Animacion Virtual de Volumenes Tomograficos en Multiprocesadores

J. Cozar
R. Asenjo
E.L. Zapata

Septiembre 1996
Technical Report No: UMA-DAC-96/18

Published in:

VII Jornadas de Paralelismo
Santiago de Compostela, Spain, September 11-13, 1996, pp. 309-318

University of Malaga
Department of Computer Architecture
C. Tecnologico • PO Box 4114 • E-29080 Malaga • Spain

Animación Virtual de Volúmenes Tomográficos en Multiprocesadores

J.R. Cózar, R. Asenjo, E.L. Zapata

Dept. de Arquitectura de Computadores

Universidad de Málaga

email: {julian, asenjo, ezapata}@atc.ctima.uma.es

Resumen

La generación de diferentes vistas tridimensionales de los órganos del cuerpo humano a partir de los datos obtenidos mediante tomografía computerizada es una de las técnicas usadas en el diagnóstico radiológico. En este artículo se abordará su creación en computadores masivamente paralelos, mediante el *renderizado* de superficies definidas con *caras de voxeles*. Los algoritmos desarrollados son generales en el sentido de no imponer ninguna restricción al tamaño del problema y ser independientes del número de procesadores. Por último, hemos verificado experimentalmente la eficiencia de los algoritmos paralelos utilizando un sistema multiprocesador basado en el procesador Intel i860, comparando los resultados que se obtienen para las distribuciones de datos por Bloques, Cíclica o la Descomposición Recursiva Múltiple.

1 Introducción

Uno de los aspectos que complica la interpretación de las imágenes obtenidas mediante tomografía computerizada [1], es la naturaleza de las mismas, pues están formadas por una serie de cortes transversales de la zona de interés. Este problema puede solventarse mediante técnicas de segmentación, que permiten extraer de ellas objetos en tres dimensiones. La visualización puede mejorarse si estos datos se tratan con técnicas 3D (tridimensionales) de iluminación, rotaciones, variación de la opacidad, etc.

Este artículo se centra en la creación de secuencias de diferentes vistas de objetos tridimensionales, extraídos de la información proporcionada por una

tomografía. La característica más destacable de los algoritmos que generan estas imágenes es su elevado requerimiento de memoria y potencia de cálculo, con lo que, si queremos obtener los resultados en un tiempo razonable, se hace imprescindible el uso de técnicas paralelas.

El algoritmo lo vamos a desarrollar para arquitecturas paralelas con memoria distribuida del tipo paso de de mensajes. A la hora de realizar la distribución de los datos se presentan muchas alternativas posibles. Implementaremos tres de las más usadas, la partición por bloques, la cíclica y la descomposición múltiple recursiva (MRD), e intentaremos establecer las ventajas e inconvenientes que presentan cada una en estos tipos de problemas.

Los códigos secuenciales de partida se obtuvieron del *3DVIENIX* [2], un sistema software muy potente que permite realizar operaciones de visualización, manipulación y análisis sobre imágenes 2D y 3D, orientadas especialmente al diagnóstico médico. Así mismo, se utilizó su entorno para implementar el interfaz con el usuario.

La sección 2 de este artículo se dedica al estudio del algoritmo secuencial, tanto desde el punto de vista del usuario como del programador, empezándose con una visión genérica del proceso de renderizado. En la tercera sección se exponen las técnicas más usuales en cada fase del renderizado en paralelo y se justificarán las alternativas que adoptamos. Finalmente, en la sección cuarta, se hará una evaluación de los algoritmos obtenidos y se compararán las tres estrategias adoptadas en el particionado de los datos.

2 El Algoritmo Secuencial

2.1 El Proceso de Renderizado

En este apartado se pretende ofrecer un repaso general al proceso de renderizado. *Renderizado* es una jerga que viene a significar *"conjunto de operaciones necesarias para proyectar una vista de un objeto o una escena sobre una superficie de visualización"*. El objeto estará iluminado y se calculará su interacción con la fuente de luz para producir una versión iluminada de la escena.

La mejor forma de dividir el proceso completo de renderizado es considerar que la descripción del objeto se va realizando a lo largo de una serie de espacios de coordenadas. Cada uno de los diferentes espacios de coordenadas por los que se van pasando facilitan ciertos procesos y especificaciones.

Para construir una escena, Fig.1(a), a los objetos, especificados en su *sistema de coordenadas local*, se les aplican algunas transformaciones tridimensionales. De este modo, se incrustan los objetos en el *espacio de coordenadas*

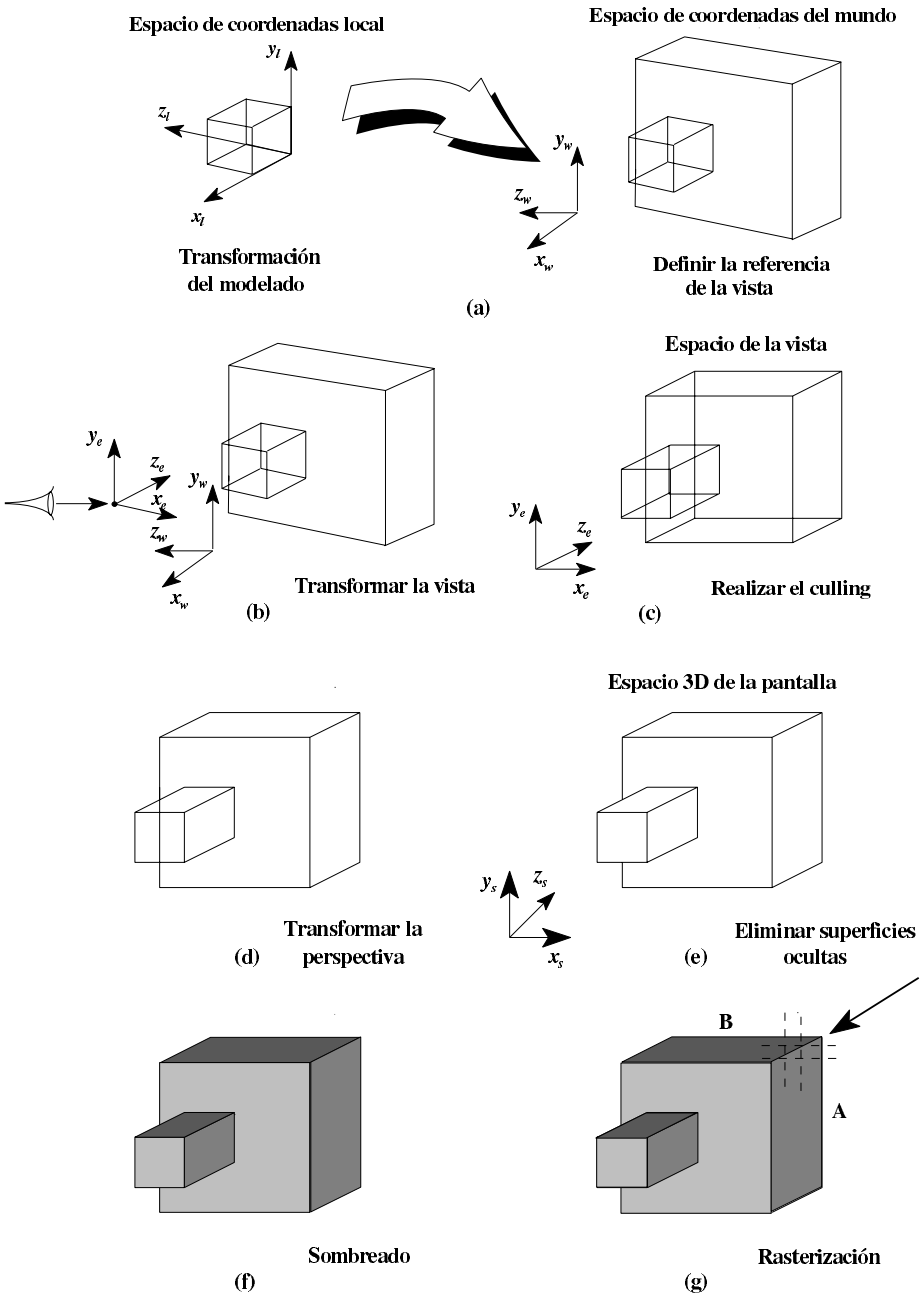


Fig. 1: *Etapas del Proceso de Renderizado*

del mundo, un espacio común a todos, Fig.1(b). En este espacio se establece la posición de la fuente o fuentes de luz que iluminan la escena y la referencia del observador.

Los objetos son transformados al *espacio de coordenadas de la vista* o el *ojo*, Fig.1(c), en el que se hacen varias especificaciones concernientes a la vista: su dirección, cómo está situado de lejos el observador de la escena, qué tipo de proyección se va a usar, etc.

En el espacio del ojo se lleva a cabo una operación que se conoce con el nombre de *culling* (escoger), Fig.1(d). Su misión es eliminar todas aquellas partes de los objetos que no sea posible ver desde el punto de vista actual. La diferencia que existe entre *culling* y eliminación de superficies ocultas puede verse comparando las Fig.1(d) y (e). Un algoritmo de *culling* determina las partes visibles de un objeto considerándolo de modo aislado, mientras que la eliminación de superficies ocultas tiene en cuenta la superposición entre sí de los objetos.

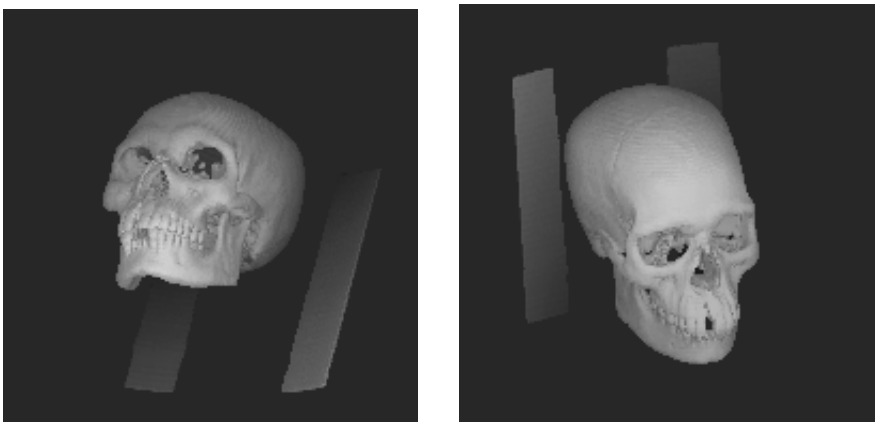
Las dos últimas operaciones que se llevan a cabo sobre los polígonos son el sombreado y la *rasterización*, normalmente en el *espacio tridimensional de la pantalla*, Fig.1(f) y (g). El sombreado compara la orientación de cada polígono con la dirección de la fuente de luz y asigna un valor de sombra al interior del polígono. La *rasterización* tiene como finalidad calcular qué valor asignar a cada *pixel* de la proyección.

Finalmente, un proceso realizado en el *espacio del ojo*, es el *clipping* (recorte), es decir, quedarnos con la parte del volumen tridimensional que queremos ver (no mostrado en la Fig.1).

2.2 Generación de Películas

Una operación bastante común en el procesado 3D de imágenes médicas [3], y muy cara computacionalmente, es la creación de secuencias de diferentes vistas de un objeto. El modo de especificarlas es el siguiente. Tras seleccionar las imágenes con las que queremos trabajar, tendremos una representación de las mismas en baja resolución. Mediante sucesivas rotaciones 3D vamos indicando las posiciones desde las que queremos ver los objetos y el numero de fotogramas intermedios. Podemos indicar también una serie de parámetros de los objetos para mejorar nuestra presentación, como el nivel de opacidad, color, tamaño, iluminación, etc.

Una vez que el usuario ha especificado la secuencia a generar, se llama a una rutina que va creando los diferentes fotogramas, los presenta por pantalla y los almacena en memoria secundaria. En la Fig.2 puede observase un ejemplo con dos vistas diferentes de un cráneo tridimensional elaboradas por el programa.



(a) (b)

Fig. 2: Ejemplos de Vistas de un Cráneo

3 Paralelización del Algoritmo

Recientemente se han desarrollado muchos algoritmos paralelos para el renderizado. La estrategia más usada en los algoritmos para paralelizar el renderizado es el paradigma divide y vencerás. Las subdivisiones pueden hacerse tanto en el espacio de los objetos como en el espacio de la imagen. Nosotros hemos usado un método híbrido, al dividir tanto el espacio de datos (durante el renderizado) como el espacio de la imagen (durante la composición).

La idea básica del algoritmo que aquí se presenta es muy simple: dividir los datos en subvolúmenes más pequeños y distribuirlos entre los múltiples procesadores, renderizarlos por separado de modo local y combinar las imágenes resultantes de un modo incremental. Los requerimientos de memoria en cada procesador son modestos, ya que cada uno contiene solo un subconjunto de los datos totales. Para cada una de las etapas se presentarán varias alternativas, justificándose las que se adoptaron . Veamos las más frecuentes.

3.1 Partición de los Datos y Balanceo de la Carga

Hay muchos modos de hacer esto. Por ejemplo, Neumann [4] comparó las tres distribuciones de datos básicas que se derivan de dividir el volumen por uno, dos o tres planos perpendiculares respectivamente. Nosotros partiremos solo una dimensión (*slices*), pues usar dos complica innecesariamente los algoritmos y las mejoras que se obtienen son insuficientes. Al ser el esquema de almacenamiento de los volúmenes disperso, usar tres es inviable.

3.2 Renderizado Local del Volumen

Cada procesador lleva a cabo un renderizado local independientemente, es decir, no se requieren comunicaciones durante el procesado de su subvolumen. A la hora de elegir el algoritmo usado para el renderizado, existen muchas soluciones posibles [5]. Nosotros usamos un algoritmo de *proyección de voxeles*, debido a los datos de los que partimos. Para implementar nuestro algoritmo de renderizado paralelo se adoptó el paradigma *host/nodo*. El *host*, además de preparar y repartir el trabajo entre los nodos y recoger sus resultados, se encargará del interfaz gráfico con el usuario. Los *nodos* serán los que realicen el proceso de renderizado en si. Su modo de proceder es el siguiente:

1. Recibir los datos del Host
2. Para cada fotograma de la secuencia
  2.1 Si hay objetos opacos:
    2.1.1 Procesarlos (rotarlos y proyectarlos)
    2.1.2 Componer los buffers opacos en el Nodo 0
    2.1.3 Hay objetos translucidos ? $\Longrightarrow$ *Radiar los buffers opacos*
  2.2 Si hay objetos translucidos:
    2.2.1 Procesarlos
    2.2.2 Componer los buffers translucidos en el Nodo 0
    2.2.3 Soy el Nodo 0 ? $\Longrightarrow$ *enviar fotograma al Host*

Para determinar correctamente la ocultación de los objetos translúcidos por los opacos, ambos tipos de objetos han de procesarse por separado.

3.3 Composición de la imagen

El paso final del algoritmo es la composición de todas las vistas parciales de los nodos en una imagen final. Las aproximaciones más usadas son:

- *Envío directo*: Hsu [6] y Neumann [4] usan esta aproximación. Se envía directamente cada *pixel* renderizado al procesador que tiene asignada la porción del plano de la imagen en la que se encuentra.
- *Composición binaria*: en cada etapa, los procesadores se van emparejando para producir una nueva subimagen, según una estructura de árbol binario. La imagen final se obtiene tras $\log n$ etapas. Un problema de este método es que hay muchos procesadores que van quedando inactivos durante el proceso de la composición.

- *Composición por intercambio binario* [7]: para explotar aún más el paralelismo, se hace una generalización del método anterior. En cada fase de la reducción, los dos procesadores involucrados en la operación de composición parten el plano de la imagen en dos partes y cada procesador se hace responsable de una.

Nosotros usamos la *composición binaria*, pues para que sea ventajoso el *intercambio binario* la operación de composición ha de ser costosa.

Se puede explotar la dispersión (*sparsity*) en los datos de la imagen si la composición se realiza solo sobre los datos de imagen que no pertenecen al fondo. Para ello, cada procesador debe de mantener un rectángulo delimitador, alineado con la pantalla, de este subárea de la imagen.

4 Evaluación

4.1 Complejidad

Una fórmula genérica para la complejidad por fotograma es:

$$O\left[opac \cdot \frac{N^3}{n} + trans \cdot \frac{N^3}{n} + (a+b+c) \cdot \log n\right]$$

en la que los dos primeros sumandos corresponden al procesado y el último a las comunicaciones. En la fórmula, *opac* y *trans* se refieren al número de objetos opacos y translúcidos; N^3 representa la tridimensionalidad de los objetos; a, b y c son constantes que dan el tiempo de comunicaciones y son distintas de cero en los casos de que haya objetos opacos, translúcidos y ambos a la vez, respectivamente; y n es el número de elementos de proceso (EPs).

- Los términos de *cálculo* son función principalmente del número de objetos, su tipo (opacos o translúcidos) y magnificación, cuántos elementos definen la superficie, posición del observador y número de EPs.

- Las *comunicaciones* dependen de bastantes variables, de un modo complejo: tipos de objetos, su forma y posición del observador (determinan el área del rectángulo que acota la proyección), tamaño del fotograma y número EPs (logarítmicamente).

La expresión de la complejidad para n=1 queda reducida al caso secuencial, ya que el algoritmo paralelo no es más que una generalización el éste.

nodo	Bloques (40563) %	Cíclica (4211) %	MDR(1D) (2337) %
0	13,5	25,2	25
1	34,7	24,9	24,6
2	28,8	23,2	24,7
3	23	25	25,7

Tab. 1: *Comparación del balanceo obtenido con las diferentes distribuciones*

4.2 Balanceo de la Carga

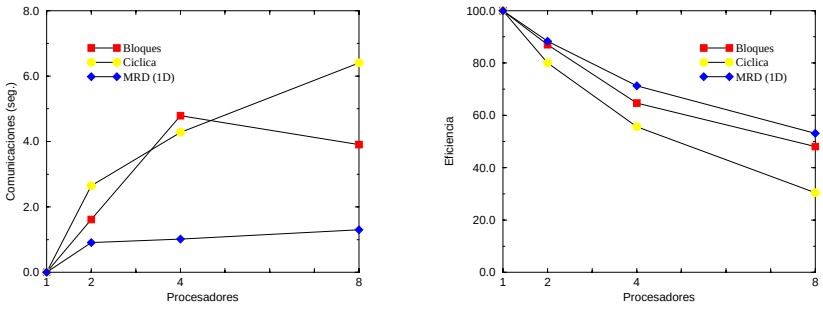
En la Tab. 1 se hace una comparación de los balanceos de carga conseguidos para las diferentes distribuciones de datos usadas. Los valores mostrados corresponden a la superficie que representa el cráneo humano (definida por 518.894 *caras de voxel*), para el caso de usar 4 nodos. Para cada nodo se indica el porcentaje de elementos que le corresponde para cada distribución. Como medida de la bondad de cada distribución, se incluye (entre paréntesis) la desviación estándar del número de elementos recibidos por cada nodo.

Observando la tabla podemos ver que la MRD(1D) distribuye mejor el uso de la memoria. La cíclica, sin embargo, reparte mejor el tiempo de cálculo. Esto se debe al hecho de que las superficies están definidas por tres tipos diferentes de *cara de voxeles* y no siempre intervienen todos los tipos en los cálculos de un fotograma. Al tomarse los elementos de *slices* consecutivos, la probabilidad de que sean del mismo tipo es mayor.

4.3 Comparación de Tiempos entre Distribuciones

Los tiempos se midieron sobre el supercomputador paralelo Paramid 16×i860 [8]. Éste está compuesto por 16 nodos o placas TTM200, cada una de las cuales está formada por un procesador Intel i860-XP dedicado al cálculo y un transputer T805 de Inmos dedicado al cálculo y/o las comunicaciones.

A continuación se muestran unas gráficas que resumen los resultados obtenidos en las pruebas para las diferentes distribuciones (Fig.3). Los tiempos corresponden a la generación de diferentes vistas de una superficie que representa un cráneo humano. Como varían bastante de unos fotogramas a otros, se generaron unos diez típicos, para estudiar el comportamiento promedio. El tamaño de la proyección, que ha de ser cuadrada, es de 256. En cuanto al número de procesadores usados, se tomaron potencias de dos, para que se tuviese el máximo rendimiento en las operaciones de comunicación en árbol binario. Los valores concretos fueron 1, 2, 4 y 8.



(a) Tiempos de Comunicación (b) Eficiencias

Fig. 3: Prestaciones del Algoritmo de Renderizado

4.3.1 Comunicaciones

La tendencia general que se observa en las comunicaciones es a aumentar con el número de procesadores (de forma logarítmica) y con el tamaño de la imagen. El pico que aparece en la distribución por bloques es debida al desbalanceo. En principio, las distribuciones consecutivas (bloques y MRD) se comportarán mejor que las alternadas (cíclica). Esto se debe a que elementos del volumen próximos se proyectarán próximos y el rectángulo que contiene a la imagen será menor, de forma que se disminuye la cantidad de mensajes a intercambiar.

4.3.2 Eficiencia

La eficiencia global del algoritmo no solo va a disminuir con el incremento del coste de las comunicaciones, sino que además, evidentemente, con el desbalanceo de la carga, por lo que conseguimos mejores rendimientos para la distribución MRD (menor coste de comunicación y menos desbalanceo). Por otro lado, las distribuciones consecutivas (bloques y MRD) consiguen un mejor aprovechamiento de la caché durante la generación de un fotograma, lo cual redunda en un incremento de la eficiencia para estas distribuciones.

Se comprobó también, cómo la eficiencia es mayor cuando en la definición del objeto interviene un mayor número de elementos.

5 Conclusiones

En este trabajo hemos estudiado principalmente el comportamiento de las diferentes distribuciones (bloques, cíclica y MRD) de forma global para este problema particular. La partición que demostró unas mejores prestaciones fue la MRD y la peor la cíclica. La repartición por bloques se encuentra en un punto intermedio, aunque más próxima a la MRD que a la cíclica. Realmente, la distribución MRD es una generalización de la distribución por bloques que optimiza el balanceo de la carga cuando trabajamos con estructuras de datos dispersas.

Referencias

[1] G.T. Herman. *Image Reconstruction from Projections. The Fundamentals of Computerized Tomography.* Academic Press, 1980.

[2] J. Udupa, R. Goncalves, K.I. Narendula, D. Odhner, S. Samarasekera y S. Sharma. *3DVIEWNIX: An Open, Transportable Software System for the Visualization and Analysis of Multidimensional, Multimodality, Multiparametric Images.* SPIE Proccedings, Vol.1897, 1991, pp. 47-58.

[3] J.K. Udupa. *The 3DVIEWNIX Software System. User Manual (version 1.0).* University of Pensilvania, Medical Imaging Group, Departament of Radiology, Philadelphia, PA, Sept. 1993.

[4] U. Neumann. *Parallel Volume-Rendering Algorithm Performance on Mesh Conected Multicomputers.* Proc. Parallel Rendering Symp., ACM, New York, 1993, pp. 253-259.

[5] A. Watt. *3D Computers Graphics.* Ed. Addison-Wesley, 2nd edition, 1994.

[6] W.M. Hsu. *Segmented Ray Casting for Data Parallel Volume Rendering.* Proc. Parallel Rendering Symp., ACM, New York, 1993, pp. 7-14.

[7] K. Ma, J.S. Painter, C.D. Hansen y M.F. Krogh. *Parallel Volume Rendering Using Binary-Swap Compositing.* Computer Graphics and Aplications, Vol.14, nº 4, 1994, pp. 59-68.

[8] R. Asenjo, M. Ujaldon. *Manual Básico del Supercomputador Paralelo Paramid.* Dpto. de Arquitectura de los Computadores, Universidad de Málaga, 1993.