UNIVERSIDAD DE GRANADA

Representación y Visualización de Datos Volumétricos

MEMORIA QUE PRESENTA

Francisco Velasco Anguita

PARA OPTAR AL GRADO DE DOCTOR

Octubre de 2002

DIRECTOR

Juan Carlos Torres Cantero

Departamento de Lenguajes y Sistemas Informáticos

La memoria titulada "*Representación y Visualización de Datos Volumétricos*", que presenta Francisco Velasco Anguita para optar al grado de doctor ha sido realizada dentro del programa de doctorado "*Especificación y Desarrollo de Software*" del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada. Ha sido dirigida por el doctor D. Juan Carlos Torres Cantero, del Departamento de Lenguajes y Sistemas Informáticos.

Granada, Octubre de 2002

El Doctorando El Director

Fdo: Francisco Velasco Anguita Fdo: Juan Carlos Torres Cantero

Agradecimientos

Quiero dedicar esta memoria a mis padres, Francisco y Mª Luisa, a los que les debo lo que soy.

También quiero expresar mi gratitud a mi director, Juan Carlos, por todo el esfuerzo y tiempo que me ha dedicado.

A Pedro y Alejandro, por sus útiles comentarios.

A los mazmorreros ;-) Sergio, Germán, Óscar, Eugenio, José Manuel y Francisco, por los buenos momentos que hemos pasado juntos.

Y en general, a todos aquellos que han mostrado su interés y me han apoyado y ayudado en la realización de este trabajo.

Por último, decir que el trabajo mostrado en esta memoria ha sido parcialmente subvencionado por la Comisión Interministerial para la Ciencia y la Tecnología a través del proyecto TIC98-0973-C03-01.

a Rosario

Índice

Índice de figuras

Índice de tablas

Motivación

¿Pensará vuesa merced ahora
que es poco trabajo hacer un libro?

Miguel de Cervantes Saavedra (1547-1616)

Queremos iniciar esta memoria situando el contexto en el que tiene aplicación la línea de trabajo que hemos seguido así como realizando una serie de definiciones intuitivas sobre los principales conceptos que usaremos a lo largo de ella.

En este sentido la presente memoria versa sobre la representación de volúmenes para su tratamiento con la ayuda de un ordenador, la visualización de estas representaciones y la transmisión de las mismas a través de una red de ordenadores.

Definimos **volumen** u **objeto volumétrico** como aquel objeto posible en el mundo real que ocupa una porción acotada del espacio tridimensional. Una característica de los volúmenes al igual que de otros objetos es que en sus infinitos puntos podemos medir unos determinados valores pertenecientes a unos dominios de propiedad concretos. Interesándonos por consiguiente, tanto la porción de espacio tridimiensional ocupada como las propiedades que presenta.

Bajo esta definición, el cuerpo de un paciente para ser diagnosticado o una porción de terreno objeto de estudio para determinar su potencial minerológico son volúmenes. Nuestro objetivo es proporcionar al especialista una herramienta informática que le permita estudiar el volumen así como planificar y simular actuaciones sobre una representación del mismo almacenado en un ordenador; de modo que este trabajo, al realizarse sobre la representación del volumen en lugar de hacerlo sobre el volumen real, permite planificar y analizar distintas alternativas de actuación reduciendo tanto el coste económico como el riesgo sobre el volumen. Una vez determinada la mejor alternativa se podría llevar a cabo sobre el volumen real.

Para obtener una representación en un ordenador de un volumen y ante la imposibilidad de representar un conjunto infinito de datos necesitamos realizar una abstracción de las características más relevantes del mismo. A esta representación abstracta de las características más relevantes de un volumen la llamamos **modelo volumétri-**

co.

Como hemos comentado un volumen es un subconjunto del espacio tridimensional cuyos puntos presentan valores de un dominio de propiedad; un volumen así definido puede ser representado mediante una función que aplique puntos de un subconjunto V del espacio euclídeo tridimensional en un dominio de propiedades Γ, tal como:

$$f : V \subset \mathbb{R}^3 \rightarrow \Gamma$$

Sin embargo, cuando trabajamos con volúmenes como el ejemplo citado del cuerpo humano dicha función f no es conocida, teniendo de ella solamente el conjunto de muestras que el aparato de medida nos proporciona. Este conjunto de muestras junto con una estimación F de la función f hecha a partir de dichas muestras es lo que se suele usar como representación del volumen.

Hoy día, debido al aumento de la precisión de los aparatos de medida de volúmenes y al incremento de las prestaciones de los ordenadores, la cantidad de información con que se trabaja se hace cada vez mayor, con lo que se hace necesaria una estructuración de la información que facilite y acelere su tratamiento. En esta memoria proponemos un esquema de representación basado en la estructuración jerárquica de la información.

Una de las operaciones más realizadas es la de obtener una visualización del modelo volumétrico. De los distintos métodos existentes para visualizar modelos volumétricos, nosotros nos hemos centrado en los basados en extracción de isosuperficie consistentes en fijar un valor de propiedad γ_0, llamado valor *umbral*, y obtener la **isosuperficie** asociada a dicho valor umbral y visualizarla.

Una isosuperficie asociada a un valor umbral γ_0 es una superficie en la que todos sus puntos cumplen la condición $F(x,y,z) = \gamma_0$.

Se pueden visualizar varios valores de propiedad obteniendo varias isosuperficies y visualizándolas conjuntamente asignándoles algún atributo de transparencia y color.

Hemos elegido este tipo de métodos de visualización por su rapidez y su independencia a los cambios del punto de vista.

Otro aspecto que tratamos en esta memoria es el de la transmisión de volúmenes a través de red, ya que este medio de comunicación permite a un equipo de trabajo no estar físicamente en un mismo lugar, además de verse beneficiados de ventajas como

la compartición de recursos, la centralización de la información, el envío/recepción de información hacia/desde otros sistemas. Sin embargo, cuando la cantidad de información es grande, o el ancho de banda escaso, el tiempo que se emplea en la transmisión completa del modelo volumétrico puede ser largo. En este sentido, puede ser interesante el realizar una **transmisión progresiva** del modelo volumétrico, de modo que el receptor pueda trabajar con él desde las primeras etapas de la transmisión, sin necesidad de esperar a que esta concluya aunque dicho trabajo lo realizará en principio sobre un modelo de peor resolución que se irá refinando a medida que la transmisión vaya avanzando.

En definitiva, estas son las tres líneas principales que se tratan en esta memoria:

- Representación de volúmenes
- Visualización de los modelos volumétricos mediante extracción de isosuperficie
- Transmisión progresiva de los mismos

Resumen de aportaciones y estructura de la memoria

La felicidad no está en la ciencia,
sino en la adquisición de la ciencia
Edgar Allan Poe (1809-1849)

Mediante el trabajo llevado a cabo y que se plasma en la redacción de esta memoria de tesis se realizan las siguientes aportaciones:

- Se discribe un marco formal para la definición y estudio de modelos volumétricos. Se formaliza tanto el dominio de propiedades donde el modelo volumétrico toma valores, como el modelo volumétrico propiamente dicho. Los cuales se clasifican en discretos y continuos, obteniéndose como resultado un álgebra de boole de modelos volumétricos discretos. Todo ello queda redactado en el capítulo 2.

- Se propone un esquema de representación para modelos volumétricos, llamado *octree de celdas*, basado por un lado en una indexación que acelera el acceso a la información y por otro lado en una agrupación de la misma donde ésta cumple ciertos criterios de uniformidad. Este esquema se describe en el capítulo 3 donde además se particulariza para el caso de realizar la visualización del modelo volumétrico por métodos de extracción de isosuperficie.

- Para el octree de celdas propuesto se describe un modo de transmisión progresiva del modelo a través de la red. El método propuesto, además de las ventajas que aporta al usuario la transmisión progresiva como es la disposición de un modelo operativo desde el primer momento, tiene la ventaja de no necesitar transmitir información extra por el hecho de hacerse de manera progresiva. El capítulo 4 muestra los detalles de esta propuesta.

- Por último, en el capítulo 5, se propone un método para la visualización del modelo volumétrico basado en la extracción de isosuperficie que, además de solucionar de un modo más apropiado los problemas de rupturas en la isosuperficie que se plantean en el capítulo 3 para la visualización de un octree de

celdas, es más simple y robusto con respecto a los métodos existentes hasta el momento.

Todas las propuestas planteadas se acompañan de datos obtenidos experimentalmente a partir de volúmenes reales, en concreto volúmenes definidos a partir de la información proporcionada por una tomografía axial computerizada (*TAC*) a un cuerpo humano. También se ha experimentado con volúmenes definidos matemáticamente.

Completamos la memoria con una revisión del estado del arte en lo concerniente a la línea en la que se encuadra el trabajo presentado. Esta revisión se redacta en el capítulo 1. Y finalizamos la memoria con el capítulo 6 donde se resumen las conclusiones que se extraen de ella y se muestran las líneas que quedan abiertas tras este trabajo, en las cuales seguiremos trabajando en un futuro próximo.

Capítulo 1

Introducción

Yo no cito a otros más que para expresar mejor mi pensamiento

Michel d'Eychem, señor de Montaigne (1533-1592)

1.1. Modelado de objetos tridimensionales

Cuando un especialista tiene que realizar algún tipo de trabajo o estudio sobre un objeto tridimensional [1] , actuar directamente sobre el objeto 3D no suele ser la mejor opción, o incluso puede que aún no exista ese objeto 3D.

Por ejemplo un neurocirujano puede planificar una intervención ayudandose de unas imágenes procedentes de una *tomografía axial computerizada (TAC)* en vez de *abrir* directamente al paciente sin saber lo que se va a encontrar. O un ingeniero que tiene que construir un puente, primeramente realiza un diseño mediante una serie de dibujos y en el que realiza una serie de cálculos antes de pasar a la fase de construcción.

En estos ejemplos, los especialistas han realizado parte de su trabajo sobre una representación del objeto real; las imágenes TAC y los dibujos han servido para que el especialista observe o diseñe unas características relevantes de los objetos 3D reales que eran el objeto de su trabajo.

A esta representación de las características relevantes de un objeto es lo que se conoce como **modelo**.

Un modelo permite a un especialista diseñar, analizar, o planificar y realizar simulaciones sobre el modelo de igual modo que si lo realizara sobre el objeto real.

Cuando aplicamos el campo de la informática al modelado nos encontramos con una gran ayuda. Un ingeniero puede realizar un dibujo en perspectiva (modelo 2D)

[1] En lo sucesivo nos referiremos a *tridimensional* como 3D, y a *bidimensional* como 2D

de una pieza mecánica (objeto 3D) que esté diseñando, pero si dispone de un modelo 3D como puede ser una maqueta, podrá disponer de las vistas que desee desde cualquier punto de observación sin ningún trabajo extra por su parte; pero si en cambio el modelo 3D que dispone ha sido realizado por ordenador, además de obtener las vistas que desee podrá realizar cálculos de manera automática sobre el modelo, podrá realizar cambios sobre el mismo de una manera más cómoda o *ver* su interior de una forma más rápida.

Un **modelo por ordenador** no es más que una estructura de datos almacenada en la máquina junto con un software que permite al usuario interactuar con dicha información.

Cuando hablamos de modelos de objetos 3D, la componente geométrica del objeto cobra especial importancia en tanto en cuanto gran parte del trabajo que podemos realizar sobre el modelo de un objeto 3D es inherentemente geométrico. Cuando un modelo representa principalmente la información geométrica de un objeto decimos que se trata de un **modelo geométrico**.

Sin embargo, almacenar de un objeto exclusivamente información geométrica puede tener una utilidad muy restringida ya que cuestiones sobre si un punto está en el interior del objeto o no, no pueden contestarse de manera automática, ya que no se tiene el concepto de objeto como tal, sino de un conjunto de elementos geométricos.

En esta línea hablamos de **modelado de sólidos** cuando se presta especial atención a tener almacenado el objeto como tal, no solo un conjunto de información geométrica, de modo que se pueda responder cualquier tipo pregunta geométrica de un modo automático sin intervención del usuario.

En un sólido, se considera que su interior es homogéneo y por tanto no se representa, el modelo del sólido se limita a representar su geometría de la forma mencionada en el párrafo anterior. En cambio si el interior del objeto no es homogéneo, por ejemplo un cuerpo humano, el modelo de ordenador debe recoger esa información del interior. Es lo que se denomina **modelo de volumen**.

En última instancia el objetivo es obtener una representación de los objetos 3D que sea implementable en un ordenador y que recoja la características del objeto que sean de interés para el usuario que necesita de dicho modelo.

En este proceso Requicha establece una etapa intermedia consistente en obtener un modelo matemático de los objetos a partir de la cual se obtiene la representación última [Requicha, 80]. Véase la figura 1.1.

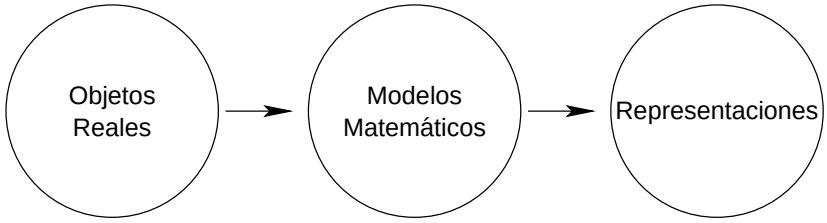


Figura 1.1: Modelado en 3 niveles

En esta línea disponemos de dos tipos de modelos matemáticos.

1.1.1. Modelos basados en topología puntual

De acuerdo con este enfoque, un sólido es un subconjunto S cerrado y acotado de puntos del espacio euclídeo R^3 [Mäntylä, 88].

Pero no cualquier subconjunto S de R^3 representa a un sólido, a S se le exigen otras condiciones [Requicha, 80]:

Rigidez: Es normal que un sólido no se deforme tras aplicarle una transformación rígida (traslación o rotación) por tanto S debe tener una configuración que permanezca invariante ante transformaciones rígidas. De hecho se puede establecer que todos los subconjuntos de puntos puedan obtenerse mediante transformaciónes rígidas de S representan al mismo sólido que S.

Regularidad: Los sólidos no poseen planos, líneas o puntos aislados. S debe ser regular, es decir, debe ser igual al conjunto resultante de calcular la clausura al interior de S. Un conjunto regular y acotado se denomina *r-set*.

Representación finita: Todo aquello que se desee almacenar en un ordenador debe tener una representación finita. Aunque S es un conjunto infinito de puntos debe poder ser representado de manera finita.

Este tipo de modelos se suelen implementar usando un buffer 3D como representación. El sólido se discretiza en una rejilla (ya sea regular o con otra estructura) para su manipulación y visualización. Todos los objetos se convierten a un meta-objeto (unidades de esa rejilla). Por tanto pueden almacenar información heterogénea del interior haciéndolos más apropiados para la representación de volúmenes. Entre sus características más relevantes nos encontramos [Kaufman, 94]:

- Insensibilidad a la complejidad de la escena una vez que los objetos se han convertido a la rejilla. Este preproceso puede verse influido por la complejidad de las escena, pero la manipulación y visualización de la misma no.
- Insensibilidad a la complejidad de los objetos, ya que todos los objetos se convierten a los meta-objetos.
- Insensibilidad a la textura (color, etc.), ya que se desarrolla sólo una vez durante el proceso de discretización quedando almacenada dicha información en el meta-objeto.
- Independencia del punto de vista, ya que los metaobjetos almacenan todos los atributos de visualización.
- La información proveniente de muestreo y simulaciones puede ser almacenada directamente en un buffer volumétrico.
- La información del interior se representa fácilmente.
- Sobre el buffer 3D se pueden hacer operaciones de bloques fácilmente así como realizar cambios de resolución en la rejilla de un modo jerárquico.

Aunque también presentan una serie de desventajas:

- Las necesidades de memoria son altas, sin embargo con los avances en memoria, tiende a no ser un problema.
- Debido a la gran cantidad de información que se maneja, se necesita una gran potencia de cálculo, que se ve aliviada con el desarrollo de hardware específico.

- La resolución finita supone una limitación para algunas operaciones tales como la medida del volumen, del mismo modo las transformaciones son difíciles sin perder información o calidad (por ejemplo las rotaciones que no sean múltiplos de 90^o).

- Una vez que el objeto ha sido discretizado se pierde su información geométrica, la cual es necesaria en algunas ocasiones (por ejemplo, para hacer medidas exactas). Una posible solución pasa por almacenar la información geométrica junto con el buffer de volumen.

1.1.2. Modelos basados en topología algebraica

La otra alternativa es representar el sólido mediante la superficie que lo delimita del exterior [Mäntylä, 88]. Este modelo matemático es posible en tanto en cuanto el interior de un sólido es homogéneo y no se almacena información sobre dicho interior.

Haciendo uso de esta alternativa se representa un objeto 3D mediante un modelo 2D. Pero no cualquier superficie va a representar a un sólido, debe cumplir con las condiciones de formar una piel completa y cerrada del objeto que no intersecte consigo misma delimitando el interior del exterior del sólido.

Una superficie que cumple estas condiciones, y por tanto representa a un sólido, se le denomina *frontera* del sólido. Formalmente se define como el espacio topológico *2-variedad*.

Los modelos basados en frontera suelen implementarse en el ordenador mediante un conjunto de primitivas geométricas, normalmente poliedros. Entre sus ventajas [Kaufman, 94] tenemos que el espacio que se requiere para su alamacenamiento suele ser poco, al mismo tiempo que realizar transformaciones de carácter geométrico es un proceso fácil y preciso, por otro lado la visualización se realiza de un modo fácil y rápido proyectando los poliedros a un dispositivo de salida 2D. En cambio, no dan la posibilidad de almacenar directamente información sobre su interior por lo que, en principio, no son adecuados para representar volúmenes.

1.2. Modelado de sólidos

Según estos modelos matemáticos, podemos establecer una serie de representaciones de sólidos sintácticamente correctas, entendiendo por representación sintácticamente correcta una estructura de símbolos construida a partir de un alfabeto cumpliendo una serie de reglas sintácticas. Al conjunto de todas las representaciones sintácticamente correctas lo llamamos *espacio de representación R*.

Del mismo modo tenemos el *espacio de modelado matemático M* cuyos elementos son modelos matemáticos de sólidos [2] .

Un *esquema de representación* es una correspondencia $s : D \rightarrow V$, llamando $D \subseteq M$ al dominio de s y $V \subseteq R$ al rango imagen de s (ver figura 1.2).

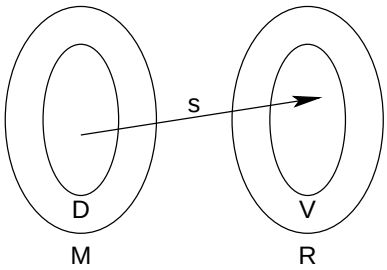


Figura 1.2: Esquema de representación

Por supuesto no hay que asumir que todos los sólidos abstractos sean representables (D no tiene por qué ser igual a M) ni todas las representaciones sintácticamente correctas se corresponderán con sólidos abstractos válidos (V no tiene por qué ser igual a R).

Cada esquema de representación lo podemos clasificar de acuerdo a las siguientes propiedades:

Dominio: Es el conjunto de sólidos abstractos representables mediante el esquema de representación. El caso ideal es que $D = M$.

[2] En esta sección llamaremos *sólidos abstractos* a los modelos matemáticos de sólidos.

Validez: Hace referencia al rango del esquema de representación. Indica qué representaciones sintácticamente correctas se corresponden con sólidos abstractos. El caso ideal es que $R = V$.

Unicidad: El esquema de representación es único si para todo sólido abstracto que es capaz de representar existe una única representación. Es decir, si la correspondencia es aplicación.

Ambigüedad: Es ambiguo si existen dos sólidos abstractos distintos con la misma representación. Interesa que no sea ambiguo, es decir, que la aplicación sea inyectiva. Cuando el esquema de representación no es ambiguo se dice que es completo en cuanto a información.

Estas dos últimas propiedades, unicidad y no ambigüedad son las más deseables, ya que hacen que el esquema de representación sea una aplicación biyectiva. Obviamente, interesa que el dominio sea lo más amplio posible.

Especialmente hay que evitar las representaciones ambiguas, ya que al poder corresponder a dos sólidos distintos pueden inducir a error. El usuario de un esquema de representación ambiguo resuelve la ambigüedad al saber lo que está haciendo, pero son totalmente inapropiados para transmitir o compartir información entre distintos usuarios.

Además de las anteriores, es conveniente tener en cuenta las siguientes cualidades de un esquema de representación:

Ser conciso: Que permita su almacenamiento en poco espacio y por tanto permitiendo una transmisión rápida.

Cerrado ante las operaciones: Que las manipulaciones y operaciones de representaciones de sólidos abstractos incluida la creación de los mismos den como resultado una representación de un sólido abstracto. Puede ser necesario el uso de herramientas que garanticen la validez.

Eficiente: Midiendo la eficiencia como la variedad y tipo de aplicaciones que podemos realizar sobre las representaciones, así como la complejidad computacional de las mismas.

Pasemos ahora a dar unas breves reseñas sobre los esquemas de representación más usuales.

1.2.1. Instanciación de primitivas

Consisten en aplicar una serie de transformaciones geométricas tales como traslaciones, rotaciones, etc. a una serie de primitivas para formar el sólido, dichas primitivas pueden ser paramétricas. No son jerárquicos, por lo que no se pueden usar componentes ya creados por instanciación para instanciarlos de nuevo en un sólido mayor.

Este esquema es inambiguo, único, fácil de validar, conciso y fácil de usar. Como desventaja presenta, además del reducido dominio, la dificultad para escribir algoritmos que calculen propiedades de los sólidos representados.

1.2.2. Enumeración de ocupación espacial

Consiste en dividir el espacio en celdas e indicar qué celdas contienen solido (etiquetadas como *negras*) y qué celdas no (etiquetadas como *blancas*) [Shirari, 81]. Un caso particular es cuando las celdas son cubos de igual tamaño resultando una descomposición regular. En este caso la implementación es fácil mediante un array en cuyas posiciones se puede almacenar información acerca de las características y propiedades que presenta el volumen en dichas celdas. Cuando la información que almacena una celda es un único valor para toda la celda la llamamos *vóxel*, llamándole celda cuando almacena varios valores, normalmente en los extremos de la misma, que nos permiten calcular o estimar el valor de propiedad de cualquier punto interior de la misma.

Este esquema de representación está especialmente indicado para volúmenes ya que podemos almacenar información sobre el interior de los mismos, aunque la superficie de los mismos no se modela de forma exacta. Además permite realizar de manera muy eficiente operaciones booleanas y calcular propiedades volumétricas. Debido al ordenamiento espacial implícito es muy apropiada para rendering. Sin embargo, las transformaciones geométricas resultan complejas.

Se trata de un método inambiguo, único y fácil de validar pero es aproximado y maneja mucha información, por lo que se suele usar otra representación para dar la entrada y después transformar.

1.2.3. Subdivisión espacial

Es un avance sobre la enumeración que reduce el espacio de almacenamiento. Consiste en realizar la división del espacio de manera recursiva obteniendo estructuras jerárquicas como las siguientes:

Bintrees: Esta representación parte de un cubo inicial (que simboliza el universo) y lo va dividiendo en dos mitades iguales sucesivamente siguiendo los ejes x, y, z de forma cíclica. Se marca como negro la celda que contiene sólido al completo, como blanco la celda que no contiene ninguna fracción de solido y el resto se subdividen [Samet, 85].

BSP: (Partición binaria del espacio) Esta representación es similar a la anterior salvo que en esta ocasión la partición se hace dependiendo del sólido, con el objeto de reducir el número de particiones [Thibault, 87]. La idea es que los planos de división vayan coincidiendo con los planos del sólido.

Octrees: Se parte de un cubo inicial (el universo) y se va subdividiendo recursivamente en ocho subcubos iguales entre sí atendiendo a criterios de pertenencia [Meagher, 80][Samet, 90b][Samet, 90a][Velasco, 96].
A raíz de este esquema han surgido otras variantes de los octrees:

Face octrees: Sus nodos hoja, además de los clásicos blancos y negros, pueden contener un plano, sin aristas ni vértices [Brunet, 91].

Face-and-edge octrees: Además de lo anterior, un nodo puede contener una arista [Navazo, 89].

Extended-octrees: Además de lo anterior, un nodo puede contener un vértice, de modo que pueden modelar de manera exacta la frontera de los poliedros [Brunet, 85]. En [Navazo, 86] se realiza una generalización a superficies libres mediante parches bicuadráticos.

Nosotros hemos realizado una propuesta de octree en el que la caja englobante inicial no es un cubo y el sistema de coordenadas usado para las subdivisiones no es el sistema de coordenadas cartesiano [Torres, 95].

Así mismo, basados en la versión 2D de los octrees, los quadtrees [Samet, 90b][Samet, 90a], hemos realizado la propuesta de quadtrees basados en sistemas de coordenadas no cartesianos, así como el uso conjunto de varios quadtrees en distintos sistemas de coordenadas para modelado bidimensional [Cano, 96].

1.2.4. Descomposición de celdas

Representa al sólido descomponiéndolo en celdas que se adaptan a él, las cuales deben describirse.

Es inambiguo pero no único. Además, comprobar la validez de la frontera es costoso, no es conciso ni fácil de crear. Aunque es una representación simple y conveniente para calcular ciertas propiedades topológicas.

1.2.5. Geometría constructiva de sólidos (CSG)

Consiste en construir el sólido mediante operaciones booleanas regularizadas y transformaciones a partir de primitivas tales como paralelepípedos, esferas, cilindros, etc. [Requicha, 82].

Se suelen representar mediante árboles donde se reflejan la estructura jerárquica de las operaciones. En las hojas se encuentran las primitivas que pueden representarse bien mediante subárboles en los que los nodos hoja son semiespacios o bien mediante otro tipo de representación.

Los semiespacios se describen por desigualdades de la forma $f(x,y,z) \geq 0$. Sustituyendo este subárbol en cada ocurrencia de la primitiva en el árbol original CSG obtendremos un árbol en el que los nodos hoja son semiespacios.

Son esquemas inambiguos pero no únicos, en donde el dominio depende de las primitivas que dispongamos y todos los objetos que pueden modelarse son válidos; son muy concisos y difíciles de visualizar al tener que evaluar la frontera. Sin embargo, el diseño de sólidos mediante este esquema de representación es muy intuitivo y la representación es bastante compacta.

1.2.6. Representaciones de barrido

Se basan en desplazar una superficie a través de una trayectoria. Se clasifican en distintos tipos según la forma de la superficie y de la trayectoria.

Son representaciones inambiguas pero no únicas y su dominio está limitado a objetos con simetría traslacional o rotacional. Como ventaja señalamos que es simple para la entrada de modelos y la representación es compacta. Su principal desventaja

se encuentra en la falta de algoritmos para calcular propiedades de los sólidos representados; además de que el dominio de las formas a representar es muy pequeño.

1.2.7. Representaciones mediante fronteras

El sólido se representa segmentando su frontera en un número finito de caras las cuales están representadas mediante sus aristas y vértices [Baer, 79]. Se le suele exigir a la frontera que sea un espacio topológico 2-variedad, es decir, que las superficies cumplan con las siguientes condiciones:

Cerradas: Cada arista lo es de dos caras.

Orientable: Cada cara tiene dos lados distinguibles —exterior e interior—. Esto se consigue dando las aristas en un orden determinado. Cada arista lo es de 2 caras, ocurrirá que cada arista se recorrerá en sus dos sentidos dependiendo de la cara que consideremos.

Que no intersecten consigo mismas: Como ocurre en el ejemplo de la botella de Klein (ver figura 1.3).

Limitadas: Una superficie que cumpla las 3 condiciones anteriores será limitada si divide el espacio en 2 subespacios uno de ellos finito.

Conectadas: Que cada objeto tenga una sola superficie.

En la representación suele haber información redundante como puede ser la normal de la cara, o información acerca de las relaciones de adyacencia con otras caras, lo que reduce el costo de las operaciones.

En definitiva especifican propiedades topológicas y geométricas del sólido, las cuales no son independientes.

Las caras planas, se representan mediante las ecuaciones de los planos que las contienen; pero las caras curvas se suelen aproximar mediante caras planas para no perder la generalidad y la velocidad de los métodos que operan sobre esta representación. Usando técnicas de sombreado como Gouraud [Gouraud, 71] y Phong [Bui-Tuong, 75] podemos visualizarlo sin notar las aristas.

Este proceso de dividir un cuerpo curvo en caras se conoce con el nombre de *lofting* [Baer, 79].

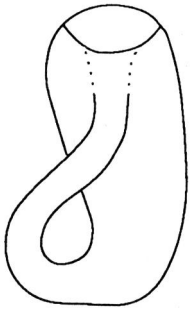


Figura 1.3: Botella de Klein

Es un esquema de representación inambiguo, pero requiere mucha información, la cual suele estar organizada de forma jerárquica: el sólido está compuesto por una serie de caras, una cara una serie de aristas, y cada arista una pareja de vértices. Además cada elemento puede pertenecer a varios elementos de nivel superior, con lo que es necesario usar una herramienta para la creación. Es muy apropiada para ciertas aplicaciones, por ejemplo para la visualización, debido a que la información que necesitan está explícitamente presente en la representación y a la cantidad de algoritmos existentes que usan dicha información.

Los modelos basados en este esquema de representación suelen construirse paso a paso haciendo uso de alguna herramienta auxiliar debido a que la cantidad de información que poseen y las relaciones existentes entre ésta lo hacen poco apropiado para una introducción a mano. Aunque las modificaciones locales son fáciles de realizar.

1.2.8. Esquemas híbridos

Se trata de usar conjuntamente varios esquemas de representación de cara a aprovechar las mejores cualidades de cada uno. En este sentido surgen dos aspectos, la conversión entre representaciones, que puede ser con o sin pérdidas; y la consistencia entre las distintas representaciones simultáneas.

1.3. Modelado y visualización de volúmenes

Cuando lo que se quiere representar es la información heterogénea del interior del objeto, ya hemos visto que el modelo matemático a usar es el basado en topología puntual. Por lo tanto el volumen va a estar representado por un conjunto de puntos, con las características que se le exigen [3] , donde cada punto va a presentar valores de unos determinados dominios de propiedad [Cano, 98, Cano, 99].

$$f : V \subset \mathbb{R}^3 \rightarrow \Gamma \tag{1.1}$$

La función f suele ser desconocida con lo que sólo tenemos el conjunto finito de muestras que nos proporciona el aparato de medida, quedando el volumen representado como una rejilla de puntos

$$\{(x_i, y_i, z_i, f_i) : f_i = f(x_i, y_i, z_i), i = 1, 2, \cdots, N\} \tag{1.2}$$

Según la distribución de los puntos la rejilla de puntos puede clasificarse en:

No estructurada: Los puntos no siguen ninguna distribución geométrica concreta. Usual en las simulaciones númericas.

Estructurada: Que se puede dividir en:

Curvilínea: Típica en las simulaciones numéricas.

Rectilínea: Común en aplicaciones médicas. Se divide en:

Irregular: Las muestras no son equidistantes.

Regular: Las muestras son equidistantes.

Isotropicas: El espaciado es igual en las tres direcciones.

Anisotropicas: El espaciado no es igual en las tres direcciones.

Una vez representado el volumen la operación más usual que se realiza con él es la visualización del mismo, obtener en un dispositivo 2D la información 3D representada. El proceso consta de 3 etapas [Haber, 90]:

[3] Véase la sección 1.1.1 (pág. 15) .

Enriquecimiento de datos: Se construye una función $F(x,y,z)$ que estima a la función $f(x,y,z)$ siendo $F(x_i,y_i,z_i) = f(x_i,y_i,z_i)$ para todas las muestras conocidas. Para las rejillas rectilíneas la interpolación trilineal es la opción más usual por el buen compromiso que presenta entre rapidez y exactitud. En cada celda se estima el interior con

$$F(x,y,z) = a + bx + cy + dz + eyz + gxz + hxy + ixyz \tag{1.3}$$

donde

$$\begin{aligned} a &= f(v_0) \\ b &= f(v_1) - f(v_0) \\ c &= f(v_2) - f(v_0) \\ d &= f(v_4) - f(v_0) \\ e &= f(v_6) - f(v_4) - f(v_2) + f(v_0) \\ g &= f(v_5) - f(v_4) - f(v_1) + f(v_0) \\ h &= f(v_3) - f(v_2) - f(v_1) + f(v_0) \\ i &= f(v_7) - f(v_6) - f(v_5) - f(v_3) + \\ &\quad f(v_4) + f(v_2) + f(v_1) - f(v_0) \end{aligned} \tag{1.4}$$

Véase la figura 1.4. En las rejillas no estructuradas, se consideran celdas tetraé-

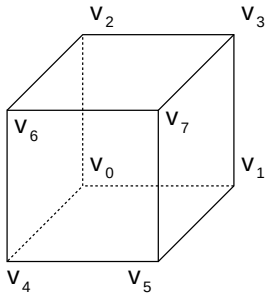


Figura 1.4: Celda con los vértices nombrados

dricas usando como interpolación lineal la ecuación

$$F(x,y,z) = a + bx + cy + dz \tag{1.5}$$

Representación: Se elige una interpretación geométrica de la función $F(x,y,z)$ que ayude a entender su comportamiento. Suelen ser objetos 3D visualizables en un dispositivo 2D. Existen 3 formas de hacerlo:

Extracción de *rebanadas*: Se trata de obtener una serie de imágenes 2D, paralelas entre sí, que pueden ser tratadas con los filtros habitualmente usados en tratamiento de imágenes para ayudar a su visionado e interpretación. El objeto a visualizar es la parte del volumen definida por

$$F(x,y,z) : (x,y,z) \in P(x,y,z) = 0 \tag{1.6}$$

donde $P(x,y,z) = 0$ es un plano.
Para visualizar el volumen completo habría que mostrar varias rebanadas a lo largo de un eje, lo cual podría hacerse sobre el mismo viewport en un período de tiempo.

Extracción de isosuperficies: En esta ocasión también se muestra solamente una porción del modelo volumétrico, en concreto, se establece un valor umbral $\gamma_0 \in \Gamma$ y se visualiza la superficie que cumple la condición

$$F(x,y,z) = \gamma_0 \tag{1.7}$$

normalmente aproximada por una malla de triángulos. Para visualizar el volumen completo habría que realizar varias visualizaciones en el tiempo variando el valor umbral a lo largo del intervalo temporal.

Visualización directa: Es la aproximación que visualiza el volumen completo en una sola imagen. Consiste en obtener un modelo 3D de los datos que combine atributos como color y opacidad de modo que el modelo es visualizado como si el volumen estuviera formado por geles translúcidos.

Visualizado: La geometría obtenida en la fase anterior es convertida en imagen usando las técnicas de informática gráfica.

Pasemos a describir un poco más las dos técnicas más usadas: la visualización directa y la extracción de isosuperficies. No obstante, también hay trabajos que combinan las dos estrategias en lo que se ha dado en llamar *modelado híbrido* [Boada, 01].

1.3.1. Visualización directa

Como se ha comentado, la visualización directa se basa en obtener a partir de los datos un *modelo de gel translúcido*. Por tanto la primera etapa consistirá en asignar atributos de color y opacidad a los datos, es lo que se conoce como *clasificación*. La clasificación se compone de dos funciones:

Opacidad: Tiene en cuenta además del valor del dato otro tipo de información como el gradiente, para hacer más opacas las zonas donde el gradiente es mayor y así resaltar la frontera entre dichas zonas.

Color: Que puede tener en cuenta información procedente de una segmentación previa donde se ha etiquetado cada dato como perteneciente a una zona concreta del volumen.

Una vez fijado el color y la opacidad de los datos, se obtiene la imagen lanzando rayos que pasan por el observador y por lo píxeles de la imagen calculando para cada rayo el color a visualizar en dicho píxel. Para ello se usa la siguiente ecuación:

$$I_\lambda = \int_0^L C_\lambda(s)\mu(s)e^{-\int_0^s \mu(t)dt}ds \tag{1.8}$$

donde I_λ indica la intesidad de luz que recibe el píxel para la longitud de onda λ, L es la longitud del rayo, $C_\lambda(s)$ es la luz reflejada por el modelo volumétrico en la posición s en la dirección del rayo (se puede calcular mediante modelos de reflexión estandar), el peso $\mu(s)$ indica la densidad en la posición s. Se van acumulando las intensidades de los puntos por los que va pasando el rayo pero cada intensidad en cada punto se ve atenuada por la densidad que ya ha sido recorrida por el rayo, esta atenuación está representada en la integral de la exponencial.

Esta ecuación debe resolverse numéricamente, tras una serie de sustituciones [Brodlie, 01] obtenemos la siguiente ecuación:

$$I_\lambda = \sum_{i=0}^{n} C_\lambda(i)\alpha(i)\prod_{j=0}^{i-1}(1-\alpha(j)) \tag{1.9}$$

donde $\alpha(p)$ representa la opacidad del punto p.

Para obtener la imagen hay dos modos de actuación: el que se conoce como *ordenado por la imagen* realiza el procesamiento desde el plano de la imagen hasta

el objeto, y el *ordenado por el objeto* donde el procesamiento se realiza en orden inverso; aunque los métodos híbridos toman las ventajas de ambos.

El método clásico **ordenado por la imagen** es el ray-casting [Levoy, 88] que se corresponde con lo comentado hasta ahora. Recientemente han habido una serie de mejoras, algunas encaminadas a mejorar la exactitud del cálculo de la integral 1.8 [Novins, 92][Jung, 98] o encaminadas a aumentar la rápidez de cálculo usando como intensidad de un pixel la del punto de mayor intensidad del rayo [Mroz, 00].

Otras líneas de trabajo sobre el modo de interpolar los puntos interiores a las celdas se dirigen a interpolar antes de clasificar en vez de a la inversa además de interpolar el color usando una ecuación ponderada con la opacidad [Gasparakis, 99].

También ha habido avances orientados al procesamiento de las rejillas curvilíneas y no estructuradas [Giertsen, 92], y orientadas a optimizar el recorrido del rayo a través del objeto [Yagel, 92][Levoy, 90]. Y por supuesto, también se ha trabajado en el campo del hardware donde se ha conseguido visualizar volúmenes en tiempo real sobre plataforma PC.

El método clásico **ordenado por el objeto** es el conocido como *splatting* [Westover, 90] mediante el cual se proyectan las celdas sobre el plano de la imagen obteniendo los llamados *splats* que se combinan para formar la imagen final.

Los avances han ido orientados a la mejora del proceso de obtención de los splats [Mueller, 99], a obtener un método híbrido como el visualizado *shear warp* [Lacroute, 94], a aplicar este método a rejillas no estructuradas [Williams, 98], orientados al hardware mediante las texturas 3D, o a conseguir visualización interactiva [Csébfalvi, 01][Pyo, 02].

1.3.2. Extracción de isosuperficie

El método clásico de visualización de volúmenes por extracción de isosuperficie es el conocido como **Marching Cubes** [Lorensen, 87]. Este método asume que el volumen está representado en una rejilla estructurada, rectilínea, regular e isotrópica; donde cada celda cúbica se puede procesar de forma independiente. El método se basa en construir para cada celda cruzada por la isosuperfice, el trozo de ésta que está en el interior de la celda, de modo que la unión de todos los trozos de isosuperficie construidos celda a celda conformarán la isosuperficie total a visualizar.

El modo que proponen para construir la isosuperficie en el interior de una celda se basa en tener unas configuraciones de celda predefinidas a las que corresponde una geometría de isosuperficie también predefinida.

Una vez fijado el valor umbral γ_0 cada vértice de la celda se puede etiquetar como *vértice positivo* si el valor de propiedad en dicho vértice es mayor o igual que el valor umbral o como *vértice negativo* si dicho valor es menor que el umbral, con lo que cada celda puede tener $2^8 = 256$ configuraciones de etiquetas distintas. Sin embargo, si se consideran equivalentes dos celdas que tengan intercambiadas las etiquetas en todos los vértices, el número de configuraciones distintas se reduce a 128; y si consideramos equivalentes a una configuración todas aquellas a las que pueda llegarse mediante rotaciones de la primera, el número de configuraciones distintas se reduce a 15.

En cada arista con etiquetas opuestas en sus extremos se puede calcular mediante interpolación lineal la posición que le corresponde al valor umbral. Los puntos de las aristas así calculados son los vértices de una malla de triángulos que representan la isosuperfice en el interior de la celda. La isosuperficie interior a cada celda se formará como la de la clase de equivalencia a la que pertenezca.

Las 15 configuraciones con sus correspondientes triangulaciones internas son las que se muestran en la figura 1.5.

1.3.2.1. Ambigüedades del marching cubes

Sin embargo, hay configuraciones que en caso de presentarse de manera adyacente construyen una isosuperficie con agujeros; en concreto se dan en celdas con caras ambiguas.

Se dice que una cara es ambigua si los vértices de una diagonal están etiquetados como positivos y los de la otra diagonal como negativos. En esta situación hay dos modos de hacer la isosuperficie, ya que la isosuperficie puede cortar doblemente la cara siguiendo la dirección de una diagonal o la de la otra.

Por ejemplo, al triangular las dos celdas del caso (a) de la figura 1.6 según el modo de triangulación de Lorensen mostrado en la figura 1.5, obtenemos un agujero en la isosuperficie.

Montani propone triangulaciones alternativas para aquellos casos donde se producen agujeros con las triangulaciones originales [Montani, 94b]; la idea es elegir una configuración o la alternativa con el objetivo de que la diagonal que une dos vértices

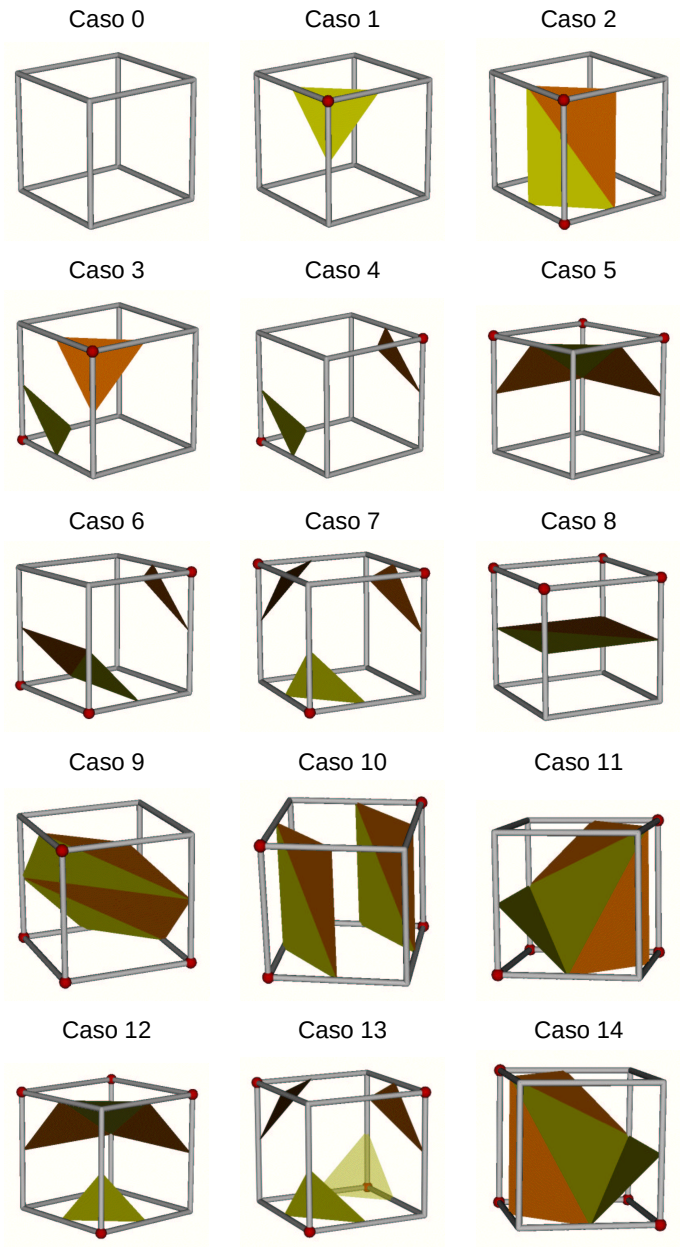
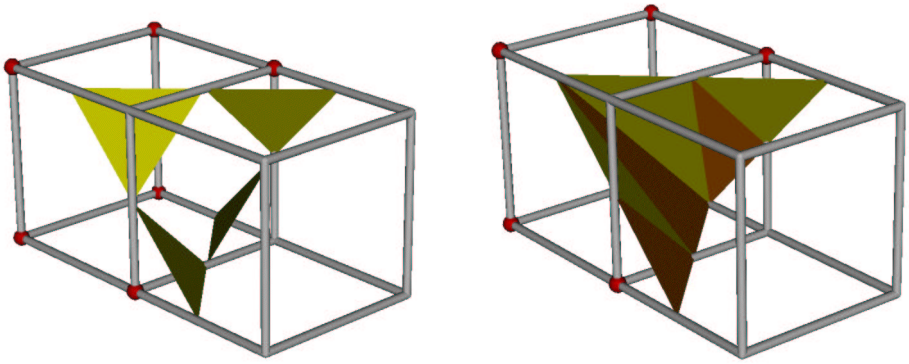


Figura 1.5: Configuraciones del método Marching Cubes



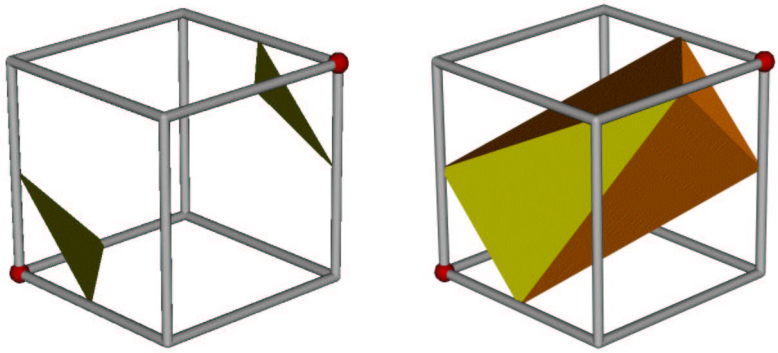
(a) (b)

Figura 1.6: Ejemplo de celda con cara ambigua

del mismo signo y que mantiene dicho signo a lo largo de su recorrido no atraviese la isosuperficie. El caso (b) de la figura 1.6 nos muestra la triangulación que no produce el agujero del caso (a).

Las ambigüedades no quedan reducidas al espacio de las caras de las celdas, en el interior también se producen situaciones ambiguas [Natarajan, 94] como puede verse en las dos triangulaciones posibles para una misma celda en la figura 1.7. El caso (a) realiza la triangulación obteniendo dos componentes de isosuperficie y el caso (b) realiza la triangulación obteniendo una sola componente con forma tubular.

Chernyaev identifica 33 configuraciones distintas teniendo en cuenta todas las ambigüedades posibles [Chernyaev, 95] mientras que Cignoni identifica 88 [Cignoni, 00]. Recientemente Lopes, tras estudiar ambos trabajos, indica que varios casos de los 88 que propone Cignoni son topológicamente equivalentes, reconociendo los 33 casos propuestos por Chernyaev, aunque indica que hay dos casos que se pueden obtener por reflexión a partir de otros dos, con lo que el número de configuraciones distintas para una celda se reduce a 31 [Lopes, 02]. No obstante, los trozos de isosuperficie interiores a la celda para esos 31 casos los caracteriza como pertenecientes a 14 tipos distintos atendiendo a que el trozo de isosuperficie sea homeomórfico a un disco, a un cilindro y teniendo en cuenta el número de aristas que corta y el número de cortes que le hace a una misma cara de la celda.



(a) (b)

Figura 1.7: Ejemplo de celda con interior ambiguo

1.3.2.2. Adecuación a la isosuperficie matemática

Otra línea de trabajo surgida a raíz del trabajo de Lorensen ha ido encaminada a mejorar la adecuación de la malla de triángulos a la trilineal definida por la función estimada $F(x,y,z)$. Básicamente, los distintos métodos propuestos consisten en dividir un triángulo en varios triángulos tomando vértices de la superficie trilineal para realizar el refinamiento. Los distintos trabajos en esta línea [Lopes, 99][Cignoni, 00][Bailey, 00] difieren en el modo de calcular los vértices o decidir si procede el refinamiento.

Tambien hay trabajos como [Parker, 99] en los que se obtiene la verdadera superficie, sin construir una malla de triángulos intermedia, visualizando la función $F(x,y,z)$ mediante ray casting.

1.3.2.3. Búsqueda de celdas activas

A las celdas que contienen isosuperficie en su interior se les denomina *celdas activas*. Uno de los problemas del marching cubes radica en que debido a que la proporción de celdas activas es baja con respecto al total de celdas, una gran parte del tiempo de visualización se emplea en clasificar celdas vacías.

De cara a localizar las celdas activas se han propuesto varios métodos que se pueden clasificar en función de en qué espacio se realiza la búsqueda.

Búsqueda basada en el dominio del espacio

Se trata de dividir el volumen en subvolúmenes de una manera jerárquica mediante un octree [Meagher, 80]. Cada nodo almacena el valor de propiedad mínimo y máximo del subvolumen que representa, de modo que al buscar las celdas activas con el valor umbral fijado se pueden descartar aquellas ramas que no van a conducir a celdas que generan isosuperficie.

Wilhelms propone realizar la subdivisión colocando los planos de división del octree a una distancia *potencia de* 2 del origen del subvolumen a dividir en vez de hacerlo por la mitad. De este modo, cuando las dimensiones del volumen no son iguales entre sí, o son distintas de una potencia de 2 el número de nodos que resulta en el árbol es menor; esta estructura propuesta se conoce como *Branch-On-Need Octree* (BONO) [Wilhelms, 92].

Búsqueda basada en el dominio de las propiedades

En este caso la busqueda se realiza usando una estructura de datos intermedia donde quedan representadas las celdas en base al rango de valores de propiedad que la convierten en activa. Usando el valor umbral como clave de busqueda se accede a través de la estructura de datos intermedia a las celdas activas para dicho umbral.

Por ejemplo, cada celda puede representarse mediante un punto 2D $(\gamma_{min}, \gamma_{max})$ formando lo que se conoce como *espacio spam*. Una vez fijado el valor umbral γ_0 todos los puntos que se encuentren en la parte superior izquierda del punto (γ_0, γ_0) son los que se corresponden a las celdas activas. Puede verse una representación geométrica en el diagrama (a) de la figura 1.8.

Para la busqueda de las celdas activas Livnat propone usar un Kd-tree con $K = 2$ [Livnat, 96]. Shen propone dividir el espacio spam en celdillas [Shen, 96] (ver el diagrama (b) de la figura 1.8), de modo que una vez localizada la celdilla que contiene el punto (γ_0, γ_0), podemos clasificar las celdillas según los 4 tipos mostrados en la figura 1.8(b). Las celdillas tipo 3 y 4 contienen claramente celdas activas y no activas respectivamente, las celdillas tipo 2 contiene celdas activas que pueden buscarse con un arbol binario unidimensional y sólo la celdilla tipo 1 es la que contiene celdas activas que deben buscarse mediante un Kd-tree bidimensional.

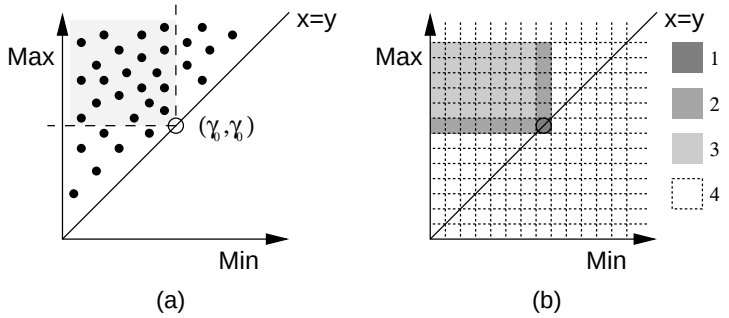


Figura 1.8: Representación geométrica del espacio spam

Una alternativa al espacio spam es el espacio de intervalos [Cignoni, 97]. Cada celda queda representada por el intervalo de los valores de propiedad que la convierten en activa. Estos intervalos se organizan en un árbol binario de intervalos, donde cada nodo tiene almacenados los intervalos que contienen un determinado valor representativo del nodo, en la rama de la izquierda el subárbol de los intervalos que son menores que dicho valor y en la rama de la derecha el subárbol de los intervalos mayores.

Busqueda basada en el espacio de la superficie

Cuando lo que se tiene una rejilla estructurada, a partir de una celda activa podemos acceder a otras celdas activas a traves de las caras y aristas que han sido atravesadas por la isosuperficie. Por tanto, una vez localizada una celda activa podríamos obtener la isosuperficie completa si ésta tuviera una sola componente.

Para localizar todas las componentes de una isosuperficie mediante este método de acceso a celdas activas es necesario un conjunto de celdas semilla que lo garantice [Bajaj, 96]. Una vez definido el conjunto de celdas semilla, se construye para ellas el Kd-tree o el árbol de intervalos para localizar las celdas semilla activas.

1.3.2.4. Reducción del número de triángulos

Otra línea de trabajo es la encaminada a reducir el número de triángulos que genera marching cubes en aras de aumentar la rapidez de proyección de los mismos

ante los cambios de punto de vista por parte del usuario en un sistema interactivo.

Existen métodos que operan sobre la malla de triángulos una vez completa con independencia de las celdas, es lo que se conoce como *decimation* [Schroeder, 92].

Este proceso se ve favorecido si la isosuperficie se ha obtenido mediante el *marching cubes discreto* [Montani, 94a]. Consiste en aplicar marching cubes pero los vértices de los triángulos se sitúan siempre en el centro de las aristas. De esta forma la probabilidad de obtener zonas formadas por varios triángulos coplanarios aumenta pudiendose retriangular sin modificar la geometría de la isosuperficie obtenida.

Otros métodos se basan en la idea de agrupar celdas en la rejilla con lo que reduciendo el número de celdas se reduce el número de triángulos. En este sentido Shu proponen un marching cubes adaptativo [Shu, 95], que consiste en aplicar marching cubes a celdas de tamaño 2^m (frente a marching cubes que usa celdas de tamaño 1) y en aquellos sitios en donde el ángulo que forman las normales de dos triangulos que comparten una arista es mayor que un valor dado (lo fijan en 30^o) se subdivide la celda en 8 celdas y se vuelve a aplicar el proceso.

Shekhar propone ir agrupando celdas a modo de octree siempre que la triangulación en el interior de las 8 celdas candidatas a ser agrupadas cumpla unas determinadas condiciones [Shekhar, 96].

Al hacer estas agrupaciones o subdivisiones de un modo no homogéneo obtenemos una rejilla multirresolución que provoca rupturas en la isosuperficie en las fronteras de cambio de resolución como la mostrada en la figura 1.9.

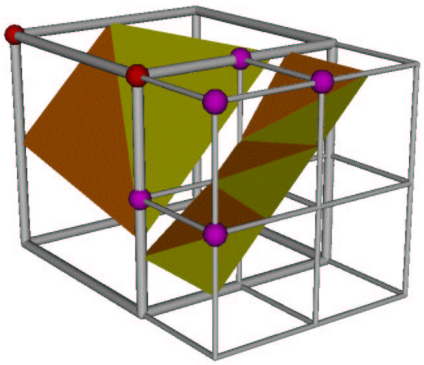


Figura 1.9: Ruptura debido a la multirresolución

Para tapar dichos agujeros, Shu propone cerrarlo con un polígono mientras que Shekhar propone desplazar los vértices de los triángulos de las celdas pequeñas hasta hacerlos coincidir con la isosuperficie en la celda mayor. Westermann propone sustituir los triángulos en la celda mayor por un *fan* de triángulos para que se adapte a los triángulos de las celdas menores [Westermann, 99].

Otra alternativa de rejilla multirresolución es la propuesta por Poston, quien agrupa celdas obteniendo nuevas celdas no necesariamente cúbicas que se adaptan a la isosuperficie [Poston, 98]. El modo de tapar los agujeros es realizar un marching 2D en las caras de la celda grande a la resolución de las celdas vecinas pequeñas. La polilínea cerrada que se obtiene es lo que se triangula.

1.3.2.5. Isosuperficie a partir de contornos

Por último, otro modo de obtener la isosuperficie, sobre todo cuando se parte de imágenes en rebanadas del volumen, es detectar los contornos en las rebanadas para el valor umbral fijado, y construir la isosuperficie conectando los contornos [Jones, 94].

1.4. Conclusiones

Se ha realizado una exposición del estado del arte en lo concerniente al modelado de objetos tridimensionales, modelado de sólidos y modelado y visualización de volúmenes; centrándonos más en este último campo. Se han comentado los trabajos más relevantes realizados por distintos autores y los últimos avances en visualización de volúmenes según diferentes enfoques.

Capítulo 2

Formalización

La tarea no es observar al individuo, sino a la especie:
observar las propiedades generales en conjunto
Joseph Addison (1672-1719)

En este capítulo vamos a describir un marco formal en el que poder estudiar los modelos de volúmenes. Del modelo mátemático que hemos visto en el capítulo anterior que representa a un volumen

$$f : V \subset \mathbb{R}^3 \rightarrow \Gamma \tag{2.1}$$

vamos a caracterizar primeramente el dominio de propiedades Γ para después estudiar y caracterizar la propia función f.

2.1. Trabajos previos

Antes de abordar la descripción de un marco formal para los volúmenes vamos a comentar los trabajos de Mallgren [Mallgren, 82], Fiume [Fiume, 89] y Torres [Torres, 93] sobre modelos matemáticos para sistemas gráficos por ordenador.

Un modelo matemático proporciona una semántica precisa para los sistemas y por tanto puede usarse para probar sus propiedades con independencia de su implementación.

La propuesta de Mallgren es 2D y los conceptos usados no son fácilmente extensibles a 3D.

Fiume formaliza el proceso de visualización, para ello define un objeto estático como

$$(Z_0, I_0) \text{ donde } Z_0 \subseteq R^3, I_0 : Z_0 \rightarrow C$$

donde C es un espacio de color, definido como un subconjunto de R^c con $c \geq 1$. Un objeto puede definirse como una primitiva dando su Z_0 e I_0 o como combinación de otros objetos.

Fiume usa estos conceptos para formalizar el proceso de visualización. Sus principales defectos es que las funciones de combinacion no son operadores booleanos y por tanto $A \cup (A \cap B)$ no es necesariamente igual a A, además se puede usar como una abstracción de rendering pero no de modelado.

Torres realiza una generallización del concepto de objeto propuesto por Fiume. Define el concepto de objeto gráfico en base a funciones de presencia y aspecto, define también una serie de operadores que le dotan de estructura de espacio vectorial y de álgebra de boole, con lo que se puede usar como una abstracción de modelado además de como abstracción de rendering.

La *presencia* describe la ocupación de espacio de un objeto gráfico, pero no se trata de un valor 0 ó 1, sino de un valor contable que permite construir objetos mediante superposición (valor de presencia > 1) o substracción (puede ser negativo) de objetos.

El *aspecto* define su apariencia y propiedades visuales. Puede contener información tales como el color, la transparencia, el coeficiente de reflexión, peso, etc.

Nuestra propuesta se centra exclusivamente en los objetos volumétricos. Hemos tomando como punto de partida la teoría de objetos gráficos de Torres pero considerando que la presencia sólo toma valores 0 ó 1. El aspecto lo hemos dividido considerando por un lado el dominio de propiedades como un dominio que recoge los valores objeto de estudio de un modelo volumétrico y por otro lado definiendo unas funciones de interpretación que dotan de atributos visuales a dichos valores del dominio de propiedad. También nos hemos centrado en los casos resultantes de considerar al dominio de propiedad como un conjunto de valores discretos o continuos y en el estudio de los modelos volumétricos que se aplican sobre estos tipos de dominios de propiedades.

2.2. Modelo matemático de dominio de propiedad

Un dominio de propiedad, no es más que un conjunto que contendrá todos los valores posibles de la propiedades representadas en el modelo del volumen.

Sin embargo, una definición tal que así sería muy general, ya que si pretendemos poder realizar con el modelo del volumen simulaciones, tendremos que operar con él y por tanto tendremos que realizar operaciones con sus valores de propiedad.

En el dominio de propiedad definiremos una serie de operaciones que cumplan unos determinados requisitos. La necesidad de estas operaciones se justifica en que las operaciones entre modelos volumétricos se van a realizar en parte mediante operaciones entre los valores de propiedad que presentan dichos modelos.

Hay que hacer notar que se pueden tener propiedades tanto con variación continua, como la temperatura, como con variación discreta, como sería el caso de representar un material.

2.2.1. Dominio de propiedad continuo

Un dominio de propiedad continuo se va a caracterizar porque entre cada dos valores van a existir infinitos valores, además de poseer una serie operaciones que cumplen unas determinadas cualidades.

Las operaciones han sido elegidas para dotar al dominio de propiedad de una operatividad amplia al mismo tiempo que son operaciones de implementación eficientes.

Definición 2.1 (Dominio de propiedad continuo)
Decimos que Γ *es un dominio de propiedad continuo sii es homeomórfico a* $\mathbb{R}^n$ *para algún* $n > 0$ *y tiene definidas las siguientes operaciones:*

- $+$*: Suma interna que le da a* $(\Gamma, +)$ *la estructura de grupo conmutativo.*
- $*$*: Producto externo sobre un cuerpo K siendo* $(\Gamma, +, *)$ *K-espacio vectorial.*
- $\times$*: Producto interno que satisface las propiedades: asociativa, conmutativa, y existencia de elemento identidad.*

$\diamond$

2.2.1.1. Composición de dominios de propiedad continuos

Cuando de un volumen queramos representar más de una propiedad, el primer paso será componer los correspondientes dominios de propiedad de cara a obtener un único dominio de propiedad que, aunque compuesto, represente todas las propiedades objeto de estudio en un volumen.

El operador de composición lo notamos como $\uplus$ y se define en base al producto cartesiano de los dominios a componer.

Definición 2.2 (Dominio de propiedad continuo compuesto)
Sean los dominios de propiedad continuos Γ_1 *y* Γ_2. *Sean* $\gamma_{1_1}, \gamma_{2_1} \in \Gamma_1$ *y* $\gamma_{1_2}, \gamma_{2_2} \in \Gamma_2$ *y sea* K *un cuerpo con* $k \in K$.
Definimos el dominio de propiedad continuo compuesto $\Gamma = \Gamma_1 \uplus \Gamma_2$ *definiendo cómo se obtiene* Γ *y sus operaciones:*

$$\Gamma = \Gamma_1 \times \Gamma_2$$

$$+ : \Gamma \times \Gamma \to \Gamma$$
$$(\gamma_{1_1}, \gamma_{1_2}) + (\gamma_{2_1}, \gamma_{2_2}) = (\gamma_{1_1} + \gamma_{2_1}, \gamma_{1_2} + \gamma_{2_2})$$

$$\times : \Gamma \times \Gamma \to \Gamma$$
$$(\gamma_{1_1}, \gamma_{1_2}) \times (\gamma_{2_1}, \gamma_{2_2}) = (\gamma_{1_1} \times \gamma_{2_1}, \gamma_{1_2} \times \gamma_{2_2})$$

$$* : K \times \Gamma \to \Gamma$$
$$k * (\gamma_{1_1}, \gamma_{1_2}) = (k * \gamma_{1_1}, k * \gamma_{1_2})$$ ◇

Teorema 2.3
El conjunto Γ *junto con las operaciones definidas anteriormente es un dominio de propiedad continuo.*
Dem:
Por ser Γ_1 *homeomórfico a* $\mathbb{R}^n$ *para algún n existe una función* $f_1 : \Gamma_1 \to \mathbb{R}^n$ *que es continua siendo* f_1^{-1} *también contínua. Por la misma razón existe* $f_2 : \Gamma_2 \to \mathbb{R}^n$ *que es continua siendo* f_2^{-1} *también contínua.*
Definiendo $f : \Gamma_1 \times \Gamma_2 \to \mathbb{R}^{n+m}$ *como*
$f(x,y) = (f_1(x), f_2(y))$ *tenemos que* Γ *es homeomórfico a* $\mathbb{R}^p$ *para algún* $p > 0$ *cum-*

pliendo sus operaciones las propiedades requeridas por la forma en que se han definido. ◇

Los dominios de propiedad compuestos pueden a su vez componerse para obtener dominios de propiedad que abarquen un mayor número de propiedades, tan solo hay que evitar, por eficiencia, que en la cadena de productos cartesianos final no haya ningún dominio de propiedad que aparezca más una vez.

2.2.2. Dominio de propiedad discreto

Un dominio de propiedad discreto se va a caracterizar por tener una cardinalidad finita y tener estructura de álgebra de boole.

Definición 2.4 (Dominio de propiedad discreto)
Dado el dominio de propiedad continuo $\overline{\Gamma}$ *decimos que* $\Gamma \subset \overline{\Gamma}$ *es un dominio de propiedad discreto extraído de* $\overline{\Gamma}$ *sii* Γ *tiene cardinalidad finita y además tiene estructura de álgebra de boole.*
El elemento neutro lo notaremos como 0*. El elemento universo lo notaremos como* ω*.* ◇

Las operaciones del álgebra de boole representan a operaciones reales de las propiedades representadas, por lo tanto deben estar definidas del mismo modo en todos los dominios de propiedad discretos extraídos de un mismo dominio de propiedad.

Los dominios de propiedad discretos van a representar propiedades cuyos valores no van a variar de forma continua, por ejemplo, el tipo de materia, o el color almacenado en la memoria de un ordenador, como podemos observar en el siguiente ejemplo.

Ejemplo 1
A partir del dominio de propiedad *color* podemos extraer el dominio de propiedad discreto $\{negro = 0, rojo, verde, azul, cian, magenta, amarillo, blanco = \omega\}$ donde la operación $\cup$ representa la mezcla aditiva de colores y la operación $\cap$ la mezcla sustractiva de colores.

El *negro* es el elemento neutro y el $blanco$ el elemento universal.

Las operaciones se definen según los siguientes cuadros:

$K = negro,\ R = rojo,\ G = verde,\ B = azul,$

$C = cian,\ M = magenta,\ Y = amarillo,\ W = blanco$

$\cup$	K	R	G	B	C	M	Y	W
K	K	R	G	B	C	M	Y	W
R	R	R	Y	M	W	M	Y	W
G	G	Y	G	C	C	W	Y	W
B	B	M	C	B	C	M	W	W
C	C	W	C	C	C	W	W	W
M	M	M	W	M	W	M	W	W
Y	Y	Y	Y	W	W	W	Y	W
W	W	W	W	W	W	W	W	W

$\cap$	K	R	G	B	C	M	Y	W
K	K	K	K	K	K	K	K	K
R	K	R	K	K	K	R	R	R
G	K	K	G	K	G	K	G	G
B	K	K	K	B	B	B	K	B
C	K	K	G	B	C	B	G	C
M	K	R	K	B	B	M	R	M
Y	K	R	G	K	G	R	Y	Y
W	K	R	G	B	C	M	Y	W

$\diamond$

Aprovechándonos de la finitud del dominio de propiedad discreto podemos seleccionar un subconjunto de valores de propiedad que nos generen el resto.

2.2.2.1. Base booleana de propiedades

Una base booleana de propiedades de un dominio de propiedad discreto será un conjunto de propiedades del dominio con las cuales podemos obtener todas las propiedades del dominio mediante la operación de unión. Se define como sigue:

Definición 2.5 (Base booleana de propiedades)
Dado un dominio de propiedad discreto Γ. *El conjunto de valores de propiedad* $\Gamma_b = \{\gamma_0, \gamma_1, \ldots, \gamma_n\} \subseteq \Gamma$ *formarán una base de* Γ *sii:*

1. $\forall \gamma_k \in \Gamma \; \exists \gamma_i, \ldots, \gamma_j \in \Gamma_b$ tal que $\gamma_k = \bigcup_{l=i}^{j} \gamma_l$
2. $\bigcup_{\gamma_i \in \Gamma_b} \gamma_i = \omega$
3. $\forall \gamma_i, \gamma_j \in \Gamma_b \; : \; \gamma_i \neq \gamma_j \Rightarrow \; \gamma_i \cap \gamma_j = 0$

$\diamond$

Teorema 2.6
Para todo dominio de propiedad discreto Γ *con al menos dos elementos, existe una base booleana.*
Dem:
Según [Permingeat, 88] sabemos lo siguiente:

1. *Podemos definir una relación* $a \leq b$ *sii* $a \cap b = a$ *tal que* $(\Gamma, \leq, \sim, \omega, 0)$ *es un retículo de boole.*
2. *Podemos definir un átomo como aquellos elementos* $a \neq 0$ *tal que* $\forall x \in \Gamma$ *o bien* $x \cap a = a$ *o bien* $x \cap a = 0$.
3. *Existe al menos un átomo.*
4. *Si* a_1 *y* a_2 *son átomos, se cumple* $a_1 \cap a_2 \neq 0 \Rightarrow a_1 = a_2$.

5. *Si* $\{a_i, \ldots, a_j\}$ *es el conjunto de átomos menores o iguales al elemento* $x \in \Gamma - \{0\}$ *se cumple que* $x = a_i \cup \ldots \cup a_j$

Por tanto, podemos afirmar que el conjunto de átomos de Γ *es una base booleana de* Γ. ◇

Teorema 2.7

Para todo dominio de propiedad discreto Γ *con al menos dos elementos, su base booleana* Γ_b *es única.*

Dem:

Supongamos que existen dos bases booleanas Γ_{b1} *y* Γ_{b2} *tales como las siguientes:*

$\Gamma_{b1} = \{\gamma_{1_1}, \ldots, \gamma_{1_n}\}$

$\Gamma_{b2} = \{\gamma_{2_1}, \ldots, \gamma_{2_m}\}$

Cualquier γ_{1_i} *por ser elemento de* Γ *puede expresarse como la unión de uno o más elementos de* Γ_{b2}*, ya que* Γ_{b2} *es base booleana de* Γ*, es decir, existen uno o varios* j *tales que* $\gamma_{1_i} = \cup_j \gamma_{2_j}$.

De forma análoga obtenemos que cualquier γ_{2_k} *puede expresarse como* $\gamma_{2_k} = \cup_l \gamma_{1_l}$*, por lo que cualquier* γ_{1_i} *podría expresarse como la unión de uno o más elementos de* Γ_{b1}*, pero un elemento de una base booleana no puede expresarse como la unión de varios elementos de la misma base, por lo que se demuestra que las dos bases booleanas son en realidad la misma.* ◇

Por tanto, un dominio de propiedad va a estar caracterizado y definido por su base booleana de propiedad.

Teorema 2.8

Sea $\overline{\Gamma}$ *un dominio de propiedad, y sea* $\Gamma_b \subset \overline{\Gamma}$ *un conjunto finito de propiedades que tiene definidas las operaciones*

$\cup : \Gamma_b \times \Gamma_b \longrightarrow \overline{\Gamma}$

$\cap : \Gamma_b \times \Gamma_b \longrightarrow \overline{\Gamma}$

$\sim : \Gamma_b \longrightarrow \overline{\Gamma}$

satisfaciendo las siguientes propiedades:

1. $\forall \gamma_i, \gamma_j \in \Gamma_b \ : \ \gamma_i \neq \gamma_j \Rightarrow \ \gamma_i \cap \gamma_j = 0$

2. $\exists\, \omega = \bigcup_{\gamma_i \in \Gamma_b} \gamma_i$ *(elemento universal)*

3. *Para cualesquiera* $x, y, z \in \Gamma_b$*, se cumple:*

   - *Asociativa*
     $x \cup (y \cup z) = (x \cup y) \cup z$
     $x \cap (y \cap z) = (x \cap y) \cap z$
   - *Conmutativa*
     $x \cup y = y \cup x$
     $x \cap y = y \cap x$
   - *Idempotente*
     $x \cup x = x$
     $x \cap x = x$
   - *Absorvente*
     $x \cup (x \cap y) = x$
     $x \cap (x \cup y) = x$
   - *Complemento*
     $x \cup \sim x = \omega$
     $x \cap \sim x = 0$
   - *Distributiva*
     $x \cup (y \cap z) = (x \cup y) \cap (x \cup z)$
     $x \cap (y \cup z) = (x \cap y) \cup (x \cap z)$

Entonces el conjunto $\Gamma = \{\gamma \in \overline{\Gamma} : \gamma = \bigcup_{\gamma_i \in P} \gamma_i \ \ \forall P \in \wp(\Gamma_b)\}$ *es un dominio de propiedad discreto generado por* Γ_b*. Y lo notamos como* $\Gamma = \mathcal{G}(\Gamma_b)$.
Dem:
Trivial por la forma en que se obtiene Γ. $\diamond$

2.2.2.2. Composición de dominios de propiedad discretos

Cuando vamos a componer dominios de propiedad discretos tenemos que hacer la distinción entre si los dominios de propiedad discretos que vamos a componer han sido extraídos de un mismo dominio de propiedad o han surgido de dominios de propiedad distintos. En ambos casos procederemos calculando la base booleana compuesta para después generar el dominio de propiedad compuesto a partir de dicha base.

Lema 2.9
Dados Γ_1 y Γ_2 dos dominios de propiedad discretos, es posible definir un dominio de propiedad discreto a partir de una composición de ellos.
Dem:

Caso 1: Dominios discretos procedentes del mismo dominio de propiedad

Si los dos dominios de propiedad discretos Γ_1 y Γ_2 han sido extraídos del mismo dominio de propiedad $\overline{\Gamma}$ siendo el 0 un elemento común y representando las operaciones lo mismo. Sean Γ_{b1} y Γ_{b2} las bases booleanas respectivamente, la base booleana Γ_b del dominio de propiedad discreto Γ composición de los otros dos se construye como sigue:

1. *Obtenemos $\Gamma'_b = \Gamma_{b1} \cup \Gamma_{b2}$*
2. *Para todo par $\gamma_i, \gamma_j \in \Gamma'_b$ tales que $\gamma_i \cap \gamma_j = \gamma_k \neq 0$ los sustituimos por los elementos $\gamma_i - \gamma_k, \gamma_j - \gamma_k, \gamma_k$ de modo que obtenemos un Γ_b tal que la intersección entre cualesquiera dos de sus elementos es cero.*

Caso 2: Dominios discretos procedentes de distintos dominios de propiedad

Si los dos dominios de propiedad discretos Γ_1 y Γ_2 han sido extraídos de distintos dominios de propiedad $\overline{\Gamma}_1$ y $\overline{\Gamma}_2$ respectivamente, y Γ_{b1} y Γ_{b2} son las respectivas bases booleanas de propiedad. La base booleana Γ_b del dominio de propiedad discreto Γ composición de los otros dos se construye como sigue:

1. *Construimos el dominio composición* $\overline{\Gamma} = \overline{\Gamma}_1 \uplus \overline{\Gamma}_2$.
2. *Definimos la base booleana* Γ_b *del dominio de propiedad discreto* Γ *como el conjunto* $\{(x,0) \ : \ x \in \Gamma_{b1}\} \cup \{(0,y) \ : \ y \in \Gamma_{b2}\}$.
3. *Las operaciones se definen de la siguiente forma:*
   $\cup : \Gamma_b \times \Gamma_b \to \overline{\Gamma}$
   $(\gamma_{1_1}, \gamma_{1_2}) \cup (\gamma_{2_1}, \gamma_{2_2}) = (\gamma_{1_1} \cup \gamma_{2_1}, \gamma_{1_2} \cup \gamma_{2_2})$
   $\cap : \Gamma_b \times \Gamma_b \to \overline{\Gamma}$
   $(\gamma_{1_1}, \gamma_{1_2}) \cap (\gamma_{2_1}, \gamma_{2_2}) = (\gamma_{1_1} \cap \gamma_{2_1}, \gamma_{1_2} \cap \gamma_{2_2})$
   $\sim : \Gamma_b \to \overline{\Gamma}$
   $\sim (\gamma_{1_1}, \gamma_{1_2}) = (\sim \gamma_{1_1}, \sim \gamma_{1_2})$

A partir de la base booleana composición obtenida como se ha descrito según el caso, se construye $\Gamma = \Gamma_1 \uplus \Gamma_2$ *como* $\Gamma = \mathcal{G}(\Gamma_b)$.

◇

2.2.3. Proyección de un dominio continuo en uno discreto

Se ha indicado que una de las características de un dominio de propiedad discreto Γ es la de ser un subconjunto de un dominio de propiedad continuo $\overline{\Gamma}$. A la función o funciones que nos permiten transformar los valores del dominio continuo en valores del dominio discreto las llamaremos funciones de proyección.

Definición 2.10 (Función de proyección de dominios continuos)
Dado un dominio de propiedad continuo $\overline{\Gamma}$ *y un dominio de propiedad discreto* $\Gamma \subset \overline{\Gamma}$, *llamaremos función de proyección de* $\overline{\Gamma}$ *en* Γ *a toda aplicación*

$$P : \overline{\Gamma} \to \Gamma$$

◇

Obviamente, la operación de proyección implica una pérdida de información, en concreto todos aquellos $\gamma \in \overline{\Gamma} - \Gamma$. Sin embargo, se representa el dominio de propiedad mediante un conjunto de cardinalidad finita y con operaciones booleanas.

2.2.4. Dominio de propiedad dependiente

Pueden darse situaciones en las que un valor de propiedad de un dominio concreto dependa directamente de otros valores de propiedad de otros dominios distintos, por ejemplo, en una película fotográfica de blanco y negro una vez revelada, el grado de transparencia de un punto depende directamente de la densidad de plata que posee, por lo tanto no tiene sentido almacenar las dos propiedades, tan solo se almacena una junto con la función que nos permita obtener valores de la otra.

Definición 2.11 (Dominio de propiedad dependiente)
Decimos que un dominio de propiedad Γ_2 *es dependiente de un dominio de propiedad* Γ_1 *sii existe una función* $\beta : \Gamma_1 \longrightarrow \Gamma_2$ *monótona con respecto a las operaciones que puedan definirse en los dominios de propiedad correspondientes.* ◇

Con esta definición se simplifica la representación de determinadas propiedades, siendo necesario sólo representar los dominios de propiedad no dependientes y las correspondientes funciones β.

2.2.5. Funciones de interpretación

Una vez representado un modelo volumétrico, cuando queramos hacer una visualización del mismo, habra que establecer una aplicación entre los valores del dominio de propiedad representado y los atributos visuales que formen la imagen obtenida.

Esta aplicación se va a hacer a través de lo que llamamos *función de interpretación*, si llamamos Λ al dominio de valores que vamos a usar para mostrar el volumen, la función de interpretación que notamos como ν se define así

$$\nu : \Gamma \longrightarrow \Lambda$$

Usualmente, Λ va a estar formado por valores correspondientes a parámetros de visualización como color, transparencia, etc. De esta forma se obtiene una representación gráfica del volumen que nos permite una mejor comprensión de las propiedades del mismo.

Las funciones de clasificación usadas en la visualización directa hacen uso de funciones de interpretación.

2.3. Modelo matemático de volumen

Recogiendo lo dicho en las secciones y capítulo anterior haremos una definición de modelo volumétrico, los cuales clasificaremos en base al dominio de propiedad sobre el que se definen en modelos volumétricos discretos y continuos.

Definición 2.12 (Modelo volumétrico)
Dado un dominio de propiedad Γ *que contiene las propiedades que vamos a representar de un volumen, definimos modelo volumétrico de dicho volumen, que notamos como* O_α*, como*

$$\alpha : V \subset \mathbb{R}^n \longrightarrow \Gamma$$

donde V *es cerrado, acotado, rígido y regular según la definición dada en la sección 1.1.1 (pág. 15) .* ◇

Puede verse fácilmente que un volumen es un objeto más general que un sólido y que añadiéndole una serie de restricciones tenemos un modelo matemático de sólidos definido por [Mäntylä, 88].

Proposición 2.13
Un modelo volumétrico con $n = 3$*, con* $\Gamma = \{no\_ocupado, ocupado\}$ *y con* $\alpha(p) = ocupado \ \ \forall p \in V$ *es un sólido.*
Dem:
El conjunto V *así definido es un subconjunto de* R^3*, cerrado, acotado, rígido y regular. Y por lo tanto, es un sólido.* ◇

Sobre un modelo volumétrico así definido podemos realizar las operaciones que tenemos definidas sobre el dominio de propiedad.

2.3.1. Equivalencia de modelos volumétricos

Veamos cómo se puede establecer una relación de equivalencia entre modelos volumétricos.

Definición 2.14 (Operador de relación $\circ$)
Sean O_{α_1} y O_{α_2} dos modelos volumétricos, con $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma$ y $\alpha_2 : V_2 \subset \mathbb{R}^n \to \Gamma$
Sea $V_1' = \{p \in V_1 : \alpha_1(p) \neq 0\}$ y $V_2' = \{p \in V_2 : \alpha_2(p) \neq 0\}$

O_{α_1} se relaciona con O_{α_2} y se nota como $O_{\alpha_1} \circ O_{\alpha_2}$ sii $V_1' = V_2'$ y además $\alpha_1(p) = \alpha_2(p) \ \ \forall p \in V_1' = V_2'$. ◇

Teorema 2.15
El operador de relación $\circ$ entre modelos volumétricos define una relación de equivalencia entre éstos.
Dem:
El operador de relación $\circ$ se ha definido en base al operador de relación $=$. El operador $=$ cumple las propiedades reflexiva, simétrica y transitiva, por tanto el operador $\circ$ también las va a cumplir y por tanto define una relación de equivalencia. ◇

Al conjunto de clases de equivalencia que se puede definir sobre $\mathbb{R}^n$ y sobre un determinado dominio de propiedad Γ lo notamos como O^n_Γ. En lo sucesivo cuando nos refiramos a un modelo volumétrico O_α nos estaremos refiriendo a la clase de equivalencia a la que pertenezca. Igualmente los operadores que definamos entre modelos volumétricos pueden verse como operaciones entre las clases de equivalencia.

2.3.2. Operaciones sobre modelos volumétricos

Sobre Γ tenemos definidas operaciones unarias internas, binarias internas, y binarias sobre un cuerpo externo. Pasemos a definir estos tipos de operaciones sobre los modelos volumétricos de una forma genérica.

Definición 2.16 (Operación unaria interna)
Dado O_{α_1} un modelo volumétrico definido como $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma_1$ y dada $op1i$ una operación unaria interna definida sobre Γ_1.
Definimos $O_\alpha = op1i(O_{\alpha_1})$ como

$$\alpha : V_1 \subset \mathbb{R}^n \to \Gamma_1$$
$$\alpha(p \in V_1) = op1i(\alpha_1(p))$$ ◇

Definición 2.17 (Operación binaria sobre un cuerpo externo)
Dado O_{α_1} un modelo volumétrico definido como $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma_1$, dado K un cuerpo y dada op2e una operación binaria definida sobre Γ_1 y K.
Definimos $O_\alpha = K\ op2e\ O_{\alpha_1}$ como

$\alpha : V_1 \subset \mathbb{R}^n \to \Gamma_1$

$\alpha(p \in V_1) = K\ op2e\ \alpha_1(p)$ ◇

Definición 2.18 (Operación binaria interna)
Dados O_{α_1} y O_{α_2} dos modelos volumetricos definidos como $\alpha_1 : V_1 \in R^n \to \Gamma_1$ y $\alpha_2 : V_2 \in R^n \to \Gamma_2$. Dada op2i una operación binaria interna definida sobre ambos dominios de propiedad. Y dada opb una operación booleana regularizada en $\mathbb{R}^n$.
Definimos $O_\alpha = O_{\alpha_1}\ op2i\ O_{\alpha_2}$ como

$\alpha : V \in R^n \to \Gamma$ *donde*

$V = V_1\ opb\ V_2$

$\Gamma = \Gamma_1 \uplus \Gamma_2$

Se extienden o restringen α_1 y α_2 obteniendo α'_1 y α'_2 para que las aplicaciones estén definidas de V en Γ

$\alpha(p \in V) = \alpha'_1(p)\ op2i\ \alpha'_2(p)$ ◇

Las operaciones binarias internas deben indicar con que operación binaria regularizada van a operar los conjuntos V. Por ejemplo, la notación $+_\cup$ indica que dos modelos volumétricos se van a operar mediante la $\cup$ booleana regularizada en $\mathbb{R}^n$ a nivel de conjuntos V y mediante $+$ a nivel de Γ.

2.3.3. Clasificación de modelos volumétricos

Los modelos volumétricos los clasificaremos en continuos o discretos según las características del dominio de propiedad sobre el que se apliquen.

Un modelo volumétrico será continuo cuando Γ sea un dominio de propiedad contínuo.

Un modelo volumétrico será discreto cuando se aplique sobre un dominio de propiedad discreto y pueda obtenerse por unión de modelos volumétricos constantes, entendiendo como tales aquellos en los que se cumple $\alpha(p) = \gamma_0 \ \forall p \in V$.

2.4. Modelos volumétricos discretos

En el estudio de los modelos volumétricos discretos hemos partido de modelos simples que son homogéneos en tanto en cuanto todo el volumen presenta un mismo valor de propiedad. A partir de ellos llegamos a definir un álgebra de boole de modelos volumétricos discretos.

Las operaciones booleanas $\cup$, $\cap$ y $-$ que usaremos en esta sección para operar modelos volumétricos de propiedad y modelos volumétricos discretos las definimos a continuación. Suponemos que los modelos volumétricos discretos a operar están definidos sobre el mismo dominio de propiedad, si no fuese el caso, se pueden componer los dominios de propiedad distintos de modo que se consigue que se estén definidas sobre el mismo dominio de propiedad.

Sean $O_{\alpha_1} = \alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma$ y $O_{\alpha_2} = \alpha_2 : V_2 \subset \mathbb{R}^n \to \Gamma$ dos modelos volumétricos y sea $O_\alpha = \alpha : V \subset \mathbb{R}^n \to \Gamma$ el resultado de la operación entre O_{α_1} y O_{α_2}.
Sea $i(V)$ el interior de V.

La extensión de α_i se realiza de la siguiente forma:

$$\alpha_i'(p) = \begin{cases} \alpha_i(p) & \text{si} \quad p \in V_i \\ 0 & \text{si} \quad p \notin V_i \end{cases}$$

Para la operación $\cup$, $O_\alpha = O_{\alpha_1} \cup O_{\alpha_2}$ donde
$V = V_1 \cup V_2$
$\alpha(p) = \alpha_1'(p) \cup \alpha_2'(p)$

Para la operación $\cap$, $O_\alpha = O_{\alpha_1} \cap O_{\alpha_2}$ donde
$V = V_1 \cap V_2$
$\alpha(p) = \alpha_1'(p) \cap \alpha_2'(p)$

Para la operación $-$, $O_\alpha = O_{\alpha_1} - O_{\alpha_2}$ donde
$V = V_1$

$$\alpha(p) = \begin{cases} \alpha_1'(p) & \text{si} \quad p \notin i(V_2) \\ \alpha_1'(p) - \alpha_2'(p) & \text{si} \quad p \in i(V_2) \end{cases}$$

La operación $\sim$ se realiza mediante la diferencia $\sim O_{\alpha_1} = O_\omega - O_{\alpha_1}$ donde:

$O_\omega = \alpha_\omega : \mathbb{R}^n \to \Gamma$

$\alpha_\omega(p) = \omega$

Del mismo modo definimos $O_0 = \sim O_\omega$, es decir, $\alpha_0 : \mathbb{R}^n \to \Gamma$ con $\alpha(p) = 0$.

2.4.1. Modelo volumétrico de propiedad

Cosideremos un modelo volumétrico en el que todos sus puntos se aplican sobre el mismo valor de propiedad $\gamma \in \Gamma$, a este modelo volumétrico lo denominamos modelo volumétrico de propiedad γ.

Definición 2.19 (Modelo volumétrico de propiedad γ)
Dado Γ dominio de propiedad discreto y dado $\gamma \in \Gamma$. Un modelo volumétrico de propiedad γ es un modelo voluétrico O_α definido como
$\alpha : V \subset \mathbb{R}^n \to \Gamma$ *donde* $\forall p \in V \Rightarrow \alpha(p) = \gamma$

La función α la notaremos como α_γ. Y al modelo volumétrico lo notaremos como O_{α_γ} u O_γ. ◇

Ejemplo 2
Podemos modelar un espacio de color rojo (figura 2.1) como un modelo volumétrico de propiedad *rojo*. ◇

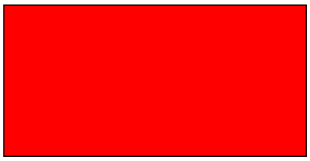


Figura 2.1: Modelo volumétrico de propiedad

Dados dos modelos volumétricos sobre propiedades distintas γ_0 y γ_1 pertenecientes al mismo dominio de propiedad Γ, podemos obtener un modelo volumétrico nuevo operando con ellos mediante la operación binaria interna $\cup$.

$$O_\alpha = O_{\alpha_{\gamma_0}} \cup O_{\alpha_{\gamma_1}}$$

De esta forma obtenemos un nuevo objeto O_α cuyos puntos pueden presentar tres propiedades distintas: γ_0, γ_1, y $\gamma_0 \cup \gamma_1$.

Ejemplo 3

Tenemos modelado un espacio rojo y un espacio verde, si los unimos podemos obtener un modelo volumétrico con tres valores de propiedad: rojo, verde y amarillo. (Figura 2.2). ◇

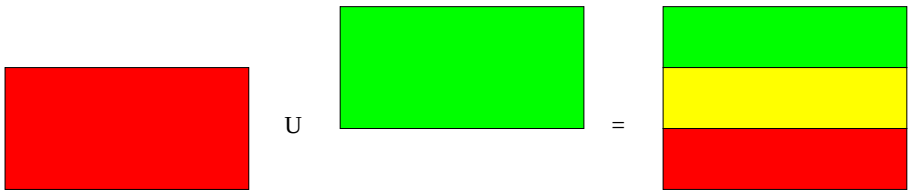


Figura 2.2: Unión de modelos volumétricos de propiedad

Antes de continuar demostremos que la $\cup$, $\cap$ y $\sim$ cumplen unas determinadas propiedades.

Proposición 2.20

Sean los modelos volumétricos de propiedad

$O_{\gamma_1} = \alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma \quad \alpha_1(p \in V_1) = \gamma_1$

$O_{\gamma_2} = \alpha_2 : V_2 \subset \mathbb{R}^n \to \Gamma \quad \alpha_2(p \in V_2) = \gamma_2$

$O_{\gamma_3} = \alpha_3 : V_3 \subset \mathbb{R}^n \to \Gamma \quad \alpha_3(p \in V_3) = \gamma_3$

Se cumplen las siguientes propiedades:

1. *Asociativa*
   $O_{\gamma_1} \cup (O_{\gamma_2} \cup O_{\gamma_3}) = (O_{\gamma_1} \cup O_{\gamma_2}) \cup O_{\gamma_3}$
   $O_{\gamma_1} \cap (O_{\gamma_2} \cap O_{\gamma_3}) = (O_{\gamma_1} \cap O_{\gamma_2}) \cap O_{\gamma_3}$
2. *Conmutativa*
   $O_{\gamma_1} \cup O_{\gamma_2} = O_{\gamma_2} \cup O_{\gamma_1}$
   $O_{\gamma_1} \cap O_{\gamma_2} = O_{\gamma_2} \cap O_{\gamma_1}$
3. *Idempotente*
   $O_{\gamma_1} = O_{\gamma_1} \cup O_{\gamma_1}$
   $O_{\gamma_1} = O_{\gamma_1} \cap O_{\gamma_1}$
4. *Absorbente*
   $O_{\gamma_1} \cap (O_{\gamma_1} \cup O_{\gamma_2}) = O_{\gamma_1}$
   $O_{\gamma_1} \cup (O_{\gamma_1} \cap O_{\gamma_2}) = O_{\gamma_1}$
5. *Complemento*
   $O_{\gamma_1} \cup \sim O_{\gamma_1} = O_{\omega}$
   $O_{\gamma_1} \cap \sim O_{\gamma_1} = O_0$
6. *Distributiva*
   $O_{\gamma_1} \cup (O_{\gamma_2} \cap O_{\gamma_3}) = (O_{\gamma_1} \cup O_{\gamma_2}) \cap (O_{\gamma_1} \cup O_{\gamma_3})$
   $O_{\gamma_1} \cap (O_{\gamma_2} \cup O_{\gamma_3}) = (O_{\gamma_1} \cap O_{\gamma_2}) \cup (O_{\gamma_1} \cap O_{\gamma_3})$

Dem:

Asociativa *para la* **unión**

$O_{\gamma_1} \cup (O_{\gamma_2} \cup O_{\gamma_3}) = O_{\gamma_1} \cup O_{\alpha_{23}}$ *donde* $\alpha_{23} : V_2 \cup V_3 \to \Gamma$ *se define como*

$$\alpha_{23}(p) \begin{cases} \gamma_2 & si \quad p \in V_2 \wedge p \notin V_3 \\ \gamma_2 \cup \gamma_3 & si \quad p \in V_2 \wedge p \in V_3 \\ \gamma_3 & si \quad p \notin V_2 \wedge p \in V_3 \end{cases}$$

$O_{\gamma_1} \cup O_{\alpha_{23}} = O_{\alpha_{123}}$ *donde* $\alpha_{123} : V_1 \cup V_2 \cup V_3 \to \Gamma$ *se define como*

$$\alpha_{123}(p) = \begin{cases} \gamma_1 & si \quad p \in V_1 \wedge p \notin V_2 \wedge p \notin V_3 \\ \gamma_2 & si \quad p \notin V_1 \wedge p \in V_2 \wedge p \notin V_3 \\ \gamma_3 & si \quad p \notin V_1 \wedge p \notin V_2 \wedge p \in V_3 \\ \gamma_1 \cup \gamma_2 & si \quad p \in V_1 \wedge p \in V_2 \wedge p \notin V_3 \\ \gamma_1 \cup \gamma_3 & si \quad p \in V_1 \wedge p \notin V_2 \wedge p \in V_3 \\ \gamma_2 \cup \gamma_3 & si \quad p \notin V_1 \wedge p \in V_2 \wedge p \in V_3 \\ \gamma_1 \cup \gamma_2 \cup \gamma_3 & si \quad p \in V_1 \wedge p \in V_2 \wedge p \in V_3 \end{cases}$$

De forma similar se muestra que a la expresión $(O_{\gamma_1} \cup O_{\gamma_2}) \cup O_{\gamma_3}$ *le corresponde una definición de* α *igual a la función* α_{123} *precedente.*

Asociativa *para la* **intersección**

Se demuestra que se cumple procediendo de manera similar a la propiedad asociativa para la unión.

Resto de propiedades

Siguiendo el mismo proceso de obtención de la función α *correspondiente a al modelo volumétrico a ambos lados de cada igualdad se demuestra que se cumplen todas las propiedades indicadas.* ◇

2.4.2. Definición de modelo volumétrico discreto

Nuestra idea es caracterizar los modelos volumétricos discretos como la unión de los modelos volumétricos de propiedad de los que están formados. Para ello usamos las propiedades de la base booleana para definir los modelos volumétricos de propiedad que van a componer los modelos volumétricos discretos.

De esta forma, dado un dominio de propiedad discreto Γ obtenemos su base Γ_b y creamos modelos volumétricos de propiedad para cada elemento de la base, y realizando uniones con ellos obtenemos modelos volumétricos que pueden presentar cualquier propiedad del dominio.

Definición 2.21 (Modelo volumétrico discreto)
Un modelo volumétrico O_α es un modelo volumétrico discreto sii existen modelos volumétricos de propiedad $O_{\alpha_{\gamma_i}}, \ldots, O_{\alpha_{\gamma_j}}$ tales que:

1. $\{\gamma_i, \ldots, \gamma_j\} \subseteq \Gamma_b$.
2. $O_\alpha = \bigcup_{l=i}^{j} O_{\alpha_{\gamma_l}}$

Al conjunto de todos los modelos volumétricos discretos que podemos definir desde $\mathbb{R}^n$ sobre el dominio de propiedad Γ lo notaremos como $O^n_{\Gamma_d}$. ◇

2.4.3. Operaciones sobre modelos volumétricos discretos

Tenemos caracterizado un modelo volumétrico discreto, pasemos a dar una serie de operaciones para obtener nuevos modelos volumétricos discretos a partir de otros.

Lema 2.22
La unión de modelos volumétricos discretos es un modelo volumétrico discreto.
Dem:
Dados O_{α_1} y O_{α_2} dos modelos volumétricos discretos, la unión de ambos es

$$O_\alpha = O_{\alpha_1} \cup O_{\alpha_2}$$

Si O_{α_1} es un modelo volumétrico discreto existen

$$O_{\gamma_{1_i}}, \ldots, O_{\gamma_{1_j}}$$

modelos volumétricos de propiedad tales que

$$\{\gamma_{1_i}, \ldots, \gamma_{1_j}\} \subseteq \Gamma_b$$

siendo Γ_b una base booleana de Γ y

$$O_{\alpha_1} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}$$

Con respecto a O_{α_2} ocurre de forma análoga, por lo tanto podemos expresar

$$O_\alpha = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}} \cup O_{\gamma_{2_i}} \cup \ldots \cup O_{\gamma_{2_j}}$$

siendo éstos, modelos volumétricos de propiedad y pertenecientes a la base booleana de Γ. ◇

Lema 2.23

La intersección regularizada de modelos volumétricos discretos es un modelo volumétrico discreto.

Dem:

Dados O_{α_1} y O_{α_2} dos modelos volumétricos discretos, la intersección de ambos es

$$O_\alpha = O_{\alpha_1} \cap O_{\alpha_2}$$

Si O_{α_1} es un modelo volumétrico discreto existen

$$O_{\gamma_{1_i}}, \ldots, O_{\gamma_{1_j}}$$

modelos volumétricos de propiedad tales que

$$\{\gamma_{1_i}, \ldots, \gamma_{1_j}\} \subseteq \Gamma_b$$

siendo Γ_b una base booleana de Γ y

$$O_{\alpha_1} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}$$

Con respecto a O_{α_2} ocurre de forma análoga. Por lo tanto podemos expresar

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap (O_{\gamma_{2_i}} \cup \ldots \cup O_{\gamma_{2_j}})$$

Aplicando la propiedad distributiva tenemos que

$$O_\alpha = \bigcup (O_{\gamma_{1_k}} \cap O_{\gamma_{2_l}})$$

$\forall O_{\gamma_{1_k}}$ perteneciente a la descomposición de O_{α_1} y $\forall O_{\gamma_{2_l}}$ perteneciente a la descomposición de O_{α_2}.

Como la intersección entre modelos volumétricos de propiedad con propiedades de la base es un modelo volumétrico de propiedad con propiedad de la base, o nada si las propiedades a intersectar no son la misma, queda hecha la demostración. ◇

Para definir la diferencia entre modelos volumétricos discretos vamos a demostrar antes que la diferencia de modelos volumétricos de propiedad da como resultado un modelo volumétrico discreto, y que la diferencia entre un modelo volumétrico discreto y un modelo volumétrico de propiedad da como resultado un modelo volumétrico discreto.

Lema 2.24

La diferencia de modelos volumétricos de propiedad da como resultado un modelo volumétrico discreto.

Dem:

Sean O_{γ_1} *y* O_{γ_2} *dos modelos volumétricos de propiedad definidos respectivamente como* $\alpha_1 : V_1 \to \Gamma$ *y* $\alpha_2 : V_2 \to \Gamma$

La diferencia entre ambos según está definida en la página 56 es $O_\alpha = O_{\gamma_1} - O_{\gamma_2}$ *donde*

$$\alpha(p) = \begin{cases} \gamma_1 & si \quad p \notin i(V_2) \\ \gamma_1 - \gamma_2 & si \quad p \in i(V_2) \end{cases}$$

O_α *se puede expresar como* $O_\alpha = O_{\alpha_a} \cup O_{\alpha_b}$ *donde* $\alpha_a : V_1 - V_2 \to \Gamma$ *con* $\alpha_a(p) = \gamma_1$ *y* $\alpha_b : V_1 \cap V_2 \to \Gamma$ *con* $\alpha_a(p) = \gamma_1 - \gamma_2$ *y por tanto es un modelo volumétrico discreto.* ◇

Lema 2.25

La diferencia entre un modelo volumétrico discreto y un modelo volumétrico de propiedad es un modelo volumétrico discreto.

Dem:

Sea O_{α_1} *un modelo volumétrico discreto y sea* O_γ *un modelo volumétrico de propiedad, la diferencia de ambos es*

$$O_\alpha = O_{\alpha_1} - O_\gamma$$

Si O_{α_1} es un modelo volumétrico discreto existen

$$O_{\gamma_{1_i}}, \ldots, O_{\gamma_{1_j}}$$

modelos volumétricos de propiedad tales que

$$\{\gamma_{1_i}, \ldots, \gamma_{1_j}\} \subseteq \Gamma_b$$

siendo Γ_b una base booleana de Γ y

$$O_{\alpha_1} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}$$

Por lo tanto podemos expresar

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) - O_\gamma$$

Sustituyendo la diferencia por la intersección con el complementario tenemos

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap \sim O_\gamma$$

Aplicando la propiedad distributiva tenemos

$$O_\alpha = \bigcup (O_{\gamma_{1_k}} \cap \sim O_\gamma)$$

Sustituyendo la intersección con el complementario por la diferencia tenemos

$$O_\alpha = \bigcup (O_{\gamma_{1_k}} - O_\gamma)$$

Quedando O_α definido mediante la union de modelos volumétricos discretos, y por tanto siendo un modelo volumétrico discreto. ◇

Lema 2.26

La diferencia de modelos volumétricos discretos es un modelo volumétrico discreto.

Dem:

Dados O_{α_1} y O_{α_2} dos modelos volumétricos discretos, la diferencia entre ambos es

$$O_\alpha = O_{\alpha_1} - O_{\alpha_2}$$

Si O_{α_1} es un modelo volumétrico discreto existen

$$O_{\gamma_{1_i}}, \ldots, O_{\gamma_{1_j}}$$

modelos volumétricos de propiedad tales que

$$\{\gamma_{1_i}, \ldots, \gamma_{1_j}\} \subseteq \Gamma_b$$

siendo Γ_b una base booleana de Γ y

$$O_{\alpha_1} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}$$

Con respecto a O_{α_2} ocurre de forma análoga. Por lo tanto podemos expresar

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) - (O_{\gamma_{2_i}} \cup \ldots \cup O_{\gamma_{2_j}})$$

Sustituyendo la diferencia por la intersección con el complementario tenemos

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap \sim (O_{\gamma_{2_i}} \cup \ldots \cup O_{\gamma_{2_j}})$$

El complementario de una unión es la intersección de los complementarios

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap (\sim O_{\gamma_{2_i}} \cap \ldots \cap \sim O_{\gamma_{2_j}})$$

Aplicando la propiedad asociativa tenemos

$$O_\alpha = ((O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap \sim O_{\gamma_{2_i}}) \cap (\sim O_{\gamma_{2_k}} \cap \ldots \cap \sim O_{\gamma_{2_j}})$$

Sustituyendo en el paréntesis de la izquierda la intersección con un complementario por la diferencia tenemos

$$O_\alpha = ((O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) - O_{\gamma_{2_i}}) \cap (\sim O_{\gamma_{2_k}} \cap \ldots \cap \sim O_{\gamma_{2_j}})$$

Es decir, en el paréntesis de la izquierda tenemos la diferencia entre un modelo volumétrico discreto y un modelo volumétrico de propiedad, que sabemos que es un modelo volumétrico discreto. Repitiendo el proceso mediante la propiedad asociativa con los otros modelos volumétricos de propiedad pertenecientes a O_{α_2} vamos realizando las diferencias una a una obteniendo como resultado un modelo volumétrico discreto. ◇

2.4.4. Álgebra de boole de modelos volumétricos discretos

Pasemos a demostrar que los modelos volumétricos discretos junto con las operaciones definidas anteriormente constituyen un álgebra de boole.

Teorema 2.27 (Álgebra de boole de modelos volumétricos discretos) *La cuádrupla* $(O^n_{\Gamma_d}, \cup, \cap, \sim)$ *es un álgebra de boole.*
Dem:
Hemos demostrado que las operaciones son cerradas, habría que demostrar que cumplen las propiedades de las álgebras de boole. Estas propiedades las cumplen ya que los modelos volumétricos discretos se basan en modelos volumétricos de propiedad y hemos visto que éstos cumplen dichas propiedades. ◇

2.4.5. Aplicaciones

Tras describir el marco formal que soporta a los modelos volumétricos discretos pasemos a describir un esquema de representación que se base en ellos.

A modo de ilustración nos vamos a centrar en el esquema de representación CSG (Constructive Solid Geometric) [Requicha, 82]. Este esquema de representación de sólidos se basa en construir el sólido mediante operaciones booleanas regularizadas a partir de primitivas (figura 2.3), estando éstas generalmente representadas como intersección de semiespacios.

Se trata de un esquema de representación bastante usado en el modelado de sólidos [Requicha, 80] por su poder expresivo, validez, no ambigüedad, facilidad de uso mediante lenguajes, es conciso y cerrado con operaciones booleanas. Entre sus desventajas se encuentra el no ser un esquema de representación único, que depende del conjunto de primitivas del que se disponga y algunos algoritmos como la evaluación de la frontera es costoso [Mäntylä, 88].

Sin embargo, considera que los sólidos poseen un único valor de propiedad repartido de forma homogénea a lo largo del sólido, sólidos homogéneos. Con el marco formal presentado podemos modelar sólidos que presenten distintos valores de un dominio de propiedad discreto. Podemos modelar volúmenes discretos.

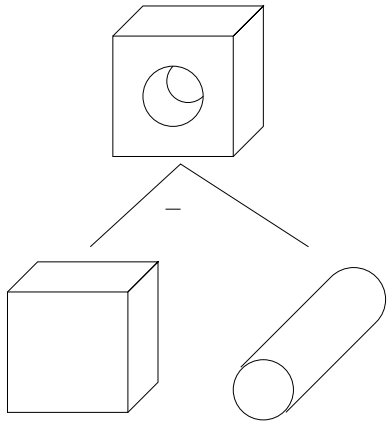


Figura 2.3: Árbol CSG de un sólido

Para ello, a las primitivas CSG hay que asignarles una propiedad del dominio de propiedades, que no necesariamente debe ser de la base booleana de propiedades, ya que una primitiva con propiedad no perteneciente a la base se puede obtener como la unión de primitivas con propiedades de la base.

Los volúmenes discretos los obtenemos operando a partir de dichas primitivas con las operaciones unión, intersección y diferencia definidas para los modelos volumétricos discretos.

Un árbol CSG discreto queda definido como sigue:

<árbol CSG discreto> ::=
<primitiva con propiedad> |
<árbol CSG discreto><operación><árbol CSG discreto> |
<árbol CSG discreto><transformación rígida>

En la figura 2.4 se muestra un volumen discreto modelado mediante CSG discreto.

2.5. Modelos volumétricos continuos

Los modelos volumétricos continuos se caracterizan por aplicarse sobre un dominio de propiedad continuo.

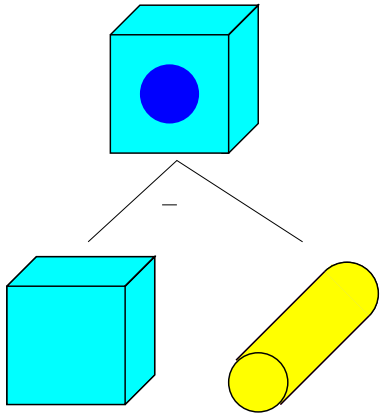


Figura 2.4: CSG discreto correspondiente a un volumen discreto

Definición 2.28 (Modelo volumétrico continuo)
Un modelo volumétrico O_α *con* $\alpha : V \subset \mathbb{R}^n \to \Gamma$ *será un modelo volumétrico continuo sii* Γ *es un dominio de propiedad continuo.*

Al conjunto de todos los modelos volumétricos continuos que podemos definir desde $\mathbb{R}^n$ *sobre el dominio de propiedad* Γ *lo notaremos como* $O^n_{\Gamma_c}$. ◇

Esta definición de modelo volumétrico continuo que no le exige a α ser una función continua nos va a permitir caracterizar modelos volumétricos del cuerpo humano donde una posible propiedad representada como es la densidad puede sufrir un cambio brusco de valor al pasar, por ejemplo, de un tejido blando a hueso.

2.5.1. Álgebra de modelos volumétricos continuos

Sobre el conjunto de todos los modelos volumétricos continuos se pueden definir operaciones internas como $+_\cup$, $\times_\cup$ y externas como $*$ sobre un cuerpo externo K, aprovechando las operaciones que tiene definidas el dominio de propiedad. La definición de dichas operaciones se realiza como se ha mostrado en la sección 2.3.2 (pág. 54) .

Definición 2.29 (Elemento neutro para $+_\cup$)
La clase de equivalencia neutra para la operación $+_\cup$ es aquella que contiene al modelo volumétrico O_α tal que $\alpha : \mathbb{R}^n \to \Gamma$ con $\alpha(p) = 0 \ \ \forall p$ donde 0 es el elemento neutro para la operación $+$ en el dominio de propiedad.
Lo notamos como O_0. ◇

Definición 2.30 (Elemento opuesto para $+_\cup$)
Dado un objeto volumétrico O_α, se define el elemento opuesto para $+_\cup$ como O_{α^-} donde $\alpha^-(p) = -\alpha(p) \ \ \forall p$. Representando $-\gamma$ al elemento opuesto a γ según la operación $+$ en el dominio de propiedad Γ. ◇

Teorema 2.31
La dupla $(O^n_{\Gamma_c}, +_\cup)$ tiene estructura de grupo aditivo.
Dem:
Las propiedades asociativa y conmutativa las cumple al basarse $+_\cup$ en las operaciones $\cup$ sobre $\mathbb{R}^n$ y $+$ sobre Γ que cumplen dichas propiedades. Veamos que posee elemento neutro y opuesto.

Dado O_{α_1} un modelo volumétrico donde $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma$ la operación
$O_{\alpha_1} +_\cup O_0 = O_\alpha$ *donde*
$$\alpha(p) = \begin{cases} 0 & si \quad p \notin V_1 \\ \alpha_1(p) + 0 & si \quad p \in V_1 \end{cases}$$
Siendo $O_\alpha \circ O_{\alpha_1}$.

$O_{\alpha_1} +_\cup O_{\alpha_1^-} = O_\alpha$ *donde*
$\alpha(p) = \alpha(p) - \alpha(p) = 0$ *siendo por tanto $O_\alpha \circ O_0$.*

Teorema 2.32
Dado un cuerpo K y la operación externa $*$ sobre K, la tupla $(O^n_{\Gamma_c}, +_\cup, *)$ tiene estructura de K-espacio vectorial.

Dem:
Se ha demostrado que $(O^n_{\Gamma_c}, +_\cup)$ *tiene estructura de grupo aditivo. Por otro lado, según la definición de operación binaria sobre un cuerpo externo realizada en la sección 2.3.2 (pág. 54) se observa que la operación* $*$ *sobre* O_Γ *y* K *se define en base a la operación* $*$ *sobre* Γ *y* K*. Como* $(\Gamma, +, *)$ *tiene estructura de K-espacio vectorial, se demuestra que la operación* $*$ *sobre* $O^n_{\Gamma_c}$ *y* K *cumple las propiedades que dotan a la tupla citada la estructura de K-espacio vectorial.*

Definición 2.33 (Elemento identidad para $\times_\cup$)
La clase de equivalencia identidad para la operación $\times_\cup$ *es aquella que contiene al modelo volumétrico* O_α *tal que* $\alpha : \mathbb{R}^n \to \Gamma$ *con* $\alpha(p) = 1 \ \ \forall p$ *donde* 1 *es el elemento identidad para* $\times$ *en el dominio de propiedad.*
Lo notamos como O_1. ◇

Teorema 2.34
El producto interno $\times_\cup$ *sobre* $O^n_{\Gamma_c}$ *cumple las propiedades asociativa, conmutativa y existencia de elemento identidad.*
Dem:
Las propiedades asociativa y conmutativa las cumple al basarse $\times_\cup$ *en las operaciones* $\cup$ *sobre* $\mathbb{R}^n$ *y* $\times$ *sobre* Γ *que cumplen dichas propiedades. Veamos que posee elemento identidad.*

Dado O_{α_1} *un modelo volumétrico donde* $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma$ *la operación*
$O_{\alpha_1} \times_\cup O_1 = O_\alpha$ *donde*

$$\alpha(p) = \begin{cases} 0 \times 1 = 0 & si \quad p \notin V_1 \\ \alpha_1(p) \times 1 = \alpha_1(p) & si \quad p \in V_1 \end{cases}$$

Siendo $O_\alpha \circ O_{\alpha_1}$.

2.5.2. Estimación de modelos volumétricos continuos

En muchas ocasiones la función α no es conocida en su totalidad, solo se tiene su definición en un conjunto finito de muestras V. Por ejemplo, la información que se pueda obtener de un paciente a partir de una tomografía axial computerizada.

En estos casos, la función

$$\alpha : V \subset \mathbb{R}^n \rightarrow \Gamma \tag{2.2}$$

no es un modelo volumétrico pues V no cumple con la condición de ser regular. Necesitamos por tanto poder definir un modelo volumétrico a partir de la función α disponible tras realizar el muestreo.

A partir de V se puede definir su envolvente convexa que notamos como $\overline{V}$, y que cumple las condiciones de ser un conjunto cerrado, acotado, rígido y regular. Por tanto podemos definir un modelo volumétrico definiendo una función

$$\overline{\alpha} : \overline{V} \subset \mathbb{R}^n \rightarrow \Gamma \tag{2.3}$$

que cumpla la condición de que $\overline{\alpha}(p) = \alpha(p)\ \forall p \in V$

La función $\overline{\alpha}$ suele ser una función de interpolación para estimar a partir de los valores conocidos de V los valores de propiedad en las posiciones no muestreadas. Siendo la *interpolación n-lineal* la función más usada.

2.5.3. Simplificación de modelos volumétricos continuos estimados

Pensando en términos de implementación y eficiencia con respecto a un modelo volumétrico continuo estimado, puede ser necesario reducir la cardinalidad de V de cara a disminuir las necesidades de almacenamiento.

A partir de V se puede obtener un subconjunto $W \subset V$ que al tener una cardinalidad inferior va a necesitar menos requerimientos de memoria. La condición que debe cumplir W es que $\overline{W} = \overline{V}$, es decir, coinciden las envolventes convexas, con lo que una vez se ha definido

$$\overline{\alpha}' : \overline{W} \subset \mathbb{R}^n \rightarrow \Gamma \tag{2.4}$$

los modelos volumétricos $O_{\overline{\alpha}}$ y $O_{\overline{\alpha}'}$ están representando la misma porción de $\mathbb{R}^n$.

Al realizar esta operación se reducen las necesidades de almacenamiento pero al mismo tiempo se comete un error de representación ya que las posiciones $p \in V - W$

se están representando con un valor de propiedad estimado por $\overline{\alpha}'(p)$ en vez de con el valor real medido por $\alpha(p)$. Ese error se puede medir por ejemplo con la expresión

$$error = \frac{1}{Card(V)} \sum_{p \in V} |\overline{\alpha}'(p) - \alpha(p)| \tag{2.5}$$

Si la función de estimación es la interpolación n-lineal, sería conveniente que W contenga todos los máximos y mínimos locales de V ya que en estas posiciones es donde se pueden producir los mayores errores en el caso de no pertenecer a W.

2.5.4. Transformación de modelos volumétricos continuos en discretos

Gracias a las funciones de proyección de dominios de propiedad continuos en discretos vamos a poder transformar los modelos volumétricos continuos en discretos.

Sean Γ_c un dominio de propiedad continuo, $\Gamma_d \subset \Gamma_c$ un dominio de propiedad discreto, Γ_{db} su base booleana de propiedades y $P : \Gamma_c \to \Gamma_d$ una proyección de Γ_c en Γ_d.

Sea $\alpha_c : V \subset \mathbb{R}^n \to \Gamma_c$ un modelo volumétrico continuo. Se podría definir una función

$$\alpha_d : V \subset \mathbb{R}^n \to \Gamma_d$$

como

$$\alpha_d(p) = P(\alpha_c(p)) \ \ \forall p \in V$$

Pero no podríamos afirmar que dicha función fuera un modelo volumétrico discreto ya que con la información disponible no podemos afirmar que se pueda obtener por unión de modelos volumétricos de propiedad. Pero si usaremos la función α_d como paso intermedio para construir el modelo volumétrico discreto resultado de la discretización de O_{α_c}.

Se construyen sendos $V_i = \{p \in V : \alpha_d(p) \cap \gamma_i = \gamma_i\} \ \ \forall \gamma_i \in \Gamma_{cb}$.

Se regulariza cada V_i obteniendo sendos conjuntos V_i', con los que se pueden definir

$$\alpha_i' : V_i' \subset \mathbb{R}^n \to \Gamma_d$$

tal que

$$\alpha_i'(p) = \gamma_i \ \ \forall p \in V_i'$$

de tal modo que los modelos $O_{\alpha'_i}$ son modelos volumétricos de propiedad sobre las propiedades de la base booleana de Γ_d.

Por tanto,

$$O'_{\alpha} = \bigcup_i O_{\alpha'_i}$$

es un modelo volumétrico discreto obtenido como transformación del modelo volumétrico continuo O_{α_c} según la proyección de propiedades P.

2.6. Conclusiones

En este capítulo se ha realizado una formalización del concepto de modelo volumétrico como una aplicación de un subcojunto de $\mathbb{R}^n$ en un dominio de propiedades.

Se ha formalizado el dominio de propiedades realizando una clasificación de los mismos en discretos y continuos y presentando modos de composición de dominios de propiedad para obtener otros nuevos.

Esta clasificación se ha extendido a los modelos volumétricos realizando una clasificación, caracterización y formalización de los mismos en discretos y continuos.

Se ha definido un álgebra de boole de modelos volumétricos discretos presentando un esquema de representación basado en dicha estructura denominado CSG discreto.

Así mismo se presenta un método de aproximación de modelos volumétricos continuos por discretos.

Capítulo 3

Octree de celdas

Aunque le arranques los pétalos,
no quitaras su belleza a la flor
Rabindranath Tagore (1861-1941)

Como hemos indicado, una representación típica de un volumen es mediante un conjunto de pares $(punto, valor\ de\ propiedad)$, siendo una distribución usual de estos puntos una rejilla estructurada, rectilínea, regular e isotrópica. Para estimar los valores en las posiciones no representadas se suele usar una interpolación trilineal $F(x,y,z)$ (ver ecuación 1.3); a esta expresión haremos referencia a lo largo de este capítulo.

Cada vez se trabaja con volúmenes de mayor tamaño y por tanto el tamaño de la rejilla es cada vez mayor. En este capítulo mostraremos una forma de agrupación de los puntos de la rejilla que permite tener representado el volumen con un número menor de celdas.

3.1. Introducción

Nuestra idea se basa en estudiar las celdas, y allí donde la propiedad presenta un gradiente uniforme, agrupar 8 celdas en una sola. Y hacer esto de una manera jerárquica a modo de octree, en lo que hemos dado en llamar *octree de celdas* [Velasco, 01a, Velasco, 01b].

> Cuando la celda almacena un solo valor de propiedad para toda la celda se le suele llamar vóxel. Los octrees *clásicos* [Meagher, 80] se basan en vóxeles, en contraposición a la estructura que usamos en este capítulo en la que las celdas que componen el octree y que denominamos *celdas* almacenan 8 valores de propiedad, uno por vértice de la celda.

Tomamos como punto de partida la estructura bono [Wilhelms, 92] por las ventajas que ofrece. La estructura bono es un índice jerárquico de acceso a las celdas de la rejilla implementado como un árbol octal.

Cada nodo del árbol almacena el valor de propiedad máximo y mínimo del subvolumen al que da acceso. De esta forma, cuando se recorre con el valor umbral como clave de búsqueda para localizar celdas activas, se pueden descartar las ramas cuyo intervalo *[propiedadMínima,PropiedadMáxima]* no contiene al umbral.

Si la rejilla tiene forma cúbica y un tamaño por dimensión igual a una potencia de 2, las subdivisiones en la subrejilla representada por un nodo interno se realizan por la mitad obteniendo las 8 subrejillas iguales en tamaño hasta llegar a las hojas del árbol. Sin embargo, cuando el tamaño de la rejilla no es potencia de 2 en alguna de sus dimensiones, o cuando los tamaños de las tres dimensiones no son iguales, las subdivisiones por la mitad no van a proporcionar subrejillas iguales entre sí siempre. Wilhelms propone realizar las subdivisiones situando los 3 planos de corte a una distancia d del origen de la subrejilla definida por la dimensión mayor D de la misma según la siguiente expresión:

$$d = \min\{2^i \geq D/2 : i \in \mathbb{N}\} \tag{3.1}$$

de este modo puede ocurrir que algún plano de corte se sitúe fuera de la subrejilla con lo que no la va a dividir. De este modo, un nodo puede tener 1, 2, 4 u 8 hijos. Esto en realidad es una ventaja pues produce un árbol con una mayor ramificación en la parte inferior (las hojas) que en la parte de la raíz obteniendo un árbol con menor número de nodos. En la figura 3.1 se muestra un ejemplo 2D con la subdivisión bono (caso (a))y la subdivisión por la mitad *por defecto* (caso (b)). Los números a los márgenes indican el número de nodos internos de cada nivel.

De cara a reducir el espacio de almacenamiento necesario para el árbol, las subdivisiones se detienen un nivel antes de que las hojas representen a una sola celda. Un nodo hoja puede dar acceso hasta a 8 celdas. A cambio de este ahorro de espacio, hay celdas no activas que deben clasificarse.

Otra característica de los árboles bono es que todas las hojas están en el mismo nivel, aunque un nodo interno represente a una sola celda, no *derivará* en nodo hoja hasta que se llegue al nivel de las hojas (ver figura 3.2).

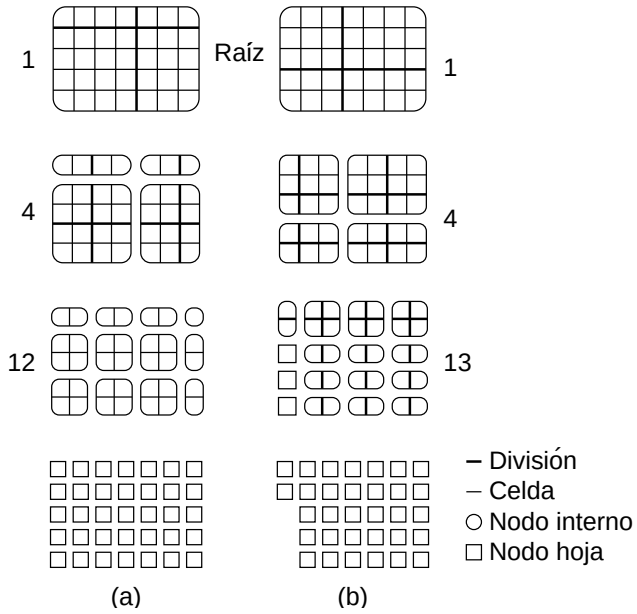


Figura 3.1: Subdivisión bono vs. subdivision por la mitad

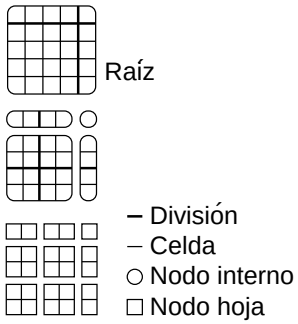


Figura 3.2: Bono con todas las hojas al mismo nivel

3.2. Octree de celdas

Con las agrupaciones que proponemos, el árbol resultante tendrá hojas en distintos niveles y por tanto representarán a celdas de distinto tamaño, las cuales podrán estar adosadas entre sí. Seguimos manteniendo el criterio de que un nodo hoja pueda representar hasta a 8 celdas. Puede haber nodos hoja que representen a una sola hoja, que denominaremos `nodoHojaUna` y nodos hoja que representen a más de una hoja, los cuales denominaremos `nodoHojaMás`.

En la figura 3.3 mostramos un ejemplo 2D de lo que sería un octree de celdas (quadtree de celdas).

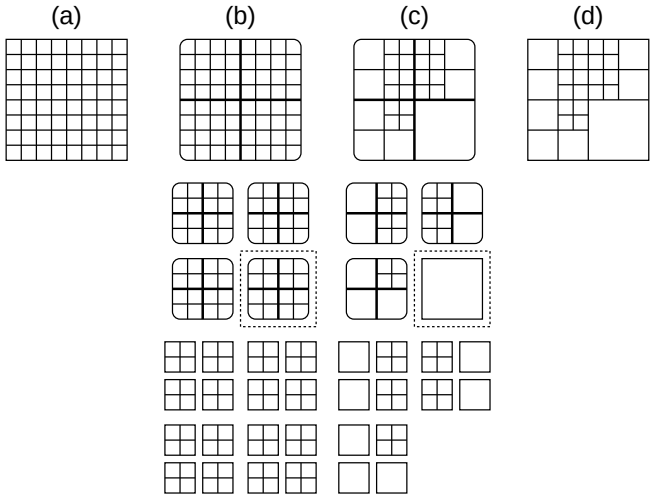


Figura 3.3: Quadtree de celdas

El gráfico (a) de dicha figura muestra la rejilla original, el bono que se obtiene de dicha rejilla se muestra en el gráfico (b). Tras realizar una posible agrupación obtenemos el árbol que se muestra en el gráfico (c). El gráfico (d) muestra la rejilla real a la que dicho árbol da acceso.

Como se observa en el último nivel del arbol (c) de la figura 3.3 se tienen 7 nodos hoja del tipo `nodoHojaUna` y 5 nodos del tipo `nodoHojaMás` que dan acceso a 20 celdas.

Aunque hablamos de agrupación de celdas, en realidad lo que se realiza es una poda en el árbol. Se sigue un proceso ascendente desde el bono de donde partimos, y cada vez que *8 celdas hermanas* cumplen las condiciones para ser agrupadas, el `nodoHojaMás` que las representa se convierte en un `nodoHojaUna` que representa a una sola celda con tamaño de arista doble. Cuando se obtienen con este proceso 8 nodos del tipo `nodoHojaUna` hermanos que representan cada uno de ellos a una sola celda, se podan convirtiendo el nodo interno padre de dichos nodos en un nodo hoja que representa o bien a 8 celdas si estas no han cumplido las condiciones para ser agrupadas, o bien a una celda si las 8 celdas pueden agruparse.

La figura 3.4 muestra el proceso completo paso a paso para el subárbol encerrado en línea de puntos en la figura 3.3.

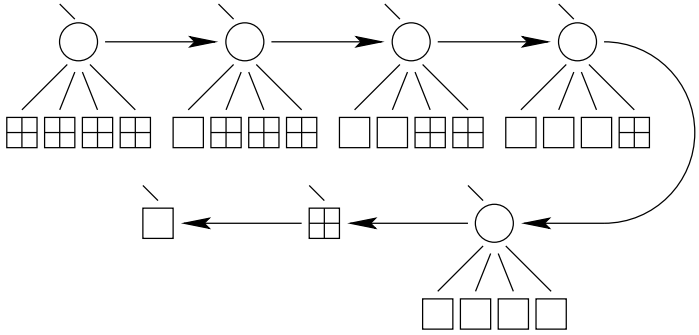


Figura 3.4: Proceso de agrupación / poda

Por tanto la rejilla original no se ve modificada, solo el árbol que da acceso a los valores de la rejilla.

3.2.1. Criterio de agrupación básico

La idea es que la rejilla representada por el octree de celdas se adapte al volumen de modo que presente celdas de mayor tamaño donde el gradiente sea uniforme y celdas a una mayor resolución donde los valores del volumen así lo requieran.

Otro objetivo perseguido es que la agrupación/poda sea independiente del valor umbral que se use para la visualización.

Para ello se ha analizado como se comporta la función estimada $F(x,y,z)$ en las caras de las celdas candidatas a ser agrupadas y en las caras de la celda resultante si la agrupación se lleva a cabo. Además hay que tener en cuenta que cuando se produce una agrupación hay valores a los que ya no se va a acceder cuando se procese la celda agrupada. En concreto los valores que estaban situados en el centro de una arista o en el centro de una cara de una celda agrupada ya no son accesibles por ésta, pero pueden ser usados por las celdas adyacentes si éstas no se han agrupado al mismo nivel. En la figura 3.5 algunos de dichos valores están marcados y pueden ser los responsables de rupturas en la visualización del volumen.

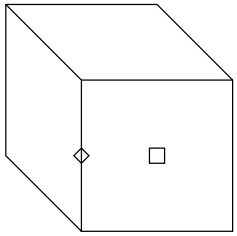


Figura 3.5: Valores no accesibles tras una agrupación

Comencemos a analizar los casos que pueden darse en la visualización por extracción de isosuperficie y que queremos evitar para que no se produzcan rupturas. Con el análisis de dichos casos extraeremos las condiciones que formarán el criterio de agrupación que evita las situaciones no deseadas.

Si el valor de propiedad en el centro de una arista de la celda agrupada es mayor que los valores existentes en los extremos de dicha arista o menor que ambos se puede producir alguna situación como la mostrada en la figura 3.6.

En dicha imagen se observa como para un determinado valor umbral la isosuperficie asociada no tiene continuidad en el interior de la celda agrupada, puesto que en dicha celda sólo son visibles los extremos de la arista y éstos son del mismo signo. En el caso concreto mostrado, las aristas marcadas en rojo pertenecen a la ruptura.

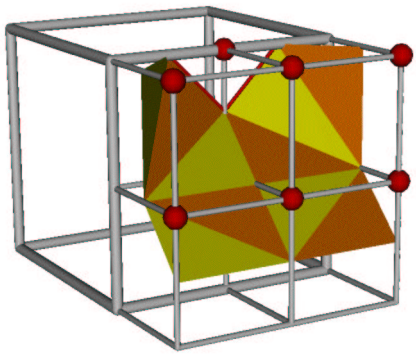


Figura 3.6: Situación no deseada tras una agrupación

Para evitar estas situaciones se le va a exigir a los puntos que van a quedar situados en el centro de una arista tras la agrupación que tengan un valor de propiedad comprendido entre los valores de propiedad de los extremos de dicha arista.

En concreto (véase la figura 3.7) para cada cara de la celda que se va a formar se debe cumplir que:

$$\begin{aligned}
&[(F(v_1) \leq F(v_2) \leq F(v_3)) \quad \vee \quad (F(v_1) \geq F(v_2) \geq F(v_3))] \quad \wedge \\
&[(F(v_1) \leq F(v_4) \leq F(v_7)) \quad \vee \quad (F(v_1) \geq F(v_4) \geq F(v_7))] \quad \wedge \\
&[(F(v_3) \leq F(v_6) \leq F(v_9)) \quad \vee \quad (F(v_3) \geq F(v_6) \geq F(v_9))] \quad \wedge \\
&[(F(v_7) \leq F(v_8) \leq F(v_9)) \quad \vee \quad (F(v_7) \geq F(v_8) \geq F(v_9))]
\end{aligned} \tag{3.2}$$

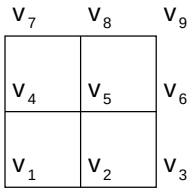


Figura 3.7: Cuatro caras coplanarias candidatas a ser agrupadas

En la figura 3.7 está representada la frontera entre una celda agrupada, formada por los vértices v_1, v_3, v_7 y v_9; y cuatro celdas adyacentes

de tamaño de arista mitad. Este tipo de diagrama lo usaremos con frecuencia en este capítulo.

Por tanto, si una isosuperficie que estando en el exterior de una celda agrupada corta a una arista, va a existir isosuperficie en el interior de la celda agrupada que corta a la misma arista.

De manera similar va a ocurrir con la posición en el centro de una cara de una celda agrupada (véase la figura 3.8).

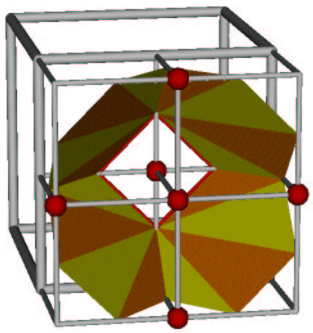


Figura 3.8: Situación no deseada tras una agrupación (2)

En este caso, para un determinado valor umbral existe una isosuperficie en el exterior de una celda agrupada que corta una cara de ésta no existiendo isosuperficie en el interior para continuarla, ya que los cuatro vértices de dicha cara tienen el mismo signo.

Al vértice central de una cara antes de agruparla le vamos a exigir que su valor de propiedad esté comprendido entre los valores de propiedad que poseen los puntos centrales de las aristas de dicha cara. Es decir:

$$\begin{aligned}&[(F(v_2) \leq F(v_5) \leq F(v_8)) \quad \vee \quad (F(v_2) \geq F(v_5) \geq F(v_8))] \quad \wedge \\ &[(F(v_4) \leq F(v_5) \leq F(v_6)) \quad \vee \quad (F(v_4) \geq F(v_5) \geq F(v_6))] \end{aligned} \tag{3.3}$$

De estas condiciones se deduce que de los nueve vértices que forman una cara antes de agruparla, los valores de propiedad máximo y mínimo van a estar situados en las esquinas de la cara que resulta después de la agrupación.

Otra situación que evitaría que se hiciera una agrupación sería la mostrada en la figura 3.9 que corresponde a que el punto central de las 8 celdas posee un valor de propiedad que es máximo o mínimo entre los 27 puntos que intervienen en el global de las 8 celdas.

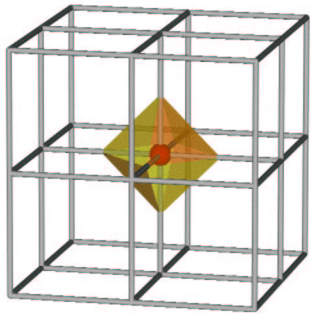


Figura 3.9: Situación que evita una agrupación

En este caso si se realizara la agrupación la componente de isosuperficie que aparece en el centro se perdería.

Por último, supongamos una rejilla de $n \times n \times n$ celdas de tamaño unidad, situada en una posición tal que en sucesivas agrupaciones puede llegar a ser una celda de tamaño n. Supongamos que un vértice situado en una esquina del grupo tiene un valor de propiedad, por ejemplo de 100, mientras que el resto de vértices del grupo tienen como valor de propiedad el 0.

Antes de realizar la agrupación, una isosuperficie creada para un valor umbral de 50 cortaría a la celda unidad situada en la esquina del grupo. Si agrupamos todas las celdas para obtener una única celda de longitud n, la isosuperficie asociada al valor 50 se situaría a una distancia considerable con respecto a la isosuperfice que pudiera resultar antes de la agrupación. Este error se puede disminuir evitando las agrupaciones que hagan que en los vértices que dejan de ser accesibles, el valor estimado de propiedad difiera del original en más de una cantidad indicada por el usuario.

Uniendo los casos mostrados surge el siguiente criterio.

Criterio de agrupación básico

Ocho celdas podrán ser agrupadas en una si:

- Para cada arista de la posible celda agrupada, el valor de propiedad del vértice central está comprendido entre los valores de propiedad de los extremos de la arista. Es decir, para cada cara se cumple la ecuación 3.2.
- Para cada cara de la posible celda agrupada, se cumple la ecuación 3.3.
- El valor de propiedad del vértice central de la posible celda agrupada no es mayor que el máximo valor entre los 8 puntos situados en las esquinas del grupo ni es menor que el mínimo.
- El valor de propiedad que se pueda estimar en los vértices que dejan de ser accesibles tras la agrupación, no difiere del valor original en más de una cantidad establecida a priori.

3.2.2. Concepto de monotonía

Otro factor a tener en cuenta para definir un criterio de agrupación es la ambigüedad, si la celda resultado de una agrupación posee alguna cara ambigua o el interior ambiguo se puede decidir no agruparla para evitar tener una celda ambigua que antes no existía. Por tanto, incluir esta condición en el criterio de agrupación lo hará más restrictivo produciéndose menos agrupaciones.

Con los trabajos de [Chernyaev, 95], [Cignoni, 00] y [Lopes, 02] no hay ningún problema en tener celdas ambiguas pues extienden el número de casos distintos del marching cubes considerando distintas triangulaciones para las celdas que presentan ambigüedades siendo además posible determinar el caso correcto con cálculos a nivel local. Queda por tanto a decisión del usuario el realizar estas agrupaciones para reducir el número de celdas o no hacerlas para reducir el número de cálculos.

Para caracterizar la ambigüedad de una celda definimos el concepto de monotonía. Una cara será monótona si para cualquier valor umbral la isosuperficie asociada no corta a la cara más de una vez. De igual modo, una celda será monótona si para cualquier valor umbral la isosuperficie asociada no corta a la celda más de una vez.

Definición 3.1 (Cara monótona)
Una cara c de una celda decimos que es monótona sii $F_{|\mathcal{F}(c)}(x,y,z)$ *sólo tiene un mínimo y un máximo.*
Siendo $\mathcal{F}(c)$ *la frontera de la cara c.* ◇

Teorema 3.2
Sea c una cara de una celda
c monótona $\Rightarrow \nexists\gamma \in \Gamma$ *tal que c sea ambigua.*
Dem:
Supongamos que $\exists\gamma \in \Gamma$ *tal que c es ambigua. Es decir dos vértices diagonalmente opuestos de la cara c son mayores o iguales a* γ *y los otros dos son menores. Es decir, la función* $F_{|\mathcal{F}(c)}(x,y,z)$ *tiene dos máximos y dos mínimos y por tanto c no sería monótona.* ◇

Con el concepto de cara monótona y con el concepto de vértice potencialmente cortado definiremos el concepto de celda monótona. Los vértices potencialmente cortados serán aquellos para los que existe al menos un valor umbral cuya isosuperficie asociada va a dejar a ese vértice a un lado de la misma, situándose los otros 7 al otro lado de la isosuperficie.

Definición 3.3 (Vértice potencialmente cortado)
Un vértice v_0 *de una celda se denomina vértice potencialmente cortado si se cumple una de las dos siguientes expresiones (véase la figura 3.10*(a)*):*
1. $F(v_0) < \min\{F(v_1),F(v_2),F(v_3)\}$
2. $F(v_0) > \max\{F(v_1),F(v_2),F(v_3)\}$
donde v_1*,* v_2 *y* v_3 *son los vértices que comparten arista con* v_0*.*

Un vértice potencialmente cortado tiene asociado un intervalo que es

$$[F(v_0), \min\{F(v_1),F(v_2),F(v_3)\}] \tag{3.4}$$

si se cumple la expresión 1. o

$$[\max\{F(v_1),F(v_2),F(v_3)\}, F(v_0)] \tag{3.5}$$

si se cumple la expresión 2. ◇

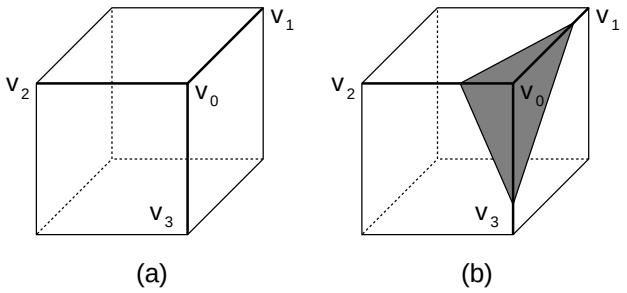


Figura 3.10: Vértices potencialmente cortados

Teorema 3.4
Una celda con un vértice potencialmente cortado será cortada por toda isosuperficie correspondiente a un valor umbral que pertenezca al intervalo asociado a dicho vértice. La isosuperfice dejará ese vértice a un lado distinto de donde van a estar situados los vértices inmediatos. Véase la figura 3.10(b).
Dem:
Trivial, pues todos los valores que pertenezcan al intervalo asociado van a hacer que v_0 *tenga un signo distinto al de* v_1, v_2 *y* v_3. ◇

Definición 3.5 (Diagonal no monótona)
Dada una celda c diremos que una diagonal de ésta, d, es no monótona si los vértices extremos de la diagonal son vértices potencialmente cortados y además la intersección entre sus intervalos asociados es no vacía.

Toda diagonal que no pueda clasificarse como no monótona *será clasificada como* monótona. ◇

Definición 3.6 (Celda monótona)
Una celda será monótona si sus 6 caras son monótonas y sus 4 diagonales son monótonas. ◇

Teorema 3.7

Sea c una celda, c monótona $\Rightarrow \nexists \gamma \in \Gamma$ *tal que c sea ambigua.*

Dem:

Por ser c monótona sus seis caras son monótonas y por tanto ningún valor umbral va a hacer que alguna de sus caras sea ambigua.

Por otro lado, al no tener diagonales no monótonas no va a tener vértices potencialmente cortados situados diagonalmente opuestos con intersección no vacía de los intervalos asociados, y por tanto no va a existir ningún valor umbral que haga de un mismo signo dos vértices diagonalmente opuestos manteniendo los adyacentes a éstos con el signo opuesto. ◇

Por tanto, en el caso de que se quiera que las celdas resultantes de una agrupación no sean ambiguas, al criterio de agrupación básico hay que añadirle la condición de que la celda resultante sea monótona.

Criterio de agrupación de monotonía

Ocho celdas podrán ser agrupadas en una si:

- Cumplen el criterio de agrupación básico.
- La celda resultante de la agrupación es monótona.

3.3. Agujeros en la isosuperficie asociados a un octree de celdas

Cuando se dan casos de celdas adosadas de distinto tamaño como el caso mostrado en la figura 3.11 se pueden producir agujeros en la isosuperficie justo en la frontera existente entre la celda de mayor tamaño y las celdas adosadas de menor tamaño.

Ello es debido a que en el lado de la celda grande la isosuperficie que atraviesa esa cara frontera se calcula a partir de 4 puntos mientras que en el lado de las celdas pequeñas la continuación de esa isosuperficie se calcula a partir de 9 puntos, dándose situaciones como la mostrada en la figura 3.12.

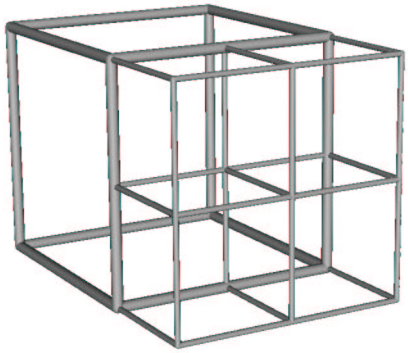


Figura 3.11: Celdas adosadas de diferente tamaño

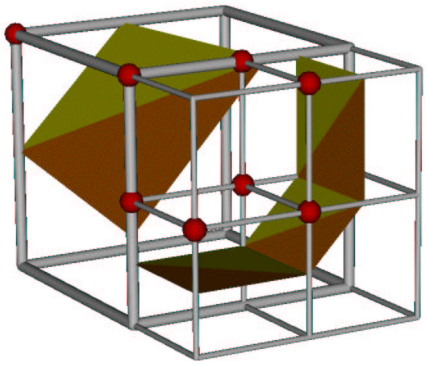


Figura 3.12: Agujero entre celdas de diferente tamaño

Hay diferentes métodos para tapar dicho agujero, como puede ser hacerlo mediante la construcción de un polígono que se adapte a dicho agujero [Shu, 95]. Otra forma de hacerlo es mover los vértices de los triángulos que se han generado en las celdas pequeñas y que están situados en la frontera que contiene el agujero para hacerlos coincidir con los triángulos que se han generado en la celda mayor [Shekhar, 96]. También se pueden tapar sustituyendo un triángulo en la celda grande por un conjunto de triángulos en *abanico* para que se adapten a la posición de los triángulos que se han generado en las celdas pequeñas [Westermann, 99].

En todos los casos se trata de un proceso que se realiza una vez fijado el valor umbral y construidos los triángulos. Nosotros proponemos un método que directamente produce la isosuperficie sin agujeros y sin necesidad de un postproceso de tapado de agujeros.

La función $F(x,y,z)$ restringida a una cara de una celda es una bilineal. En este capítulo nos referiremos como *bilineal de una cara* a la bilineal que se define restringiendo $F(x,y,z)$ a esa cara.

En la situación planteada, la frontera que contiene el agujero está definida por una bilineal por parte de la celda grande y por 4 por parte de las celdas pequeñas; de modo que cada punto de esa frontera puede ser evaluado mediante dos expresiones que pueden dar dos resultados distintos.

La primera aproximación para conseguir el objetivo que perseguimos es modificar los valores de los vértices que correspondan para que las 4 bilineales definidas por las caras de las celdas pequeñas se ajusten a la bilineal definida por la celda grande. Véase la figura 3.13 donde se representa la frontera entre una celda grande y 4 celdas de tamaño menor.

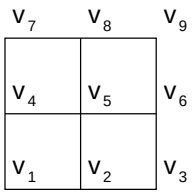


Figura 3.13: Frontera entre celdas de diferente tamaño

Realizando las 5 sustituciones de las ecuaciones 3.6, que no son más que asignarle a dichos vértices el valor que le corresponde según la bilineal de la celda grande, se

consigue que la unión de las 4 bilineales definidas por las caras de las celdas pequeñas coincidan con la bilineal de la celda mayor, con lo que podemos afirmar ahora que la frontera que posee el agujero está definida mediante una única bilineal.

$$\begin{aligned} F(v_2) &\leftarrow \frac{F(v_1)+F(v_3)}{2} \\ F(v_4) &\leftarrow \frac{F(v_1)+F(v_7)}{2} \\ F(v_6) &\leftarrow \frac{F(v_3)+F(v_9)}{2} \\ F(v_8) &\leftarrow \frac{F(v_7)+F(v_9)}{2} \\ F(v_5) &\leftarrow \frac{F(v_1)+F(v_3)+F(v_7)+F(v_9)}{4} \end{aligned} \tag{3.6}$$

Sin embargo esto no arregla el problema, ya que la isocurva definida por un valor umbral en una bilineal puede ser una hipérbola. Esta curva se aproxima mediante un segmento recto en el lado de la celda grande mientras que se puede aproximar hasta por 3 segmentos rectos en el lado de las celdas pequeñas, resultando por tanto un agujero. Ver figura 3.14.

Los nuevos valores que han adoptado los puntos centrales de las aristas sí permiten que los triángulos de las celdas pequeñas se generen ajustándose al triángulo de la celda grande, es el valor del vértice central de la cara el que no permite que el agujero se cierre completamente.

Analicemos los distintos casos de isocurvas que pueden darse en una cara frontera para buscar que posible valor se le puede dar al centro para que los triángulos pequeños se ajusten al grande.

Caso 1: Bilineal plana

El caso trivial se da cuando la bilineal de la cara grande define un plano, con lo cual la isocurva asociada a cualquier valor umbral que corte a la cara va a ser un segmento recto en vez de una hipérbola. Por tanto, asignandole a v_5 el valor que le corresponda según dicho plano hacemos coincidir los triángulos pequeños con el grande.

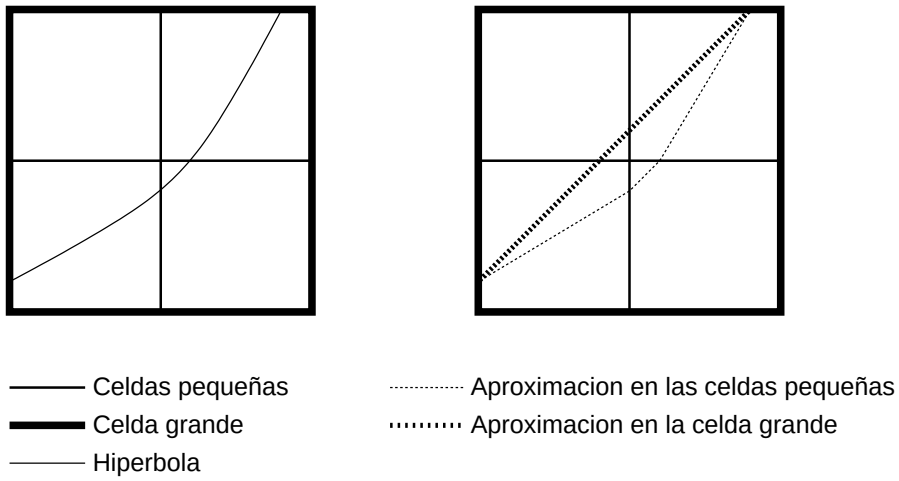


Figura 3.14: Aproximación de una hipérbola

Para este caso una vez modificado el vértice 5 con dicho valor, se puede visualizar la isosuperficie asociada a cualquier valor umbral sin que en dicha cara aparezcan agujeros.

Caso 2: Isocurva desde una arista a la contigua

Supongamos una celda con una configuración de valores como la mostrada en la figura 3.15(a) donde para cualquier valor umbral que haga a la isocurva atravesar la cara, esta isocurva va a ir desde una arista a la contigua.

Si representamos todas las isocurvas variando el valor umbral desde 10 hasta 40 obtenemos una representación que aproxima la bilineal mediante dos triángulos (figura 3.15(b)).

De modo que asignándole a v_5 el valor que le corresponde según esa aproximación de dos triángulos (el valor medio entre los extremos de la diagonal que hace de doblez entre los triángulos), hacemos que cuando calculemos las isocurvas en las celdas pequeñas, éstas se ajustarán a la isocurva en la celda grande.

En este caso ideal, una vez fijado el valor de v_5, se puede visualizar la isosuperficie asociada a cualquier valor umbral sin que en dicha cara aparezcan agujeros.

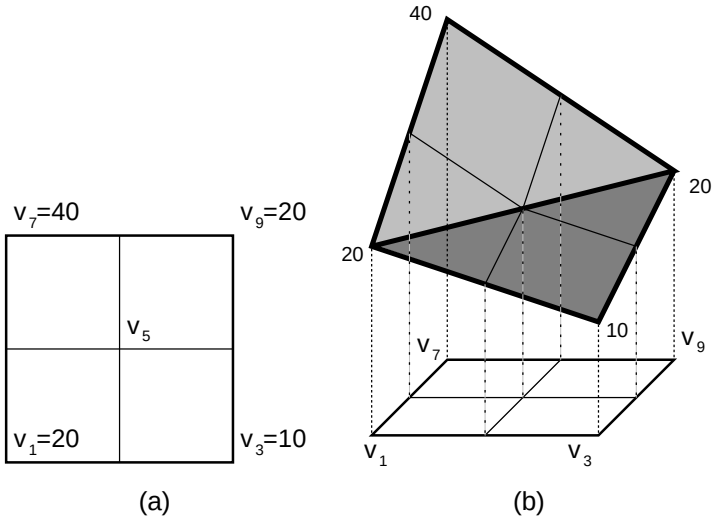


Figura 3.15: Configuración de cara cuyas isocurvas son diagonales a la misma

En el caso no ideal en el que los extremos de la diagonal no tienen el mismo valor, los valores umbral que pertenezcan al intervalo abierto definido por dichos extremos hará que la isocurva corte a dicha diagonal y no conecte aristas contiguas sino opuestas. La isosuperfice asociada se dibujará con un agujero. Esta situación se contempla en el caso 3.

Sin embargo, una vez fijado el valor de v_5 con el valor promedio de los extremos de la diagonal, se puede visualizar la isosuperficie a cualquier valor umbral que no pertenezca al intervalo abierto definido por dichos extremos sin que aparezcan agujeros en dicha cara.

Caso 3: Isocurva desde una arista a la opuesta

Pensemos ahora en un caso de cara en la que para un determinado valor umbral γ_0 la isocurva va desde una arista a la opuesta. En este caso podemos distinguir dos subcasos.

Caso 3.1: La isocurva sólo pasa por 2 celdas pequeñas

En este caso se trata de una isocurva como la que se muestra en la figura 3.16.

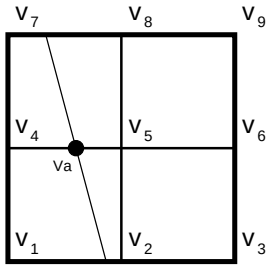


Figura 3.16: Configuración de cara con isocurva entre lados opuestos

El valor que hay que asignarle a v_5 para que el punto v_a calculado en las celdas pequeñas se sitúe en la isocurva generada por la celda grande es dependiente de γ_0 y se calcula con la expresión

$$F(v_5) \leftarrow \frac{\gamma_0 - F(v_4)}{x_a} + F(v_4) \tag{3.7}$$

donde

$$x_a = \frac{\frac{\gamma_0 - F(v_1)}{F(v_2) - F(v_1)} + \frac{\gamma_0 - F(v_7)}{F(v_8) - F(v_7)}}{2} \tag{3.8}$$

Los valores a usar para $F(v_2)$, $F(v_4)$ y $F(v_8)$ son los que se les ha asignado en las expresiones 3.6 (pág. 92) .

Caso 3.2: La isocurva pasa por 3 celdas pequeñas

En este caso se trata de una isocurva como la que se muestra en la figura 3.17.

Ahora v_5 sigue siendo dependiente de γ_0 y además puede necesitar 2 valores distintos simultáneamente, ya que el valor que necesita para hacer coincidir el punto v_a calculado en las celdas pequeñas con la isocurva de la celda grande se calcula con la expresión

$$F(v_5) \leftarrow \frac{\gamma_0 - F(v_4)}{x_a} + F(v_4) \tag{3.9}$$

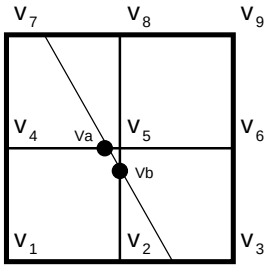


Figura 3.17: Configuración de cara con isocurva entre lados opuestos (y 2)

donde

$$x_a = \frac{\frac{\gamma_0 - F(v_1)}{F(v_2) - F(v_1)} + \frac{\gamma_0 - F(v_7)}{F(v_8) - F(v_7)}}{2} \tag{3.10}$$

mientras que el valor que hay que asignarle a v_5 para hacer coincidir el punto v_b calculado en las celdas pequeñas con la isocurva de la celda grande se calcula con la expresión distinta

$$F(v_5) \leftarrow \frac{\gamma_0 - F(v_2)}{y_b} + F(v_2) \tag{3.11}$$

donde

$$y_b = 2 \cdot \frac{\frac{\gamma_0 - F(v_1)}{F(v_2) - F(v_1)} - 1}{\frac{\gamma_0 - F(v_1)}{F(v_2) - F(v_1)} - \frac{\gamma_0 - F(v_7)}{F(v_8) - F(v_7)}} \tag{3.12}$$

Los valores a usar para $F(v_2)$, $F(v_4)$ y $F(v_8)$ son los que se les ha asignado en las expresiones 3.6 (pág. 92) .

En este caso, cada vez que se cambie el valor umbral hay que hacer un preproceso para calcular los nuevos valores que le corresponden al vértice 5. En ese preproceso habría que localizar las celdas con fronteras en las que hay un cambio de resolución, para clasificar la cara según los casos expuestos anteriormente y asignarle el valor al vértice central según las expresiones indicadas.

Un caso particularmente conflictivo es el caso 3.2, ya que requiere almacenar para una misma posición dos valores distintos para usar uno u otro según el punto de corte isosuperficie-arista que se vaya a calcular.

A continuación daremos un criterio de agrupación que permite obtener isosuperficies sin agujeros sin necesidad de realizar un preproceso.

3.4. Tapado de agujeros

Como acabamos de ver no existe un único valor e independiente del valor umbral para asignarle al vértice central de una cara agrupada para que cualquier isosuperficie se genere sin agujeros en las fronteras de cambio de resolución.

Posteriormente, en el capítulo 5 de esta memoria presentaremos un método de triangulación, llamado *marching de aristas*, que sí genera la isosuperficie sin agujeros en la frontera de cambio de resolución y además de ser independiente del valor umbral no necesita que ningún valor sea modificado.

Sin embargo, en este capítulo, hemos querido presentar un método, que aunque depende de un conjunto de valores umbral, una vez aplicado puede obtenerse la isosuperficie sin agujeros, pudiendose cambiar el valor umbral de visualización sin necesidad de repetir el preproceso.

La idea consiste en hacer uso de los casos 1 y 2 de la sección anterior para los cuales existe un rango de valores que no requieren recalcular el valor para el vértice 5.

Llamemos C_Γ al conjunto de valores umbral importante para el usuario del sistema de modelado y visualización de volúmenes.

Llamaremos **cara válida** a la cara de una celda agrupada que, o bien pertenece al caso 1 de la sección anterior, o bien pertenece al caso 2 de la sección anterior para todos los valores de C_Γ.

El criterio de agrupación se va a modificar en el sentido de que si alguna cara de la celda agrupada no es considerada *cara válida*, la agrupación no se realiza.

Como hemos comentado, las caras cuya bilineal define un plano pertenecen al caso 1 y por tanto son válidas. Consideremos el resto haciendo una distinción entre caras monótonas y no monótonas.

Caso de cara monótona

Si la cara es monótona para cada valor umbral comprendido entre el máximo y el mínimo valor de la cara sólo va a haber una isocurva que corte a la cara. Las isocurvas que vayan de una arista a la contigua (caso 2) sólo van a cortar a una diagonal.

Se puede definir $I_{(v_1,v_9)}$ como el intervalo abierto que definen los valores representados en los vértices de la diagonal $\overline{v_1v_9}$. Las isocurvas correspondientes a los valores

de dicho intervalo cortarán a dicha diagonal.

De igual modo se puede definir $I_{(v_3,v_7)}$ como el intervalo abierto que contiene a los valores cuyas isocurvas van a cortar a la diagonal opuesta.

Por tanto, $I_{(v_1,v_9)} \cap I_{(v_3,v_7)}$ va contener a los valores cuyas isocurvas van a cortar a las dos diagonales y por tanto hace que para esos valores la cara sea del caso 3.

Una cara monótona será válida si se cumple que $I_{(v_1,v_9)} \cap I_{(v_3,v_7)} \cap C_\Gamma = \emptyset$

En este caso a $F(v_5)$ se le asigna el valor medio entre los valores extremos de la diagonal con intervalo menor.

Caso de cara no monótona

Las caras no monótonas además del mínimo y máximo global, contienen un mínimo y un máximo local. Estando los mínimos en los extremos opuestos de una diagonal y los máximos en los extremos opuestos de la otra diagonal.

Este tipo de cara siempre va a tener las isocurvas que la cruzan entre una arista y la contigua. De hecho si hacemos variar el umbral entre el mínimo y el máximo valor de la cara (ver el ejemplo de la figura 3.18) comienzan las isocurvas uniendo las aristas contiguas que convergen en el mínimo, cuando el umbral llega al mínimo local (situado en el vértice opuesto) hay dos isocurvas que unen aristas contiguas sin cortar a la diagonal que une los máximos, al llegar el umbral a un valor conocido como *valor silla* [Cignoni, 00] [1] la cara es cortada por 4 isocurvas definiendo un polígono cerrado, a partir de este valor a cada valor umbral tan solo le corresponden 2 isocurvas que unen aristas contiguas sin cortar a la diagonal que une los mínimos, hasta que se llega al máximo local. A partir de éste y hasta llegar al máximo de la cara, cada valor umbral solo tiene una isocurva que conecta las dos aristas contiguas que convergen en el máximo.

En este caso una cara será válida si es cierta alguna de las dos expresiones siguientes:
1. $valor\ silla < \min(C_\Gamma)$
2. $valor\ silla > \max(C_\Gamma)$

[1] El punto silla se corresponde con el origen de las asimptotas de la hipérbola que define la bilineal. Su posición en la cara se define mediante la expresión $(\frac{F(v_1)-F(v_7)}{F(v_1)+F(v_9)-F(v_3)-F(v_7)}, \frac{F(v_1)-F(v_3)}{F(v_1)+F(v_9)-F(v_3)-F(v_7)})$ y el valor de la bilineal en dicho punto es $\frac{F(v_1)\cdot F(v_9)-F(v_3)\cdot F(v_7)}{F(v_1)+F(v_9)-F(v_3)-F(v_7)}$

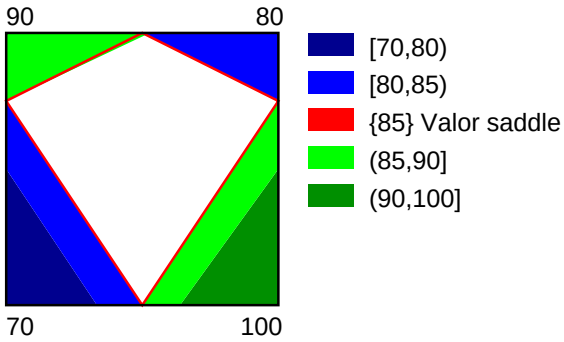


Figura 3.18: Rango de isocurvas en una cara no monótona

Si se cumple la expresión 1. a $F(v_5)$ se le debe asignar el valor promedio de los mínimos de la cara, y si se cumple la expresión 2. a $F(v_5)$ se le debe asignar el valor promedio de los máximos de la cara.

Criterio de agrupación sin agujeros

Ocho celdas podrán ser agrupadas en una si:

- Cumplen algún criterio de agrupación de los mostrados
- Todas las caras resultantes de la agrupación son válidas para el conjunto de valores C_Γ especificado por el usuario.

Si se realiza la agrupación el valor de los vértices v_2, v_4, v_6 y v_8 de cada cara se debe modificar según las expresiones 3.6 (pág. 92) y el valor de v_5 según la clasificación que se haga de la cara según lo expuesto en esta sección.

Esta modificación de valores no se lleva a cabo inmediatamente después de realizar la agrupación ya que afectaría a agrupaciones posteriores. Se realiza una vez se ha obtenido el octree de celdas. Para ello ser realiza un recorrido *primero en anchura* para asegurarnos que los valores que vayamos necesitando para calcular nuevos ya hayan sido modificados previamente.

Otra posible opción consiste en no restringir el criterio de agrupación y realizar un preproceso que coloque los valores adecuados en los vértices centro de cara agrupada dependiendo del valor que se va a visualizar en ese momento. Se pierde en

velocidad de visualización pero se gana en espacio al permitir una mayor poda del árbol. Además tiene el inconveniente de tener que manejar dos posibles valores para los vértices centrales de las caras del caso 3.2.

3.5. Medida del error

En cada agrupación estamos sustituyendo 8 porciones de volumen que se está aproximando mediante 8 funciones trilineales por una sola porción de volumen que aunque ocupa el mismo espacio de $\mathbb{R}^3$ se está aproximando mediante una sola función trilineal. Por tanto, en la mayoría de los casos se puede estar cometiendo un error.

Para medir el error se puede comparar la trilineal resultante con las trilineales originales. Aunque no debemos olvidar que las trilineales originales son ya de por sí una aproximación a una función no conocida. Lo ideal sería poder hacer la comparación con la función real, con lo que el error cometido sería, en términos relativos, menor.

Otro modo de medir el error sería hacerlo sobre la isosuperficie una vez construida, midiendo cómo se ha separado la superficie obtenida mediante un octree de celdas de la superficie que define la trilineal para el mismo valor umbral. O al igual que hemos comentado hace un momento, lo ideal sería comparar la isosuperficie obtenida con la que define la función real para el valor umbral considerado.

3.6. Resultados con volúmenes reales

Las propuestas realizadas en este capítulo han sido implementadas y probadas con volúmenes reales y simulados. Como volúmenes reales se han usado modelos de diferentes partes del cuerpo humano construidos con información procedente de una tomografía axial computerizada (TAC). Dicha información se ha normalizado para que varíe entre 0 y 255. En concreto se han usado los datos del *Visible Human Project* sin filtrar. La figura 3.19 muestra imágenes de dichos modelos estableciendo el umbral de visualización en el valor de 60. Las resoluciones son de 100 x 80 x 80 para el cuello, 150 x 120 x 120 para la rodilla y 200 x 240 x 240 para la cabeza.

Hemos querido probar también las propuestas con modelos para los que se conociera con exactitud la función que los define. En ese sentido hemos usado tres modelos según la siguiente definición:

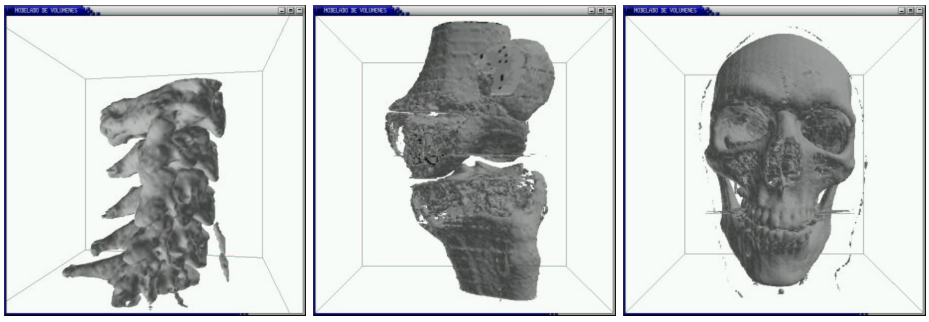


Figura 3.19: Imágenes de modelos reales

$$\alpha(x,y,z) : V \subset \mathbb{R}^3 \rightarrow [0,255]$$
$$\alpha(x,y,z) = \begin{cases} 255 \cdot (1 - r_i) & \text{si} \quad r_i \leq 1 \\ 0 & \text{en} \quad \text{otro caso} \end{cases} \tag{3.13}$$

donde V es un cubo de lado 2 centrado en el origen y alineado con los ejes y r_i se define según el modelo como:

- **Modelo 1**

$$r_1 = \sqrt{x^2 + y^2 + z^2} \tag{3.14}$$

- **Modelo 2**

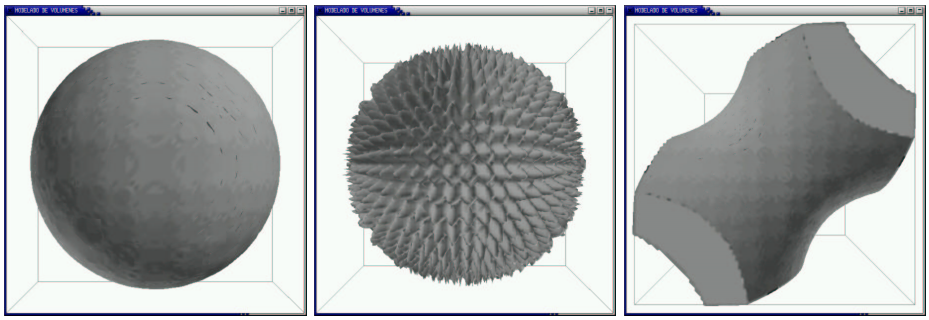
$$r_2 = \sqrt{x^2 + y^2 + z^2} + 0.05 \cdot (\sin(50 \cdot \arctan\frac{z}{x}) + \cos(40 \cdot \arctan\frac{y}{x})) \tag{3.15}$$

- **Modelo 3**

$$r_3 = \sqrt{x^2 + 2 \cdot y \cdot z} \tag{3.16}$$

Se ha construido un octree de celdas a partir de estos modelos con una resolución de 100 x 100 x 100. Usando como umbral de visualización el valor de 60 obtenemos las imágenes de la figura 3.20.

Dichos modelos se han representado mediante un bono y un octree de celdas usando el criterio de agrupación que hemos llamado *sin agujeros* y permitiendo cualquier diferencia de valor en los vertices no accesibles tras la agrupación (punto 4 del criterio de agrupación básico).



Modelo 1 Modelo 2 Modelo 3

Figura 3.20: Imágenes de modelos matemáticos

Se ha medido el tamaño del árbol en cada caso, el tiempo empleado en construirlo, el tiempo empleado en generar la malla de triangulos para la isosuperficie mostrada y el número de triángulos que la componen. En las tablas 3.1 y 3.2 y en los diagramas de la figura 3.21 se muestran los resultados obtenidos.

	Construcción del Árbol		Generación de la Malla	
Modelo	Bono	Oct.Celdas	Bono	Oct.Celdas
Cuello	54	168	115	113
Rodilla	430	1,330	327	314
Cabeza	436	1,373	1,103	1,099
Modelo 1	55	176	60	42
Modelo 2	56	162	187	184
Modelo 3	55	179	121	186

Tabla 3.1: Bono vs. Octree de celdas (Tiempos en ms)

Se observa que si bien el tiempo empleado en la construcción del árbol es significativamente mayor en el octree de celdas que en el bono, es un proceso que solo hay que realizar una vez, pudiendose cambiar el umbral y el punto de vista sin que haya que reconstruir el árbol. El tiempo empleado para generar la isosuperficie y el número

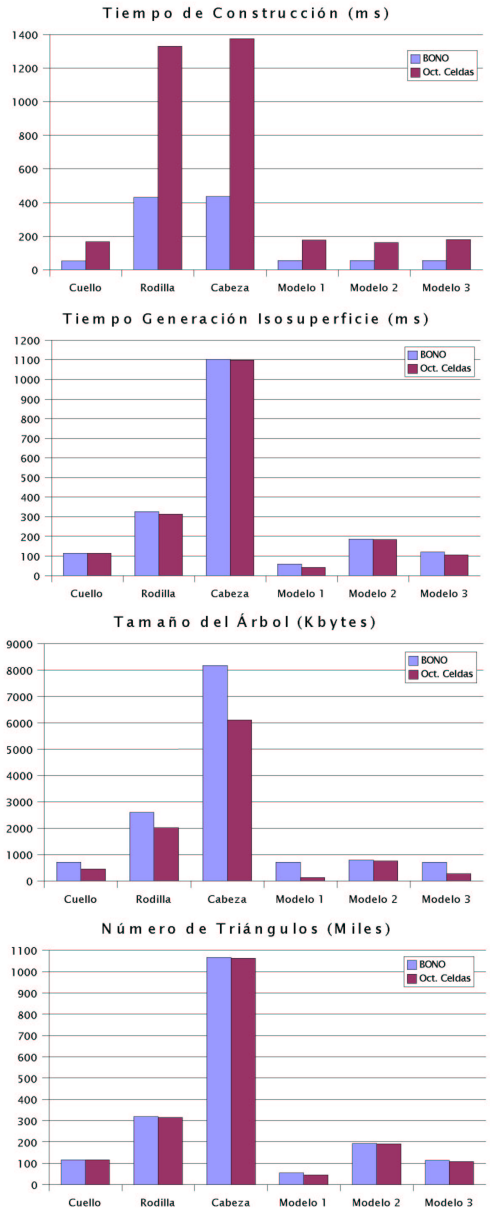


Figura 3.21: Bono vs. Octree de Celdas

	Tamaño del Árbol (bytes)		Nº de Triángulos	
Modelo	Bono	Oct.Celdas	Bono	Oct.Celdas
Cuello	737,296	466,768	115,876	115,670
Rodilla	2,671,184	2,061,072	318,768	315,196
Cabeza	8,374,544	6,255,952	1,067,412	1,063,086
Modelo 1	727,312	135,632	55,724	45,068
Modelo 2	809,232	783,056	192,592	190,370
Modelo 3	726,672	275,664	114,672	108,178

Tabla 3.2: Bono vs. Octree de celdas (Tamaños)

de triángulos que la forman permanece prácticamente igual, sólo ligeramente inferior en el octree de celdas, lo cual es un indicativo de la calidad de la imagen generada como después corroboraremos con las medidas de error que hemos realizado. Donde si se aprecia una ventaja es en el tamaño del árbol del octree de celdas con respecto al bono. Inferior a éste con lo que las necesidades de espacio disminuyen.

Al representar el volumen mediante un octree de celdas se está cometiendo un error, por el hecho de representarlo con menos información de la que se dispone originalmente. Hemos realizado dos mediciones del error, por un lado el error que se comete sobre el conjunto de muestras con independencia del valor umbral que se use para las visualizaciones. Y el error que se comete medido en la isosuperficie una vez se ha fijado un valor umbral concreto.

En el primer caso se ha medido la diferencia media (en valor absoluto) del valor de propiedad original que presenta una muestra con respecto al valor estimado en dicho punto por el octree de celdas. Esta medida se ha vuelto a calcular teniendo en cuenta solamente aquellas muestras en las cuales el valor de propiedad original no coincide con el calculado por el octree. También se ha estimado para estas posiciones la distancia a la que se encontraría en el octree de celdas una isosuperficie que con la información original pasara por dichas posiciones.

Esta estimación de distancia se ha calculado dividiendo la diferencia en valor de propiedad por el modulo del gradiente en dicha posición. Cuando el módulo del

gradiente es muy pequeño, lo que indica una variación muy suave del valor de propiedad, la estimación de la distancia así calculada no es adecuada. Cuando la distancia así calculada supera la longitud de la mitad de una diagonal de la celda que se está procesando se asume como valor de distancia la mitad de dicha diagonal.

La tabla 3.3 muestra los datos sobre el error en los modelos que estamos tratando. La columna etiquetada como *Error 1* muestra el error medido en unidades de propiedad según la siguiente expresión:

$$Error\ 1\ = \frac{\sum_{\forall p \in V} |F(p) - f(p)|}{Card(V)} \tag{3.17}$$

donde V es el conjunto de muestras del volumen, $f(p)$ es el valor de propiedad original de esa muestra y $F(p)$ el valor de propiedad que según el octree de celdas presenta el punto p.

La columna etiquetada como *Error 2* muestra el error medido en unidades de propiedad con la expresión 3.17 pero considerando solamente aquellos puntos para los que $|F(p) - f(p)| \neq 0$.

La columna etiquetada como *Error 3* muestra el error medido en distancia donde la unidad es el lado de una celda mínima, medido con la siguiente expresión:

$$Error\ 3\ = \frac{\sum_p \min(\frac{|F(p)-f(p)|}{|Grad(p)|}, mediaDiagonal)}{n} \tag{3.18}$$

donde $p \in \{p \in V : |F(p) - f(p)| \neq 0\}$, n es el cardinal de este conjunto, $Grad(p)$ es el gradiente de p y *mediaDiagonal* es la mitad de la longitud de la diagonal de la celda que contiene al punto dado.

La columna etiquetada como *Proporción* muestra la proporción de muestras para las que en la expresión de cálculo de *Error 3* se ha usado el valor *mediaDiagonal*.

En cuanto a la medición del error para un valor umbral concreto se ha medido el error en los vértices de los triángulos y en el centro geométrico de los mismos. Las medidas de error que se han usado son las representadas por las expresiones 3.17 y 3.18 sobre el conjunto de vértices indicados, es decir, la diferencia en valor de propiedad y la distancia estimada.

En este caso hacemos una comparación con bono, en tanto en cuanto, tanto en un bono (igual que un marching cubes en lo que a la generación de triángulos se refiere) como en un octree de celdas, los vértices de los triángulos se calculan mediante una

Modelo	Error 1	Error 2	Error 3	Proporción
Cuello	0.19	3.05	3.09	82.23 %
Rodilla	0.15	2.34	1.70	50.09 %
Cabeza	0.07	1.70	2.19	62.61 %
Modelo 1	1.27	2.67	3.44	21.60 %
Modelo 2	0.07	3.11	0.71	11.53 %
Modelo 3	0.82	3.80	2.30	16.82 %

Tabla 3.3: Error con independencia del umbral

estimación lineal a lo largo de las aristas. Para ello comparamos el valor de propiedad de un vértice calculado (es decir, el valor umbral) con el valor de propiedad que posee dicho vértice en el volumen real. Para los modelos 1, 2 y 3 podemos realizar esa comparación de manera exacta pues se dispone de la función contínua que representa a dichos volúmenes. En el caso de los modelos del cuerpo humano, hemos construido tanto el bono, como el octree de celdas a un nivel de resolución menor que el máximo disponible y usamos como función de referencia los modelos a la máxima resolución.

La tabla 3.4 muestra los resultados obtenidos.

Hemos querido mostrar este error de manera visual presentando imágenes de los modelos coloreadas en función del *error 3* de los vértices de los triángulos. La interpretación de los colores se muestra en la figura 3.22 donde un error positivo indica que la isosuperfice se ha desplazado hacia el interior de la isosuperficie real y un error negativo que se ha desplazado en el sentido opuesto.

Las imágenes coloreadas de la manera comentada se muestran en la figura 3.23, visualmente las imágenes del cuello, rodilla y cabeza muestran zonas con un gran error, ello es debido a que al construirse a un nivel menos de resolución que el modelo que se usa como referencia, se han construido con únicamente la octava parte de la información disponible, con lo que se pueden haber eliminado puntos que fuesen máximos o mínimos locales, lo que provoca un gran error en la comparación con el modelo de referencia. La figura 3.24 muestra la imagen coloreada del cuello modelada con bono a un nivel de resolución menor; donde se observa un coloreado prácticamente igual al de la imagen del cuello modelado con un octree de celdas.

Modelo	Esquema 1	Error 1	Error 3	Proporción
Cuello	Bono	4.17	0.38	4.00 %
	Oct.Celdas	4.18	0.38	4.01 %
Rodilla	Bono	3.37	0.37	3.41 %
	Oct.Celdas	3.40	0.38	3.49 %
Cabeza	Bono	5.79	0.31	2.59 %
	Oct.Celdas	5.81	0.31	2.62 %
Modelo 1	Bono	0.00	0.00	0.00 %
	Oct.Celdas	0.17	0.04	0.03 %
Modelo 2	Bono	1.83	0.16	4.27 %
	Oct.Celdas	1.88	0.17	4.33 %
Modelo 3	Bono	45.17	0.30	20.06 %
	Oct.Celdas	45.49	0.35	21.87 %

Tabla 3.4: Error para un valor umbral concreto

Podemos concluir a la vista de los datos mostrados, que el error cometido es tolerable y que las imágenes obtenidas a partir de un octree de celdas son de una calidad similar a las obtenidas a partir de un bono.

Por último, mostramos en la tabla 3.5 como varía la medida del error 1 en función de la diferencia de valor permitida para los vértices no accesibles tras la agrupación.

Se observa cómo la variación de dicho parámetro no es muy significativa, debido a que se trata de una condición más en un criterio que se compone de varias condiciones. Donde más útil se muestra es en aquellos modelos que presentan una mayor suavidad, como los modelos 1 y 3; ya que en modelos con esas características de suavidad se suele producir una mayor agrupación produciendose una mayor diferencia de valor, siendo por tanto necesario controlarlo mediante este parámetro.

También mostramos una tabla similar a la anterior en la que mostramos el tamaño (en KB) del árbol en función de la variación de ese parámetro (tabla 3.6). Se observa cómo siempre se obtiene un tamaño inferior a un árbol bono.

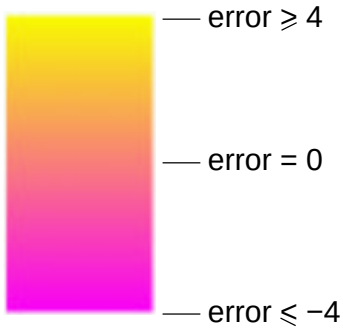


Figura 3.22: Gama de color para interpretar los errores

Dif.Permitida	Cuello	Rodilla	Cabeza	Mod. 1	Mod. 2	Mod. 3
5	0.05	0.05	0.04	0.30	0.00	0.13
10	0.06	0.07	0.04	0.52	0.02	0.21
25	0.06	0.09	0.06	0.98	0.06	0.42
50	0.19	0.15	0.07	1.27	0.07	0.54
100	0.19	0.15	0.07	1.27	0.07	0.54
150	0.19	0.15	0.07	1.27	0.07	0.54
200	0.19	0.15	0.07	1.27	0.07	0.54
Sin límite	0.19	0.15	0.07	1.27	0.07	0.81

Tabla 3.5: Error 1 en función de la diferencia en valor permitida

3.7. Conclusiones

En este capítulo se ha presentado un esquema de representación de volúmenes basado en estructuración jerárquica de la información. El esquema permite la representación de un volumen mediante celdas de diferente tamaño en función del comportamiento de las propiedades del volumen en las diferentes zonas del mismo.

Se presentan varios criterios para realizar la agrupación jerárquica de la información, permitiendo uno de ellos la obtención de isosuperficies sin agujeros.

Se presenta una comparación con métodos existentes mediante la visualización de modelos de volúmenes reales y modelos matemáticos. También se presentan datos del

Dif.Permitida	Cuello	Rodilla	Cabeza	Mod. 1	Mod. 2	Mod. 3
Bono	720	2,609	8,178	710	790	710
5	476	2,083	6,164	174	789	319
10	476	2,053	6,143	136	775	291
25	476	2,047	6,127	132	766	282
50	456	2,013	6,109	132	765	275
100	456	2,013	6,109	132	765	273
150	456	2,013	6,109	132	765	273
200	456	2,013	6,109	132	765	273
Sin límite	456	2,013	6,109	132	765	269

Tabla 3.6: Tamaño del árbol (KB) en función de la diferencia en valor de propiedad permitida

error cometido al realizar la agrupación de información. En todos los casos el espacio ocupado es significativamente menor, manteniendo la calidad de la visualización.

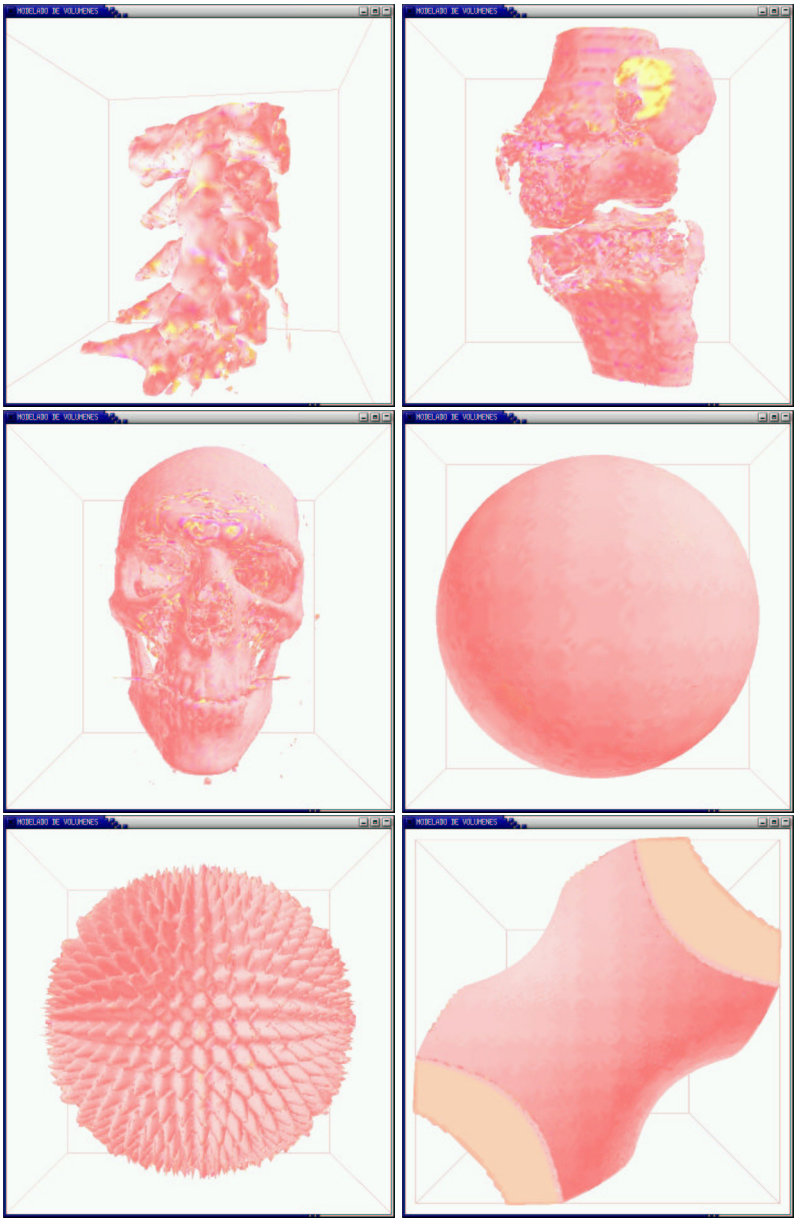


Figura 3.23: Imágenes coloreadas según el error cometido

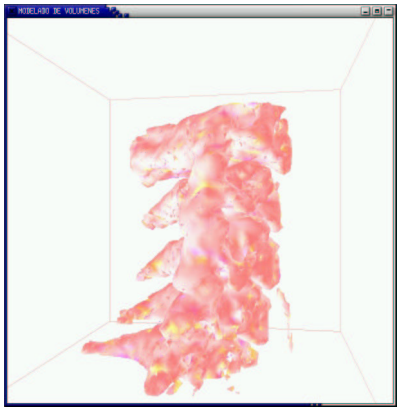


Figura 3.24: Imagen procedente de bono coloreada según el error cometido

Capítulo 4

Transmisión Progresiva de Octrees de Celdas

Piensa en lo que tienes
y no en lo que te falta
Marco Aurelio

Hoy día es habitual en cualquier ámbito de aplicación de la informática tener los sistemas interconectados en red, de modo que se ven beneficiados de ventajas como la compartición de recursos, la centralización de la información, el envío/recepción de información hacia/desde otros sistemas, etc.

Los sistemas de modelado y visualización de volúmenes conectados en red se benefician por tanto de esas ventajas mencionadas, entre ellas la de poder transmitir o recibir un modelo volumétrico hacia o desde otros sistemas. El tamaño de los modelos volumétricos es cada vez mayor, con lo que la transmisión se convierte en una operación lenta. Ante esta situación se puede recurrir a realizar la transmisión de manera progresiva.

Se transmite el modelo a diferentes niveles de resolución, se realiza una transmisión a un nivel pobre de resolución con lo que el receptor recibe un modelo completo en poco tiempo con el que poder operar desde el principio. En sucesivos envíos se va transmitiendo la información necesaria para que el receptor vaya disponiendo del modelo a una resolución cada vez mayor.

En este capítulo proponemos un método para realizar la transmisión progresiva de un volumen a través de la red usando octrees de celdas, de modo que el receptor va recibiendo el volumen en diferentes fases, aumentando la resolución en cada una de ellas pero estando operativo desde el principio [Velasco, 02b].

4.1. Trabajos previos

En el contexto de la visualización de volúmenes, Engel et al. proponen un sistema cliente-servidor de visualización de volúmenes que permite que el ordenador cliente

que visualiza el volumen no tenga la potencia de cálculo que un volumen de gran tamaño exigiría, el cual estaría almacenado en un ordenador servidor que realiza cálculos y proporciona al cliente la información que éste necesita para realizar la visualización [Engel, 99].

En esta línea realizan una división del proceso de visualización en 7 etapas haciendo una clasificación del sistema cliente-servidor en 6 casos distintos según qué etapas del proceso se llevan a cabo el servidor y cuales el cliente.

Las etapas del proceso de visualización que definen son las siguientes:

1. *Almacenamiento*: No hay proceso, sólo un volumen almacenado.
2. *Filtrado*: Se localizan las celdas activas una vez fijado el umbral.
3. *Interpolación*: En cada celda activa se localizan los puntos de corte de la isosuperficie con las aristas. Estos puntos serán los vértices de los triángulos.
4. *Triangulación*: Con los vértices obtenidos en la etapa anterior se procede a ir construyendo la malla de triángulos.
5. *Proyección*: Los triángulos 3D se proyectan sobre el plano de visualización.
6. *Rasterización*: Se procede al relleno pixel a pixel sobre un frame buffer de los triángulos 2D obtenidos.
7. *Visualizado*: Se vuelca sobre el dispositivo de salida la imagen del frame buffer.

Con estas 7 etapas resultan 6 clases de sistema cliente-servidor en atención a donde se coloque la división de tareas entre servidor (S) y cliente (C). Estas clases quedan recogidas en la tabla 4.1.

En todos ellos el volumen queda almacenado en el servidor necesitandose una transmisión de información cada vez que el cliente necesite hacer una visualización.

4.2. Transmisión de un octree de celdas

Nosotros proponemos una clase más (la clase 7 en la tabla 4.2) consistente en realizar la transmisión del volumen completo de modo que a partir de ese momento

Etapa	Clase 1	Clase 2	Clase 3	Clase 4	Clase 5	Clase 6
Almacenamiento	S	S	S	S	S	S
Filtrado	S	S	S	S	S	C
Interpolación	S	S	S	S	C	C
Triangulación	S	S	S	C	C	C
Proyección	S	S	C	C	C	C
Rasterización	S	C	C	C	C	C
Visualizado	C	C	C	C	C	C
Se transmite	Imagen	Triáng. 2D	Triáng. 3D	Vértices triáng.	Celdas activas	Modelo volum.
Retransmisión si cambia	Umbral Observ.	Umbral Observ.	Umbral	Umbral	Umbral	Umbral

Tabla 4.1: Clasificación realizada en [Engel, 99]

todas las operaciones sobre el modelo puedan hacerse en el cliente sin necesidad de nuevas transmisiones. Además, proponemos que la transmisión se realice de manera progresiva, de modo que desde el principio el cliente posea un modelo del volumen completo aunque a baja resolución con el que poder hacer visualizaciones y ajustes del umbral, en envíos sucesivos el modelo del volumen se irá refinando mejorándose las visualizaciones.

El octree de celdas es un esquema de representación idóneo para realizar una transmisión progresiva del mismo, ya que está soportado por una estructura en árbol el cual, si se podan todas las ramas al mismo nivel, sigue representado el volumen completo pero a menor resolución.

Por ello, si se realiza la transmisión del árbol nivel a nivel, desde el principio se tiene un modelo del volumen completo, y con la llegada de cada nuevo nivel el modelo del volumen representado aumenta de resolución. Junto con los nodos del árbol correspondientes a un nivel se envían los valores de la rejilla a los que dichos nodos dan acceso

El cliente no tiene por qué recibir el modelo del volumen a la resolución máxima. Puede indicar el nivel de resolución máximo que desea recibir. Así mismo, el esquema permite que después de recibir el modelo de volumen a una resolución media, el

Etapa	Clase 7
Almacenamiento	C
Filtrado	C
Interpolación	C
Triangulación	C
Proyección	C
Rasterización	C
Visualizado	C
Se transmite	Modelo volumétrico
Retransmisión si cambia	*No hay retransmisión*

Tabla 4.2: Nuestra propuesta (clase 7)

usuario seleccione un área y que la transmisión continúe enviando solamente la rama seleccionada por el usuario.

4.3. Implementación

Para llevar a cabo la transmisión de un octree de celdas, tendremos que transmitir tanto los valores de la rejilla como el árbol que da acceso a los valores de la rejilla. De igual modo, es importante que tanto el emisor como el receptor estén sincronizados para que la información que éste esté esperando sea la que aquel le está enviando. Hay que tener en cuenta que no se envía ningún tipo de información ajena a la propia del esquema de represetnación por el hecho de hacer la transmisión de manera progresiva.

Vamos a mostrar como sería la transmisión/recepción de valores, la transmisión/recepción de nodos del árbol y por último el algoritmo global de transmisión/recepción de un octree de celdas.

4.3.1. Transmisión/recepción de valores

La transmisión/recepción de un volumen debe comenzar con el envío del tamaño del volumen, su caja englobante, que denominamos `MAXX`, `MAXY` y `MAXZ`. A partir de esos tres datos calculamos un nuevo dato, llamado `distancia` que nos va a guiar la transmisión de valores y que calculamos como sigue:

$$distancia = \min\{x = 2^n : n \in \mathbb{N} \wedge x >= \max\{MAXX, MAXY, MAXZ\}\} \quad (4.1)$$

`distancia` define la separación que hay entre dos valores consecutivos en cada dirección (x, y, z) a una determinada resolución. El envío comienza con la distancia que corresponde a la resolución peor y en cada refinamiento la distancia se irá reduciendo a la mitad, hasta que a la máxima resolución la distancia sea 1.

Con el tamaño transmitido y el dato `distancia` calculado hay que realizar un envio por cada nivel, siendo el envío de peor resolución distinto a los demás por ser el primero, en el resto de envíos hay que tener la precaución de no enviar valores ya enviados.

El pseudocódigo para la transmisión/recepción del primer nivel es el mostrado en la figura 4.1.

```
FUNCTION enviarValoresNivelRaiz ()
{
   for (i = 0; i <= MAXX; i += distancia)
      for (j = 0; j <= MAXY; j += distancia)
         for (k = 0; k <= MAXZ; k += distancia)
            enviar (rejilla[i][j][k]);
}
```

Figura 4.1: Transmisión de valores para el nivel raiz

El código de recepción es igual, solo se cambia `enviar` por `recibir`.

Para los siguientes niveles, los valores estarán espaciados la mitad de las unidades (`distancia /= 2`), y además tenemos que evitar enviar valores doblemente.

En la figura 4.2 se observa para un ejemplo pequeño qué valores hay que transmitir en cada envío. Si transmitiéramos cada vez los valores que nos indicara la variable `distancia` los valores marcados con un cuadrado se hubieran transmitido 3 veces y los valores marcados con un círculo se hubieran transmitido 2 veces. Para evitar esa redundancia hay que tener en cuenta qué fila se está transmitiendo; por ejemplo, cuando en la figura `distancia` valía 1 en el 3^{er} envío en las filas pares había que transmitir los valores (marcados como rombos) que estaban distanciados 1 unidad, pero en las filas impares había que transmitir los valores que estaban distanciados 2 unidades.

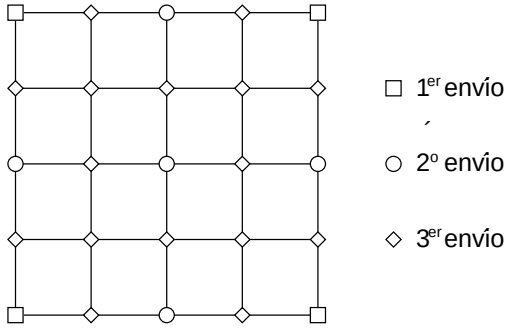


Figura 4.2: Transmisión progresiva de valores

Para controlar esta situación usamos las variables lógicas auxiliares `auxi` y `auxj`. Cuando `auxi` es `TRUE` indica que `i` tiene un valor tal que puede dar acceso a una fila que contiene valores ya enviados, lo mismo podemos decir de `auxj` con respecto a `j`; por tanto cuando ambas variables son `TRUE` nos encontramos que cuando variemos `k` en su bucle estaremos ante una fila que contiene valores ya enviados y se realiza la transmisión enviando sólo los valores intermedios a los ya transmitidos.

Todo ello queda recogido en el pseudocódigo de la figura 4.3.

En las filas que contienen valores ya enviados, no se empieza en el extremo de la fila y se avanza con el doble de la distancia que le corresponde a ese nivel. En el resto de filas, se comienza en el extremo y se avanza según la distancia que le corresponde a ese nivel.

```
FUNCTION enviarValoresNivel ()
{
  antiguaDistancia = distancia;
  distancia /= 2;
  for (i = 0, auxi = TRUE; i <= MAXX; i += distancia, auxi = ! auxi)
    for (j = 0; auxj = TRUE; j <= MAXY; j += distancia, auxj = ! auxj)
      if (auxi && auxj)
        for (k = distancia; k <= MAXZ; k += antiguaDistancia)
          enviar (rejilla[i][j][k]);
      else
        for (k = 0; k <= MAXZ; k += distancia)
          enviar (rejilla[i][j][k]);
}
```

Figura 4.3: Transmisión de valores

4.3.2. Transmisión/recepción del árbol

Para transmitir el árbol hay que recorrerlo según el orden *primero en anchura*, vamos a necesitar una estructura de datos *cola* y una marca `iniciaNivel` que nos indica el final de un nivel y el comienzo del siguiente. Las funciones para el manejo de la cola son:

- `vaciaCola()`: Devuelve `TRUE` si la cola está vacía y `FALSE` en caso contrario.
- `ultimoCola(elemento)`: Introduce `elemento` al final de la cola.
- `primeroCola()`: Devuelve el elemento que está situado en la primera posición de la cola y lo saca de ésta.

Los elementos que puede contener la cola pueden ser nodos y la marca `iniciaNivel`.

El pseudocódigo de la transmisión completa del árbol es el mostrado en la figura 4.4.

El algoritmo es el clásico de recorrido en anchura de un árbol en el que se comienza introduciendo el nodo raiz en la cola y mientras haya elementos en la cola, se saca el que está en primer lugar, y se introducen los hijos de éste si los tuviera. Todo

```
enviar (raiz);
ultimoCola (iniciaNivel);
if (esNodoInterno (raiz))
   ultimoCola (raiz);
salir = FALSE;
do {
   elemento = primeroCola ();
   if (elemento == iniciaNivel)
      if (!vaciaCola())
         ultimoCola (iniciaNivel);
      else
         salir = TRUE;
   else {   /* elemento es un nodo interno */
      for each hijo de elemento {
         enviar (hijo);
         if (esNodoInterno (hijo))
            ultimoCola (hijo);
      }
   }
} while (!salir)
```

Figura 4.4: Transmisión del árbol

nodo, antes de ser introducido en la cola es procesado; en nuestro caso, el procesamiento consiste en enviarlo al receptor. Una modificación añadida al algoritmo es el tratamiento de la marca `iniciaNivel` que indica cuándo se produce un cambio de nivel. Se introduce en la cola junto con el nodo raíz y cada vez que llega al primer lugar de la cola indicando la finalización de un nivel se vuelve a introducir al final.

El pseudocódigo para la recepción es muy similar al de transmisión, sólo que se cambian las instrucciones de acceso a hijos del árbol por instrucciones de creación del árbol (ver figura 4.5).

4.3.3. Transmisión/recepción progresiva de un octree de celdas

Ya sólo queda integrar los pseudocódigos correspondientes a los valores y al árbol para tener el algoritmo completo.

En el servidor, se empieza enviando los límites de la caja englobante y calculando `distancia` que va a controlar el envío de los valores. En general se van a enviar los valores de un nivel antes que los nodos de ese nivel. Por tanto se continúa enviando los valores del nivel raíz y el nodo raíz, iniciando la cola con la marca de comienzo de nivel y con la raíz si ésta es nodo interno para después pasar al bucle principal de recorrido del arbol.

Recordemos del capítulo anterior que tenemos dos tipos de nodos hoja: `nodoHojaUna` que sólo da acceso a una celda, y `nodoHojaMás` que da acceso a más de una celda. Por lo tanto si en el ultimo nivel que se transmita hay nodos tipo `nodoHojaMás` habrá que enviar un nivel más de valores para que las celdas accesibles a través de ese `nodoHojaMás` tengan los valores que necesitan. Eso va a ocurrir siempre salvo que el árbol esté formado solamente por la raíz y esta sea de tipo `nodoHojaUna`. Para ello usamos la variable `otroNivel` que se inicia a `TRUE` y se cambia a `FALSE` solo si la raíz es del tipo `nodoHojaUna`.

Por lo demas, el bucle principal del algoritmo es similar al de envío del arbol pero realizando una transmision de valores de la rejilla cada vez que empezamos un nuevo nivel, circunstancia que nos indica la marca que se introduce en la cola. El pseudocódigo lo tenemos en la figura 4.6.

En lo que respecta al cliente, en el caso de que se realice una visualización durante la recepción del modelo, va recibiendo valores y nodos del árbol. Ante la llegada de cada nodo si se cumplen las condiciones para ser visualizado, es decir, la cota esta comprendida entre el mínimo y el máximo almacenados en el nodo (trabajo realizado por la función `esProcesable(celda(hijo))`), se generan los triángulos que corresponden a la celda representada por dicho nodo.

Cuando la dimensión de la caja englobante no es potencia de 2 habrá nodos internos para los cuales no hay una celda en la rejilla del tamaño adecuado. En esa situación, se construye una celda con los valores que sí están en la rejilla, y los vértices de la celda que están fuera de la caja englobante se inician con un valor de propiedad *por defecto*. La celda así construída se evalúa como las otras.

Se puede mantener una lista de triángulos por cada nivel, de modo que en la lista de un nivel se almacenan los triángulos que han generado los nodos tipo `nodoHojaUna` de ese nivel y los nodos tipo `nodoHojaMás` del nivel inmediatamente superior, y esa lista no se tiene que recalcular, los triángulos de esa lista son válidos para los niveles siguientes. Los triángulos que se han generado por los nodos internos y los nodos tipo `nodoHojaMás` de un nivel se almacenan en otra lista que se destruye cada vez que iniciamos un nuevo nivel.

Una variación con respecto al algoritmo de recepción del árbol descrito previamente es que ahora en la cola tambien se introducen los nodos tipo `nodoHojaMás` para encontrarlos en el siguiente nivel y procesar las celdas a las que da acceso. El pseudocódigo lo tenemos en la figura 4.7.

Por último, indicar que aunque los pseudocódigos realizan la visualización según se van recibiendo datos, esto no tiene porqué hacerse así. Se puede mantener de manera independiente la visualización de la recepción.

4.4. Resultados

La idea propuesta se ha implementado y se han tomado datos con respecto al tamaño de la información que se transmite en cada nivel, el tiempo que se emplea en realizar una visualización en cada nivel y el número de triángulos que se generan. También mostramos las imágenes del volumen visualizado.

El volumen usado tiene un tamaño de $220 \times 230 \times 220$. Mostramos los datos e imágenes obtenidos en la tabla 4.3 y figuras 4.8, 4.9 y 4.10.

En la tabla 4.3 la primera columna del tamaño alude a la cantidad de información que se transmite en cada nivel, mientras que la segunda columna de tamaño alude a la cantidad de información que se ha transmitido hasta el momento desde el principio. Las dos columnas de tiempo hacen alusión al tiempo empleado en generar los triángulos de la isosuperficie y al tiempo empleado en proyectar los triángulos generados anteriormente.

Mostramos también una gráfica donde se muestra el tiempo de proyección de triángulos en función del tamaño del modelo (figura 4.11). Cuando el tiempo de proyección es muy alto dificulta la visualización interactiva cuando el usuario, por ejemplo, reposiciona la cámara. Para mantener el frame rate en estos casos, se puede

Nivel	Tamaño (bytes)	Tamaño total (bytes)	Tiempo (ms) generación	Tiempo (ms) Proyección	Triángulos
0	10	10	0.0	0.0	0
1	78	88	0.0	0.4	14
2	624	712	0.1	0.8	85
3	4.752	5.464	0.3	1.5	386
4	28.776	34.240	1.6	3.5	1.702
5	186.856	221.096	7.0	8.8	7.640
6	1.141.012	1.362.108	29.7	30.1	34.050
7	7.339.684	8.701.792	125.1	115.7	150.576
8	21.896.104	30.597.896	694.6	459.0	658.766

Tabla 4.3: Datos de la transmisión

implementar el tener almacenadas dos mallas de triángulos para el modelo a diferente resolución. Cuando no se pueda mantener el frame rate a la mayor resolución ante una acción del usuario se puede mostrar la malla de peor resolución temporalmente mientras dura la operación del usuario, restableciendo la malla de mayor resolución al finalizar.

En la figura 4.12 se muestra gráficamente los instantes de tiempo en los que está disponible el modelo a cada nivel de resolución. Suponemos que la velocidad de transferencia es constante, por tanto, hemos tomado como referencia para confeccionar la gráfica el tamaño del modelo a cada resolución.

Por último, hemos querido mostrar en la figura 4.13 las imágenes de los últimos 4 niveles coloreadas en función del error cometido en la isosuperficie al estar ésta a menor resolución. La tabla 4.4 muestra el error cometido en diferencia de valor (*Error 1*), estimación de distancia según el criterio usado en el capítulo 3 (*Error 3*) y la *proporción* de vértices a los que se les ha estimado la distancia usando la mitad de la diagonal de la celda donde se encontraban.

Puede verse que con poco más de 1 Mbyte transmitido (hasta el nivel 6) se obtiene una visualización aceptable y con un tiempo corto de visualización. Lo que permite al usuario realizar un trabajo previo sobre el volumen teniendo en cuenta que posee el volumen completo de modo local con lo que un cambio en la cota de visualiza-

Nivel	Error 1	Error 3	Proporción
5	14.48	3.20	34.33 %
6	9.72	1.28	19.22 %
7	4.41	0.34	6.48 %
8	1.05	0.07	2.01 %

Tabla 4.4: Error cometido por niveles

ción no exige nuevas transmisiones. Estas se harán cuando el usuario necesite mayor resolución. El sistema permite visualización rápida de volúmenes complejos.

4.5. Conclusiones

En este capítulo hemos presentado el uso del octree de celdas como representación de volúmenes para la transmisión progresiva del mismo. La transmisión se realiza con el proceso de evitación de cracks (criterio de poda por monotonía del grupo de celdas y modificación de valores) ya realizado en el servidor, con lo que el cliente no debe preocuparse de ello. No obstante, hay que tener en cuenta que debido a que los primeros niveles representan el volumen con una resolución exponencialmente peor que el volumen original, las visualizaciones obtenidas presentan poco más que una envolvente del volumen transmitido. No obstante, ya permite al usuario seleccionar una zona de estudio y recibir a mayor resolución solo un subvolumen. Además el volumen queda almacenado de manera local, los cambios en el umbral de visualización no exigen nuevas transmisiones.

```
recibir (raiz);
crearArbol (raiz);
ultimoCola (iniciaNivel);
if (esNodoInterno (raiz))
   ultimoCola (raiz);
salir = FALSE;
do {
   elemento = primeroCola ();
   if (elemento == iniciaNivel)
      if (!vaciaCola())
         ultimoCola (iniciaNivel);
      else
         salir = TRUE;
   else {   /* elemento es un nodo interno */
      for each hijo de elemento {
         recibir (hijo);
         colgar hijo de elemento;
         if (esNodoInterno (hijo))
            ultimoCola (hijo);
      }
   }
} while (!salir)
```

Figura 4.5: Recepción del árbol

```
 1:   enviar (MAXX, MAXY, MAXZ);
 2:   dimArbol = calculaDistancia (MAXX,MAXY,MAXZ);
 3:
 4:   call enviarValoresNivelRaiz ()
 5:   enviar (raiz);
 6:
 7:   ultimoCola (iniciaNivel);
 8:
 9:   otroNivel = TRUE;
10:   if (esNodoInterno (raiz))
11:      ultimoCola (raiz);
12:   else if (esNodoHojaUna (raiz))
13:      otroNivel = FALSE;
14:
15:   salir = FALSE;
16:   do {
17:      elemento = primeroCola ();
18:      if (elemento == iniciaNivel)
19:         if (!vaciaCola()) {
20:            call enviarValoresNivel();
21:            ultimoCola (iniciaNivel);
22:         } else
23:            salir = TRUE;
24:      else {   /* elemento es un nodo interno */
25:         for each hijo de elemento {
26:            enviar (hijo);
27:            if (esNodoInterno (hijo))
28:               ultimoCola (hijo);
29:         }
30:      }
31:   } while (!salir)
32:
33:   if (otroNivel)
34:      call enviarValoresNivel();
```

Figura 4.6: Pseudocódigo global de transmisión

```
 1:   recibir (MAXX, MAXY, MAXZ);
 2:   dimArbol = calculaDistancia (MAXX,MAXY,MAXZ);
 3:
 4:   call recibirValoresNivelRaiz ()
 5:   recibir (raiz);
 6:   crearArbol (raiz);
 7:   if (esProcesable (celda(raiz))
 8:      procesar (celda);
 9:
10:   ultimoCola (iniciaNivel);
11:
12:   if (esNodoInterno (raiz) || esNodoHojaMás(raiz))
13:      ultimoCola (raiz);
14:
15:   salir = FALSE;
16:   do {
17:      elemento = primeroCola ();
18:      if (elemento == iniciaNivel)
19:         if (!vaciaCola()) {
20:            call recibirValoresNivel();
21:            ultimoCola (iniciaNivel);
22:         } else
23:            salir = TRUE;
24:      else if (esNodoInterno (elemento)) {
25:         for each hijo de elmento {
26:            recibir (hijo);
27:            colgar hijo de elemento;
28:            if (esProcesable (celda(hijo))
29:               procesar (celda)
30:            if (esNodoInterno (hijo) || esNodoHojaMás(hijo))
31:               ultimoCola (hijo);
32:         }
33:      } else {   /* elemento es un nodoHojaMás */
34:         for each celda accesible por elemento
35:            if (esProcesable (celda))
36:               procesar (celda)
37:      }
38:   } while (!salir)
```

Figura 4.7: Pseudocódigo global de recepción

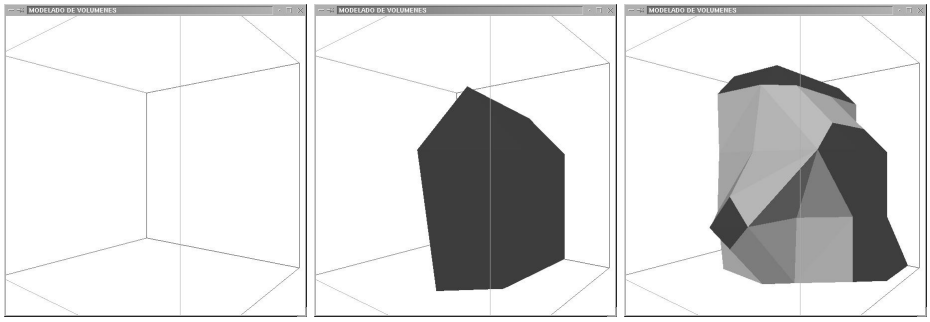


Figura 4.8: Imágenes de los niveles 0, 1 y 2

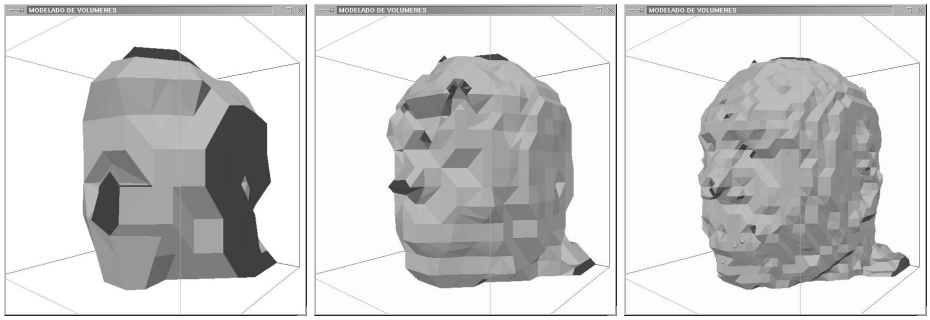


Figura 4.9: Imágenes de los niveles 3, 4 y 5

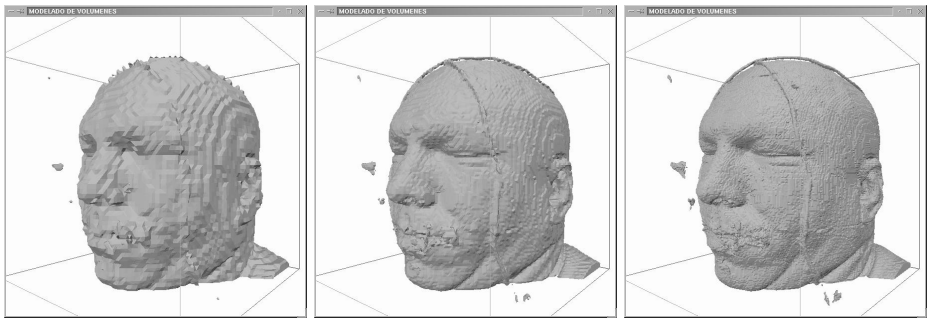


Figura 4.10: Imágenes de los niveles 6, 7 y 8

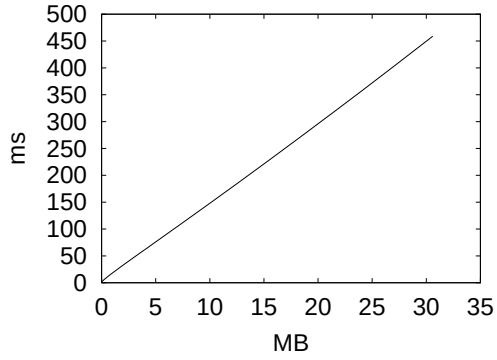


Figura 4.11: Tiempo de proyección en función del tamaño

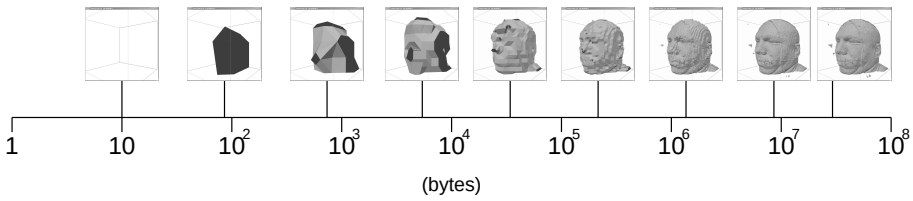
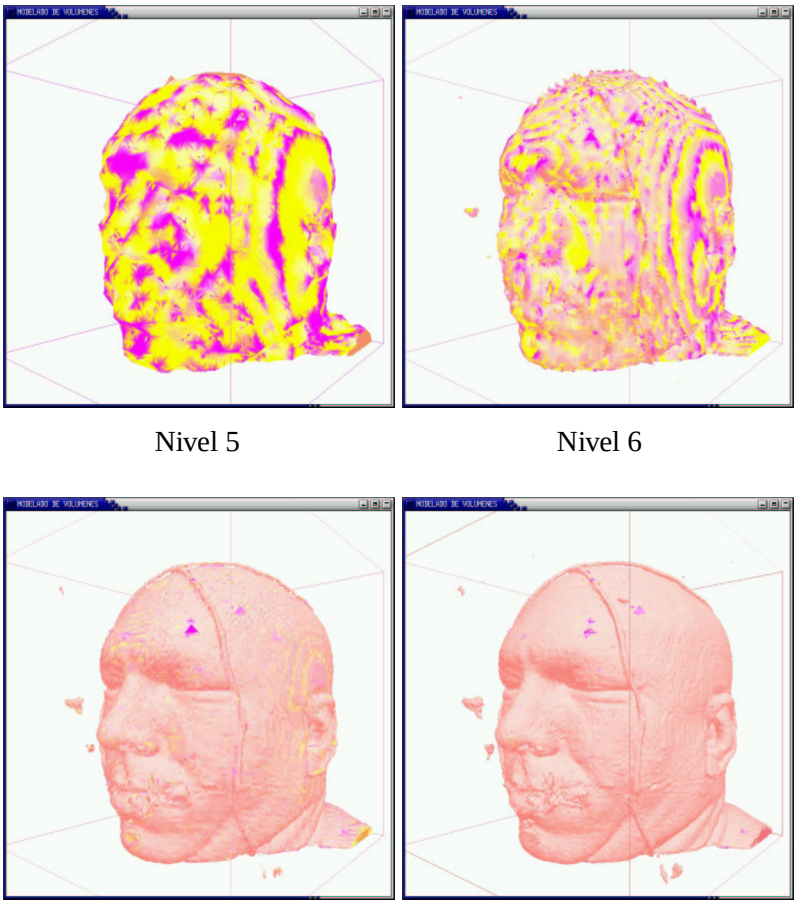


Figura 4.12: Niveles disponibles según el tiempo



Nivel 5 Nivel 6

Nivel 7 Nivel 8

Figura 4.13: Imágenes mostrando el error

Capítulo 5

Procesamiento de Aristas

El pintor no debe llevar al lienzo lo que ve,
sino lo que se verá
Paul Valéry (1871-1945)

Hemos contado en el capítulo 3 la problemática del octree de celdas en lo referente a los agujeros que pueden surgir en las isosuperficies cuando éstas atraviesan una frontera entre celdas de diferente tamaño. El problema viene ocasionado porque dicha frontera contiene información geométrica como son los vértices de los triángulos que forman la isosuperfice. Estos vértices, cuando son calculados con la información que contienen las celdas pequeñas no se suelen situar de manera coincidente con los que se calculan con la información de la celda grande.

En este capítulo proponemos un metodo para construir los triángulos de la isosuperficie que se caracteriza por no tener información geométrica en la fronteras de las celdas, por tanto, evitándose los problemas ocasionados por la no coincidencia de la información calculada desde ambos lados de la frontera [Velasco, 02a].

5.1. Presentación del método

Marching cubes se basa en obtener una triangulación de la isosuperfice procesando cada celda de manera individual, en dicho proceso, atendiendo a los valores que presentan los vértices de la celda y al valor umbral, se obtiene una triangulación de la isosuperficie en el interior de la celda. Es un proceso que podríamos nombrar como *generación de triángulos dirigido por las celdas*.

En nuestra propuesta, ese rol de dirección se lo otorgamos a las aristas. Cada arista se va a clasificar atendiendo al valor de propiedad de sus extremos, a su posición con respecto a las celdas que la comparten y al valor umbral. Una vez clasificada va a generar una triangulación de modo que juntando los triángulos generados por todas las aristas vamos a tener la triangulación de la isosuperficie. Este método, que hemos llamado *procesamiento de aristas* lo podemos llamar también, por analogía con mar-

ching cubes, *marching de aristas*. En este capítulo lo nombraremos indistintamente de ambas formas.

5.1.1. Clasificación y triangulación de aristas

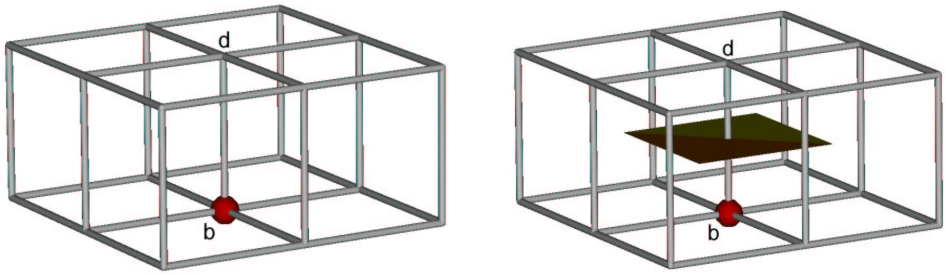
En un octree de celdas las aristas pueden clasificarse de acuerdo a los valores de propiedad de sus extremos y según su posición relativa en las celdas.

Consideremos primero el caso trivial, es decir, aquel en que ambos extremos de la arista tienen su valor de propiedad mayor que el umbral o ambos menor que el umbral. En este caso la isosuperficie no atraviesa dicha arista y por tanto no se generan triángulos.

Veamos ahora los casos que surgen para las aristas cuyos extremos tienen valores de propiedad a ambos lados del valor umbral. Por similitud con marching cubes las llamamos **aristas activas**. Estudiemos los casos que surgen en función de que la arista activa esté contenida en celdas de igual tamaño o en celdas de diferente tamaño.

Celdas de igual tamaño

Una arista compartida por celdas de igual tamaño es la mostrada en la figura 5.1 (a). La arista está contenida en la frontera de cuatro celdas; tomando un sólo punto del interior de cada celda obtenemos 4 puntos con los que construir 2 triángulos como los mostrados en la figura 5.1 (b).



(a) (b)

Figura 5.1: Arista activa en celdas de igual tamaño

Celdas de diferente tamaño

Cuando tenemos celdas de diferente tamaño nos encontramos con dos tipos de aristas. Veamos la figura 5.2.

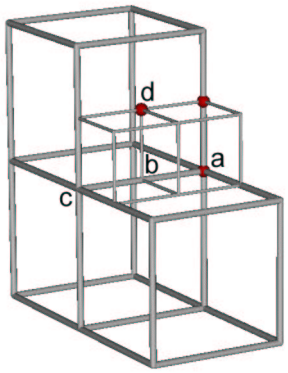


Figura 5.2: Arista activa en celdas de diferente tamaño

Podemos encontrarnos con aristas grandes que contienen a aristas pequeñas, como es el caso de la arista $\overline{ac}$ de la celda grande que contiene a la arista $\overline{ab}$ de la celda pequeña. En estas situaciones sólo se clasificarán las aristas contenidas en lugar de la contenedora. La clasificación depende del número de celdas vecinas a dicha arista.

Con este planteamiento, una arista puede ser clasificada en dos tipos:

1. Aquella que está en la frontera de 4 celdas, como la arista $\overline{ab}$.
2. Aquella que está en la frontera de 3 celdas, como la arista $\overline{bd}$.

La triangulación para el caso 1 se hace como en el caso de celdas de igual tamaño. La arista activa pertenece a la frontera de 4 celdas, cada celda aporta un punto y con los 4 puntos se construyen 2 triángulos. Véase la figura 5.3.

La triangulación para el caso 2 se hace tomando, de igual modo, un punto de cada celda que contiene a la arista en su frontera y con los tres puntos se construye 1 triángulo. Véase la figura 5.4.

Por tanto, sólo hay tres tipos distintos en los que podemos clasificar una arista.

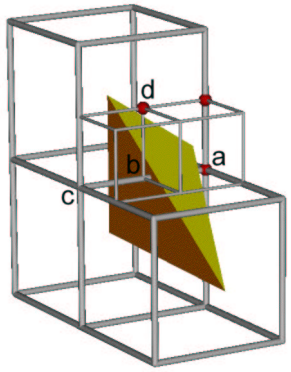


Figura 5.3: Triangulación para una arista fronteriza a 4 celdas

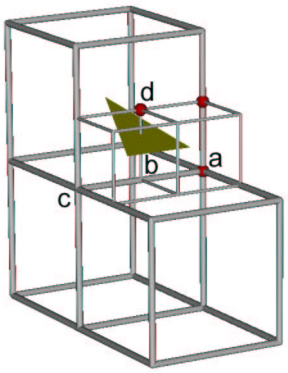


Figura 5.4: Triangulación para una arista fronteriza a 3 celdas

Clasificación de las aristas

Toda arista no contenida en una arista de mayor longitud puede clasificarse en:

1. Arista no activa: el valor umbral no está comprendido entre los valores de los extremos de la arista.
   No genera triángulos.
2. Arista activa contenida en la frontera de 3 celdas. Por ejemplo la arista $\overline{ab}$ en la figura 5.4.
   Se genera 1 triángulo.
3. Arista activa contenida en la frontera de 4 celdas. Por ejemplo la arista $\overline{bd}$ en la figura 5.3 o en la figura 5.1(b).
   Se generan 2 triángulos.

5.1.2. Etiquetado de aristas

Es necesario garantizar que no se va a procesar ninguna arista contenedora y que cada arista contenida sólo se va a procesar una vez, ya que puede pertenecer hasta a 4 celdas. Para ello se realiza un etiquetado de las celdas una vez contruído y agrupado el árbol. La etiqueta de una celda indicará que aristas se van a procesar desde esta celda y cuales no.

Este postproceso es independiente del valor umbral y del punto de vista y por tanto sólo es realizado una vez.

El etiquetado de aristas proporciona una ventaja adicional: si en una celda sólo se van a procesar las aristas indicadas por su etiqueta, sólo es necesario usar los vértices de estas aristas para determinar los valores de propiedad máximo y mínimo que se usan para la construcción del índice de acceso a las celdas. Se obtendrá un índice que descartará más celdas y ramas en el árbol que el que teníamos originalmente para obtener la triangulación mediante marching cubes.

El proceso de generación de la malla de triángulos mediante marching de aristas haciendo uso del árbol índice y del etiquetado mencionado, consiste en hacer un recorrido del árbol descartando aquellas ramas cuyo intervalo $[min, max]$ no contiene al valor umbral, y para los nodos hoja, solo se van a clasificar las aristas indicadas por su etiqueta. Sólo las aristas clasificadas como activas generarán uno o dos triángulos según la clasificación mostrada anteriormente.

5.1.3. Concepto de isopunto

En todos los casos se ha usado de cada celda un punto para contruir los triángulos. Este punto, que denominamos **isopunto**, es único para cada celda y es el punto con el que contribuye esa celda para la triangulación de todas sus aristas activas.

5.1.4. Ausencia de agujeros

Por la forma en que se construyen los triángulos se garantiza la no existencia de agujeros en la frontera entre celdas de diferente tamaño. Ello es debido a que los vértices de los triángulos (isopuntos) se encuentran en el interior de las celdas y además son únicos para cada celda, con lo que no hay que hacer coincidir ningún tipo de información en la frontera problemática.

Recordemos el ejemplo que usábamos en el capítulo 3 y que reproducimos en las figura 5.5.

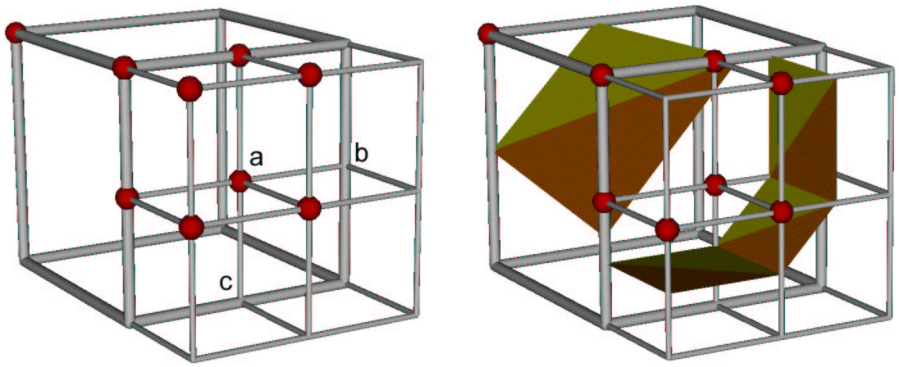


Figura 5.5: Agujero por triangulación mediante marching cubes

Cuando triangulamos mediante marching de aristas, las dos aristas de tipo 2, $\overline{ab}$ y $\overline{ac}$, generan un triángulo cada una, como se muestra en la figura 5.6, que atraviesan la frontera de cambio de resolución.

Para triangular el resto de aristas activas que aparecen en la figura tendríamos que saber cómo son las celdas adyacentes a las mostradas. En la figura 5.7 mostramos una posible prologación de celdas vecinas con la triangulación correspondiente.

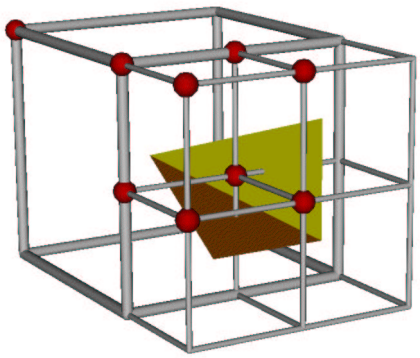


Figura 5.6: Triangulación mediante marching de aristas

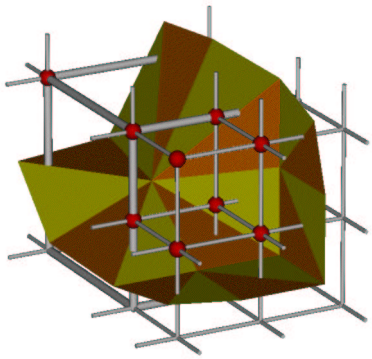


Figura 5.7: Triangulación mediante marching de aristas (y 2)

5.2. Elección de isopuntos

El último aspecto que hay que concretar con relación a este método de obtención de los triángulos es la obtención de los isopuntos, los puntos que van a representar a las celdas para la construcción de los triángulos y donde van a converger todos los triángulos que corten a la celda.

Buscando rapidez se podría considerar un punto fijo por celda, como puede ser el centro de la misma, o incluso uno de sus vértices, por ejemplo el más cercano al origen. Esta forma de elegir los isopuntos produciría isosuperficies escalonadas aunque sería rápido.

De cara a obtener isosuperficies más suaves, se puede considerar los valores de los vértices de la celda y el valor umbral de modo que la posición del isopunto venga determinado por estos valores, y por tanto no tenga una posición relativa fija en todas las celdas.

Al mismo tiempo deseamos que no se necesiten excesivos cálculos para obtener los isopuntos ya que ralentizarían el proceso.

Partimos de la base de que toda isosuperficie que corte a una celda va a cortar al menos a una de sus 4 diagonales, por tanto se puede buscar como candidatos a isopuntos aquellos puntos de las diagonales que posean el valor umbral.

La función $F(x,y,z)$ restringida a una diagonal es un polinomio de tercer grado, con lo que calcular las soluciones y determinar cuales están en el interior de la celda requiere un número elevado de cálculos, más de los que desearíamos en pro de la rapidez.

Por ello, cada polinomio de tercer grado de cada diagonal lo vamos a aproximar mediante 2 segmentos rectos: desde un vértice al centro de la celda y desde este centro al otro vértice. En concreto vamos a buscar el isopunto en una de las 8 subdiagonales que surgen de unir el centro de la celda con cada uno de sus vértices según la figura 5.8.

El proceso de calculo del isopunto que proponemos es el siguiente:

1. Calculamos el valor que le corresponde al vértice central v_c según la trilineal del interior de la celda.

2. Elegimos como subdiagonal de cálculo $\overline{v_i v_c}$ aquella a la que le corresponda el intervalo $[F(v_i), F(v_c)]$ de mayor longitud y que contiene al umbral γ_0.

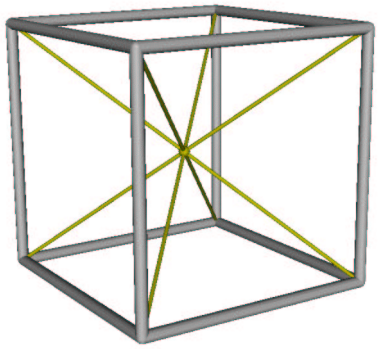


Figura 5.8: Subdiagonales para el cálculo de isopuntos

3. Calculamos el isopunto p mediante interpolación lineal en la subdiagonal elegida.

Para poder realizar un visualizado de la isosuperficie con técnicas de sombreado como por ejemplo Phong [Bui-Tuong, 75], es necesario calcular el vector gradiente que según la información del volumen en la celda le corresponde al isopunto calculado.

Dicho vector $g = (g.x, g.y, g.z)$ en el isopunto p lo calculamos como:

$$\begin{aligned} g.x(P) &= (g.x^0 \cdot (1-\Delta y) + g.x^1 \cdot \Delta y) \cdot (1-\Delta z) + \\ &\quad (g.x^2 \cdot (1-\Delta y) + g.x^3 \cdot \Delta y) \cdot \Delta z \end{aligned} \tag{5.1}$$

siendo

$$\Delta x = \frac{p.x - c.x}{c.t} \tag{5.2}$$

$$\Delta y = \frac{p.y - c.y}{c.t} \tag{5.3}$$

$$\Delta z = \frac{p.z - c.z}{c.t} \tag{5.4}$$

y

$$g.x^0 = F(c.x + c.t,\ c.y,\ c.z) - F(c.x,\ c.y,\ c.z) \tag{5.5}$$

$$g.x^1 = F(c.x + c.t,\ c.y + c.t,\ c.z) - F(c.x,\ c.y + c.t,\ c.z) \quad (5.6)$$

$$g.x^2 = F(c.x + c.t,\ c.y,\ c.z + c.t) - F(c.x,\ c.y,\ c.z + c.t) \quad (5.7)$$

$$g.x^3 = F(c.x + c.t,\ c.y + c.t,\ c.z + c.t) - F(c.x,\ c.y + c.t,\ c.z + c.t) \quad (5.8)$$

donde $(c.x, c.y, c.z)$ representa las coordenadas del vértice de la celda más cercano al origen y $c.t$ el tamaño de lado de la celda.

$g.y$ y $g.z$ se calculan de una manera similar.

Por último, indicar que los isopuntos y sus gradientes sólo es necesario calcularlos una vez, ya que se pueden almacenar temporalmente para usarlos cuando se necesiten al procesar otras aristas activas.

5.3. Comparación con marching cubes

En esta sección vamos a considerar los 15 casos del marching cubes original en lo que a triangulaciones y ambigüedades se refiere.

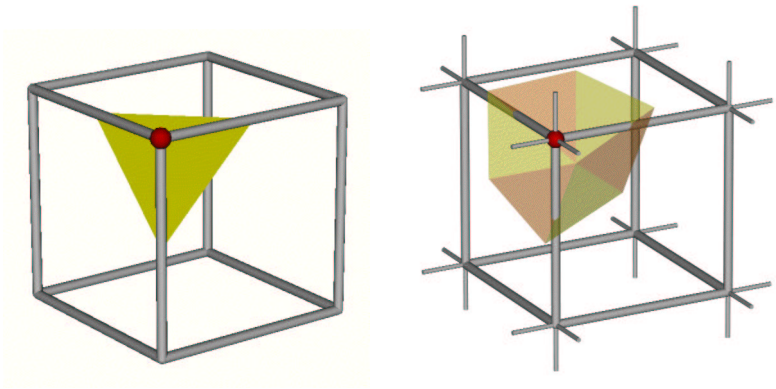
Comencemos mostrando, caso a caso, la triangulación propuesta por marching cubes y la triangulación resultante para la misma celda mediante marching de aristas realizando la triangulación de todas las aristas activas de dicha celda.

Las figuras desde la 5.9 a la 5.22 muestran la comparación caso a caso. En cada figura la imagen (a) muestra el caso marching cubes y la imagen (b) el mismo caso triangulado mediante marching de aristas. En este caso a los triángulos se le ha dado un cierto grado de transparencia que ayude a ver la totalidad de los mismos.

5.3.1. Ambigüedades

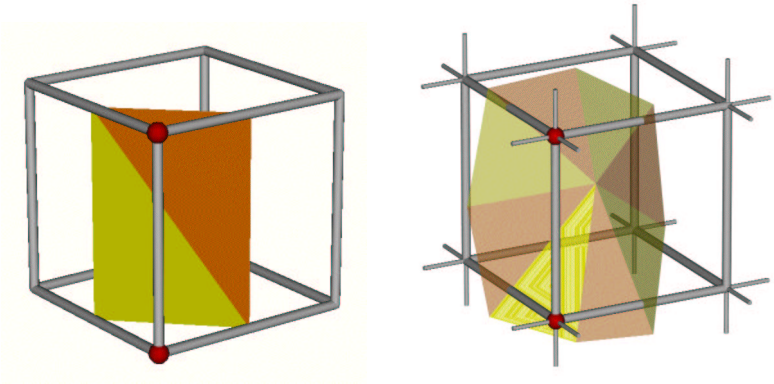
Los casos de celdas ambiguas (números 3, 4, 6, 7, 10, 12 y 13) observamos que se obtiene una triangulación que podemos considerar de compromiso entre las posibles triangulaciones que se han propuesto para las celdas ambiguas [Chernyaev, 95].

En cualquier caso, usando marching de aristas no existe ambigüedad ya que para cada arista activa, una vez determinado su tipo, sólo existe una única triangulación posible.



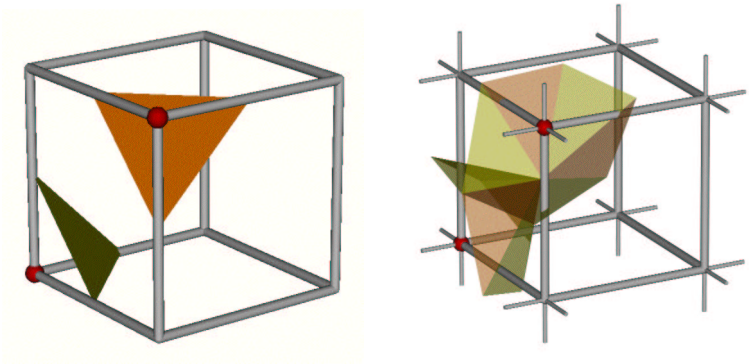
(a) (b)

Figura 5.9: Caso 1 mediante marching cubes y marching de aristas



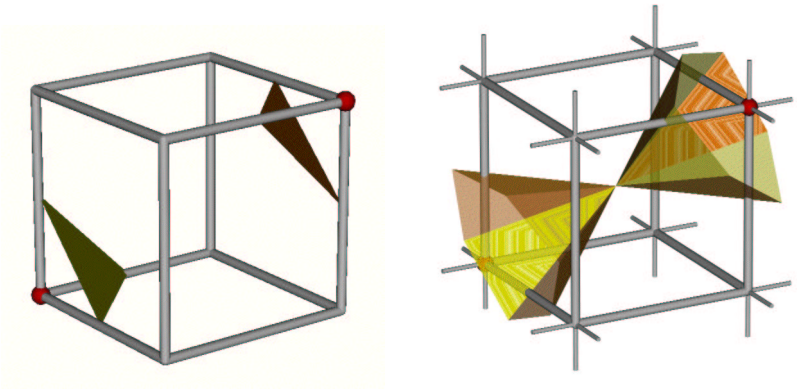
(a) (b)

Figura 5.10: Caso 2 mediante marching cubes y marching de aristas



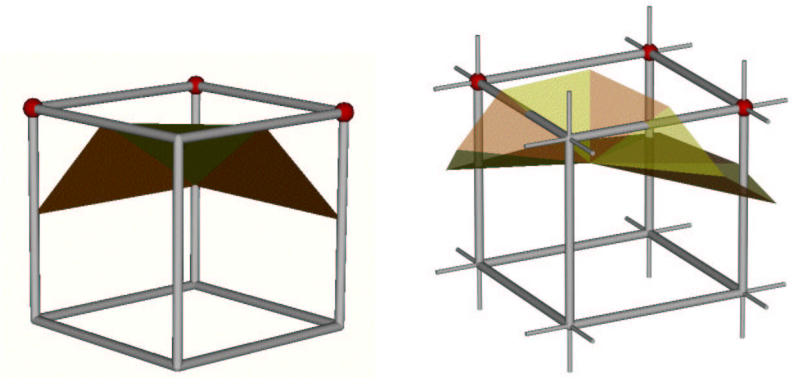
(a) (b)

Figura 5.11: Caso 3 mediante marching cubes y marching de aristas



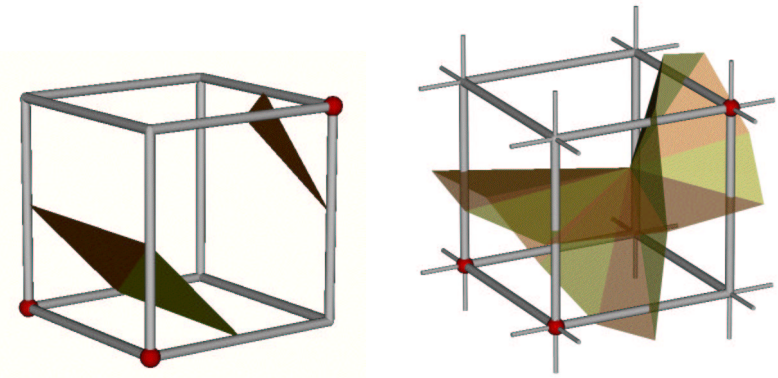
(a) (b)

Figura 5.12: Caso 4 mediante marching cubes y marching de aristas



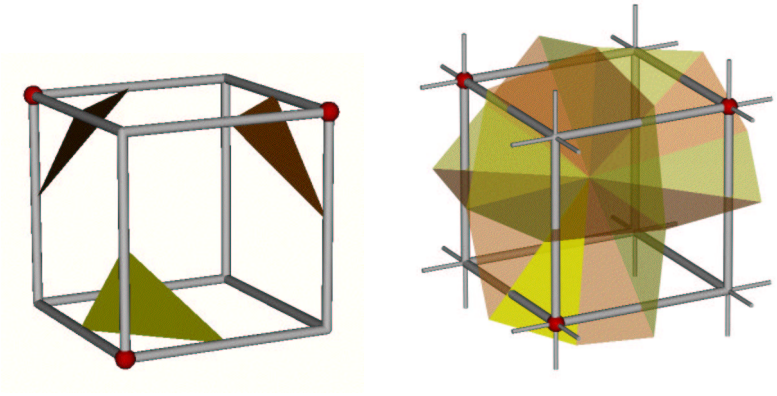
(a) (b)

Figura 5.13: Caso 5 mediante marching cubes y marching de aristas



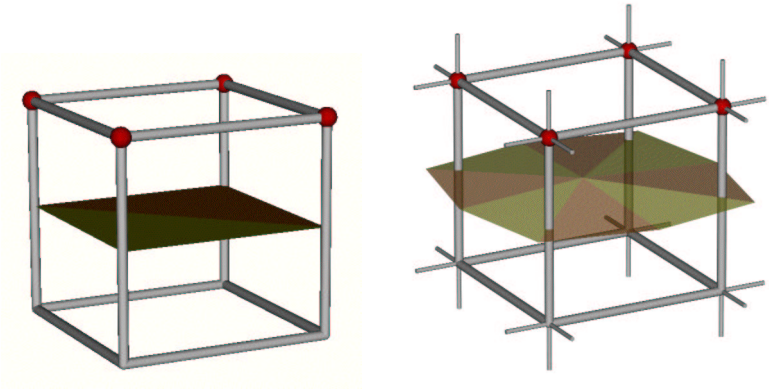
(a) (b)

Figura 5.14: Caso 6 mediante marching cubes y marching de aristas



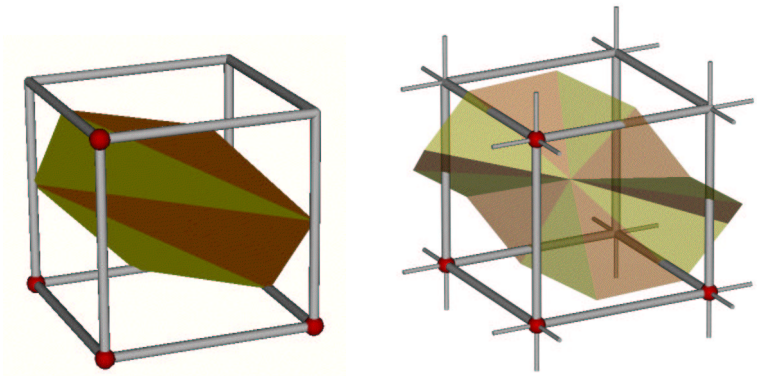
(a) (b)

Figura 5.15: Caso 7 mediante marching cubes y marching de aristas



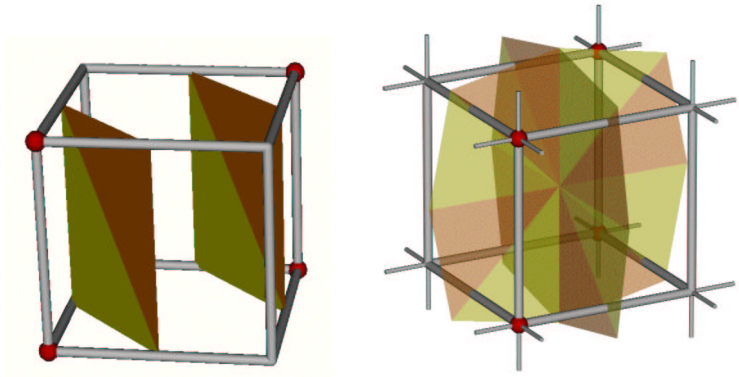
(a) (b)

Figura 5.16: Caso 8 mediante marching cubes y marching de aristas



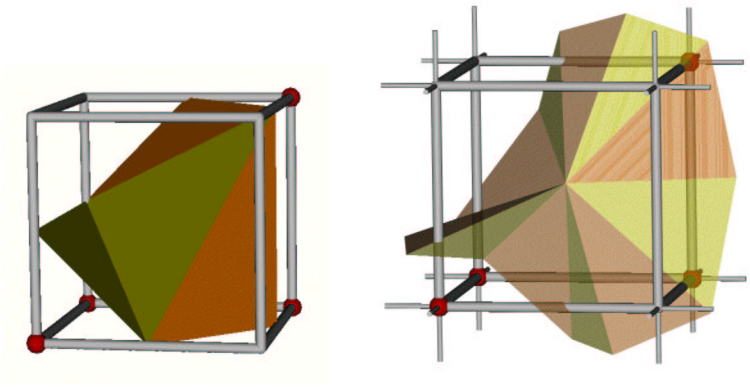
(a) (b)

Figura 5.17: Caso 9 mediante marching cubes y marching de aristas



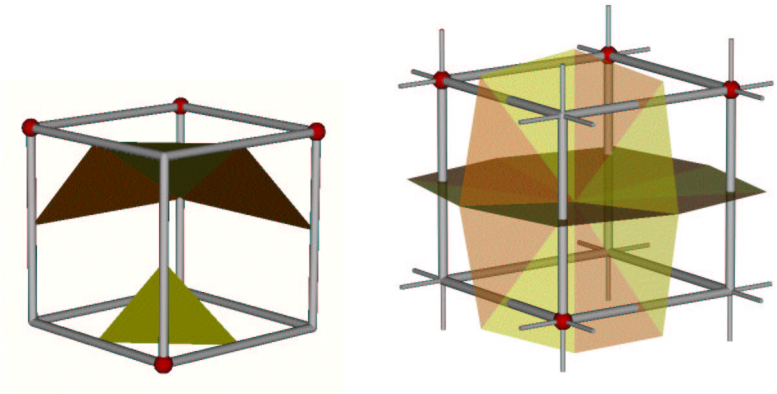
(a) (b)

Figura 5.18: Caso 10 mediante marching cubes y marching de aristas



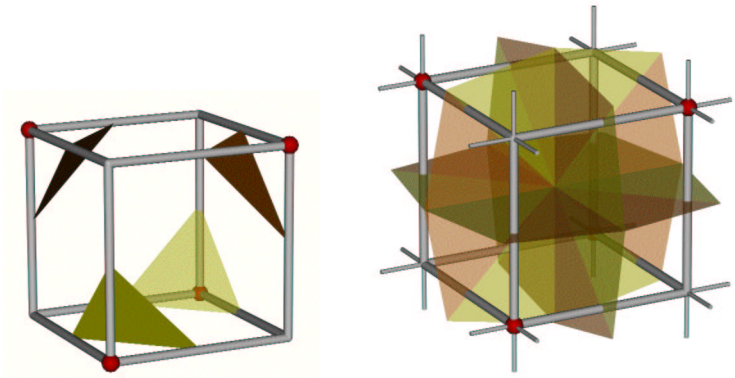
(a) (b)

Figura 5.19: Caso 11 mediante marching cubes y marching de aristas



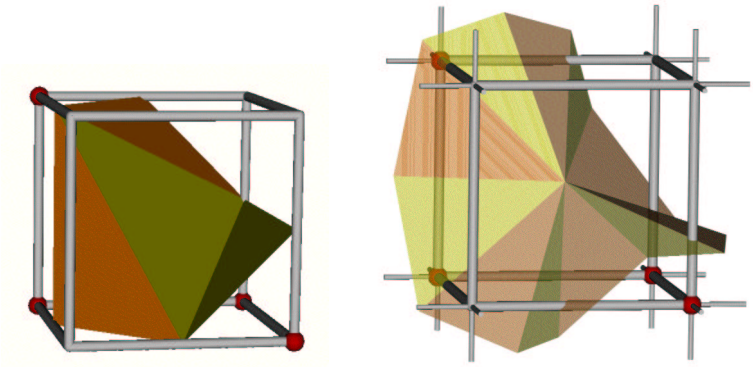
(a) (b)

Figura 5.20: Caso 12 mediante marching cubes y marching de aristas



(a) (b)

Figura 5.21: Caso 13 mediante marching cubes y marching de aristas



(a) (b)

Figura 5.22: Caso 14 mediante marching cubes y marching de aristas

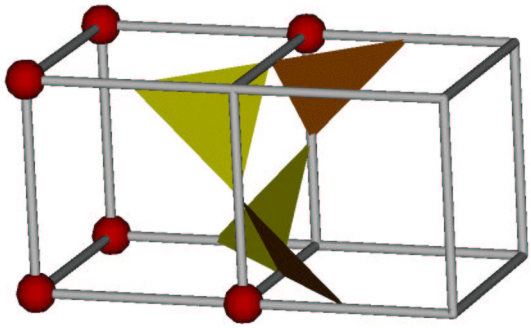


Figura 5.23: Triangulación marching cubes no adecuada

Otra de las ventajas que proporciona la *no ambigüedad* es la ausencia de agujeros en la isosuperficie que originalmente eran provocados por considerar una única triangulación por celda ambigua (la propuesta por [Lorensen, 87]). Por ejemplo el mostrado en la figura 5.23.

Ese mismo caso, mediante marching de aristas se triangularía como muestra la figura 5.24.

No se producen agujeros debido a la no ambigüedad de las aristas y la unicidad de los isopuntos.

5.3.2. Número de triángulos

A la hora de comparar el número de triángulos por ambos procedimientos debemos tener en cuenta que debido a que cada arista está compartida por 4 celdas, debemos dividir el número de triángulos que para una celda se genera mediante marching de aristas por 4. La tabla 5.1 muestra los resultados.

Se observa cómo el número de triángulos con nuestra propuesta oscila entre un 25% menos (en los casos 9, 11 y 14) hasta un 50% más (en los casos 1, 3, 4, 7 y 13). Ese gran incremento se produce principalmente en aquellas celdas que contienen algún vértice de un signo siendo los tres vértices adyacentes de signo contrario, como en el caso 1 (figura 5.25 (a)).

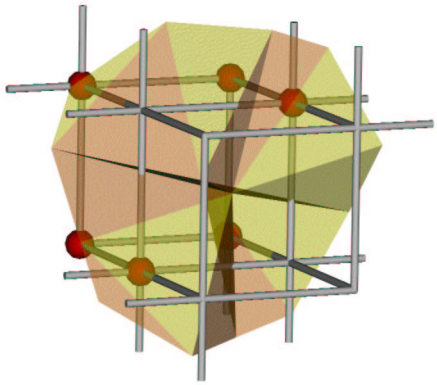


Figura 5.24: Triangulación marching de aristas para el caso anterior

En este caso, se podría hacer una retriangulación como la mostrada en la figura 5.25 (b) en la que prescidiendo del isopunto de la propia celda se obtiene la porción de isosuperficie usando $4/4 = 1.0$ triángulo.

En la actualidad estamos trabajando en esta idea, no obstante se pueden usar los métodos de retriangulación con reducción existentes como [Schroeder, 92].

5.4. Resultados

Las propuestas de este capítulo han sido probadas con los mismos volúmenes usados en el capítulo 3. Debido a que la generación de la malla de triángulos por procesamiento de aristas la genera sin agujeros, se ha usado como método de agrupación el criterio de agrupación básico, con lo que se obtiene una reducción mayor en el tamaño del árbol. Las tablas 5.2, 5.3 y 5.4 muestran los datos de tamaño del árbol, tiempo de generación de la isosuperficie y número de triángulos de la misma con las condiciones llevadas a cabo en el capítulo 3 y las condiciones planteadas en este capítulo. Los diagramas de la figura 5.26 muestran dichos datos gráficamente. Las imágenes de la figura 5.27 muestran los modelos visualizados mediante procesamiento de aristas.

Caso	M. Cubes	M. de Aristas
Caso 1	1	6/4 = 1.5
Caso 2	2	8/4 = 2.0
Caso 3	2	12/4 = 3.0
Caso 4	2	12/4 = 3.0
Caso 5	3	10/4 = 2.5
Caso 6	3	14/4 = 3.5
Caso 7	3	18/4 = 4.5
Caso 8	2	8/4 = 2.0
Caso 9	4	12/4 = 3.0
Caso 10	4	16/4 = 4.0
Caso 11	4	12/4 = 3.0
Caso 12	4	16/4 = 4.0
Caso 13	4	24/4 = 6.0
Caso 14	4	12/4 = 3.0

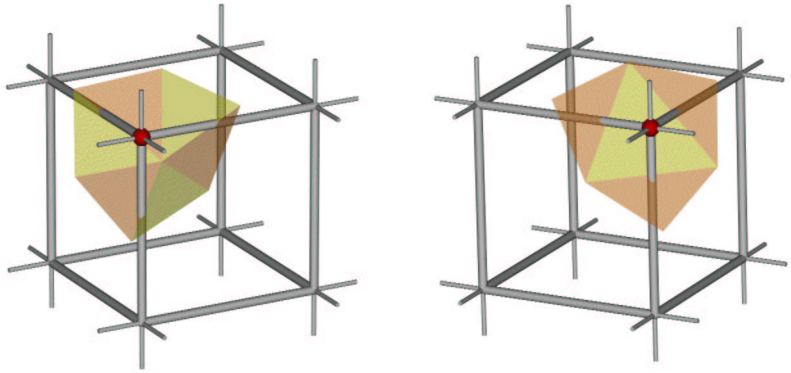
Tabla 5.1: Número de triángulos mediante marching cubes y marching de aristas

Se observa como se reduce el tamaño del árbol y el tiempo de generación de la isosuperficie, especialmente en aquellos modelos que presentan la isosuperficie con mayor suavidad. En cuanto al número de triángulos generados se ha sufrido un incremento en los modelos con isosuperficie menos suave, mientras se reduce dicho número en los modelos con isosuperficie suave.

Hemos medido de igual modo el error cometido en la isosuperficie con las mediciones ya conocidas de capítulos anteriores: diferencia de valor de propiedad (error 1) y distancia estimada en isosuperficie (error 3).

La tabla 5.5 muestra los datos obtenidos. Se incluyen así mismo los datos que para los mismos volúmenes se obtenían con las condiciones y visualización del capítulo 3.

Se observa, que la calidad de las imágenes obtenidas es buena, en tanto en cuanto la desviación media estimada de la isosuperficie es inferior a la unidad (lado de la



(a) (b)

Figura 5.25: Retriangulación para reducir el número de triángulos

celda mínima) para todos los modelos estudiados.

5.5. Conclusiones

En este capítulo se ha presentado un método de generación de la malla de triángulos para la visualización de volúmnes por extracción de isosuperfice. Dicho método opera generando triángulos arista a arista en lugar de hacerlo celda a celda como hacen los métodos empleados hasta ahora.

Con dicho método se reducen a 3 el número de configuraciones distintas a considerar, así mismo se garantiza la no existencia de agujeros en la malla de triángulos sin necesidad de realizar ningún proceso dependiente del valor umbral de visualización.

De igual modo, las celdas ambiguas obtienen una triangulación sin rupturas sin necesidad de considerar casos especiales.

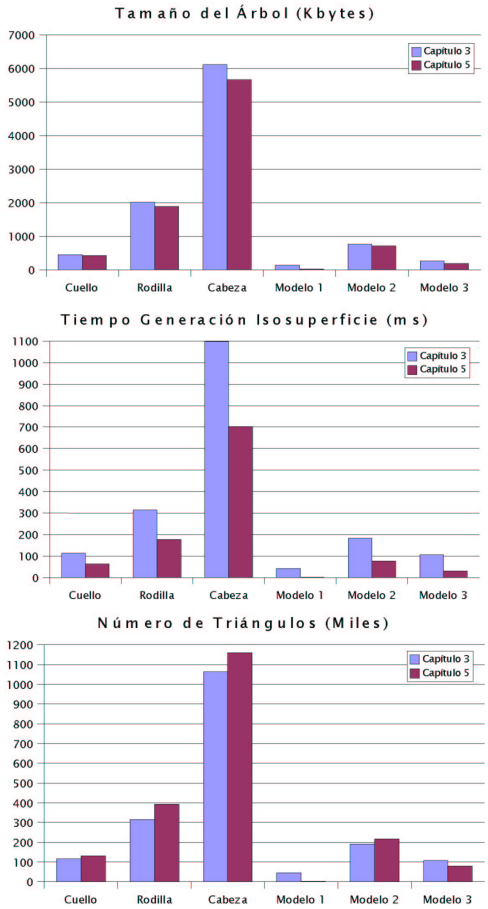


Figura 5.26: Marching cubes vs. Marching de aristas (4)

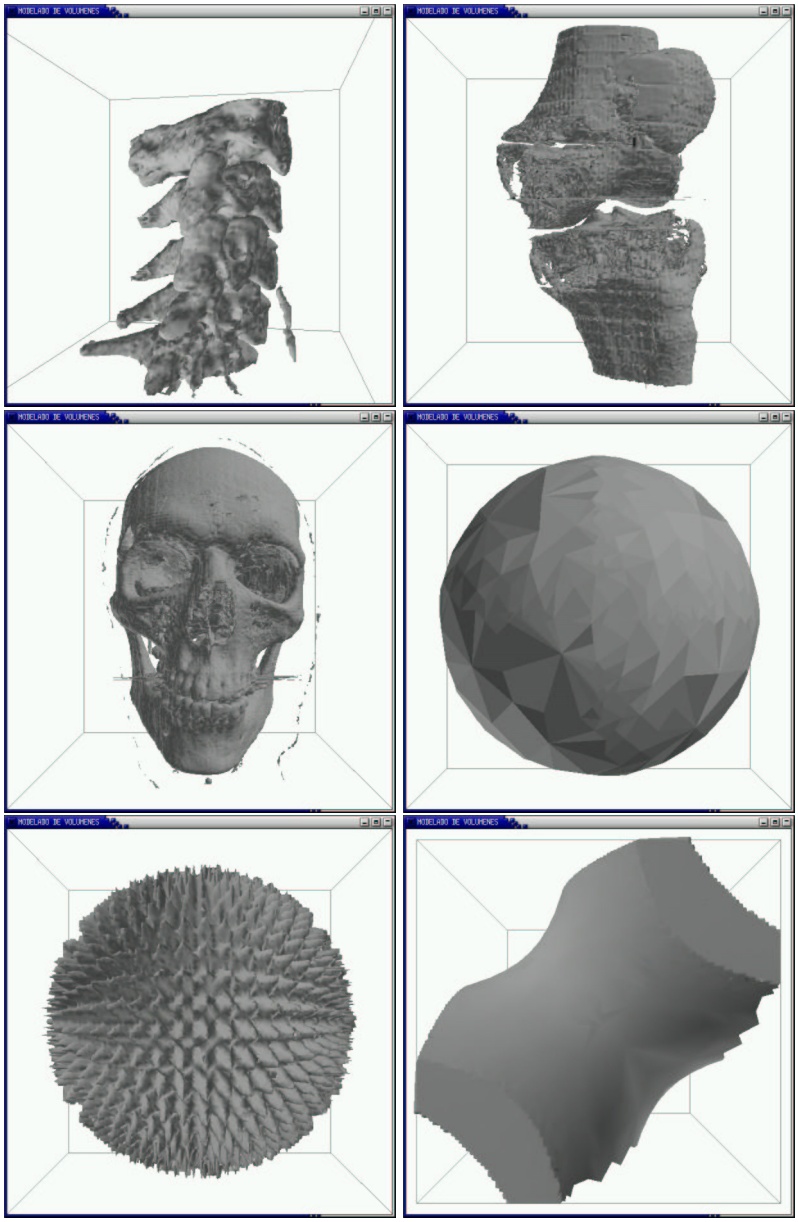


Figura 5.27: Imágenes de modelos mediante marching de aristas

	Tamaño del Árbol (bytes)	
Modelo	Capítulo 3	Capítulo 5
Cuello	466,768	435,009
Rodilla	2,061,072	1,929,227
Cabeza	6,255,952	5,806,723
Modelo 1	135,632	22,611
Modelo 2	783,056	733,343
Modelo 3	275,664	192,621

Tabla 5.2: Marching cubes vs. Marching de aristas

	Generación de la Malla (ms)	
Modelo	Capítulo 3	Capítulo 5
Cuello	113	63
Rodilla	314	178
Cabeza	1,099	703
Modelo 1	42	2
Modelo 2	184	76
Modelo 3	106	31

Tabla 5.3: Marching cubes vs. Marching de aristas (2)

	Nº de Triángulos	
Modelo	Capítulo 3	Capítulo 5
Cuello	115,670	130,944
Rodilla	315,196	392,941
Cabeza	1,063,086	1,159,252
Modelo 1	45,068	1,782
Modelo 2	190,370	216,498
Modelo 3	108,178	79,645

Tabla 5.4: Marching cubes vs. Marching de aristas (3)

Capítulo 3			
Modelo	Error 1	Error 3	Proporción
Cuello	4.18	0.38	4.01 %
Rodilla	3.40	0.38	3.49 %
Cabeza	5.81	0.31	2.62 %
Modelo 1	0.17	0.04	0.03 %
Modelo 2	1.88	0.17	4.33 %
Modelo 3	45.49	0.35	21.87 %
Capítulo 5			
Modelo	Error 1	Error 3	Proporción
Cuello	6.00	0.24	0.67 %
Rodilla	4.14	0.21	0.51 %
Cabeza	8.73	0.22	0.60 %
Modelo 1	1.96	0.04	0.00 %
Modelo 2	3.27	0.20	3.56 %
Modelo 3	40.02	0.21	1.02 %

Tabla 5.5: Error para un valor umbral concreto

Capítulo 6

Conclusiones y Trabajos Futuros

Los problemas sin resolver, no los resultados,
son los que mantienen activa la mente
Rewin Guido

Concluiremos este trabajo resumiendo las principales aportaciones realizadas y las líneas abiertas en las que seguir trabajando.

Se ha desarrollado un marco formal para describir los dominios de valores de propiedad medibles en un volumen, haciendo una diferenciación entre dominios de propiedad continuos y discretos. Se describe cómo se pueden componer dominios de propiedad, como establecer dependencias entre propiedades y cómo aplicarles atributos visuales que ayuden a su comprensión e interpretación.

Se ha desarrollado un marco formal para describir un modelo matemático de volumen, llamados modelos volumétricos, definiendo una relación de equivalencia entre modelos volumétricos, definiendo operaciones entre modelos volumétricos y realizando una clasificación de los mismos en discretos y continuos.

En el caso de los modelos volumétricos discretos se han definido las operaciones y teoremas necesarios para definir un álgebra de boole de modelos volumétricos discretos, presentando como aplicación de este resultado un esquema de representación de volúmenes discretos denominado CSG discreto.

En el caso de los modelos volumétricos continuos se definen las operaciones que lo dotan de estructura de grupo aditivo y de K-espacio vectorial. Así mismo se formaliza la representación de un modelo volumétrico continuo mediante un conjunto finito de muestras. Se presenta la posibilidad de representar el modelo volumétrico a partir de un número inferior de muestras, definiendo una medida del error y dando una condición para reducir el error cometido.

Basándonos en la idea de reducir el número de muestras para representar un volumen se presenta el esquema de representación de volúmenes denominado *octree de celdas*. Básicamente consiste en un árbol octal que permite un acceso rápido a las celdas que contienen la información del volumen. Estas celdas pueden agruparse en

celdas de mayor tamaño permitiendo representar el volumen con menos información. Se definen tres criterios de agrupación de celdas: un criterio básico, otro basado en el concepto que presentamos de *monotonía* que garantiza que la celda resultante no es ambigua [1] , y un tercero que permite obtener una isosuperficie directamente sin agujeros.

Este esquema puede ser usado para realizar la transmisión progresiva del modelo. Presentándose un modo de realizarla.

Se ha presentado un método para realizar la visualización de un octree de celdas mediante extracción de isosuperficie que se basa en el procesamiento de aristas en lugar de realizar un procesamiento de celdas. Se trata de un método simple que solo requiere considerar 2 tipos distintos de configuración de cada arista para generar la isosupercie. El método permite definir un índice de acceso a las celdas más restrictivo además de garantizar la ausencia de agujeros en la isosuperficie sin necesidad de postprocesos. Sin necesidad de hacer consideraciones de casos especiales, las celdas ambiguas obtienen una triangulación sin rupturas.

6.1. Líneas de trabajo futuro

Las líneas de trabajo que quedan abiertas y en las que continuaremos trabajando en el futuro próximo son:

- La definición formal de la combinación de dominios de propiedad discretos y continuos que nos permitan definir modelos volumétricos mixtos.
- La definición de un esquema de representación basado en el octree de celdas que pueda prescindir de la rejilla de valores.
- El estudio de nuevos criterios de agrupación.
- Estudiar los métodos de visualización directa de un octree de celdas.
- Estudiar la visualización multirresolución adaptativa de un octree de celdas.

[1] Entendiendo por ambiüedad de celdas el concepto establecido en el contexto de visualización de volúmenes mediante extracción de isosuperficie.

- Estudiar la mejora del método de generación de isosuperficies *marching de aristas* para reducir el número de triángulos generados.

Referencias

La lectura es a la inteligencia
lo que el ejercicio es al cuerpo
Richard Steele (1672-1729)

[Baer, 79] Baer, A., Eastman, C., and Henrion, M. (1979). Geometric modelling: A survey. *Computer Aided Design*, 11(5):253–272.

[Bailey, 00] Bailey, M. (2000). Manufacturing isovolumes. In Chen, M., Kaufman, A., and Yagel, R., editors, *Volume graphics*, pages 79–93. New York: Springer.

[Bajaj, 96] Bajaj, C., Pascucci, V., and Schikore, D. (1996). Fast isocontouring for improved interactivity. In *ACM Symposium on Volume Visualization*, pages 39–46,99, San Francisco, USA.

[Boada, 01] Boada, I. (2001). *Towards Multiresolution Integrated Surface and Volume Data Representations*. PhD thesis, Technical University of Catalonia, Spain.

[Brodlie, 01] Brodlie, K. and Wood, J. (2001). Recent advances in volume visualization. *Computers Graphics forum*, 20(2):125–148.

[Brunet, 85] Brunet, P. and Navazo, I. (1985). Geometric modelling using exact octree representation of polyhedral objects. In *Computer graphics forum*, volume 5. Eurographics.

[Brunet, 91] Brunet, P., Navazo, I., and Vinacua, A. (1991). Octree detection of closed compartments. *ACM*.

[Bui-Tuong, 75] Bui-Tuong, P. (1975). Ilumination for computer generated pictures. *CADM*, 18(6):311–317.

[Cano, 98] Cano, P., Velasco, F., and León, A. (1998). Modelado y visualización de volúmenes. In *I Jornadas de informática gráfica*, pages 21–30, Granada.

[Cano, 99] Cano, P., Velasco, F., and León, A. (1999). Diseño orientado a objetos de una arquitectura para modelado y visualización de volúmenes. In *Congreso español de informática gráfica (CEIG)*, pages 63–76, Jaén. Eurographics'SE.

[Cano, 96] Cano, P., Velasco, F., and Torres, J. (1996). Modelado 2d mediante jerarquía de quadtrees en distintos sistemas de coordenadas. In *II Jornadas de informática*, pages 153–162, Almuñecar, Granada.

[Chernyaev, 95] Chernyaev, E. (1995). Marching cubes 33: construccion of topologically correct isosurfaces. Technical Report CN/95-17. Avaiable as `http://wwwinfo.cern.ch/asdoc/psdir/mc.ps.gz`, CERN.

[Cignoni, 00] Cignoni, P., Ganovelli, F., Montani, C., and Scopigno, R. (2000). Reconstruction of topologically correct and adaptive trilineal surfaces. *Computer and Graphics*, 24(3):399–418.

[Cignoni, 97] Cignoni, P., Marino, P., Montani, C., Puppo, E., and Scopigno, R. (1997). Speeding up isosurface extraction using interval trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):158–170.

[Csébfalvi, 01] Csébfalvi, B. and Gröller, E. (2001). Interactive volume rendering based on a "bubble model". In *Proceedings of Graphics Interface*, pages 209–216, Ottawa, Ontario, Canada.

[Engel, 99] Engel, K., Westermann, R., and Ertl, T. (1999). Isosurface extraction techniques for web-based volume visualization. In *IEEE Visualization*, pages 139–146.

[Fiume, 89] Fiume, E. (1989). *The mathematical structure of raster graphics*. Academic Press, Boston.

[Gasparakis, 99] Gasparakis, C. (1999). Multi-resolution multi-field ray tracing: a mathematical overview. In *Proceedings of IEEE Visualization*. ACM Press.

[Giertsen, 92] Giertsen, C. (1992). Volume visualization of sparse irregular meshes. *IEEE Computer Graphics and Applications*, 12(2):40–48.

[Gouraud, 71] Gouraud, H. (1971). Continuous shading of curved surfaces. *IEEE Transactions on Computers*, C-20(6):623–629.

[Haber, 90] Haber, R. and McNabb, D. (1990). *Visualization idioms: a conceptual model for scientific visualization systems*, chapter of Visualization in Scientific Computing, pages 74–93. B. Shriver, G.M. Nielson and L.J. Rosenblum (eds).

[Jones, 94] Jones, M. and Chen, M. (1994). A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):C75–C83.

[Jung, 98] Jung, M., Park, H., and Paik, D. (1998). An analytical ray casting of volume data. In *Pacific Graphics*, pages 79–86.

[Kaufman, 94] Kaufman, A. (1994). *Trends in volume visualization and volume graphics*, chapter Scientific visualization, pages 3–19. Academic Press Ltd.

[Lacroute, 94] Lacroute, P. and Levoy, M. (1994). Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Siggraph*, pages 451–458.

[Levoy, 88] Levoy, M. (1988). Display of surfaces from volume data. *IEEE Computer Graphics & Applications*, pages 29–37.

[Levoy, 90] Levoy, M. (1990). Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261.

[Livnat, 96] Livnat, Y., Shen, H., and Johnson, C. (1996). A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84.

[Lopes, 99] Lopes, A. (1999). *Accuracy in Scientific Visualization*. PhD thesis, Univiersity of Leeds, Leeds, UK.

[Lopes, 02] Lopes, A. and Brodlie, K. (2002). Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transaction on Visualization and Computer Graphics*, Pendiente de publicación.

[Lorensen, 87] Lorensen, W. and Cline, H. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *ACM Computer Graphics*, 21(4):163–169.

[Mallgren, 82] Mallgren, W. (1982). *Formal specification of interactive graphics programming languages*. MIT Press, Cambridge.

[Mäntylä, 88] Mäntylä, M. (1988). *An Introduction to Solid Modelling*. Computer Science Press.

[Meagher, 80] Meagher, D. (1980). Octree encoding: a new tecnique for the representation, manipulation and display of arbitrary three dimensional objects by computer. Technical Report IPL-TR-80-111, Polytechnic Inst., Revisselaer.

[Montani, 94a] Montani, C., Scateni, R., and Scopigno, R. (1994a). Discretized marching cubes. In *Proceedings of Visualization*, pages 281–287, Los Alamitos, CA, USA.

[Montani, 94b] Montani, C., Scateni, R., and Scopigno, R. (1994b). A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer*, 10:353–355.

[Mroz, 00] Mroz, L., Hauser, H., and Groeller, E. (2000). Interactive high-quality maximum intensity projection. *Computer Graphics Forum*, 19(3):C341–C350.

[Mueller, 99] Mueller, K., Shareef, N., Huang, J., and Crawfis, R. (1999). High-quality splatting on rectilinear grids with efficient culling of occluded voxels. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):116–134.

[Natarajan, 94] Natarajan, B. (1994). On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11:52–62.

[Navazo, 89] Navazo, I. (1989). Extended octree representation of general solids with plane faces: Model structure and algorithms. *Computer and graphics*, 13:5–16.

[Navazo, 86] Navazo, I., Ayala, D., and Brunet, P. (1986). A geometric modeller based on the exact octree representation of polyhedra. In *Computer graphics forum*, volume 5. Eurographics.

[Novins, 92] Novins, K. and Arvo, J. (1992). Controlled precision volume integration. In *Workshop on Volume Visualization*, pages 83–89. ACM SIGGRAPH.

[Parker, 99] Parker, S., Parker, M., Livnat, Y., Sloan, P., Hansen, C., and Shirley, P. (1999). Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):238–250.

[Permingeat, 88] Permingeat, N. and Glaude, D. (1988). *Álgebra de Boole. Teoría, métodos de cálculo, aplicaciones.* Vicens Vives.

[Poston, 98] Poston, T., Wong, T., and Heng, P. (1998). Multiresolution isosurface extraction with adaptive skeleton climbing. *Computer Graphics Forum*, 17(3).

[Pyo, 02] Pyo, S. and Shin, Y. (2002). Fast volume carving. In *Proceedings of Eurographics*, pages 105–112, Saarbrücken, Germany.

[Requicha, 80] Requicha, A. (1980). Representations for rigid solids: theory, methods and systems. *Computer surveys*, 12(4).

[Requicha, 82] Requicha, A. and Voelcker, H. (1982). Solid modelling: A historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, 2(2):9–24.

[Samet, 90a] Samet, H. (1990a). *Applications of spatial data structures*. Addison-Wesley.

[Samet, 90b] Samet, H. (1990b). *The design and analysis of spatial data structures*. Addison-Wesley.

[Samet, 85] Samet, H. and Tamminen, M. (1985). Bintrees, csg trees and time. *Computer Graphics of the ACM*, 19:121–130.

[Schroeder, 92] Schroeder, W., Zarge, J., and Lorensen, W. (1992). Decimation of triangle meshes. *ACM Computer Graphics*, 25(2):65–70.

[Shekhar, 96] Shekhar, R., Fayyad, E., Yagel, R., and Cornhill, F. (1996). Octree-based decimation of marching cubes surfaces. In *Visualization'96*, pages 335–342,499, San Francisco, USA. IEEE.

[Shen, 96] Shen, H., Hansen, C., Livnat, Y., and Johnson, C. (1996). Isosurfacing in span space with utmost efficiency. In *Proceedings of IEEE Visualization*, pages 287–294. IEEE Press.

[Shirari, 81] Shirari, S. (1981). Representation of three-dimensional digital images. *ACM Computing Surveys*, 13(4):399–423.

[Shu, 95] Shu, R., Zhou, C., and Kankanhalli, M. (1995). Adaptive marching cubes. *The Visual Computer*, 11:202–217.

[Thibault, 87] Thibault, W. and Naylor, B. (1987). Set operations on polyhedra using binary partitioning trees. *ACM Computing Graphics*, 21(4):153–162.

[Torres, 95] Torres, J., Cano, P., Velasco, F., and Conde, F. (1995). Using octrees in non cartesian coordinate systems. Technical Report LSI-95-2, Dpto. Lenguajes y Sistemas Informáticos, Universidad de Granada.

[Torres, 93] Torres, J. and Clares, B. (1993). Graphics objects: A mathematical abstract model for computer graphics. *Computer Graphics Forum*, 12(5):311–327.

[Velasco, 96] Velasco, F., Rodríguez, M., and Cabrera, M. (1996). Gralpla: Un lenguaje de especificación algebraica. In *I Jornadas de trabajo en ingeniería del software*, pages 121–132, Sevilla.

[Velasco, 01a] Velasco, F. and Torres, J. (2001a). Cell octree: A new data structure for volume modeling and visualization. In *VI Fall Workshop on Vision, Modeling and Visualization*, pages 151–158, Stuttgart, Germany.

[Velasco, 01b] Velasco, F. and Torres, J. (2001b). Uso de octrees de celdas para visualización de volúmenes. In *XI Congreso Español de Informática Gráfica*, pages 99–111, Girona.

[Velasco, 02a] Velasco, F., Torres, J., and Cano, P. (2002a). Marching edges: A method for isosurface extraction. In *I Ibero-American Symposium on Computer Graphics*, pages 199–208, Guimarães, Portugal.

[Velasco, 02b] Velasco, F., Torres, J., and Leon, A. (2002b). Transmisión progresiva de octrees de celdas. Technical Report LSI-02-2, Dpto. Lenguajes y Sistemas Informáticos, Universidad de Granada.

[Westermann, 99] Westermann, R., Kobbelt, L., and Ertl, T. (1999). Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15:100–111.

[Westover, 90] Westover, L. (1990). Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–376.

[Wilhelms, 92] Wilhelms, J. and Gelder, A. V. (1992). Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227.

[Williams, 98] Williams, P., Max, N., and Stein, C. (1998). A high accuracy volume renderer for unstructured data. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):37–54.

[Yagel, 92] Yagel, R., Cohen, D., and Kaufman, A. (1992). Discrete ray tracing. *IEEE Computer Graphics and Applications*, 12(5):19–28.