V el conjunto de vértices del mallado final, una relación de orden total sobre V consiste en un predicado $\rho(u, v)$ que permita comparar cada par de vértices $u, v \, \varepsilon \, V$, con el fin de determinar si el vértice u precede a v. Este predicado debe cumplir las siguientes propiedades:

- **Propiedad reflexiva:** Para todo vértice v, el predicado ρ debe indicar que v precede a v, es decir:

$$\bigwedge_{u,v \in V} \rho(v, v)$$

- **Propiedad antisimétrica:** Para todo par de vértices $u, v \in V$, si u precede a v y v precede a u, entonces u debe ser igual a v, es decir:

$$\bigwedge_{u,v \in V} \rho(u, v) \wedge \rho(v, u) \Longrightarrow u = v$$

- **Propiedad transitiva:** Para todo trío de vértices $u, v, w \in V$, si u precede a v y v precede a w, entonces u debe preceder a w, es decir:

$$\bigwedge_{u,v,w \in V} \rho(u, v) \wedge \rho(v, w) \Longrightarrow \rho(u, w)$$

- **Propiedad de orden total:** Para todo par de vértices $u, v \in V$, u precede a v o v precede a u, es decir:

$$\bigwedge_{u,v \in V} \rho(u, v) \vee \rho(v, u)$$

Sea $W \subseteq V$ tal que $W \neq \emptyset$ y sea $u \in W$, u es un elemento minimal de W si y solo si no existe otro elemento $v \in W$ tal que $v \neq u$ y v preceda a u, es decir:

$$\bigwedge_{\substack{v \in V \\ v \neq u}} \overline{\rho(v, u)}$$

Debido a la propiedad de orden total W sólo posee un elemento minimal, debido a que si existieran dos elementos $u, v \in W$ entonces u no debe preceder a v y v no debe preceder a u, lo cual contradice la propiedad de orden total del predicado ρ.

Después de establecer el predicado de ordenamiento, éste se utiliza para determinar unívocamente la triangulación a utilizar en cada cara de cada poliedro- $\alpha\beta$, independientemente del poliedro al que pertenezca. Como se puede observar en la Figura 3.8, sólo existen tres tipos de cara a procesar: triángulos, cuadriláteros y pentágonos, los cuales se triangulan de la siguiente manera (ver **Figura 2.30**):

- **Triángulos:** En este caso no existe ningún problema, debido a que sólo hay una triangulación posible.

- **Cuadriláteros:** Se elige aquella triangulación donde se utiliza la diagonal entre el menor vértice y su vértice opuesto.

- **Pentágonos:** En la Figura 2.28 se puede observar que todas las caras pentagonales de los poliedros- $\alpha\beta$ tienen un vértice neutro, por lo cual se divide con la triangulación que utiliza las diagonales entre este vértice y sus vértices opuestos.

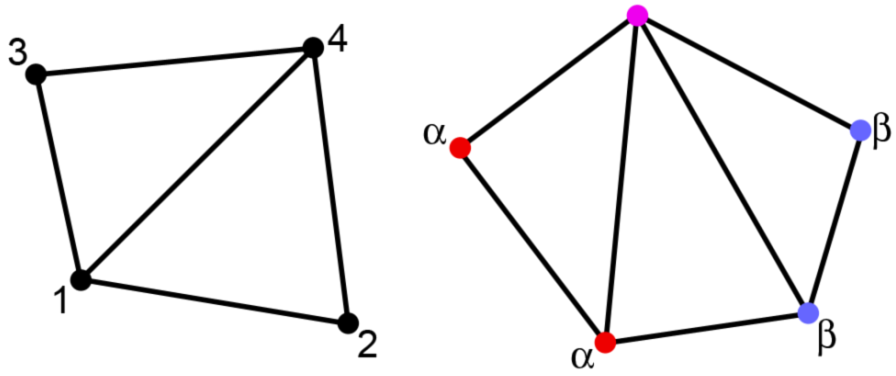

Figura 2.30: Regla de conexión por índices para cuadriláteros (izquierda) y para pentágonos (derecha).

A continuación, se analizarán cada uno de los casos posibles para las triangulaciones de cada tipo de poliedro en la Figura 3.8, los cuales se clasifican como tetraedros, prismas, cristales y cubos.

A. TETRAEDROS

En los casos 040, 310 y 013 el poliedro-$\alpha\beta$ es un tetraedro, el cual consiste en un poliedro de cuatro vértices con cuatro triángulos. En estos casos, no es necesario ningún procesamiento o análisis adicional, debido a que éste no debe ser dividido.

B. PRISMAS

En los casos 031, 022, 103, 130, 220 y 301 el poliedro-$\alpha\beta$ es un prisma, el cual consiste en un poliedro de seis vértices, con dos triángulos opuestos y tres cuadriláteros. Para cada prisma existe un total de $2^3 = 8$ tetraedrizaciones posibles, de las cuales solo seis son válidas.

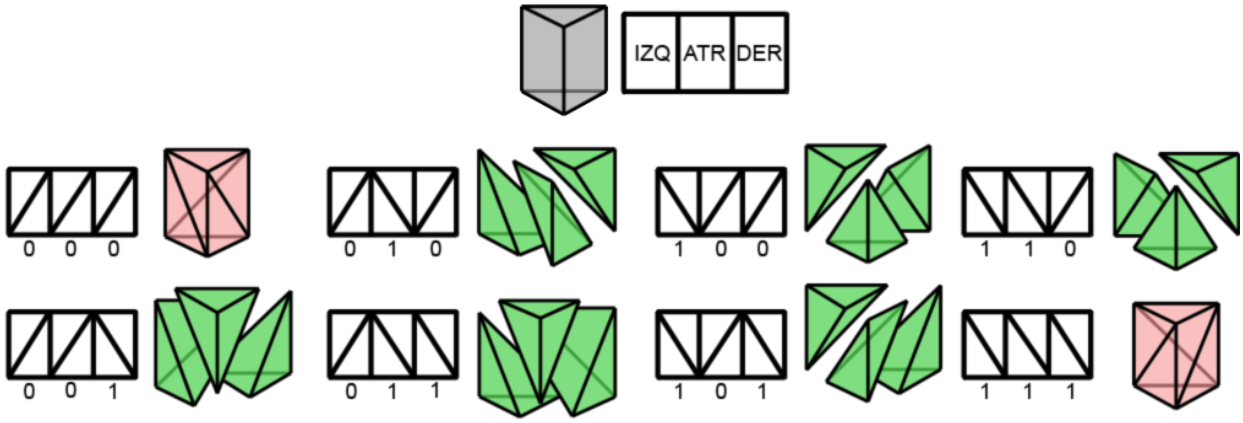

Figura 2.31: Posibles divisiones para un prisma. Las configuraciones 000 y 111 no son tetraedrizables. IZQ, ATR y DER indican la dirección de la tetraedrización a utilizar para la cara izquierda, trasera y derecha, respectivamente.

Como se puede observar en la Figura 2.31, en las seis configuraciones válidas se puede dividir el prisma en tres tetraedros de salida. Además, las dos configuraciones inválidas no ocurren al aplicar la regla de conexión por índices, debido a que esos casos no son consistentes con la propiedad transitiva del predicado de ordenamiento ρ.

C. CRISTALES

En los casos 121, 112 y 211 el poliedro-$\alpha\beta$ es un cristal, el cual consiste en un poliedro de ocho vértices con dos triángulos opuestos, dos cuadriláteros y dos pentágonos. Para este tipo de poliedros, primero se procede a descomponerlo en un tetraedro y dos pirámides de base cuadrada utilizando la regla de conexión por índices en los pentágonos, como se muestra en la Figura 2.32.

Luego, cada pirámide puede ser dividida en dos tetraedros utilizando la regla de conexión por índices en cada cuadrilátero.

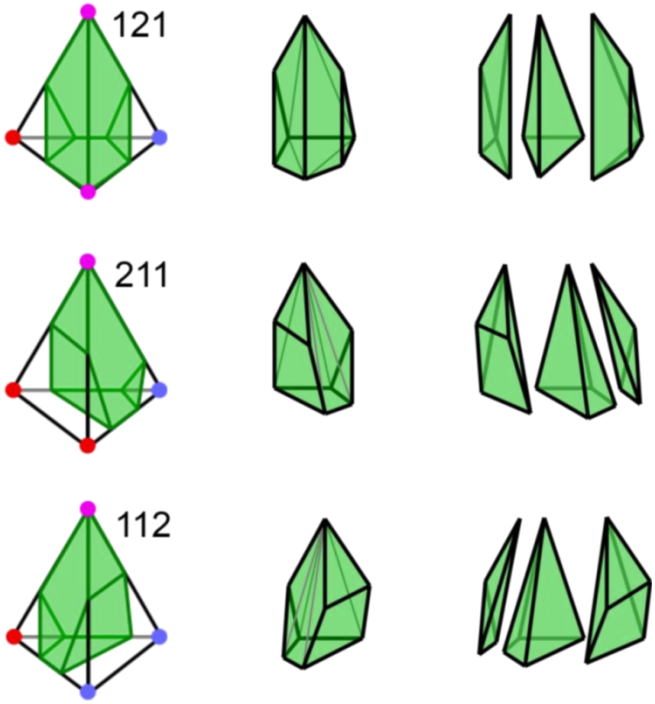

Figura 2.32: Posibles divisiones de un cristal en dos pirámides y un tetraedro.

D. CUBOS

En el caso 202 el poliedro- $\alpha\beta$ tiene forma de cubo, que consiste en ocho vértices unidos por seis cuadriláteros opuestos entre sí. Debido a que el cubo se encuentra compuesto por seis cuadriláteros, existe un total de $2^6 = 64$ triangulaciones de las caras. Para verificar que todas las triangulaciones de las caras generadas por la regla de conexión por índices son tetraedrizables, se analizará caso por caso en base a los dos menores vértices del cubo, los cuales serán identificados con **1** y **2** en las Figuras 2.33, 2.34 y 2.35.

El primer caso es cuando los vértices **1** y **2** se encuentran en esquinas opuestas del cubo. En este caso, cada una de las caras del poliedro contiene exactamente uno de los vértices mínimos, por lo que las diagonales a utilizar se encuentran definidas. En la Figura 2.33 se puede observar la división del cubo en seis tetraedros.

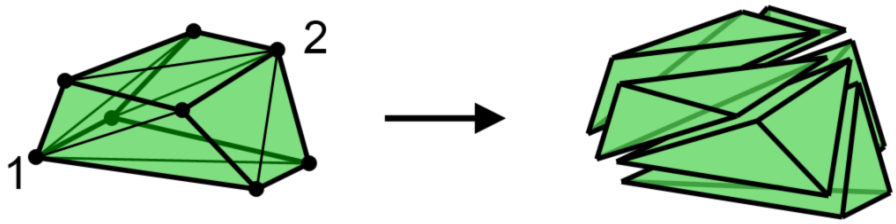

Figura 2.33: División de un cubo cuando los dos vértices se encuentran en esquinas opuestas del cubo.

El segundo caso es cuando los vértices **1** y **2** se encuentran en esquinas opuestas de la misma cara del cubo. En este caso, todas las caras tienen el menor vértice definido, excepto la opuesta a la cara con los vértices mínimos, por lo cual quedan dos casos por verificar (ver Figura 2.34). En el primer caso, se puede dividir el cubo en cinco tetraedros, y en el segundo caso, se puede dividir el cubo en seis tetraedros.

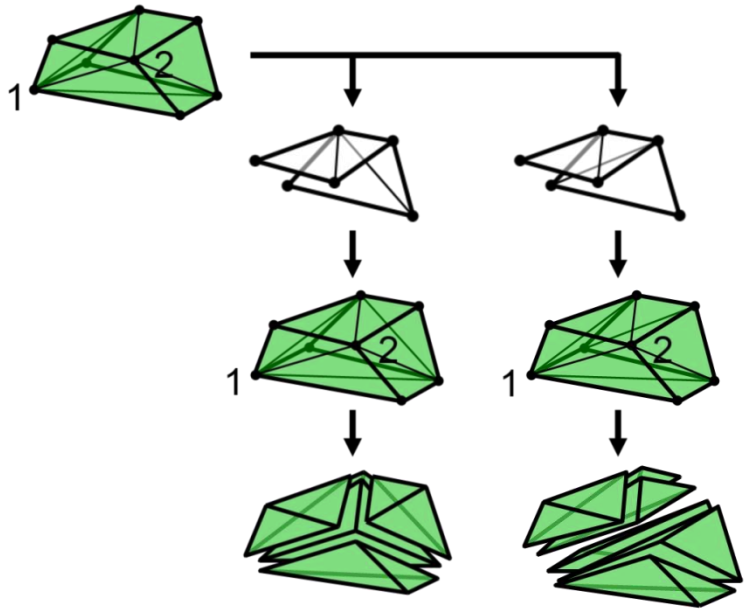

Figura 2.34: División de un cubo cuando los dos menores vértices se encuentran en una misma cara de éste.

Finalmente, el último caso es cuando los vértices **1** y **2** se encuentran en la misma arista del cubo. En este caso, quedan cuatro casos por considerar, debido a que la regla de conexión por índice permite establecer la triangulación de cuatro caras del cubo. En la Figura 2.35 se pueden observar las divisiones del cubo en seis tetraedros para los cuatro casos restantes.

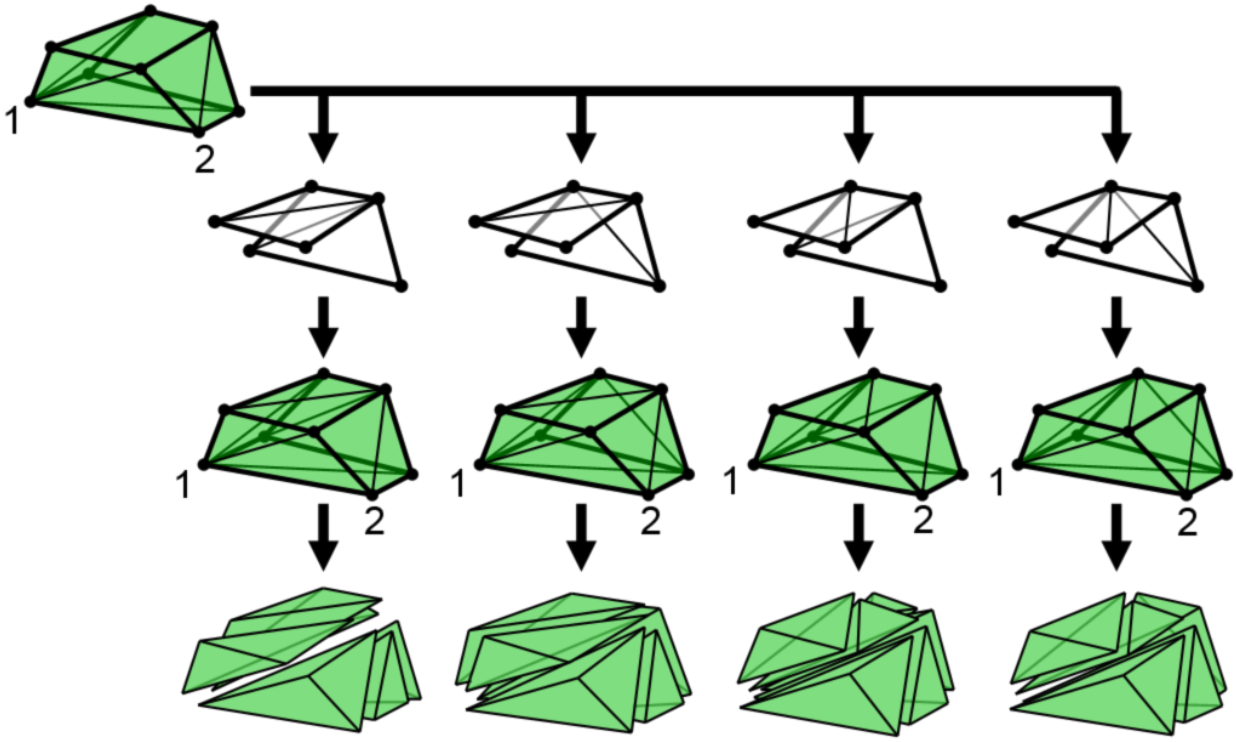

Figura 2.35: División de un cubo cuando los dos vértices mínimos están en una misma arista de éste.

2.4. MÉTRICAS DE CALIDAD

La razón principal para el uso de mallados de triángulos y tetraedros a la hora de trabajar con volúmenes de datos tridimensionales es que facilita su visualización y permite realizar simulaciones sobre partes de ellas sin la necesidad de tener que manipular todo el volumen. Muchos de los procedimientos utilizados para llevar a cabo visualizaciones o simulaciones sobre mallados triangulares implican procedimientos matemáticos que requieren que los triángulos a procesar posean formas regulares y no tengan degeneraciones. Por ejemplo, para visualizar la isosuperficie utilizando el modelo de sombreado Gouraud se realiza interpolación trilineal para calcular el color dentro del triángulo, por lo cual es preferible maximizar el mínimo ángulo interno de cada triángulo, de forma que el error generado por el modelo de iluminación sea minimizado.

Debido a esto, se han diseñado e implementado diversas métricas para evaluar la calidad de triángulos y tetraedros [STI07]. En este trabajo, sólo se tomarán en cuenta aquellas métricas que permitan la evaluación de los elementos de un mallado independientemente, sin necesidad de otro mallado de referencia.

Las métricas utilizadas para evaluar la calidad de las primitivas se pueden clasificar en dos tipos diferentes: propias y no propias. Las métricas propias consisten en un valor real sin unidad, el cual es igual a uno (1) para subregiones ideales – un triángulo equilátero con área unitaria o un tetraedro regular con volumen unitario – y tiende a infinito (∞) para elementos degenerados o de baja calidad, como por ejemplo, un triángulos con aristas cuyas longitudes difieren significativamente o un elemento con un par de vértices iguales. Las métricas no propias consisten en valores reales con unidad, como por ejemplo, longitudes, ángulos, áreas o volúmenes, cuyo valor ideal depende de la métrica.

A continuación, se describirán las métricas más importantes a la hora de evaluar la calidad de triángulos y tetraedros. La descripción de cada métrica incluye la explicación del factor que evalúa, la forma de calcularla, posibles casos de borde, el rango donde se considera que el elemento es de buena calidad, así como el rango donde se considera que el elemento es degenerado o de baja calidad, los cuales fueron extraídos de [STI07]. Luego, se describe cómo evaluar la calidad de un conjunto de primitivas en base a la calidad de cada una de las primitivas.

2.4.1. MÉTRICAS DE CALIDAD PARA TRIÁNGULOS

Las métricas descritas en esta sección están definidas sobre un elemento triangular como el que se muestra a continuación en la Figura 2.36.

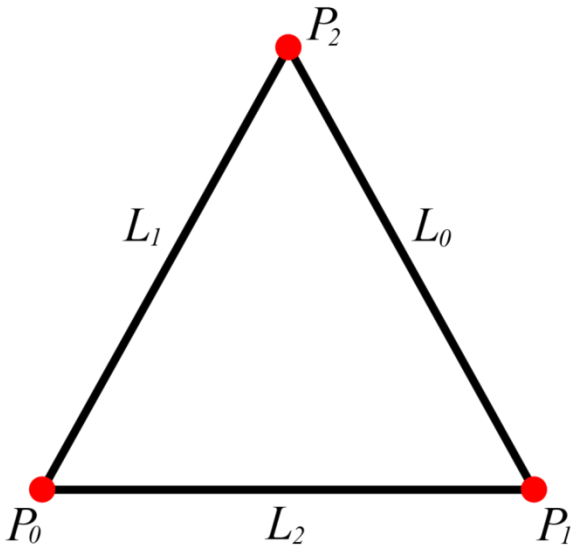

Figura 2.36: Elemento triangular utilizado para describir las métricas de calidad.

A continuación se definirán varios elementos del triángulo, lo cual facilitará la descripción de las métricas a evaluar. En todas las definiciones y métricas descritas a continuación, si el índice de un punto o arista del triángulo se encuentra fuera de $\{0, 1, 2\}$, se tomará el resto de dividir éste entre tres, lo cual facilita la expresión de algunas ecuaciones en las métricas.

Las aristas del triángulo se encuentran definidas de la siguiente manera:

$$\overrightarrow{L_0} = P_2 - P_1, \quad \overrightarrow{L_1} = P_0 - P_2, \quad \overrightarrow{L_2} = P_1 - P_0$$

Las longitudes de las aristas del triángulo se definen de la siguiente manera:

$$L_0 = \left\|\overrightarrow{L_0}\right\|, \quad L_1 = \left\|\overrightarrow{L_1}\right\|, \quad L_2 = \left\|\overrightarrow{L_2}\right\|$$

y las longitudes de la arista más pequeña y más grande son, respectivamente:

$$L_{min} = \min(L_0, L_1, L_2), \quad L_{max} = \max(L_0, L_1, L_2)$$

El área de un triángulo es la mitad de la magnitud del producto cruz de cualquier par de aristas adyacentes:

$$A = \frac{1}{2}\|L_0 \times L_1\| = \frac{1}{2}\|L_1 \times L_2\| = \frac{1}{2}\|L_2 \times L_0\|$$

Además, se define r como el radio del círculo inscrito y R como el radio del círculo circunscrito del triángulo, los cuales también son llamados radio interno y radio circunscrito:

$$r = \frac{2A}{L_0 + L_1 + L_2}, \quad R = \frac{L_0 L_1 L_2}{2r(L_0 + L_1 + L_2)} = \frac{L_0 L_1 L_2}{4A}$$

2.4.1.1. RELACIÓN DE ASPECTO

La relación de aspecto de un triángulo representa la proporcionalidad que tiene respecto a su ancho y alto. Para calcular la relación de aspecto de un triángulo se divide la longitud de la arista más larga entre el radio del círculo inscrito. Sin embargo, el valor se normaliza de forma tal que un triángulo equilátero de área unitaria tenga $q = 1$.

$$q = \frac{L_{max}}{2\sqrt{3}r} = \frac{L_{max}\,(L_0 + L_1 + L_2)}{4\sqrt{3}A}$$

La relación de aspecto del triángulo q se encuentra dentro del rango $[1, \infty)$, y se considera que un triángulo es de buena calidad si ésta se encuentra dentro del rango $[1, 1.3]$.

2.4.1.2. PROPORCIÓN DE ARISTAS

La proporción de las aristas se calcula como:

$$q = \frac{L_{max}}{L_{min}}$$

La proporción de las aristas q se encuentra en el rango $[1, \infty)$. Un triángulo se considera de buena calidad si ésta se encuentra en el rango $[1, 1.3]$.

2.4.1.3. PROPORCIÓN DE RADIOS

La proporción de los radios de un triángulo se calcula como:

$$q = \frac{R}{2r}$$

donde r y R son los radios inscritos y circunscritos del triángulo a evaluar. La proporción de los radios de un triángulo q se encuentra en el rango $[1, \infty)$. Sin embargo, un triángulo sólo se considerará de buena calidad si su proporción de radios se encuentra en el rango $[1, 1.3]$.

2.4.2. MÉTRICAS DE CALIDAD PARA TETRAEDROS

Las métricas descritas en esta sección están definidas sobre un elemento tetraédrico como el que se muestra a continuación en la Figura 2.37.

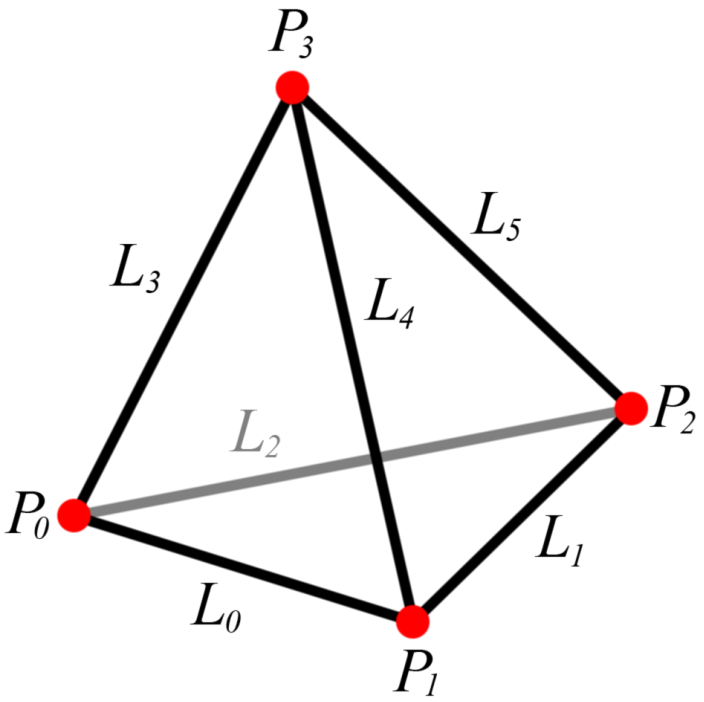

Figura 2.37: Elemento tetraédrico utilizado para describir las métricas de calidad.

A continuación se definirán varios elementos del tetraedro, lo cual facilitará la descripción de las métricas a evaluar.

Las aristas del tetraedro se encuentran definidas de la siguiente manera:

$$\overrightarrow{L_0} = \overrightarrow{L_{01}} = P_1 - P_0, \quad \overrightarrow{L_3} = \overrightarrow{L_{03}} = P_3 - P_0$$
$$\overrightarrow{L_1} = \overrightarrow{L_{12}} = P_2 - P_1, \quad \overrightarrow{L_4} = \overrightarrow{L_{13}} = P_3 - P_1$$
$$\overrightarrow{L_2} = \overrightarrow{L_{20}} = P_0 - P_2, \quad \overrightarrow{L_5} = \overrightarrow{L_{23}} = P_3 - P_2$$

Las longitudes de las aristas del triángulo se definen de la siguiente manera:

$$L_0 = \left\|\overrightarrow{L_0}\right\|, \quad L_1 = \left\|\overrightarrow{L_1}\right\|, \quad L_2 = \left\|\overrightarrow{L_2}\right\|, \quad L_3 = \left\|\overrightarrow{L_3}\right\|, \quad L_4 = \left\|\overrightarrow{L_4}\right\|, \quad L_5 = \left\|\overrightarrow{L_5}\right\|$$

y las longitudes de la arista más pequeña y más grande son, respectivamente:

$$L_{min} = \min(L_0, L_1, L_2, L_3, L_4, L_5), \quad L_{max} = \max(L_0, L_1, L_2, L_3, L_4, L_5)$$

El área de la superficie de un tetraedro es igual a la suma del área de las cuatro caras que lo componen:

$$A = \frac{1}{2}\left(\left\|\overrightarrow{L_2} \times \overrightarrow{L_0}\right\| + \left\|\overrightarrow{L_3} \times \overrightarrow{L_0}\right\| + \left\|\overrightarrow{L_4} \times \overrightarrow{L_1}\right\| + \left\|\overrightarrow{L_3} \times \overrightarrow{L_2}\right\|\right)$$

El volumen del tetraedro se puede definir en base a las aristas como:

$$V = \frac{\left(\overrightarrow{L_2} \times \overrightarrow{L_0}\right) \cdot \overrightarrow{L_3}}{6}$$

Además, se define r como el radio de la esfera inscrita y R como el radio de la esfera circunscrita del tetraedro, los cuales también son llamados radio interno y radio circunscrito:

$$r = \frac{3V}{A}, \quad R = \frac{\left\|L_3^2 \cdot \left(\overrightarrow{L_2} \times \overrightarrow{L_0}\right) + L_2^2 \cdot \left(\overrightarrow{L_3} \times \overrightarrow{L_0}\right) + L_0^2 \cdot \left(\overrightarrow{L_3} \times \overrightarrow{L_2}\right)\right\|}{12V}$$

2.4.2.1. RELACIÓN DE ASPECTO

La relación de aspecto de un tetraedro representa la proporcionalidad que tiene respecto a su longitud, ancho, y alto. Para calcular la relación de aspecto de un tetraedro se divide la longitud de la arista más larga entre el radio de la esfera inscrita. Sin embargo, el valor se normaliza de forma tal que un tetraedro equilátero de área unitaria tenga $q = 1$.

$$q = \frac{L_{max}}{2\sqrt{6}r} = \frac{AL_{max}}{6\sqrt{6}V}$$

La relación de aspecto q del tetraedro se encuentra dentro del rango $[1, \infty)$, y se considera que un tetraedro es de buena calidad si ésta se encuentra dentro del rango $[1, 3]$.

2.4.2.2. PROPORCIÓN DE ARISTAS

La proporción de aristas de un tetraedro se define como la longitud de la arista más larga entre la longitud de la arista más corta, es decir:

$$q = \frac{L_{max}}{L_{min}}$$

La proporción de las aristas q se encuentra en el rango $[1, \infty)$, pero para que un tetraedro sea considerado de buena calidad ésta se debe encontrar en el rango $[1, 3]$.

2.4.2.3. PROPORCIONALIDAD DE RADIOS

Esta métrica se define como el radio de la esfera circunscrita entre el radio de la esfera inscrita en el tetraedro, normalizada de forma tal que un tetraedro equilátero tenga calidad uno, es decir:

$$q = \frac{R}{3r} = \frac{\left\| L_3^2 \cdot \left(\overrightarrow{L_2} \times \overrightarrow{L_0}\right) + L_2^2 \cdot \left(\overrightarrow{L_3} \times \overrightarrow{L_0}\right) + L_0^2 \cdot \left(\overrightarrow{L_3} \times \overrightarrow{L_2}\right) \right\| A}{108 V^2}$$

La proporcionalidad de los radios de un tetraedro q se encuentra en el rango $[1, \infty)$, y se considera que un tetraedro es de buena calidad si dicha proporcionalidad se encuentra en el rango $[1, 3]$.

2.4.3. MÉTRICAS PARA CONJUNTOS DE PRIMITIVAS

La salida generada por los algoritmos para la extracción de isosuperficies e intervalos de volumen no es una sola primitiva, sino un conjunto ellas, por lo cual se deben diseñar métodos para el análisis de conjuntos de primitivas en base a la calidad de cada uno de los elementos que componen dicho conjunto.

El primer método [STI07] para evaluar la calidad del conjunto consiste en clasificar cada primitiva de éste en base a los rangos y determinar si es buena, normal o degenerada en base a alguna métrica y contar cuántas existen de cada tipo. Luego, la calidad del modelo será directamente proporcional a la cantidad de primitivas buenas e inversamente proporcional a la cantidad de primitivas degeneradas.

El segundo método [STI07] consiste en calcular la calidad del conjunto de primitivas como la media de la calidad de cada uno de los elementos de éste, y tomar la desviación estándar como el grado de esparcimiento de la calidad de los elementos. Para las métricas no propias se recomienda utilizar la media y desviación estándar aritmética, las cuales se definen como:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} q_i \,, \quad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (q_i - \mu)^2}$$

donde q_i es la calidad del *i*-ésimo elemento del conjunto. Para las métricas propias se recomienda utilizar la media y desviación estándar geométrica, las cuales se definen como:

$$\mu_g = \left(\prod_{i=1}^{n} q_i\right)^{1/n}, \quad \sigma_g = \exp\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\ln q_i - \ln \mu_g\right)^2}$$

CAPÍTULO 3. EXTRACCIÓN DE INTERVALOS DE VOLUMEN MEDIANTE MODIFICACIÓN DE CUBOS MARCHANTES

A continuación se explicará el algoritmo desarrollado para la extracción de intervalos de volumen $[\alpha, \beta]$ mediante el uso de una tabla de casos similar a la utilizada por cubos marchantes para la extracción de isosuperficies (ver Sección 2.2.1). Luego, se explicarán las ventajas y desventajas de este algoritmo, así como el procedimiento utilizado para generar la tabla de conectividad y el post-procesamiento a realizar sobre el mallado resultante. Finalmente, se describirán ciertas consideraciones que se deben tener en cuenta para el manejo de los casos ambiguos en aquellos algoritmos basados en este procedimiento.

3.1. ALGORITMO PARA LA EXTRACCIÓN DEL INTERVALO DE VOLUMEN

El algoritmo desarrollado se encuentra basado en Cubos Marchantes [LOR87] y, al igual que éste, reconstruye cada celda de la malla independientemente mediante el uso de una tabla de conectividad. Cada celda se encuentra delimitada por ocho vóxeles y doce aristas, y puede aportar hasta treinta y dos (32) vértices diferentes al mallado final, debido a que cada vóxel neutro aporta un vértice y a que cada arista aporta dos vértices diferentes cuando un vóxel que la delimita es positivo y el otro es negativo. En la Figura 3.1 se puede observar la enumeración de la celda utilizada por el algoritmo propuesto.

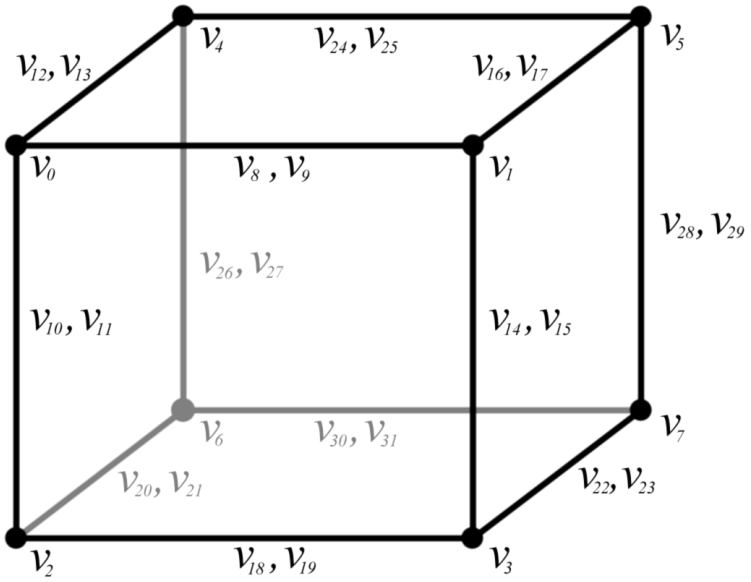

Figura 3.1: Enumeración de la celda utilizada por el algoritmo propuesto.

Es importante observar que en la enumeración utilizada los puntos de intersección se encuentran ordenados en base a los siguientes criterios:

- Los vértices provenientes de vóxeles neutros preceden aquellos provenientes de intersecciones en las aristas de la celda.

- Los vértices provenientes de vóxeles se encuentran ordenados por la coordenada del vóxel dentro de la celda, que equivale a ordenarlos en base la posición del vóxel dentro del volumen.

- Los vértices provenientes de aristas diferentes se encuentran ordenados en base a los vóxeles que las delimitan, es decir, si el menor vértice de ambas aristas es diferente estas se ordenan en base a éste, si no se ordenan en base al otro extremo.

- Los vértices provenientes de una misma arista se encuentran ordenadas por el isovalor en el punto de intersección, es decir, aquellos vértices con isovalor α preceden aquellos con isovalor β.

Este ordenamiento garantiza el ordenamiento de los vértices a nivel global, es decir, si un vértice v_1 precede a otro vértice v_2 dentro de una celda, entonces v_1 precede a v_2 en el mallado completo, lo cual es importante debido a que este predicado de ordenamiento es necesario para prevenir el problema expuesto en la Figura 2.29 en la Sección 2.3.3.2, durante la generación de la tabla de casos.

El primer paso para extraer el intervalo de volumen $[\alpha, \beta]$ de una celda consiste en clasificar cada uno de los vóxeles que la delimita en tres grupos, positivos, negativos y neutros, en base a la siguiente ecuación:

$$\text{clase}(\delta_i, \alpha, \beta) = \begin{cases} -1, & si\ \delta_i < \alpha \\ +1, & si\ \delta_i > \beta \\ 0, & si\ \alpha \leq \delta_i \leq \beta \end{cases}$$

donde δ_i es el isovalor del vóxel a clasificar. Como se puede observar, un vóxel será neutro si su isovalor se encuentra dentro del intervalo extraer, o será positivo o negativo en base al signo de $\delta_i - \alpha$ si su isovalor no se encuentra dentro del intervalo. Luego, con la clasificación de los

vóxeles se construye un índice de clasificación interpretando la clase de cada vóxel como un dígito en base-3 (ver Figura 3.2).

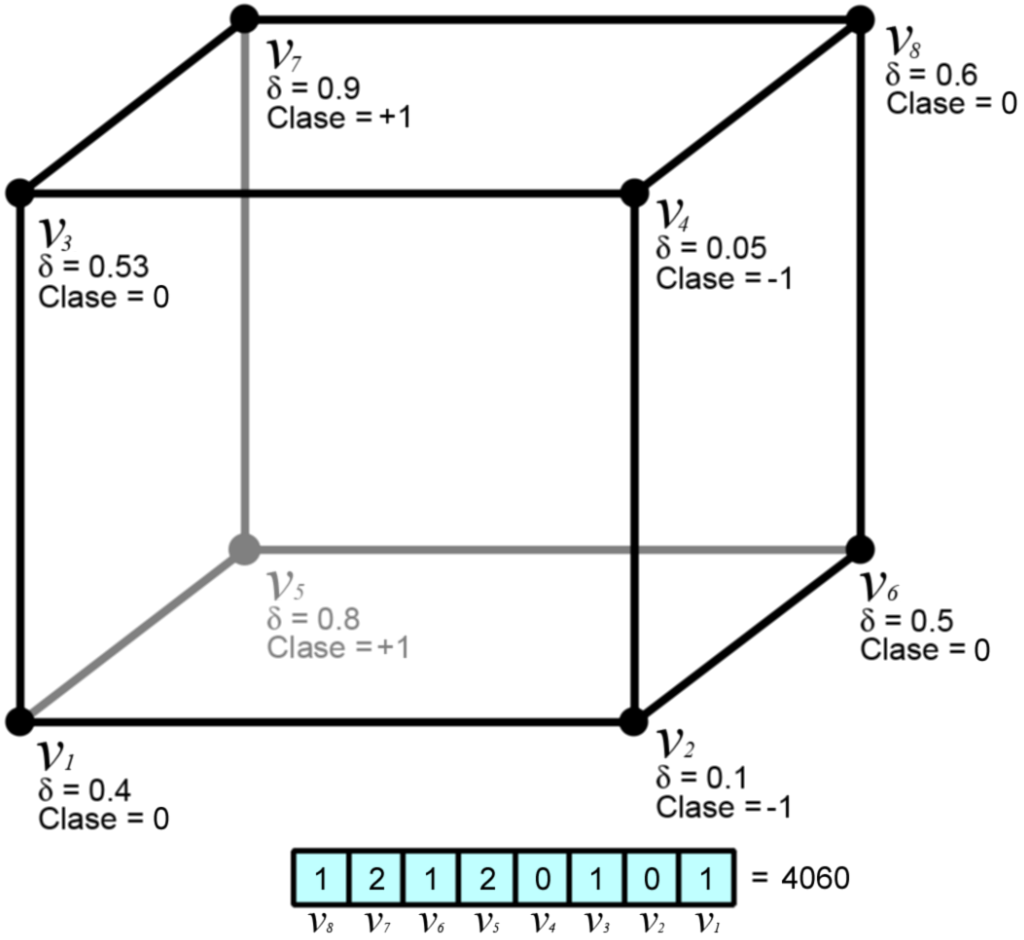

Figura 3.2: Clasificación de una celda en base a los isovalores de los vóxeles que la delimitan, con $\alpha = 0.3$ y $\beta = 0.6$, donde δ_i representa el isovalor de cada vóxel.

Debido a que cada uno de los vóxeles tiene tres estados posibles, existe un total de $3^8 = 6561$ formas diferentes en que el intervalo de volumen a extraer intersecte la celda. Sin embargo, a diferencia del algoritmo para la extracción de isosuperficies, no se puede reducir la cantidad de casos por simetría, reflexión o rotación, debido a que esto interfiere con el ordenamiento de los puntos de intersección. En la Figura 3.3 se pueden observar algunas posibles intersecciones entre la celda y el intervalo de volumen a extraer.

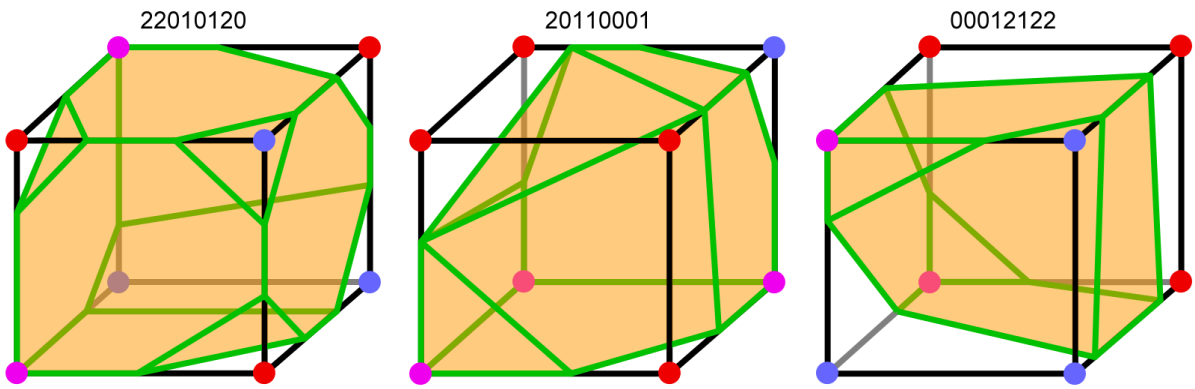

Figura 3.3: Algunas posibles intersecciones entre la celda y el intervalo de volumen a extraer. Los puntos rojos representan los vóxeles negativos, los puntos azules representan los vóxeles positivos y los puntos violetas representan los vóxeles neutros.

El siguiente paso es determinar cuáles vóxeles producen vértices y cuáles aristas de la celda son intersectadas por el volumen a extraer, así como la posición de estos puntos de intersección. Un vóxel produce un vértice si este es neutro, cuya posición P_x equivale a la posición del vóxel, es decir:

$$P_x = P, \quad x \in \{0,1,\cdots,6,7\}$$

donde P es la posición del vóxel neutro. Una arista produce un vértice con isovalor α si un extremo es negativo y el otro no es negativo, donde la posición P_x de esta intersección se calcula mediante el uso de interpolación lineal sobre la arista de la siguiente manera:

$$P_x = P_0 + (P_1 - P_0)\left(\frac{\alpha - \delta_0}{\delta_1 - \delta_0}\right), \quad x \in \{8,10,12,\cdots,26,28,30\}$$

donde, δ_0 y δ_1 son los isovalores de los vóxeles que delimitan la arista y P_0 y P_1 son las posiciones de los extremos de la arista. De forma similar, una arista produce un vértice con isovalor β si un extremo es positivo y el otro no es positivo, donde la posición P_x de esta intersección se calcula mediante el uso de interpolación lineal sobre la arista de la siguiente manera:

$$P_x = P_0 + (P_1 - P_0)\left(\frac{\beta - \delta_0}{\delta_1 - \delta_0}\right), \quad x \in \{9,11,13,\cdots,27,29,31\}$$

Después de calcular todos los puntos de intersección del intervalo de volumen con la celda se generan los tetraedros que lo componen utilizando una tabla de conectividad de $3^8 = 6561$ casos, la cual se indexa utilizando el índice obtenido de la clasificación de los vóxeles.

Mediante este algoritmo también se pueden calcular otros atributos de los vértices del mallado final, extrayendo el atributo del vóxel para aquellos vértices provenientes de vóxeles neutros, y mediante interpolación lineal de los atributos en los extremos de cada arista para aquellos vértices generados por intersecciones en las aristas. Por ejemplo, se puede calcular el gradiente G_x para vértice P_x de la siguiente manera:

$$
\begin{aligned}
&G_x = G, && x \in \{0,1,\cdots,6,7\} \\
&G_x = G_0 + (G_1 - G_0)\left(\frac{\alpha - v_0}{v_1 - v_0}\right), && x \in \{8,10,12,\cdots,26,28,30\} \\
&G_x = G_0 + (G_1 - G_0)\left(\frac{\beta - v_0}{v_1 - v_0}\right), && x \in \{9,11,13,\cdots,27,29,31\}
\end{aligned}
$$

donde G es el gradiente el vóxel neutro y G_0 y G_1 son los gradientes en los extremos de la arista intersectada.

3.2. VENTAJAS Y DESVENTAJAS DEL ALGORITMO PROPUESTO

Como se puede observar en la Sección 3.1, el algoritmo propuesto es muy similar al algoritmo propuesto por Lorensen [LOR87] para la extracción de isosuperficies, por lo cual éste posee ventajas similares a aquellas del algoritmo original (ver Sección 2.2.1.1), entre las cuales se encuentran:

- **Sencillez:** El algoritmo es sencillo de implementar, ya que sólo se necesita una tabla de 3^8 casos para procesar cada celda en base a la clasificación de sus vóxeles, y no se requiere ningún algoritmo complejo para el procesamiento de la celda, como triangulaciones de Delaunay [GUO95] o intersección de poliedros [FUJ95].

- **Eficiencia:** El algoritmo sólo calcula las intersecciones en aquellas aristas donde es necesario, reutilizando las intersecciones de las celdas anteriores de ser posible, igual que

el algoritmo para extraer isosuperficies. Además, el algoritmo sólo necesita la construcción de un índice y búsquedas de orden constante en tablas precalculadas.

- **Paralelizable:** El algoritmo es fácil de paralelizar, debido a que el volumen extraído de una celda es independiente del extraído de otras celdas. Esto brinda una ventaja sobre los algoritmos basados en tetraedrizaciones de Delaunay [GUO95], ya que la paralelización de estos es mucho más complicada.

Sin embargo, el algoritmo posee desventajas similares a aquellas del algoritmo original, entre las cuales se encuentran:

- **Manejo de casos ambiguos:** El algoritmo no realiza ningún procesamiento adicional para determinar casos ambiguos, ya que éste asume que el intervalo de volumen a extraer sólo puede intersectar las celdas de una sola forma por cada índice de clasificación. Sin embargo, el algoritmo maneja las celdas de forma consistente, de forma que no existan huecos en el volumen similares a aquellos generados en el algoritmo original propuesto por Lorensen [LOR87]. En la Sección 3.5 se describen las consideraciones que se deben tener en cuenta para manejar casos ambiguos en el algoritmo.

- **Tetraedros degenerados:** En ciertos casos, el algoritmo genera tetraedros degenerados o con volumen negativo, lo cual se puede corregir durante el post-procesamiento (ver Sección 3.4).

- **Complejidad de la salida:** El algoritmo genera una cantidad excesiva de tetraedros cuando la resolución de los modelos de entrada es muy alta, lo que ocasiona que el volumen final sea más difícil de procesar. Sin embargo, el mallado generado está compuesto por una cantidad de tetraedros mucho menor en comparación a los mallados generados por el algoritmo propuesto por Nielson y Sung [NIE97b] (ver Sección 4.2).

3.3. GENERACIÓN DE LA TABLA DE CONECTIVIDAD

Para extraer tetraedros de cada celda el algoritmo necesita una tabla de conectividad que le indique los vértices del mallado a unir en cada celda para obtener los tetraedros que componen el mallado resultante. Esta lista consiste en un conjunto de 4-tuplas de la forma (a, b, c, d), donde

cada tupla indica que se deben unir los vértices a, b, c y d para generar un tetraedro de salida. La 4-tupla será válida si y sólo si los cuatro elementos de ésta son diferentes y pertenecen a los vértices intersectados para el caso que está siendo procesado.

Con el objetivo de preservar la eficiencia del algoritmo final, se deben precalcular los tetraedros de salida para cada caso de intersección posible, lo cual se hace mediante la creación de celdas sintéticas y la extracción de tetraedros de éstas. El procedimiento utilizado para la generación de la tabla de conectividad consiste en obtener el poliedro-$\alpha\beta$ que representa el intervalo $[\alpha, \beta]$ a extraer, para luego tetraedrizar éste mediante el algoritmo propuesto por Max [MAX01] para la tetraedrización de poliedros convexos de forma coherente.

A continuación se explicarán las consideraciones tomadas en cuenta para crear las celdas sintéticas sobre las cuales se extraen los tetraedros de salida, así como el procedimiento utilizado para la extracción del poliedro-$\alpha\beta$ y su tetraedrización.

3.3.1. CREACIÓN DE LA CELDA SINTÉTICA

El procedimiento para la creación de una celda sintética recibe como entrada el índice de clasificación M de la celda que se desea construir, y produce como salida una celda C y un intervalo $[\alpha, \beta]$ de forma tal que al clasificar los vóxeles de la celda C en base al intervalo $[\alpha, \beta]$ utilizando la fórmula descrita en la Sección 3.1 se obtenga el índice M, es decir:

$$\delta_i \in \begin{cases} (-\infty, \alpha), & si\ M_i = 0 \\ [\alpha, \beta], & si\ M_i = 1 \\ (\beta, \infty), & si\ M_i = 2 \end{cases}$$

donde δ_i es el isovalor del i-ésimo vóxel de C y M_i es el i-ésimo dígito de M si éste se interpreta en base-3.

Para generar cada uno de los isovalores de los vóxeles que delimitan la celda, se fija el intervalo $[\alpha, \beta]$, y se calculan los valores de desplazamiento k y tolerancia Δ de la siguiente manera:

$$k = \frac{\beta - \alpha}{2}, \quad \Delta = \frac{k}{2}$$

donde k se utiliza para calcular el punto base de los isovalores y Δ se utiliza para añadir una variación aleatoria sobre los isovalores, con el propósito de que los puntos de intersección se encuentren en posición general [EDE90].

Finalmente, cada isovalor C_i se calcula en base al valor en la posición correspondiente en el índice de la siguiente manera:

$$\delta_i = \begin{cases} \alpha - k + \text{rand}(-\Delta, \Delta), & si\ M_{\text{i}} = 0 \\ \alpha + k + \text{rand}(-\Delta, \Delta), & si\ M_i = 1 \\ \beta + k + \text{rand}(-\Delta, \Delta), & si\ M_i = 2 \end{cases}$$

donde $\text{rand}(A, B)$ representa una función que retorna un valor aleatorio uniformemente distribuido en el intervalo $[A, B]$.

3.3.2. EXTRACCIÓN DEL POLIEDRO ALFA-BETA DE LA CELDA SINTÉTICA

El primer paso para la extracción del poliedro-$\alpha\beta$ consiste en calcular los puntos donde el intervalo de volumen $[\alpha, \beta]$ a extraer intersecta la celda sintética, en base a los isovalores generados en el paso anterior. El procedimiento para calcular estos puntos es el mismo utilizado en el algoritmo final, el cual se puede observar en la Sección 3.1.

Luego, se calcula la cápsula convexa de los puntos de intersección mediante el uso de un algoritmo de fuerza bruta con tiempo de ejecución $O(n^4)$, donde n es la cantidad de puntos de intersección entre la celda y el intervalo de volumen a extraer. El uso de fuerza bruta no afecta el desempeño del algoritmo final debido a que este procedimiento sólo es realizado durante el preprocesamiento de la tabla de conectividad. La cápsula convexa obtenida será representada como un conjunto de caras que delimitan el poliedro-$\alpha\beta$, donde cada cara consiste en una lista ordenada de vértices que la delimitan. Este procedimiento consiste en los siguientes pasos, los cuales se pueden observar en la Figura 3.4:

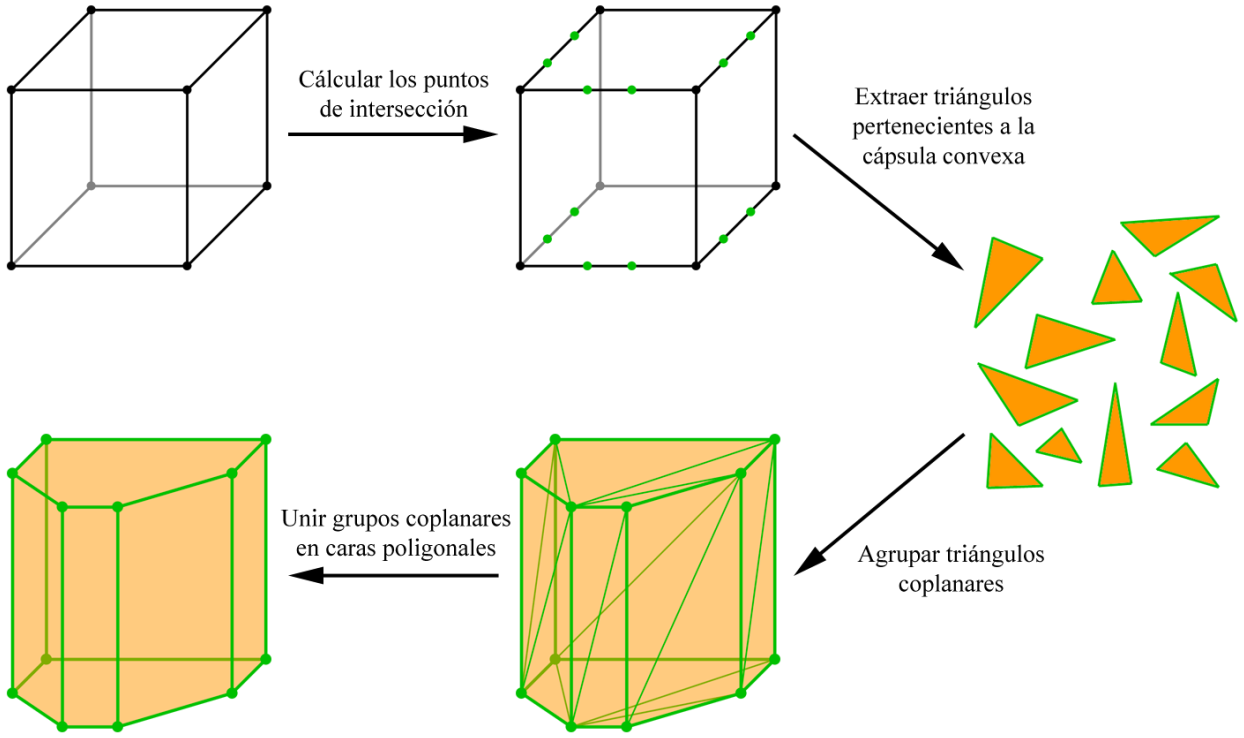

Figura 3.4: Pasos necesarios para el cálculo del poliedro-$\alpha\beta$ a partir de la celda sintética.

- Calcular los triángulos que pertenecen a la cápsula convexa, los cuales serán todos aquellos delimitados por tres puntos de intersección P_a, P_b y P_c, tal que no tengan ningún punto de intersección del lado positivo del plano definido por éstos. La ecuación del plano T al que pertenece el triángulo que está siendo considerado está definida por:

$$T = (N_x, N_y, N_z, D)$$
$$N = (N_x, N_y, N_z) = (P_b - P_a) \times (P_c - P_a)$$
$$D = -N \cdot P_a$$

  donde N es la normal del plano y D es la distancia de éste al origen. Luego, el triángulo pertenecerá a la cápsula convexa si y sólo si no existe un punto de intersección P_i tal que:

$$N \cdot P_i + D > 0$$

- Clasificar los triángulos en distintas clases de equivalencia τ_k, donde dos triángulos pertenecerán a la misma clase de equivalencia si y sólo si los planos T_1 y T_2 a los que pertenecen son equivalentes, es decir:

$$T_1 = \left(N_{x,1}, N_{y,1}, N_{z,1}, D_1\right)$$
$$T_2 = \left(N_{x,2}, N_{y,2}, N_{z,2}, D_2\right)$$
$$T_1 \equiv T_2 \Leftrightarrow \exists k \in \mathbb{R}^+ [T_1 = kT_2]$$

- Unir los triángulos de cada clase de equivalencia τ_k en una cara poligonal del poliedro-$\alpha\beta$, mediante la aplicación de un predicado de ordenamiento sobre los puntos en τ_k. Para realizar esto, se extrae el conjunto A con los puntos de intersección pertenecientes al grupo que está siendo procesado, donde A se define como:

$$A = \{a_1, a_2, \cdots, a_n\}$$
$$a_i < a_{i+1}, \quad \forall i \in \{1,2, \cdots, n-1\}$$

  donde a_i es el i-ésimo vértice perteneciente a la cara. Luego, se elige un vértice pivote a_p, el cual será el vértice inicial a partir del cual se reconstruirá la cara del poliedro-$\alpha\beta$. Una vez que se ha seleccionado el vértice pivote, se procede a ordenar el resto los puntos a partir de éste, mediante el uso de un predicado de ordenamiento $\rho_F(a_i, a_j)$ que establece que a_i precede a a_j en el grupo τ_k si y sólo sí el triángulo (a_p, a_i, a_j) existe en τ_k, es decir:

$$\rho_F(a_i, a_j) \Leftrightarrow (a_p, a_i, a_j) \in \tau_k$$

  Por ejemplo, en la Figura 3.5 se puede observar una cara hexagonal del poliedro-$\alpha\beta$, donde las flechas rojas indican la precedencia entre los vértices distintos al pivote, y los triángulos pertenecientes a τ_k son aquellos que utilizan dos aristas grises y una roja.

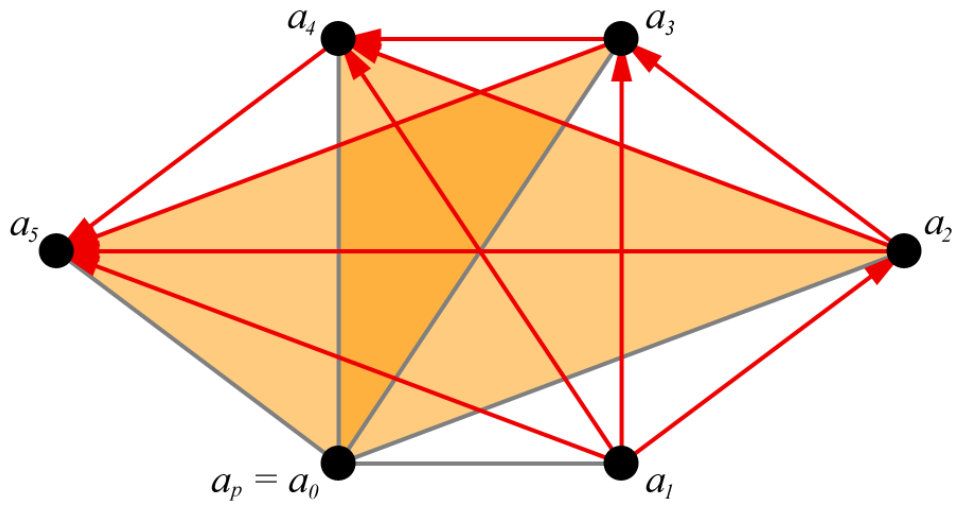

Figura 3.5: Ordenamiento de los puntos en una cara del poliedro-$\alpha\beta$.

Una vez que se obtienen todas las caras del poliedro-$\alpha\beta$, este es tetraedrizado utilizando el procedimiento explicado en la Sección 3.3.3. Para que este procedimiento se realice de forma satisfactoria, es importante verificar que el poliedro-$\alpha\beta$ es convexo e incluye todos los puntos de intersección calculados.

Para verificar que el poliedro-$\alpha\beta$ es convexo, se puede observar que todas las caras generadas durante la construcción tienen a todos los demás vértices del mallado de un sólo lado, lo cual es condición suficiente para garantizar que los poliedros generados son convexos, ya que de ser cóncavos existiría una cara con vértices del mallado a ambos lados [GRA72].

Para verificar que el poliedro-$\alpha\beta$ contiene todos los puntos de intersección en la celda, es suficiente verificar que no existe un punto de intersección tal que no pertenezca a algún triángulo perteneciente a la cápsula convexa. En la Figura 3.6 se muestran los polígonos extraídos de las caras de las celdas para los $3^4 = 81$ casos posibles. Como se puede observar, todos los polígonos no vacíos poseen al menos tres vértices, por lo cual cada vértice del mallado final va a pertenecer al menos a una cara externa de la celda, ya que no existen vértices estrictamente dentro de la celda y cada vértice posee al menos dos vecinos en una misma cara, con los cuales forma una cara externa perteneciente a la cápsula convexa.

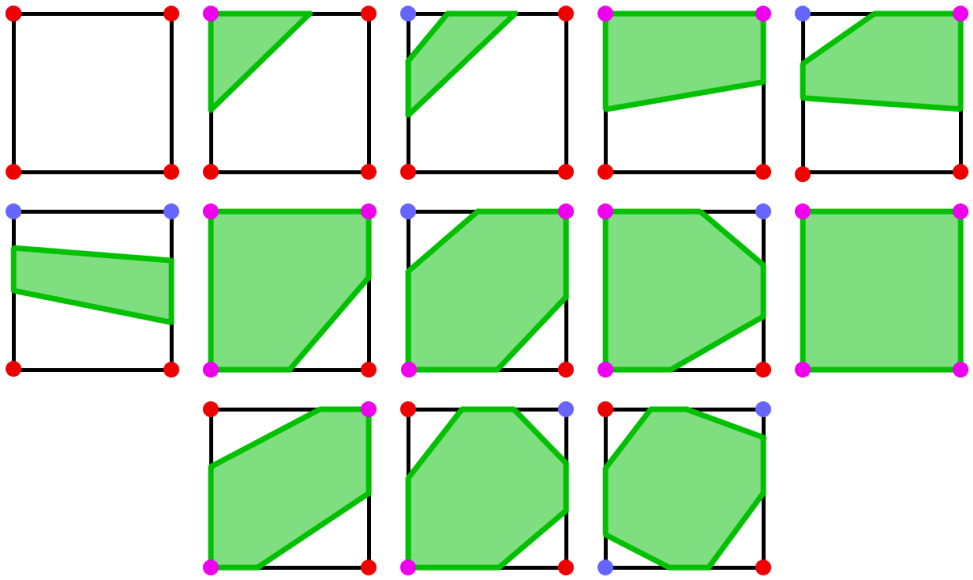

Figura 3.6: Posibles intersecciones entre el intervalo $[\alpha, \beta]$ a extraer y cada cara de la celda.

3.3.3. TETRAEDRIZACIÓN DEL POLIEDRO ALFA-BETA

El paso final para la extracción del intervalo de volumen consiste en la tetraedrización del poliedro-$\alpha\beta$ de la celda sintética, mediante el uso del algoritmo propuesto por Max [MAX01], el cual permite la tetraedrización de poliedros convexos de forma consistente en todas las caras. Esto evita el problema descrito en la Sección 2.3.3.2 para el algoritmo basado en tetracubos marchantes, debido a que este algoritmo garantiza que la triangulación utilizada para cada cara del poliedro-$\alpha\beta$ será la misma para ambos poliedros que la comparten.

Este algoritmo consta de dos pasos principales: en la primera fase éste divide cada cara del poliedro en triángulos, utilizando las diagonales que parten desde el menor punto de la cara hacia los demás puntos de ésta; en la segunda fase el algoritmo tetraedriza el poliedro trazando diagonales desde el menor punto del poliedro hacia todos los demás puntos del poliedro.

Como se puede observar, para la ejecución de este algoritmo se necesita un predicado de orden total $\rho(a_i, a_j)$ para determinar si un punto de intersección a_i precede a otro punto de intersección a_j. Para este algoritmo se utiliza el ordenamiento descrito en la Sección 3.1, el cual cumple con todas las propiedades descritas en la Sección 2.3.3.2, las cuales son necesarias y suficientes para establecer un predicado de orden total.

Para implementar este algoritmo, se utilizó un grafo de conectividad $G = (A, E)$, donde A es el conjunto con los puntos de intersección de la celda y E es el conjunto con las aristas (a_i, a_j) que unen los vértices dentro del modelo. Las aristas existentes pertenecientes a E son las siguientes:

- Todas aquellas aristas (a_i, a_j) tal que $a_i \neq a_j$ y a_i sea adyacente a a_j en alguna cara del poliedro a tetraedrizar. Estas aristas corresponden a las aristas originales del poliedro-$\alpha\beta$.

- Todas aquellas aristas (a_i, a_j) tal que $a_i \neq a_j$ y a_i comparta alguna cara con a_j donde a_i sea el menor vértice de dicha cara. Estas aristas corresponden a aquellas generadas por la primera fase del algoritmo de tetraedrización propuesto por Max [MAX01].

- Todas aquellas aristas (a_i, a_j), tal que $a_i \neq a_j$ y a_i sea el menor vértice del poliedro a tetraedrizar. Estas aristas corresponden a aquellas generadas por la segunda fase del algoritmo de tetraedrización propuesto por Max [MAX01].

En la Figura 3.7 se pueden observar las distintas aristas generadas por el algoritmo de tetraedrización, donde las aristas negras representan las aristas originales del poliedro, las aristas azules representas las aristas generadas durante la primera fase y las aristas rojas representan las aristas internas generadas durante la segunda fase.

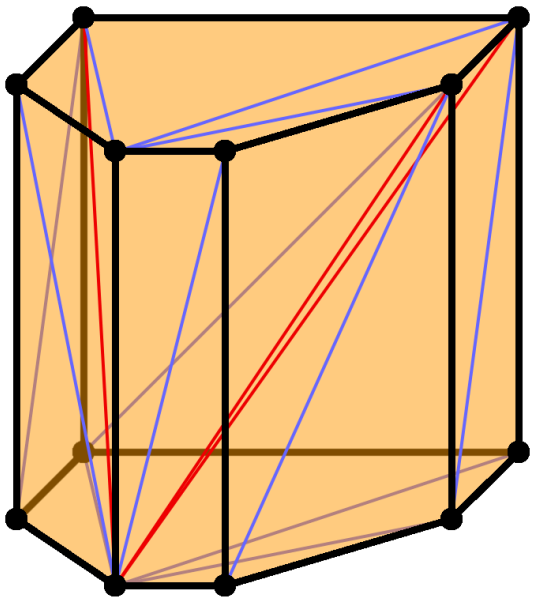

Figura 3.7: Aristas generadas durante la tetraedrización del poliedro-$\alpha\beta$.

Luego, una vez construido el grafo $G = (A, E)$ con las aristas del poliedro, se procede a la extracción de los tetraedros generados, mediante la búsqueda de subgrafos isomorfos a G_4, donde G_4 representa un grafo completo[12] con cuatro vértices, ya que un tetraedro es isomorfo a G_4. Este procedimiento se lleva a cabo probando cada cuarteto de vértices (a, b, c, d) posible, añadiéndolo a la salida si y sólo sí las seis aristas (a, b), (a, c), (a, d), (b, c), (b, d) y (c, d) pertenecen a E. Este algoritmo de fuerza bruta tiene complejidad en tiempo igual a $O(n^4)$; sin embargo, esto no afecta la eficiencia del algoritmo final, ya que este procedimiento sólo se lleva a cabo durante la generación de la tabla de conectividad.

3.4. FASE DE POST-PROCESAMIENTO

Como se mencionó en la Sección 3.2, el algoritmo tiene la desventaja de generar tetraedros degenerados o con volumen negativo, lo cual se debe a que en ciertos casos de intersección entre el intervalo de volumen a extraer y las celdas se generan tetraedros cuya orientación no depende únicamente de los índices de los extremos dentro de la celda, si no de la posición final de estos en el espacio.

Por ejemplo, para el caso $(10000011)_3 = 2191$, el tetraedro $(0,10,16,22)$ es necesario para reconstruir el intervalo de volumen dentro de la celda. Sin embargo, no se puede determinar la orientación del tetraedro sin calcular las posiciones donde las aristas son intersectadas, ya que la orientación de éste depende de la posición de las intersecciones $P_{16,\alpha}$ y $P_{22,\alpha}$, como se puede observar en la Figura 3.8.

[12] Un grafo completo de n vértices es aquél que contiene aristas entre todo par de vértices.

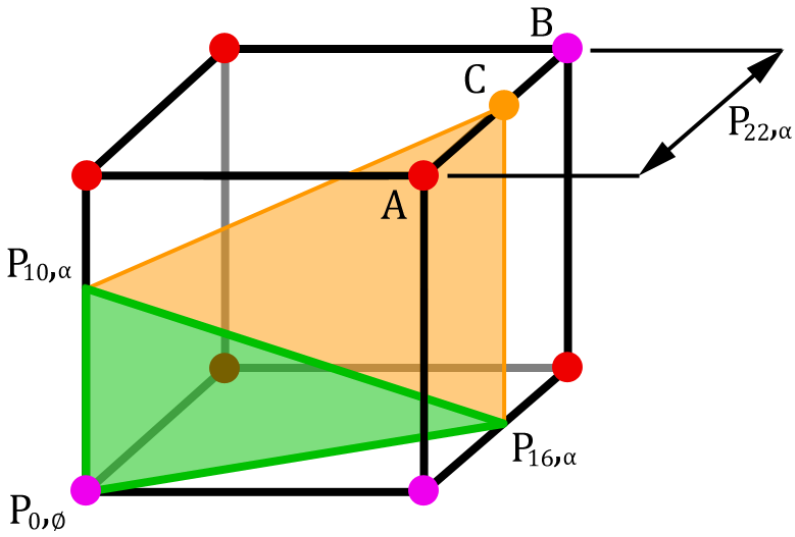

Figura 3.8: Caso donde la orientación de un tetraedro no se puede determinar sin calcular la posición de las intersecciones en el espacio.

Para observar por qué ocurre esto, se pueden fijar los puntos $P_{0,\emptyset}$, $P_{10,\alpha}$ y $P_{16,\alpha}$ y mover el punto $P_{22,\alpha}$ a lo largo de la arista $\overline{AB}$ de la celda; mientras éste se acerque al vóxel A el tetraedro tendrá volumen positivo y mientras éste se acerque al vóxel B el tetraedro tendrá volumen negativo (ver Sección 2.4.2 para más información sobre la definición del volumen de un tetraedro). Inclusive, existe un punto intermedio C en cual el tetraedro tendrá volumen nulo, por lo cual se considerará un tetraedro degenerado.

Para solucionar esto, se puede verificar cada tetraedro antes de añadirlo al mallado final, y en caso de tener volumen negativo se intercambian los dos últimos vértices del tetraedro, y en caso de tener volumen nulo simplemente no se añade al mallado final.

3.5. CONSIDERACIONES PARA EL MANEJO DE CASOS AMBIGUOS

Como se mencionó en la Sección 3.2, el algoritmo propuesto no realiza ningún procesamiento adicional para manejar casos ambiguos en la celda, sino asume que la intersección del intervalo de volumen con la celda consiste en un único poliedro convexo o el vacío. En esta sección se analizarán los distintos tipos de ambigüedades existentes y las posibles soluciones para el manejo de estos casos, para finalmente analizar la salida del algoritmo propuesto así como las limitaciones de éste que impiden el manejo de casos ambiguos de forma sencilla.

3.5.1. MODELOS TOPOLÓGICAMENTE CORRECTOS

Para analizar los casos ambiguos que se pueden encontrar durante la ejecución del algoritmo propuesto basado en cubos marchantes, primero se debe definir qué es una superficie topológicamente correcta. En [CHE95] se define que una superficie es topológicamente correcta si y sólo si la topología de los tetraedros generados coincide con la topología de la función $F(x, y, z)$ dentro de cada celda.

Para obtener un intervalo de volumen topológicamente correcto, dos vóxeles de la misma clase (positivo, negativo o neutro) deben estar unidos dentro de una celda si y sólo si existe un camino dentro de ésta que conecte ambos vóxeles y no cambie de clase. El caso más simple para determinar si dos vóxeles están conectados es cuando existe un camino a través de las aristas de la celda que sólo contiene vóxeles de la misma clase. Sin embargo, hay dos casos donde la regla anterior no es suficiente. Por ejemplo, cuando se tiene una cara con dos vóxeles positivos y dos vóxeles negativos en esquinas opuestas que no están conectados a través de las demás aristas de la celda, no se puede decir directamente si los vóxeles positivos están unidos, si los vóxeles negativos están unidos o los cuatro vóxeles están completamente separados, debido a que éstos podrían estar unidos mediante un camino que pase por dentro de la cara. A este tipo de caras se les denomina caras ambiguas, como la que se puede observar en la Figura 3.9.

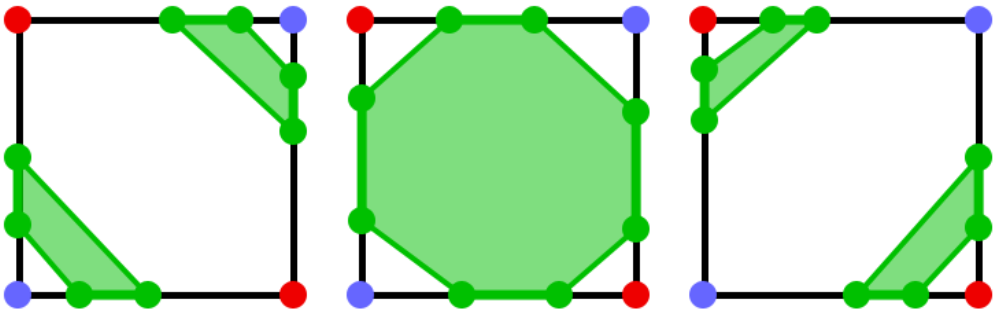

Figura 3.9: Ejemplo de una cara ambigua. A la izquierda se muestra el caso donde los vóxeles positivos se encuentran unidos, a la derecha se muestra el caso donde los vóxeles negativos se encuentran unidos, y en el centro se muestra el caso donde éstos son separados.

Debido a la existencia de tres tipos diferentes de vóxeles, existen varios tipos de caras ambiguas, entre las cuales se encuentran:

- Dos vóxeles positivos y dos vóxeles neutros, con los vóxeles positivos en esquinas opuestas. En este caso se puede elegir entre unir los vóxeles positivos o separarlos.

- Dos vóxeles negativos y dos vóxeles neutros, con los vóxeles negativos en esquinas opuestas. En este caso se puede elegir entre unir los vóxeles positivos o separarlos.

- Dos vóxeles positivos, un vóxel neutro y un vóxel negativo, con los vóxeles positivos en esquinas opuestas. En este caso se puede elegir entre unir los vóxeles positivos o separarlos.

- Dos vóxeles negativos, un vóxel neutro y un vóxel positivo, con los vóxeles negativos en esquinas opuestas. En este caso se puede elegir entre unir los vóxeles negativos o separarlos.

- Dos vóxeles positivos y dos vóxeles negativos, con los vóxeles positivos en esquinas opuestas. En este caso se puede elegir entre unir los vóxeles positivos, unir los vóxeles negativos o separar los cuatro vóxeles.

En la Figura 3.10 se pueden observar las distintas caras ambiguas posibles, así como todas las soluciones posibles para cada uno de ellos. Cualquier método a utilizar para la resolución de ambigüedades debe tomar en cuenta todos los casos posibles, de forma que se pueda obtener la topología correcta de la función $F(x, y, z)$.

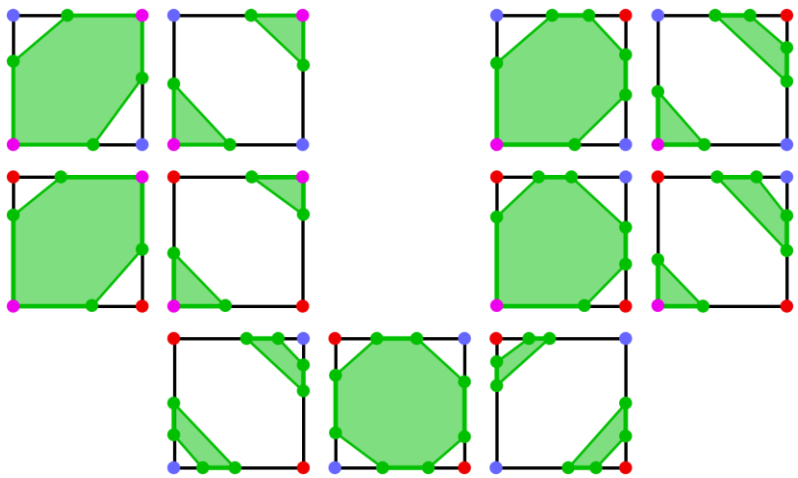

Figura 3.10: Posibles caras ambiguas durante la extracción de intervalos de volumen mediante cubos marchantes.

3.5.2. MANEJO DE AMBIGÜEDADES EN LAS CARAS DE LA CELDA

Para manejar las ambigüedades en las caras de la celda, se puede extender el método de la decisión asintótica propuesto por Nielson [NIE91] para la resolución de los casos ambiguos durante la extracción de isosuperficies (ver Sección 2.2.1.4). Este método consiste en elegir entre separar o no separar dos vóxeles opuestos de una cara ambigua en base a la evaluación de un punto medio de ésta por medio de interpolación bilineal, de forma tal que la superficie generada sea topológicamente correcta (ver Sección 3.5.1).

La interpolación bilineal a través de una cara es la extensión natural a realizar interpolación lineal a través de un segmento en dos dimensiones. Después de realizar un cambio de variables, se puede asumir que el dominio de la cara es un cuadrado unitario $\{(s,t)|0 \leq s,t \leq 1\}$, lo cual lleva a la siguiente fórmula para realizar interpolación bilineal:

$$B(s,t) = (1-s \quad s)\begin{pmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \end{pmatrix}\begin{pmatrix} 1-t \\ t \end{pmatrix}$$

donde $B_{0,0}$, $B_{0,1}$, $B_{1,0}$ y $B_{1,1}$ son los isovalores de los vóxeles que delimitan la cara ambigua.

Al igual que en el método original propuesto por Nielson [NIE91], se puede verificar fácilmente que las curvas $H_\alpha = \{(s,t)|B(s,t) = \alpha\}$ y $H_\beta = \{(s,t)|B(s,t) = \beta\}$ son hipérbolas. Hay varias formas en que el dominio de la cara puede intersectar (o no intersectar) estas hipérbolas, como se puede observar en la Figura 3.11.

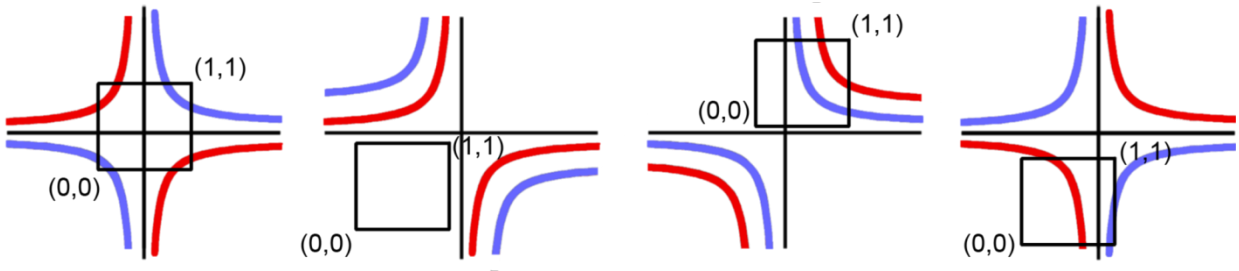

Figura 3.11: Distintos resultados posibles de intersección con las hipérbolas H_α y H_β, donde H_α está representada por la curva de color rojo y H_β está representada por la curva de color azul.

Los casos ambiguos se dan cuando el dominio intersecta ambas componentes de la hipérbola, en cuyo caso no se puede determinar a priori si los vóxeles positivos están dentro de la misma región de las hipérbolas. Por ejemplo, en la Figura 3.12 se pueden observar tres formas posibles

en que el dominio de la cara intersecte las hipérbolas H_α y H_β en la cara ambigua mostrada en la Figura 3.9.

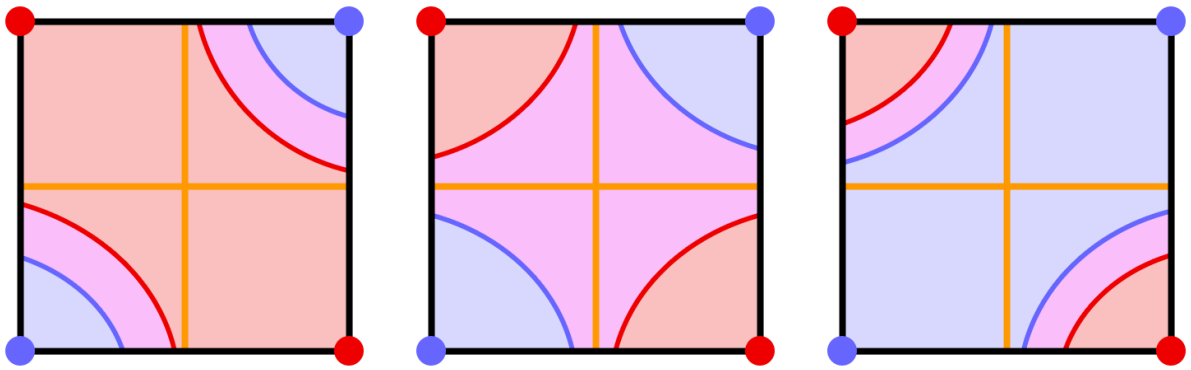

Figura 3.12: Posibles hipérbolas para la cara ambigua mostrada en la Figura 3.9.

La solución propuesta por Nielson [NIE91] puede ser adaptada para la extracción de intervalos de volumen utilizando mediante la evaluación de $F(x, y, z)$ en el punto de la cara donde las asíntotas de las hipérbolas se intersectan. La evaluación de este punto permite determinar el par de vóxeles opuestos a unir, o si éstos deben ser separados en su totalidad.

El primer paso consiste en buscar las asíntotas $\{(s,t)|s = S_h\}$ y $\{(s,t)|t = T_h\}$, donde:

$$S_h = \frac{B_{0,0} - B_{0,1}}{B_{0,0} + B_{1,1} - B_{0,1} - B_{1,0}}, \quad T_h = \frac{B_{0,0} - B_{1,0}}{B_{0,0} + B_{1,1} - B_{0,1} - B_{1,0}}$$

por lo cual se puede calcular el valor en $B(S_\alpha, T_\alpha)$ mediante interpolación bilineal:

$$B_{S,T} = B(S_h, T_h) = \frac{B_{0,0}B_{1,1} - B_{1,0}B_{0,1}}{B_{0,0} + B_{1,1} - B_{0,1} - B_{1,0}}$$

Finalmente, para determinar a qué caso de la Figura 3.12 corresponde la cara ambigua se compara $B_{S,T}$ con el intervalo $[\alpha, \beta]$ a extraer: Si $B_{S,T}$ se encuentra por debajo del intervalo ($B_{S,T} < \alpha$), entonces los vóxeles negativos deben ser unidos; si $B_{S,T}$ se encuentra por encima del intervalo ($B_{S,T} > \beta$), entonces los vóxeles positivos deben ser unidos; en caso contrario ($\alpha \leq B_{S,T} \leq \beta$) los cuatro vóxeles deben ser separados. Estos tres casos se pueden observar en la Figura 3.13.

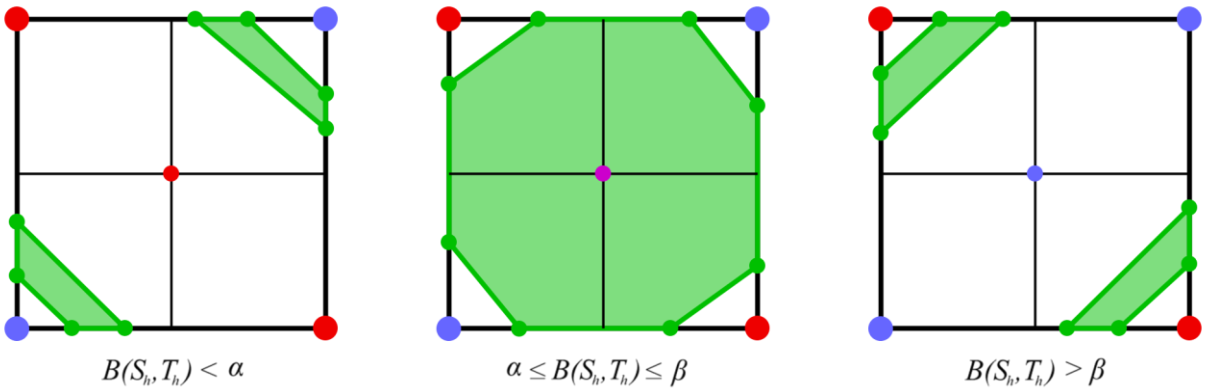

Figura 3.13: Triangulación final después de evaluar la intersección de las asíntotas.

3.5.3. SALIDA DEL ALGORITMO PROPUESTO

Como se mencionó en la Sección 3.2, el algoritmo propuesto para la extracción de intervalos de volumen mediante cubos marchantes no realiza ningún procesamiento adicional para el manejo de los casos ambiguos. El algoritmo propuesto, en lugar de determinar cuál solución de la Figura 3.10 es topológicamente correcta, el algoritmo elige resolver las ambigüedades separando todos los vóxeles de la celda, ya que éste construye el poliedro-$\alpha\beta$ como la cápsula convexa [GRA72] de los puntos de intersección.

Es importante mencionar que a pesar de que todas las resoluciones posibles de los casos ambiguos consisten en piezas convexas, no se puede garantizar que el poliedro-$\alpha\beta$ sea convexó o esté compuesto por piezas convexas únicamente. Por ejemplo, en la Figura 3.14 se puede observar el caso $(10110111)_3$, en el cual se obtiene un poliedro cóncavo si se unen los vóxeles negativos de la cara superior, ya que la línea $\overline{AB}$ no se encuentra completamente dentro del poliedro.

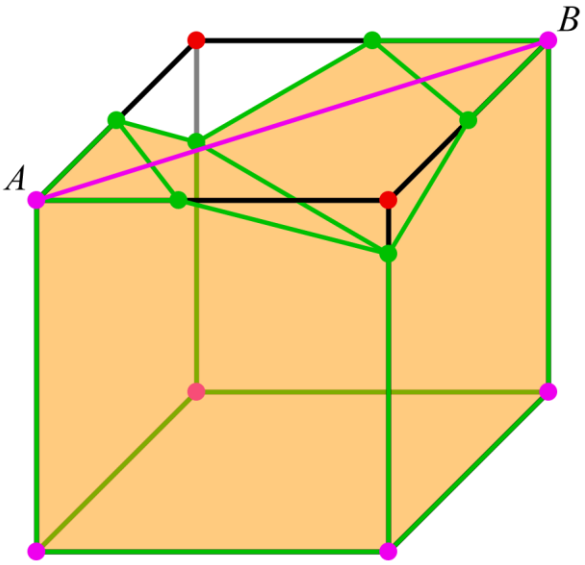

Figura 3.14: Caso donde el poliedro-$\alpha\beta$ es cóncavo debido a la unión de los vóxeles negativos.

Debido a que no se puede garantizar que el poliedro-$\alpha\beta$ es convexo si se manejan los casos ambiguos, entonces es imposible tetraedrizar este poliedro sin conocer la posición final de los puntos de intersección [MAX01]. Esto impide el uso de una tabla de conectividad para generar los tetraedros de salida, lo cual ocasiona que se requiera el uso de un algoritmo para la tetraedrización de poliedros cóncavos durante el procesamiento de cada celda del volumen.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBAS

Para implementar los algoritmos mencionados en los objetivos específicos (ver Sección 1.2), se utilizó el lenguaje de programación C++, debido a que éste provee las funcionalidades necesarias para la implementación eficiente de estos, así como diversas bibliotecas con código abierto reusable, lo que facilita la implementación de las partes no relacionadas con el algoritmo, como la interfaz gráfica de usuario. Entre las herramientas y bibliotecas utilizadas para la implementación se encuentran:

- **Visual Studio 2008[13]:** Consiste en el entorno de desarrollo (IDE) proporcionado por Microsoft para la creación de aplicaciones y bibliotecas en C++.
- **STLport 5.2.1[14]:** Consiste en una implementación de la Biblioteca Estándar de Plantillas de C++ (*Standard Template Library*, STL), la cual implementa extensiones importantes de la misma, como lo son los contenedores asociativos por *hashing*.
- **Boost 1.38.0[15]:** Biblioteca código abierto con extensiones para la biblioteca estándar de C++, incluyendo funcionalidades que permiten el manejo de múltiples hilos, programación genérica, arreglos estáticos y manejo del sistema de archivos.
- **Qt Open Source 4.5.1[16]:** Consiste en un conjunto de bibliotecas para C++, las cuales implementan funcionalidades para el diseño de interfaces gráficas de usuario (GUIs). Se escogió esta biblioteca debido a que es de código abierto y portable entre distintos tipos de sistemas.
- **Doxygen 1.5.8[17]:** Consiste en una aplicación para generar documentación HTML para código fuente, el cual soporta C, C++ y Java, entre otros lenguajes.

Para evaluar el desempeño del algoritmo propuesto, se desarrollaron dos aplicaciones de prueba para el manejo de datos volumétricos; la primera para la extracción de isosuperficies y la segunda para la extracción de intervalos de volumen. Para la extracción de isosuperficies se implementó cubos marchantes (ver Sección 2.2.1), tetracubos marchantes (ver Sección 2.2.2) y

[13] http://www.microsoft.com/visualstudio
[14] http://www.sourceforge.net/projects/stlport
[15] http://www.boost.org
[16] http://www.qtsoftware.com
[17] http://www.stack.nl/~dimitri/doxygen

cubos marchantes con resolución de ambigüedades (ver Secciones 2.2.1.4 y 2.2.1.5), y para la extracción de intervalos de volumen se implementó tetracubos marchantes (ver Sección 2.3.3) y el algoritmo propuesto basado en cubos marchantes (ver Capítulo 3).

Con el objetivo de comparar los algoritmos implementados, se utilizaron cuatro volúmenes de datos, donde los tres primeros fueron obtenidos a partir de tomografías computarizas (CT) y el último fue obtenido a partir de la ecuación implícita de una hipérbola, cuyos datos básicos se pueden observar en la Tabla 4.1.

ID	Volumen	Fuente	Vóxeles	Tipo de dato	Bytes
1	Brazo derecho	CT	492×240×155	Entero de 8 bits.	53MB
2	Cabeza	CT	256×256×113	Entero de 16 bits.	42MB
3	Muñeca derecha	CT	256×256×183	Entero de 8 bits.	34MB
4	Hipérbola	$F(x,y,z) = xyz$	128×128×128	Flotante simple.	24MB

Tabla 4.1: Volúmenes de datos utilizados para probar los algoritmos implementados.

Para cada volumen se realizó la extracción de diversas isosuperficies e intervalos de volumen, utilizando los algoritmos implementados y evaluando tanto el tiempo requerido para la extracción del mallado final así como la calidad de éste utilizando las métricas descritas en la Sección 2.4.

A continuación se mostrarán los resultados obtenidos de las pruebas realizadas sobre los algoritmos de extracción de isosuperficies y de intervalos de volumen utilizando los volúmenes descritos en la Tabla 4.1, para finalmente realizar comparaciones cualitativas sobre los modelos obtenidos de los algoritmos para la extracción de intervalos de volumen.

4.1. RESULTADOS OBTENIDOS DE LA EXTRACCIÓN DE ISOSUPERFICIES

Para evaluar el desempeño de los algoritmos de extracción de isosuperficies, se realizaron extracciones de distintas isosuperficies de los cuatro volúmenes de prueba descritos en la Tabla 4.1, de las cuales se obtuvo el tiempo necesario para la extracción de las isosuperficies así como la calidad de los mallados que los aproximan. En los siguientes gráficos se muestra el comportamiento del tiempo de ejecución de los distintos algoritmos de extracción de isosuperficies en base al isovalor extraído.

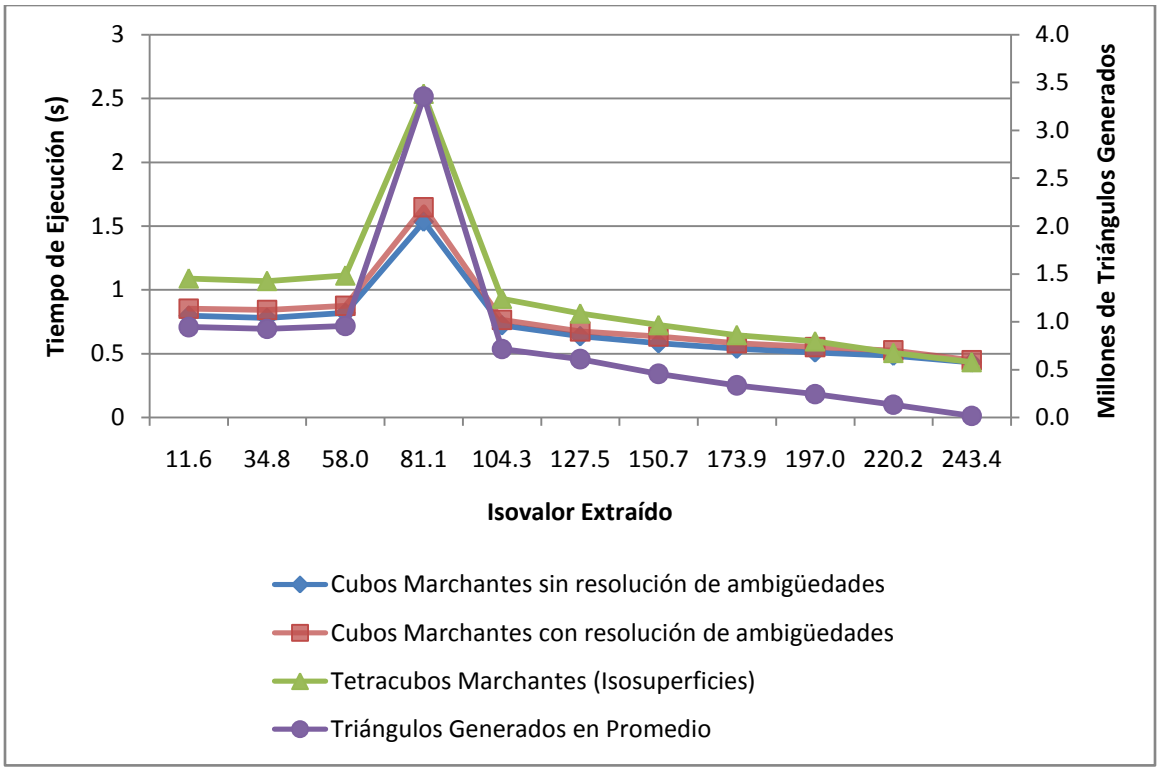

Gráfico 4.1: Comparación del tiempo de ejecución por los algoritmos de extracción de isosuperficies del Volumen #1 (Brazo derecho).

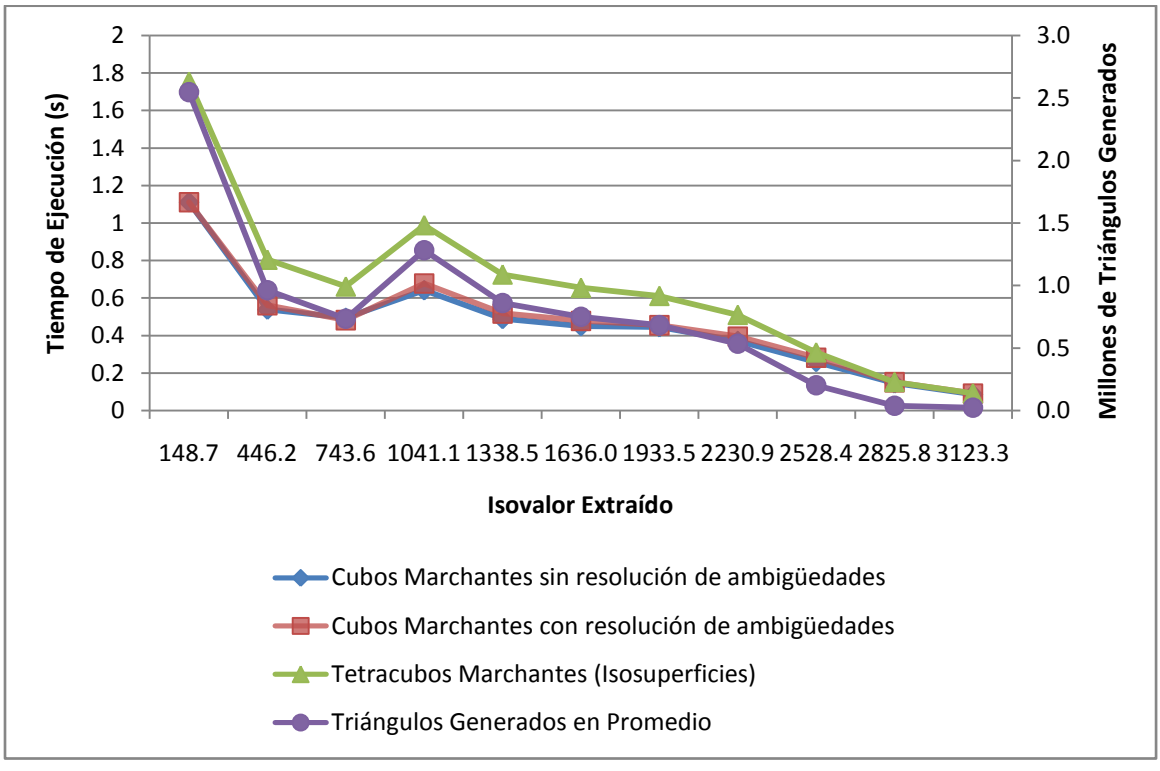

Gráfico 4.2: Comparación del tiempo de ejecución por los algoritmos de extracción de isosuperficies del Volumen #2 (Cabeza).

En ambos gráficos se puede observar que el tiempo de ejecución requerido por tetracubos marchantes es claramente superior al tiempo de ejecución de cubos marchantes, en un 25% si se aplica resolución de ambigüedades y en un 30% si no se aplica. También se puede resaltar que el costo en tiempo de ejecución para realizar la resolución de ambigüedades en cubos marchantes es insignificante, ya que ambos algoritmos mostraron el mismo desempeño. Finalmente, se puede deducir que el tiempo de ejecución del algoritmo es directamente proporcional al tamaño del mallado generado, debido a que las curvas del tiempo de ejecución siguen el mismo patrón de la curva de la cantidad de triángulos generados.

Otro aspecto importante a tomar en cuenta a la hora de evaluar un algoritmo para la extracción de isosuperficies consiste en calcular el número de primitivas utilizadas por el algoritmo para representar el mallado final, ya que esto indicará la facilidad con la cual se podrán visualizar y manipular los mallados obtenidos. A continuación se puede observar un gráfico donde se muestra la cantidad de primitivas generadas por los algoritmos de extracción de isosuperficie para el Volumen #1 (Brazo derecho):

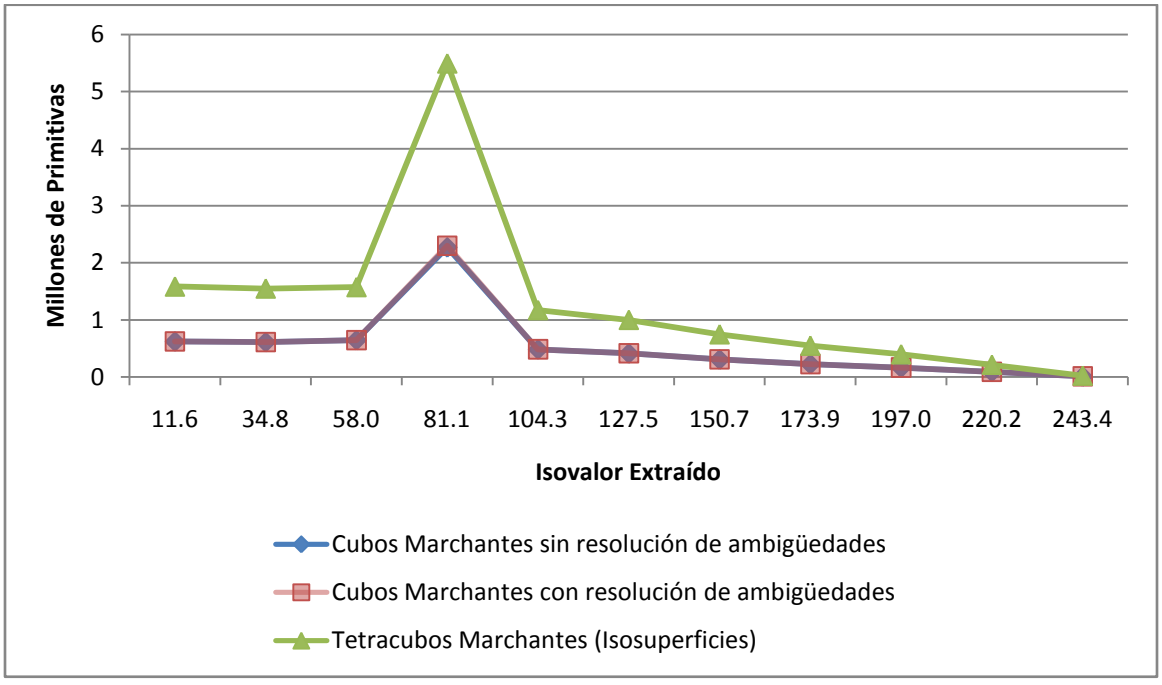

Gráfico 4.3: Comparación del número de primitivas generadas por los algoritmos de extracción de isosuperficies del Volumen #1 (Brazo derecho).

Como se puede observar en el Gráfico 4.3, la complejidad de los modelos generados por tetracubos marchantes es aproximadamente un 145% más compleja que aquellos modelos

generados por cubos marchantes, se aplique o no resolución de ambigüedades. Esto se debe a que tetracubos marchantes requiere dividir cada celda del volumen en tetraedros (ver Sección 2.2.2.1) para luego extraer triángulos de cada uno de ellos, mientras que cubos marchantes no requiere esta división previa, lo cual ocasiona que genere una salida más sencilla. También se puede destacar que la aplicación de resolución de ambigüedades en cubos marchantes no requiere la generación de una cantidad significativa de primitivas adicionales, lo que se debe a que los casos ambiguos son poco frecuentes en general [NIE91]. El valor elevado de primitivas generado para el isovalor 81.1 se debe a que este valor representa la densidad intermedia entre la densidad de los tejidos blandos y la de los huesos.

Finalmente, se evaluaron los mallados obtenidos por los tres algoritmos utilizando las métricas expuestas en la Sección 2.4.1, con el fin de predecir la calidad de la visualización y manipulación que se puede obtener de estos mallados. En el siguiente gráfico se puede observar el porcentaje de primitivas de buena calidad generadas por los tres algoritmos a partir del Volumen #3, utilizando la relación de aspecto como valor de referencia (ver Sección 2.4.1.2):

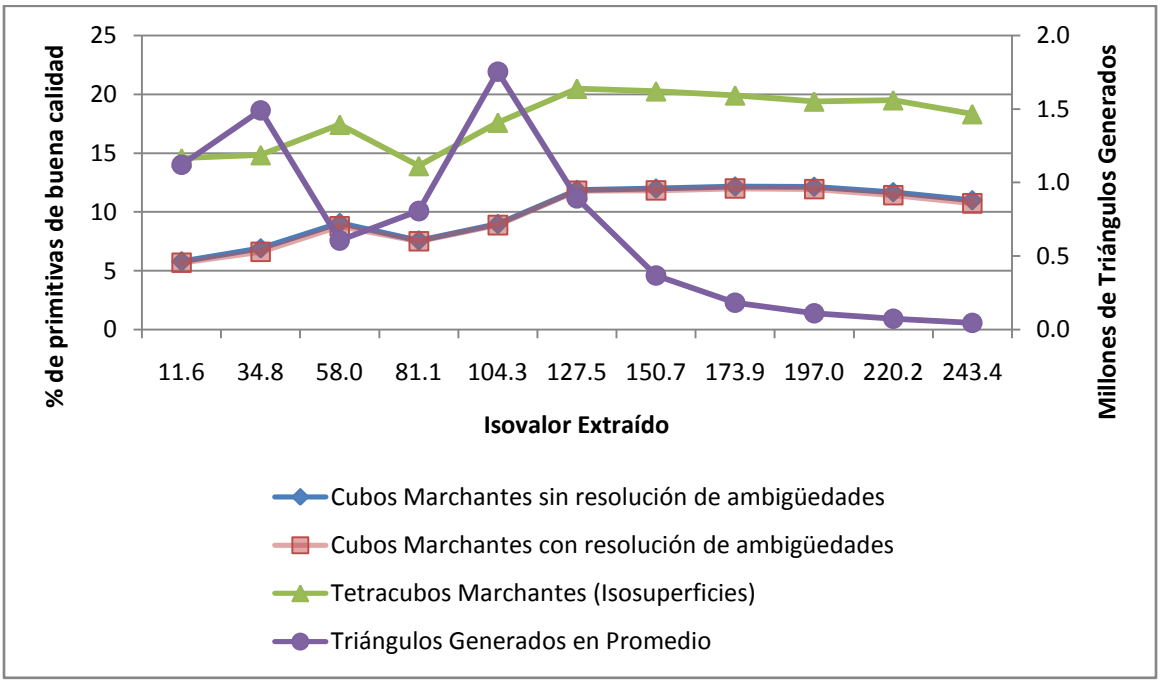

Gráfico 4.4: Comparación del número de primitivas de buena calidad generadas por los algoritmos de extracción de isosuperficies del Volumen #3 (Muñeca), utilizando la relación de aspecto como referencia.

En el Gráfico 4.4 se puede observar que tetracubos marchantes extrae un mallado triangular con una cantidad de primitivas de alta calidad 85% mayor en comparación con cubos marchantes. Esto se puede confirmar en el Gráfico 4.5, donde se puede verificar que la calidad de los triángulos generados por tetracubos marchantes es 4% superior en promedio. Es importante recordar que la relación de aspecto es una métrica propia, es decir, para las primitivas ideales el valor de la métrica se aproxima a 1, y para las primitivas degeneradas el valor de la métrica se aproxima a ∞ (ver Sección 2.4).

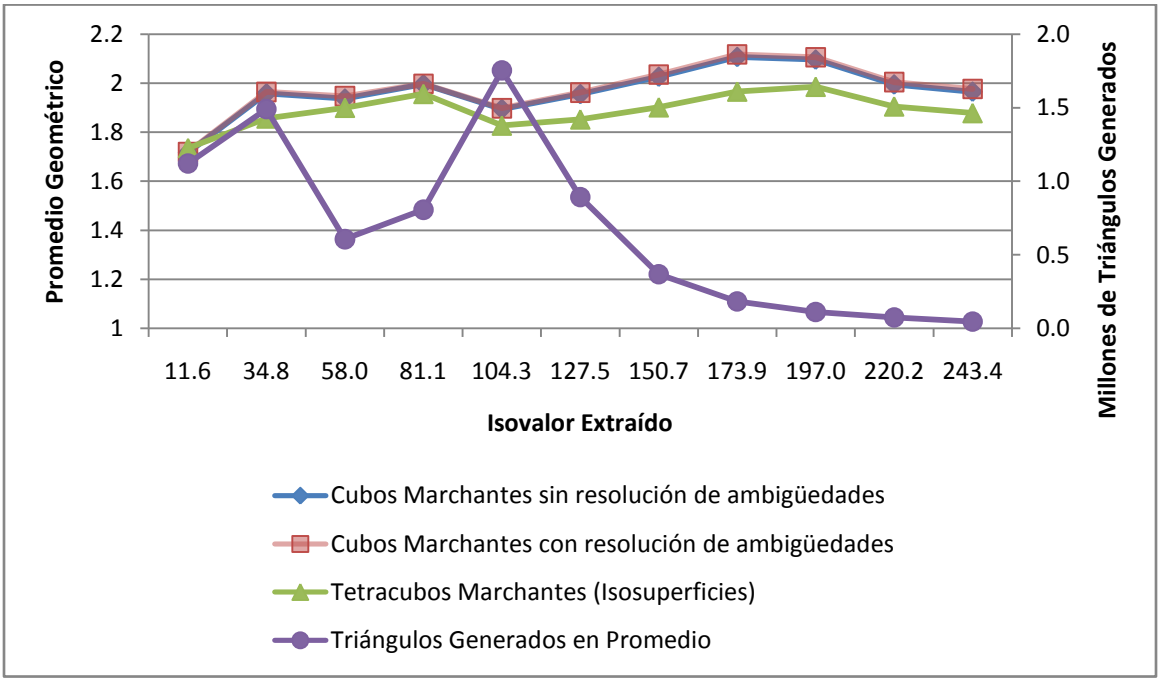

Gráfico 4.5: Comparación de la calidad de las primitivas generadas por los algoritmos de extracción de isosuperficies del Volumen #3 (Muñeca), utilizando el promedio geométrico de la relación de aspecto como referencia.

En base al Gráfico 4.4 y al Gráfico 4.5 también se puede deducir que la calidad del mallado generado es independiente a la cantidad de primitivas que lo componen, ya que el porcentaje de triángulos de buena calidad y el promedio geométrico de la calidad de éstos no varían considerablemente, a pesar del tamaño del mallado final.

4.2. RESULTADOS OBTENIDOS DE LA EXTRACCIÓN DE INTERVALOS DE VOLUMEN

Para evaluar el desempeño de los algoritmos de extracción de intervalos de volumen, se realizaron extracciones de distintos intervalos de los cuatro volúmenes de prueba descritos en la

Tabla 4.1, de los cuales se obtuvo el tiempo necesario para la extracción de éstos así como la calidad de los mallados que las aproximan. En los siguientes gráficos se muestra el comportamiento del tiempo de ejecución de los distintos algoritmos de extracción de intervalos de volumen en base al isovalor extraído.

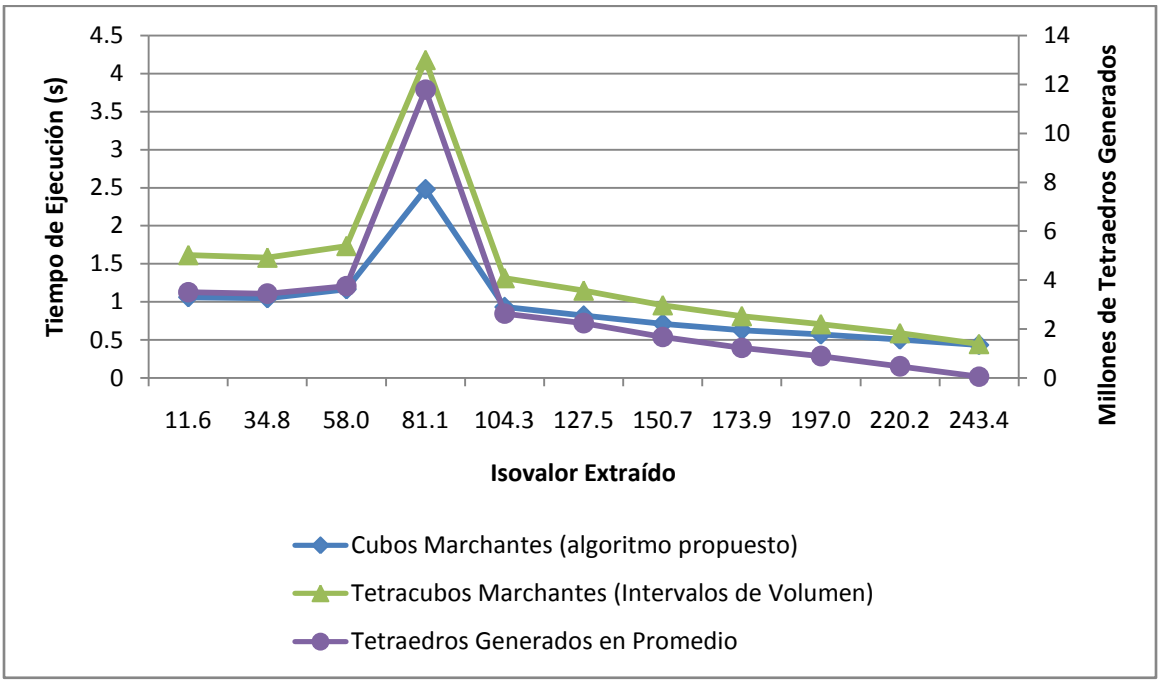

Gráfico 4.6: Comparación del tiempo de ejecución por los algoritmos de extracción de intervalos de volumen del Volumen #1 (Brazo derecho).

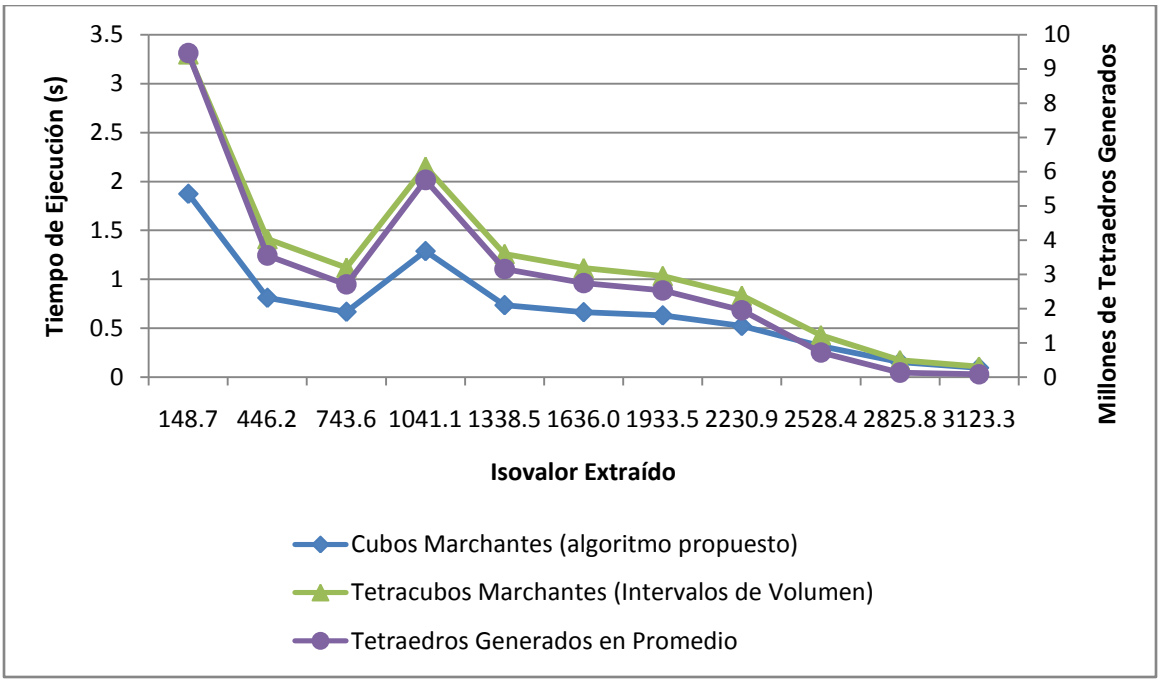

Gráfico 4.7: Comparación del tiempo de ejecución por los algoritmos de extracción de intervalos de volumen del Volumen #2 (Cabeza).

En ambos gráficos se puede observar que el tiempo de ejecución requerido por tetracubos marchantes es superior en un 40% al tiempo de ejecución de cubos marchantes, lo cual se debe a que el algoritmo propuesto realiza un procesamiento menor al realizado por tetracubos marchantes, ya que no tiene que calcular las intersecciones en las diagonales de las caras de la celda, las cuales son necesarias para dividir la celda en tetraedros. Finalmente, se puede deducir que el tiempo de ejecución del algoritmo es directamente proporcional al tamaño del mallado generado, debido a que las curvas del tiempo de ejecución siguen el mismo patrón de la curva de la cantidad de tetraedros generados.

Otro aspecto importante a tomar en cuenta a la hora de evaluar un algoritmo para la extracción de intervalos de volumen consiste en calcular el número de primitivas utilizadas por el algoritmo para representar el mallado final, ya que esto indicará la facilidad con la cual se podrán visualizar y manipular los mallados obtenidos. A continuación se puede observar un gráfico donde se muestra la cantidad de primitivas generadas por los algoritmos de extracción de intervalos de volumen para el Volumen #1 (Brazo derecho):

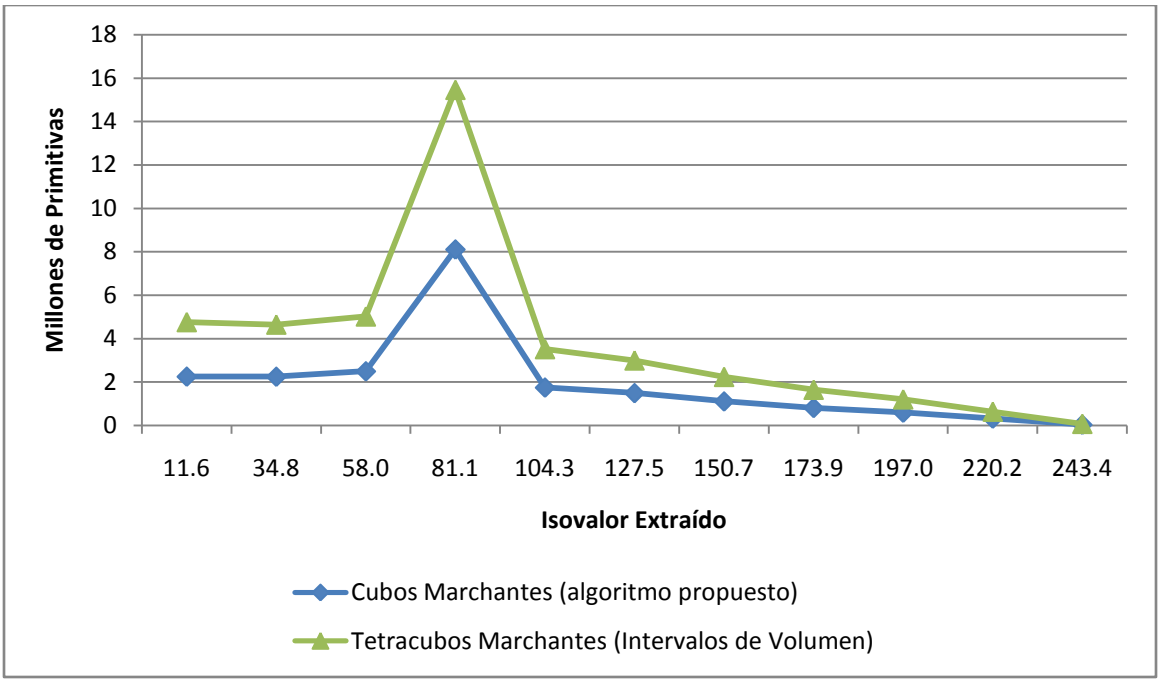

Gráfico 4.8: Comparación del número de primitivas generadas por los algoritmos de extracción de intervalos de volumen del Volumen #1 (Brazo derecho).

Como se puede observar en el Gráfico 4.8, los modelos generados por el algoritmo propuesto basado en cubos marchantes sólo requieren un 50% de las primitivas utilizadas por los modelos generados mediante el uso de tetracubos marchantes. Esto se debe a que tetracubos marchantes requiere dividir cada celda del volumen en tetraedros (ver Sección 2.2.2.1) para luego extraer tetraedros de cada uno de ellos, mientras que el algoritmo propuesto no requiere esta división previa, lo cual ocasiona que genere una salida más sencilla. Como se mencionó anteriormente, el valor elevado de primitivas generado para el isovalor 81.1 se debe a que este valor representa la densidad intermedia entre la densidad de los tejidos blandos y la de los huesos.

Finalmente, se evaluaron los mallados obtenidos por ambos algoritmos utilizando las métricas expuestas en la Sección 2.4.2, con el fin de predecir la calidad de la visualización y manipulación que se puede obtener de estos mallados. En el siguiente gráfico se puede observar el porcentaje de métricas de buena calidad generadas por ambos algoritmos a partir del Volumen #3, utilizando la relación de aspecto como valor de referencia (ver Sección 2.4.2.1):

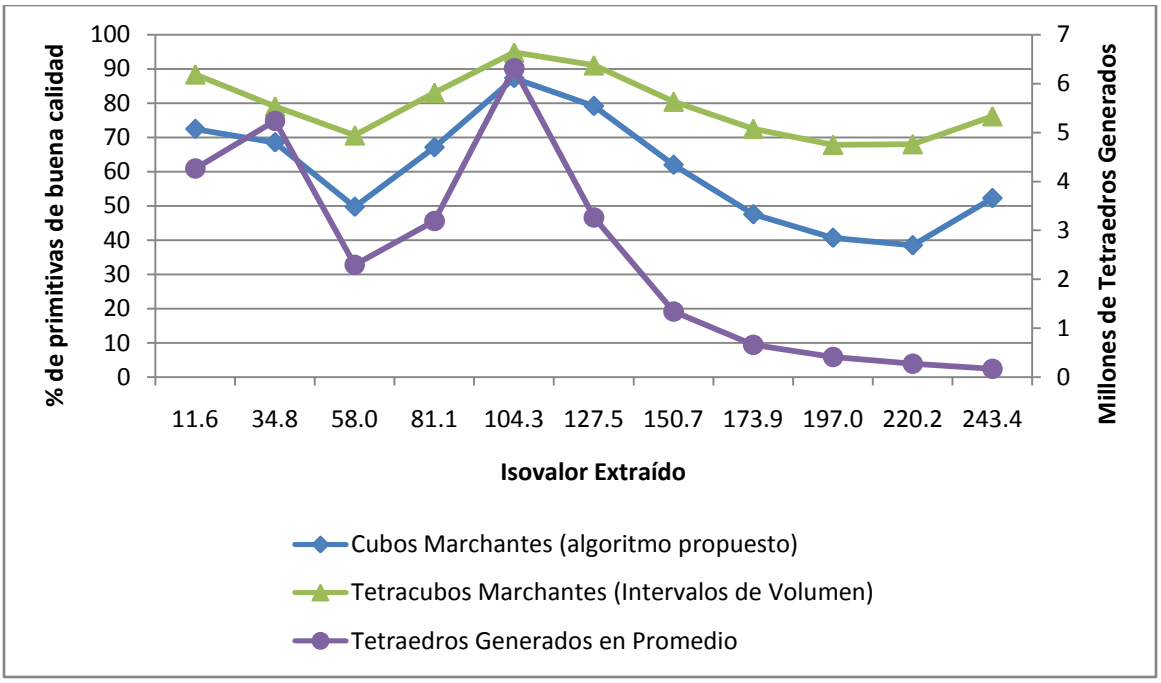

Gráfico 4.9: Comparación del número de primitivas de buena calidad generadas por los algoritmos de extracción de intervalos de volumen del Volumen #3 (Muñeca), utilizando la relación de aspecto como referencia.

En el Gráfico 4.9 se puede observar que tetracubos marchantes extrae un mallado tetraédrico con una cantidad 35% mayor de primitivas de alta calidad en comparación con cubos marchantes, lo cual se puede confirmar en el Gráfico 4.10, donde se puede verificar que la calidad promedio de los triángulos generados por tetracubos marchantes es superior en un 70%. Es importante recordar que la relación de aspecto es una métrica propia, es decir, para las primitivas ideales el valor de la métrica se aproxima a 1, y para las primitivas degeneradas el valor de la métrica se aproxima a ∞ (ver Sección 2.4).

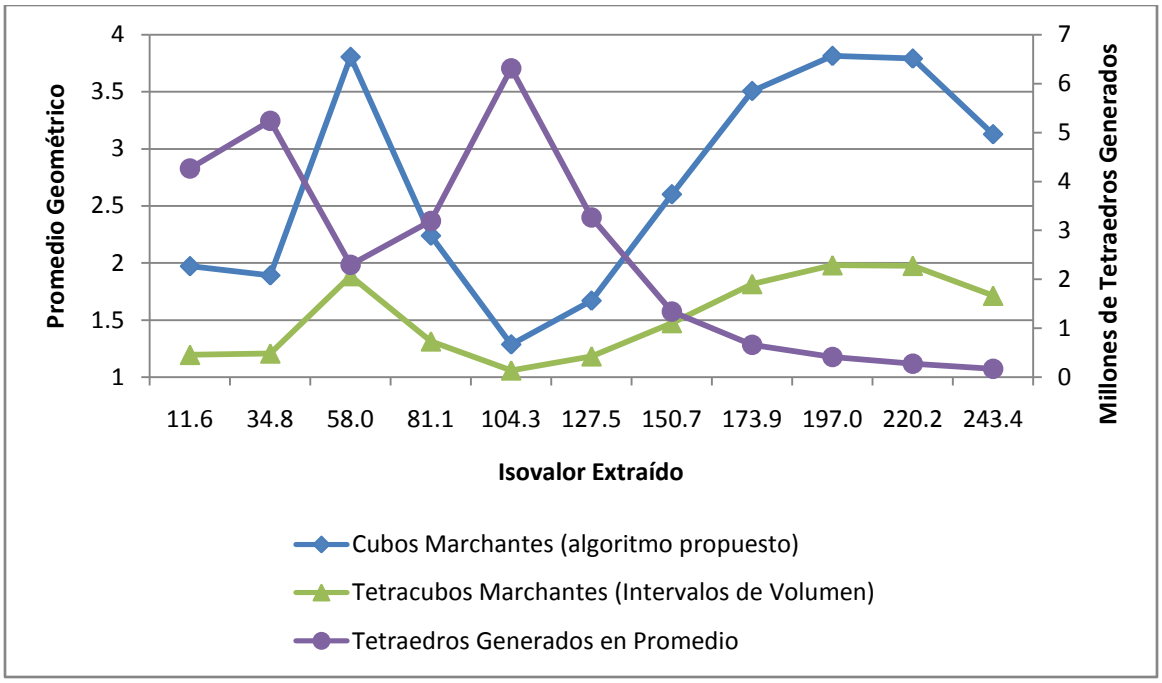

Gráfico 4.10: Comparación de la calidad de las primitivas generadas por los algoritmos de extracción de intervalos de volumen a partir del Volumen #3 (Muñeca), utilizando el promedio geométrico de la relación de aspecto como referencia.

En base al Gráfico 4.9 y al Gráfico 4.10 se puede deducir que la calidad del mallado generado es directamente proporcional a la cantidad de primitivas que lo componen, ya que, a medida que el tamaño del volumen aumenta, el porcentaje de tetraedros de buena calidad aumenta y el promedio geométrico de la calidad de éstos disminuye. Esto se debe a que los volúmenes más grandes están compuestos por una mayor cantidad de celdas internas[18], las cuales generan tetraedros de buena calidad.

4.3. ANÁLISIS CUALITATIVO SOBRE LAS ISOSUPERFICIES EXTRAÍDAS

Después de analizar cuantitativamente los resultados obtenidos por los algoritmos de extracción de isosuperficies en la Sección 4.1, se procedió a realizar un análisis cualitativo de la calidad de las isosuperficies extraídas. Este análisis consiste en observar y comparar las características de los mallados generados visualmente, con el fin de determinar cuál algoritmo produce el mejor resultado.

[18] Una celda interna es aquella delimitada por ocho (8) vóxeles neutros.

Para visualizar el resultado obtenido a partir de las superficies extraídas, se utilizaron los modelos de sombreado plano (*flat shading*) y por gradiente (*gradient-shading*), como se explicó en la Sección 2.2.1. Para implementar la interfaz gráfica de usuario se utilizaron las bibliotecas Qt y OpenGL[19], lo cual permitió la implementación sencilla y eficiente del visualizador de isosuperficies.

A continuación se analizarán varias capturas de pantalla obtenidas de algunas superficies extraídas mediantes las técnicas mencionadas anteriormente.

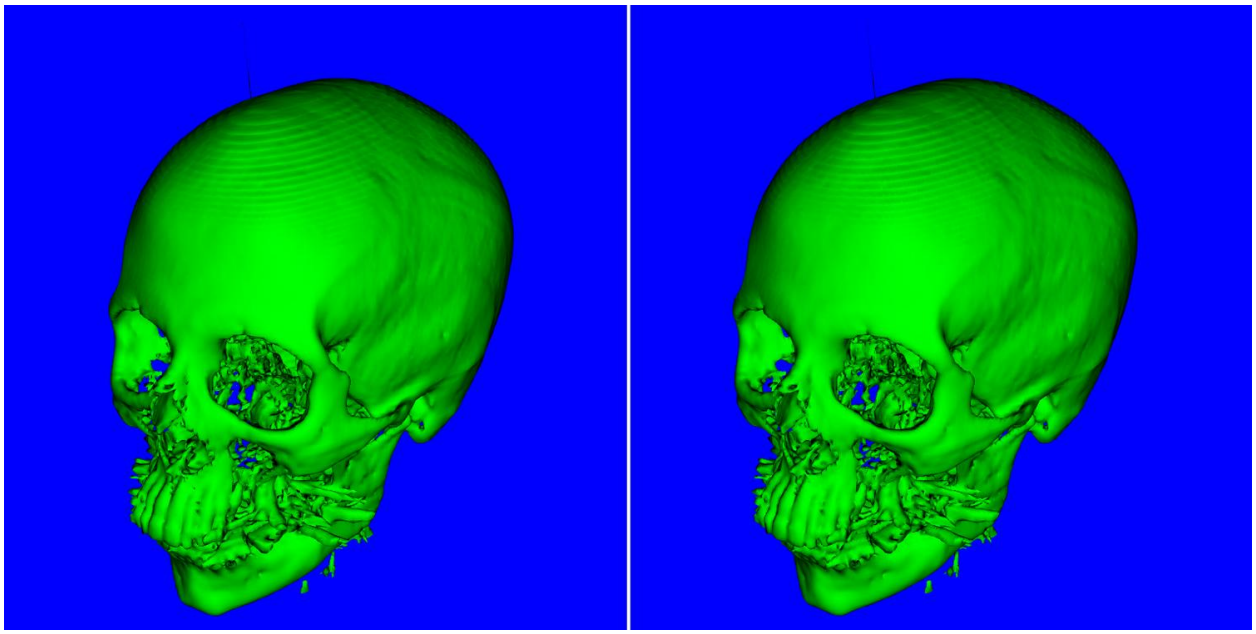

Figura 4.1: Isosuperficies obtenidas a partir del Volumen #2 (Cabeza) utilizando cubos marchantes sin resolución de ambigüedades (izquierda) y cubos marchantes con resolución de ambigüedades (derecha), rasterizadas mediante el sombreado por gradiente (*gradient-shading*).

Como se puede observar en la Figura 4.1, la salida de cubos marchantes no difiere notablemente independientemente de si se aplica resolución de ambigüedades o no. Esto se debe a que la frecuencia con la que ocurren los casos ambiguos representa menos del 1% del volumen total [NIE91].

[19] http://www.opengl.org

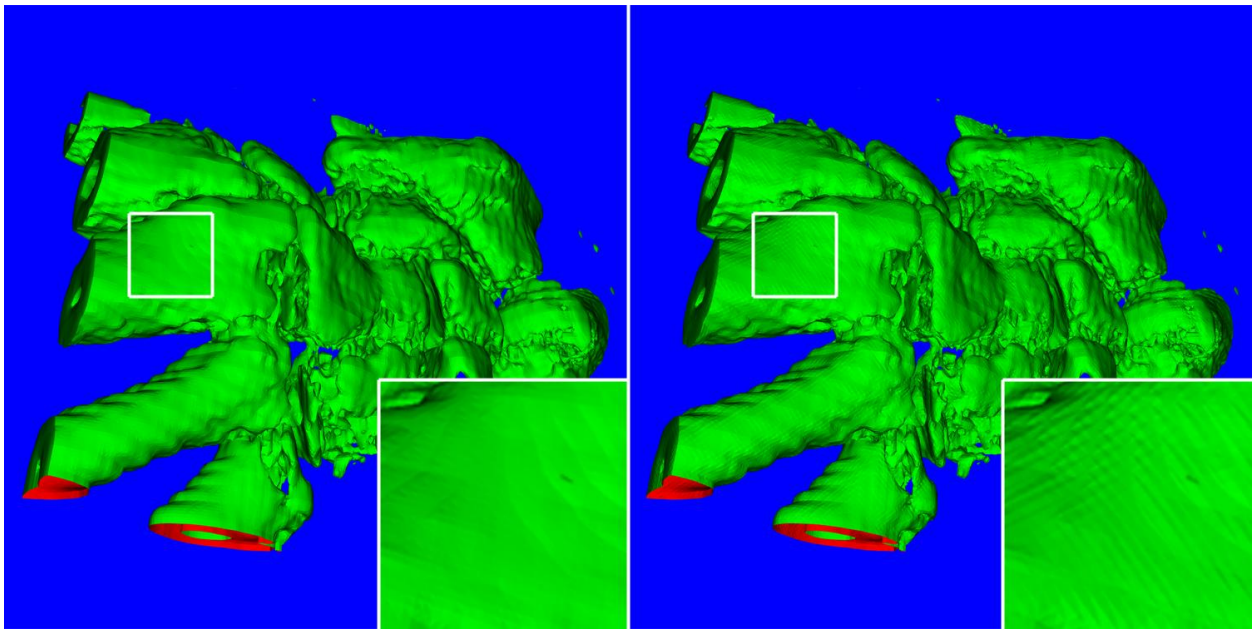

Figura 4.2: Isosuperficies obtenidas a partir del Volumen #3 (Muñeca) utilizando cubos marchantes con resolución de ambigüedades (izquierda) y tetracubos marchantes (derecha), rasterizadas mediante el sombreado plano (*flat-shading*).

En la Figura 4.2 se puede observar la extracción de una isosuperficie proveniente del Volumen #3 (Muñeca) mediante el uso de cubos marchantes y tetracubos marchantes, donde se muestra la ampliación de una sección de las superficies extraídas, en las cuales se observa que cubos marchantes produce una superficie con mucho menos ruido que aquella producida por tetracubos marchantes. La explicación a este fenómeno se encuentra en la Sección 4.4.

4.4. ANÁLISIS CUALITATIVO SOBRE LOS INTERVALOS DE VOLUMEN EXTRAÍDOS

Después de analizar cuantitativamente los resultados obtenidos por los algoritmos de extracción de intervalos de volumen en la Sección 4.2, se procedió a realizar un análisis cualitativo de la calidad de los intervalos extraídos. Este análisis consiste en observar y comparar las características de los mallados generados visualmente, con el fin de determinar cuál algoritmo produce el mejor resultado.

Para visualizar el resultado obtenido a partir de las superficies extraídas, se utilizaron los modelos de sombreado plano (*flat shading*) y por gradiente (*gradient-shading*), como se explicó en la Sección 2.2.1. Para implementar la interfaz gráfica de usuario se utilizaron las bibliotecas

Qt y OpenGL[20], lo cual permitió la implementación sencilla y eficiente del visualizador de intervalos de volumen.

En la Figura 4.3 se puede observar la extracción de un intervalo de volumen proveniente del Volumen #1 (Brazo derecho) mediante el uso del algoritmo propuesto basado en cubos marchantes y tetracubos marchantes, donde se muestra la ampliación de una sección de los intervalos de volumen extraídos. En estas ampliaciones se observa que el algoritmo propuesto produce un volumen con mucho menos ruido que aquel producido por tetracubos marchantes.

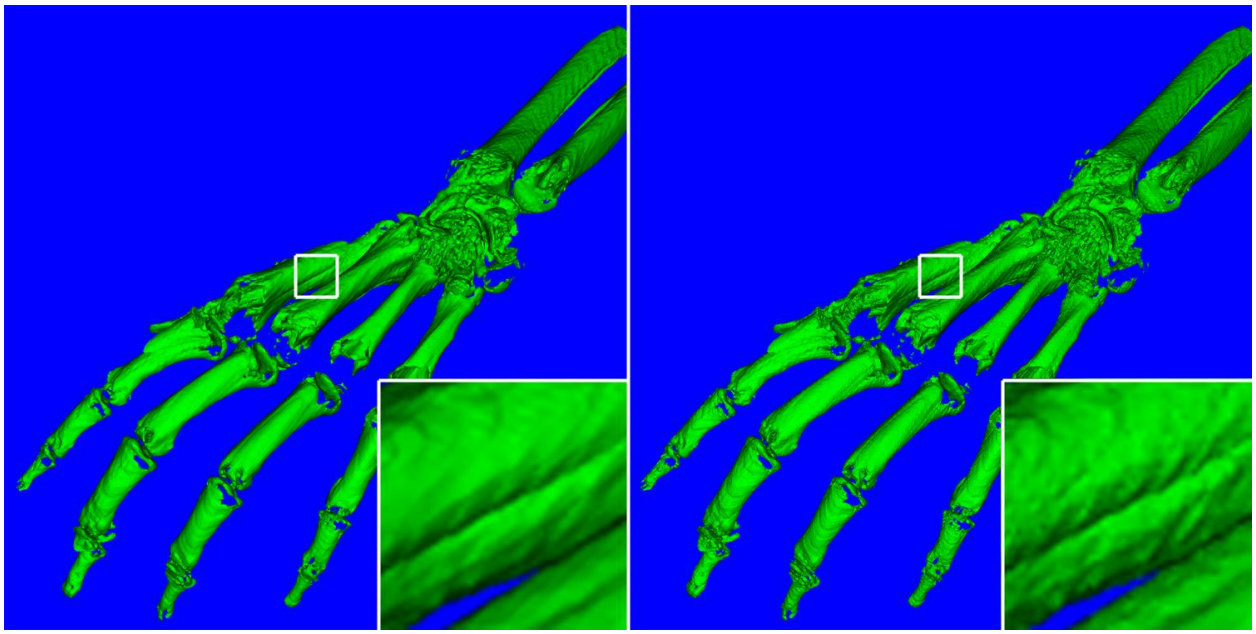

Figura 4.3: Intervalos de volumen obtenidos a partir del Volumen #1 (Brazo derecho) utilizando cubos marchantes (izquierda) y tetracubos marchantes (derecha), rasterizadas mediante el sombreado plano (*flat-shading*).

Este ruido en el mallado generado por tetracubos marchantes ocurre debido a la división adicional de la celda, la cual se requiere para su división en tetraedros. Este ruido se puede observar con más detalle en un volumen sintético de 4×2×4 vóxeles, cuyos intervalos de volumen extraídos se pueden observar en la Figura 4.4 y la Figura 4.5.

[20] http://www.opengl.org

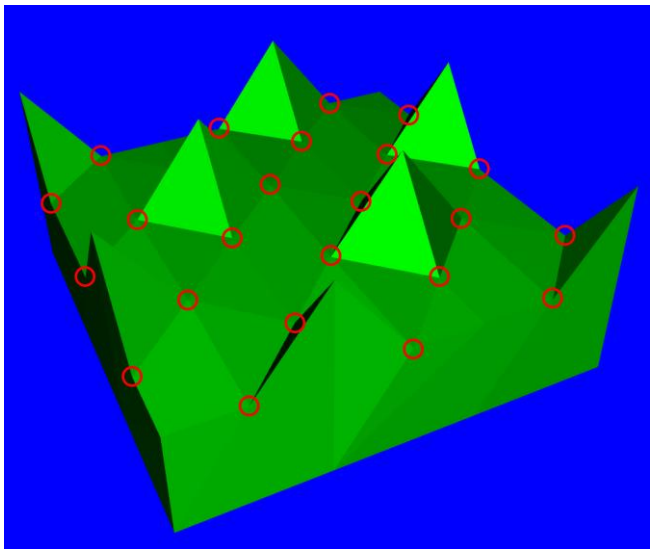

Figura 4.4: Intervalo de volumen extraído de volumen sintético de 4×2×4 vóxeles mediante tetracubos marchantes, donde los círculos rojos indican los vértices adicionales generados por intersecciones en las diagonales de las caras.

En la Figura 4.4 se pueden observar los vértices adicionales creados por tetracubos marchantes durante la extracción del intervalo de volumen, los cuales deforman el mallado extraído en cada celda, mientras que en la Figura 4.5 se observa el mismo intervalo de volumen extraído utilizando cubos marchantes, el cual produce un mallado más suave debido a que no utiliza las intersecciones en las diagonales de las caras.

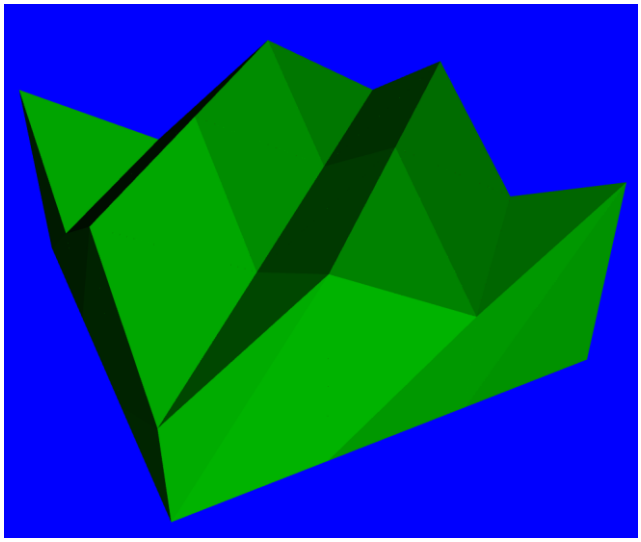

Figura 4.5: Intervalo de volumen extraído de volumen sintético de 4×2×4 vóxeles mediante cubos marchantes.

En la Figura 4.6 se muestra la extracción de una isosuperficie de un corte del cubo sintético utilizado la Figura 4.4 y la Figura 4.5 mediante el uso del algoritmo propuesto basado en cubos

marchantes y tetracubos marchantes. Como se puede observar, el algoritmo propuesto no requiere el punto de intersección en la diagonal de la cara para reconstruir la isosuperficie, lo cual ocasiona que la salida generada sea mucho más suave que aquella generada por tetracubos marchantes.

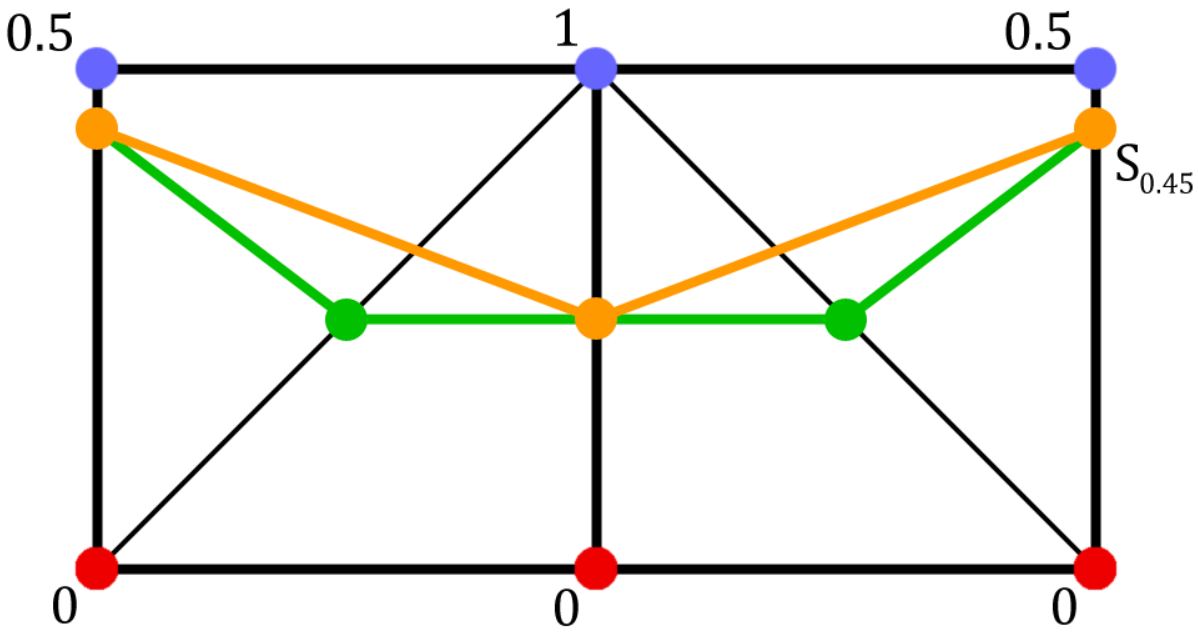

Figura 4.6: Comparación de la isosuperficie extraída de un corte del volumen sintético mediante cubos marchantes (línea anaranjada) y tetracubos marchantes (línea verde).

En la Figura 4.7 en la cual se muestran dos intervalos de volumen extraídos del Volumen #4 (Hipérbola), el primero mediante el algoritmo propuesto basado en cubos marchantes y el segundo mediante tetracubos marchantes. En esta figura se puede observar que tetracubos marchantes genera huecos en los planos de las asíntotas de la hipérbola, lo cual no ocurre en el algoritmo propuesto. En la Figura 4.8 se puede observar un volumen sintético donde se generan estos huecos, lo cual ocurre debido a los puntos adicionales que requiere tetracubos marchantes para extraer el intervalo de volumen.

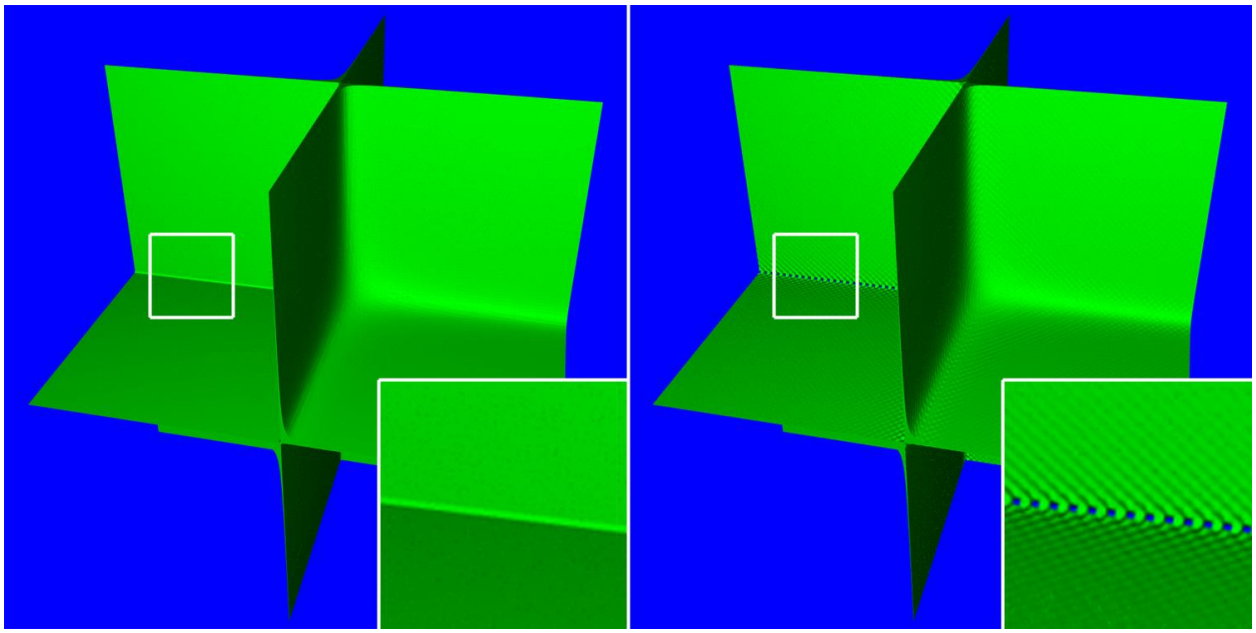

Figura 4.7: Intervalos de volumen obtenidos a partir del Volumen #4 (Hipérbola) utilizando el algoritmo propuesto basado en cubos marchantes (izquierda) y tetracubos marchantes (derecha).

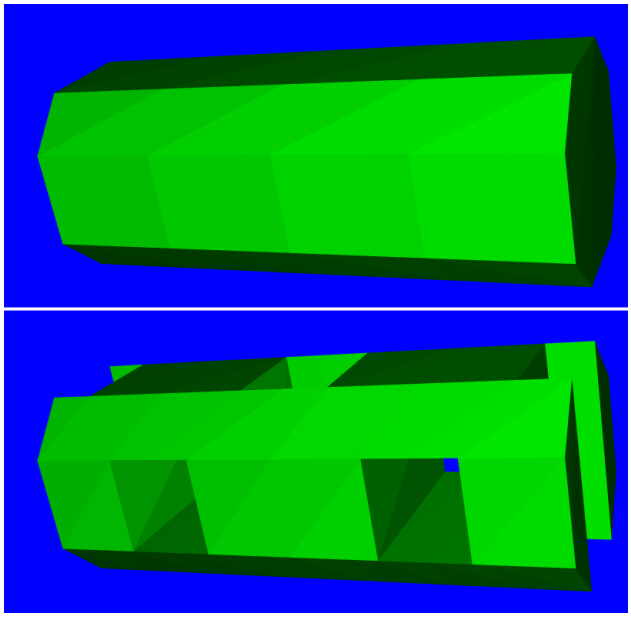

Figura 4.8: Intervalos de volumen obtenidos a partir de volumen sintético de 4×2×2 vóxeles, utilizando el algoritmo propuesto basado en cubos marchantes (arriba) y tetracubos marchantes (abajo).

CAPÍTULO 5. CONCLUSIONES

En este trabajo se propone un algoritmo para la extracción de intervalos de volumen basado en cubos marchantes, con el objetivo de obtener mallados tetraédricos de mayor calidad visual en menor tiempo. Para llevar a cabo la extracción sólo se utiliza una tabla de conectividad, sin requerir el uso de ningún algoritmo complejo durante el procesamiento del volumen, como tetraedrizaciones de Delaunay [GUO95] o la intersección de poliedros [FUJ95]. El uso de algoritmos simples y una tabla de conectividad implica una implementación más sencilla y un tiempo de ejecución menor respecto a otros algoritmos más complejos.

El algoritmo propuesto mostró un desempeño superior que el algoritmo basado en tetracubos marchantes [NIE97b], ya que genera mallados más simples en un tiempo de ejecución considerablemente menor. Además, este algoritmo no requiere una gran cantidad adicional de espacio, ya que sólo requiere almacenar cuatro cortes del volumen a procesar a la vez, y una tabla de conectividad precalculada de $3^8 = 6561$ casos.

Sin embargo, el algoritmo basado en tetracubos marchantes genera mallados con primitivas de mejor calidad en promedio, lo cual se debe a que utiliza elementos lineales, como los tetraedros, para aproximar una función lineal, mientras que el algoritmo propuesto utiliza elementos lineales para aproximar una función hiperbólica. No obstante, el algoritmo propuesto genera mallados con una mejor calidad visual que aquellos generados por el algoritmo basado en tetracubos marchantes [NIE97b], ya que no necesita utilizar las diagonales de las caras para extraer el intervalo de las celdas del volumen.

Por otra parte, el algoritmo propuesto no realiza ningún procesamiento adicional para el manejo de las ambigüedades, ya que este asume que todos los puntos de intersección dentro de una celda se encuentran conectados. La implementación del manejo de ambigüedades permitiría la generación de modelos topológicamente correctos, lo cual es importante para algunas aplicaciones de los intervalos de volumen, como la simulación de procesos mediante elementos finitos y aplicaciones médicas.

En conclusión, el algoritmo propuesto presenta diversas ventajas sobre el algoritmo basado en tetracubos marchantes, con respecto al tiempo de ejecución y al tamaño de los mallados

generados, pero sería necesario realizar comparaciones con otros algoritmos más complejos, con el fin de determinar qué aspectos de éste podrían ser mejorados.

CAPÍTULO 6. TRABAJOS A FUTURO

Como se mencionó en la Sección 3.2, el algoritmo propuesto no realiza ningún procesamiento adicional para el manejo de los casos ambiguos, sino que asume que todos los poliedros-$\alpha\beta$ son convexos y une todos los vóxeles neutros dentro de la celda. La implementación del manejo de los casos ambiguos permitiría la generación de modelos topológicamente correctos. En la Sección 3.5 se explican las consideraciones a tomar en cuenta para la resolución de los casos ambiguos, tanto para el manejo de ambigüedades en las caras como ambigüedades internas.

En las Secciones 4.2 y 4.4 se compararon los resultados obtenidos del algoritmo propuesto y la extracción de intervalos de volumen basada en tetracubos marchantes [NIE97b]. Sin embargo, no se realizaron comparaciones entre el algoritmo propuesto y las otras técnicas descritas en la Sección 2.3, como la extracción basada en formas alfa [GUO95] y la extracción basada en cubos marchantes e intersección de poliedros [FUJ95]. No obstante, se espera que el algoritmo propuesto sea más rápido que dichos algoritmos debido a que sólo se necesita indexar una tabla de conectividad, sin necesidad de recurrir a algoritmos complejos durante el procesamiento de cada celda.

Como se mencionó en la Sección 4.4, la comparación cualitativa entre los algoritmos para la extracción de intervalos de volumen se realizó mediante la rasterización directa de los tetraedros, utilizando sombreado por gradiente y (*gradient-shading*) sombreado plano (*flat-shading*). Sin embargo, la visualización correcta de estos mallados se realiza mediante el uso de técnicas de visualización directa de volúmenes (DVR), como por ejemplo, proyección de tetraedros [KRA04]. La implementación de estas técnicas permitiría realizar una visualización más precisa sobre los mallados generados por los algoritmos para la extracción de intervalos de volumen.

Finalmente, la implementación desarrollada asume la variación lineal de los isovalores en las aristas de las celdas del volumen, lo cual impide el uso de técnicas numéricas como *regula-falsi* [FOR95] para el cálculo de los puntos de intersección en volúmenes generados a partir de ecuaciones implícitas. Modificar la implementación para el uso de *regula-falsi* permitiría obtener mallados de mejor calidad a partir de ecuaciones implícitas, lo cual haría innecesario el uso de altas resoluciones para aproximar dichas ecuaciones de forma correcta.

CAPÍTULO 7. REFERENCIAS

[ANT98] H. Anton. "Calculus: A New Horizon, Combined". 6ta edición. John Wiley & Sons Inc. 1998.

[CAR95] B.P. Carneiro, C. Silva y A.E. Kaufman, "Tetra-Cubes: An algorithm to generate 3D isosurfaces based upon tetrahedra", *Anais do IX SIBGRAPI*, pp. 205-210, 1995.

[CHE95] E.V. Chernyaev, "Marching Cubes 33: Construction of Topologically Correct Isosurfaces", *Reporte Técnico CERN CN 95-17*, CERN, 1995.

[DÜR88] M. J. Dürst, "Letters: Additional reference to "marching cubes"", *Computer Graphics*, vol. 22, num. 2, pp. 72-73, 1988.

[EDE90] H. Edelsbrunner y E. Mucke, "Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms", *ACM Transactions on Graphics*, vol. 9, pp. 66-104, 1990.

[EDE94] H. Edelsbrunner y E. Mucke, "Three-Dimensional Alpha Shapes", *ACM Transactions on Graphics*, vol. 13, pp. 43-72, 1994.

[ELV92] T. T. Elvins, "A Survey of Algorithms for Volume Visualization", *Computer Graphics*, vol. 26, num. 3, pp. 194-201, 1992.

[FUC77] H. Fuchs, Z.M. Kedem y S.P. Uselton, "Optimal Surface Reconstruction from Planar Contours", *Communications of the ACM*, vol. 20, num. 10, pp. 693-702, 1977.

[FOR95] J.A. Ford, "Improved Algorithms of Illinois-type for the Numerical Solution of Nonlinear Equations", *Technical Report CSM-257, University of Essex*, 1995.

[FUJ95] I. Fujishiro, Y. Maeda y H. Sato, "Interval Volume: A Solid Fitting Technique for Volumetric Data Display and Analysis", *Proceedings of the 6th Conference on Visualization '95*, pp. 151-158, 1995.

[GRA72] R. L. Graham, "An efficient algorithm for determining the convex hull of a planar set", *Information Processing Letters*, pp. 132-133, 1972.

[GRA94] R. L. Graham, D. E. Knuth y O. Patashnik, "Concrete Mathematics: A Foundation for Computer Science", *Addison-Wesley Longman Publishing Co., Inc.*, Boston, MA, 1994.

[GUO95] B. Guo, "Interval Set: A Volume Rendering Technique Generalizing Isosurface Extraction", *Proceedings of the Conference on Visualization '95*, pp. 3-10, 1995.

[HER79] G.T. Herman y H.K. Liu, "Three-Dimensional display of Human Organs from Computed Tomograms", *Computer Graphics and Image Processing*, vol. 9, num. 1, pp. 1-21, 1979.

[KEP75] E. Keppel. "Approximating Complex Surfaces by Triangulation of Contour Lines", *IBM Journal of Research and Development*, vol. 19, num. 1, pp. 2-11, 1975.

[KRA04] M. Kraus, W. Qiao y D. S. Ebert, "Projecting Tetrahedra without Rendering Artifacts", *Proceedings of the Conference of Visualization '04*, pp. 27-34, Octubre 2004.

[LEV88] M. Levoy, "Display of Surfaces from Volume Data", *IEEE Computer Graphics and Applications*, vol. 8, num. 3, pp. 29-37, Julio 1988.

[LEV90a] M. Levoy, "Volume Rendering: A Hybrid Ray Tracer for Rendering Polygon and Volume Data", *IEEE Computer Graphics and Applications*, vol. 10, num. 2, pp. 33-40, Julio 1990.

[LEV90b] M. Levoy, "Efficient Ray Tracing from Volume Data", *ACM Transactions on Graphics*, vol. 9, num. 3, pp. 245-261, Julio 1990.

[LOR87] W.E. Lorensen y H.E. Cline, "Marching Cubes: A High-Resolution 3D Surface Construction Algorithm", *SIGGRAPH 87 Conference Proceedings, Computer Graphics*, vol. 21, num. 4, pp. 163-169, Julio 1987.

[MAX01] N. Max, "Consistent subdivision of convex polyhedra into tetrahedra", *Journal of Graphics Tools*, vol. 6, num. 3, pp. 29-36, 2001.

[MON94] C. Montani, R. Scateni y R. Scopigno. "A modified look-up table for implicit disambiguation of Marching Cubes.", *The Visual Computer*, vol. 10, 1994.

[NIE91] G.M. Nielson y B. Hamman, "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes", *Proceedings of the Conference of Visualization '91*, IEEE Computer Society Press, pp. 83-90, 1991.

[NIE97a] G.M. Nielson, "Tools for Triangulations and Tetrahedrizations", *Scientific Visualization: Overviews, Methodologies, and Techniques*, IEEE Computer Society Press, 1997.

[NIE97b] G.M. Nielson y Junwon Sung, "Interval Volume Tetrahedrization", *Proceedings of the Conference of Visualization '97*, pp. 221-228, 1997.

[SAW07] A.P. Sawant y C.G. Healey, "Visualizing Flow Data using Assorted Glyphs". *The ACM Student Journal, Crossroads*, vol. 14, num. 2, pp.3-7, 2007.

[SCH92] W.J. Schroeder, J.A. Zarge y W.E. Lorensen, "Decimation of triangle meshes". *SIGGRAPH 92 Conference Proceedings*, Computer Graphics, vol. 26, pp. 65-70, Julio 1992.

[SCH03] J. P. Schulze, M. Kraus, U. Lang y T. Ertl. "Integrating Pre-Integration into the Shear-Warp Algorithm". *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pp. 109-118, Julio 2003.

[STI07] C.J. Stimpson, C.D. Ernst, P. Knupp, P.P. Pébay y D. Thompson, "The Verdict Geometric Quality Library", *Sandia National Laboratories*, SAND2007-1751, 2007.

[WES89] L. Westover, "Interactive Volume Rendering", *Proceedings of the Chapel Hill Workshop on Volume Visualization*, vol. 1, pp. 9-16, Mayo 1989.

[WES90] L. Westover, "Footprint Evaluation for Volume Rendering", *Computer Graphics*, vol. 24, num. 4, pp. 367-376, 1990.

[WIL92] P. Williams y N. Max. "A volume density optical model". *Proceedings of the 1992 Workshop on Volume Visualization*, pp. 61-68, Octubre 1992.

[YON95] Yong Zhou, Weihai Chen y Zesheng Tang, "An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes", *Computer & Graphics*, pp. 355-364, 1995.