Mejoras al Algoritmo de *Marching Cubes*

Andrea Silvetti[1], Claudio Delrieux[2] y Silvia Castro[1]

[1]Dpto. de Ciencias de la Computación
[2]Dpto. de Ingeniería Eléctrica
Universidad Nacional del Sur – Alem 1253 – (8000) Bahía Blanca
silvetti@criba.edu.ar

1. Introducción

Uno de los algoritmos de rendering de volúmenes más difundido y utilizado es el denominado *marching cubes,* propuesto por Lorensen y Cline en 1987 [3]. En el mismo se busca extraer una superficie umbral a partir de una matriz volumétrica de datos escalares. Una *celda* en el espacio está delimitada por los ocho valores de sus vértices. Cada celda se clasifica en distintas *configuraciones* según los valores de sus vértices respecto al valor umbral. Una celda pertenece a la superficie umbral si por lo menos uno de sus vértices está por debajo del valor umbral y por lo menos otro está por encima. En este caso, cada uno de los ocho vértices de una celda puede asumir un valor por debajo o por encima del umbral. El total de todas las configuraciones posibles es 2^8=256, pero por consideraciones de simetría se reducen a solo 15 (ver Fig. 1).

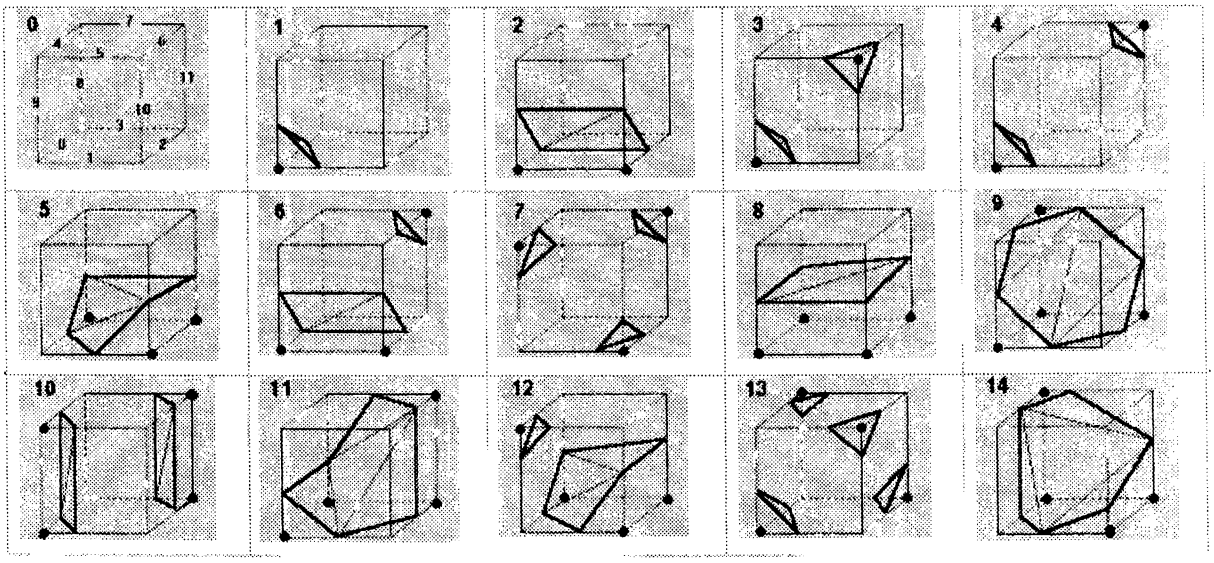

Figura 1.

Sin modificaciones en el algoritmo original [2, 6], algunos casos resultan en superficies con "agujeros"[4]. Cuando una celda tiene por lo menos una cara tal que dos de sus vértices tienen valores por encima del umbral y los otros dos por debajo, y estos vértices están diagonalmente separados, entonces es imposible decidir si el volumen pasa "por dentro" de la cara o por fuera (es decir, los vértices están unidos por el volumen o separados por un espacio vacío, ver Fig. 2(a)). Esta situación suele denominarse *cara ambigua*. No es posible determinar a priori que una configuración con caras ambiguas debe ser separada o unida. Si tal determinación fuese hecha, cuando dos celdas adyacentes comparten una cara ambigua y originalmente una de las celdas tiene a lo sumo cuatro puntos marcados mientras que la otra tiene al menos cuatro, es decir, cuando una de ellas debe invertir los puntos marcados antes de ser procesada y luego invertir las normales de los triángulos generados para esa celda, de este modo, la cara compartida es tal que en una de las celdas queda con los dos puntos

marcados separados mientras que en el otro quedan unidos. En la Fig. 2(b) se ilustra de qué manera esto sucede cuando una celda en configuración 6 es adyacente a otra en configuración 2 invertida.

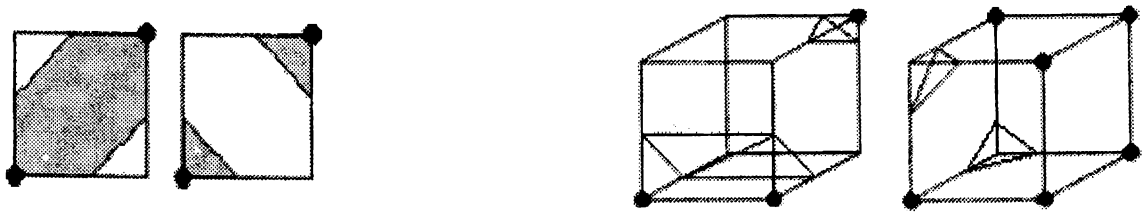

Figura 2

2. Soluciones parciales

Se ve entonces la necesidad de distinguir entre una cara ambigua separando los puntos marcados y una cara ambigua uniendo dichos puntos, es decir, hay dos posibles conexiones para aparear los cuatro puntos que dividen las aristas de la cara. Para lograr una superficie topológicamente correcta, las dos celdas en cuestión deben optar por la misma conexión y en función de esta decisión elegir la triangulación correcta en cada caso. En Nielson y Hamann [] se presenta una forma de optar por una conexión para una cara ambigua. Se usa una variación bilineal de los valores escalares en direcciones paramétricas r y s sobre la cara ambigua. El punto (r,s) podría tomarse como el centro de la cara y por lo tanto, calcular el Valor(r,s) como ¼ (Valor(0,0)+Valor(0,1)+Valor(1,0)+Valor(1,1)). Si este valor es mayor o igual al umbral, es una cara 'Unida' (también referenciada como U), de lo contrario es 'Separada' (referenciada como S). De esa manera, cada configuración con caras ambiguas debe considerar dos casos, el caso S y el caso U. Esto lleva a una proliferación de casos (50 o más), dado que varias configuraciones tienen caras ambiguas, e inclusive algunas configuraciones tienen dos o más caras ambiguas, lo que obliga a considerar cuatro casos u ocho casos para la configuración.

Pero lo peor de esta solución es que es incompleta, dado que hay configuraciones ambiguas en las que no hay caras ambiguas, por ejemplo, si tenemos dos vértices opuestos por la diagonal principal de la celda. En este caso la celda es ambigua porque podemos tener una situación U (la celda es atravesada por un delgado cilindro del volumen, o una situación S (el volumen corta dos veces a la celda, en los vértices determinados). La solución de Nielson y Hamann se podría extender para estos casos, considerando el valor en el centro de la celda como un octavo del valor de la sumatoria de los vértices, pero esto implicaría una proliferación de casos que no justificaría simplificar por condiciones de simetría.

Otra forma de manejar el problema fue propuesta por Bouvier [1]. En este caso se propone encontrar una aproximación cruda de la superficie,considerando que esta *pasa* directamente por los voxels. De esta forma, aparentemente no es necesario considerar las situaciones de ambiguedad, aunque al costo de perder exactitud y resolución en la superficie resultante.

3. Nuevas Triangulaciones

Como fuera propuesto en [5], en algunos casos, además de las seis caras que delimitan a un voxel, necesitamos considerar una cara 'diagonal' para evitar obtener superficies no esperadas (ver Fig. 3(a)). También es preciso distinguir entre caras ambiguas orientadas hacia la derecha y caras ambiguas orientadas hacia la izquierda (FIg. 3(b)).

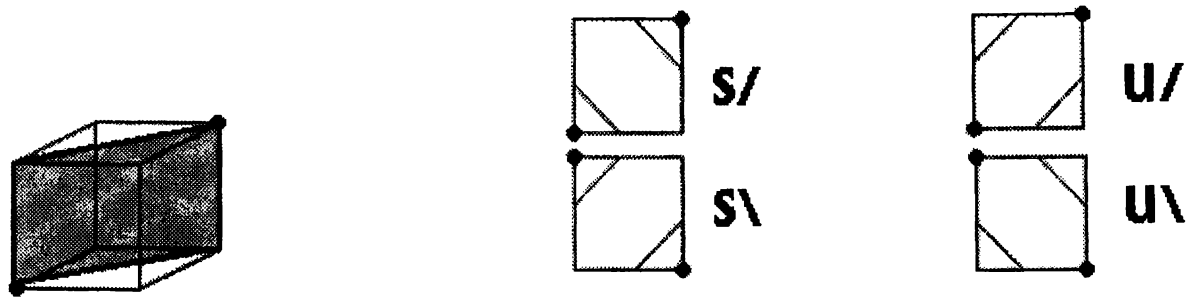

Figura 3

Una cuidadosa consideración de cada caso permite obtener la tabla con todas las posibles configuraciones de cada celda con caras ambiguas:

Configuración	N° de caras	Combinaciones posibles						
3, 4, 6	1 /	1S	1U					
7	2/ + 1\	3S	3U	2S y 1U	2U y 1S			
10, 12	1/ + 1\	2S	2U	1S y 1U	1U y 1S			
13	2/ + 4\	6S	6U	5S y 1U	1S y 5U	4S y 2U	2S y 4U	3S y 3U

De todas estas configuraciones, solamente 30 son necesarias (no mostradas aquí por razones de espacio, consultar [5]). En la Fig. 4 es posible comparar el resultado de procesar un conjunto de datos con el algoritmo original de Lorensen y Cline, y con las configuraciones aquí propuestas. Es notoria la desaparición de los "agujeros", así como la mejoría en tiempo de ejecución.

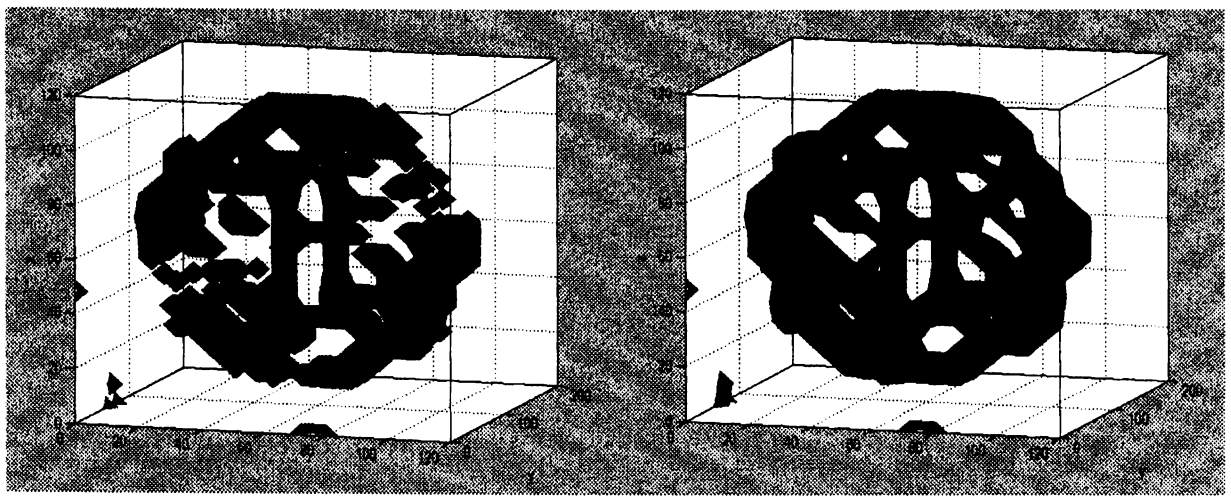

Figura 4

4. Referencias

[1] Denis Bouvier. Double-Time Cubes: A fast 3D Surface Construction Algorithm for Volume Visualization. En CISST'97, International Conference on Image Science, Systems and Technology, Las Vegas, 1997, pp.82-89.

[2] T. T. Elvins. A Survey of Algorithms for Volume Visualization. Computer Graphics 26(3), pp. 194-201, 1992.

[3] Lorensen W.E., y Cline H.E., 'Marching Cubes: A Hight-Resolution 3D Surface Construction Algorithm', SIGGRAPH 87 Conference Proceedings, *Computer Graphics*, Vol. 21, N° 4, July 1987, pp.163-169.

[4] Nielson, G. M., y Hamann B., 'The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes', Proceedings of the *IEEE* visualization '91 Conference, October 1991, 83-91

[5] Silvetti, A., Castro, S y Delrieux, C, 'Una Solución Definitiva para el Algoritmo de Marching Cubes ', En ICIE 99, International Conference on Information Engineering, Buenos Aires, 1999, pp. 759-772.

[6] Alan Watt y Mark Watt, 'Advanced Animation and Rendering Techniques', Addison-Wesley, London, 1992.