Reconstrucción Iterativa de Imágenes TAC basada en GPUs*

Vicent Vidal, Liubov A. Flores, Patricia Mayo, Francisco Rodenas, Gumersindo Verdú

***Abstract*— Aunque se usa ampliamente en medicina nuclear (gamma-cameras, SPECT- single photon emission computed tomography, PET- positron emission tomography), la reconstrucción iterativa de imágenes no está difundida en tomografía axial computarizada (TAC). La mayor razón de esto es que el conjunto de datos requeridos en TAC es mucho mayor que en la medicina nuclear y la reconstrucción iterativa se hace computacionalmente muy intensa. Las unidades gráficas de procesamiento (GPUs) proporcionan la posibilidad de reducir el alto costo computacional de reconstrucción en una forma efectiva. El objetivo de este trabajo es desarrollar el algoritmo de reconstrucción de imágenes basado en GPUs.**

I. INTRODUCCIÓN

En medicina, el diagnóstico basado en imágenes TAC es fundamental en la detección de anormalidades debido a la diferente atenuación de los rayos -X, que frecuentemente no es distinguida por los radiólogos. En la tomografía axial computarizada (TAC), un conjunto de proyecciones tomadas por un escáner se usa para reconstruir la estructura interna de un objeto.

El problema de reconstrucción fue resuelto por Johan Radon en 1917 [1]. Desde entonces, los avances tecnológicos y teóricos han motivado continuas investigaciones sobre diferentes métodos de reconstrucción y sus implementaciones. En la implementación de un algoritmo es posible plantear como optimizar su ejecución y lograr mejores resultados. Por esta razón la computación paralela que distribuye los procesos de cómputo en una forma efectiva es importante. Es reconocido que las unidades de procesamiento gráfico (GPUs) pueden ser explotadas para mejorar la eficiencia computacional [2] y su uso es muy popular.

El método de retroproyección filtrada (FBP) es uno de los métodos analíticos de reconstrucción y se usa en la mayoría de escáneres como un método estándar. Es interesante notar que la implementación de FBP en GPUs ha sido investigada ampliamente en la literatura computacional [3].

Por otro lado, métodos iterativos proporcionan la reconstrucción óptima en condiciones de ruido en una imagen. En TAC, es común encontrar un conjunto incompleto de proyecciones espaciadas desigualmente. En estos casos la reconstrucción iterativa proporciona imágenes de mejor calidad ([6], [7], [8]).

La aceleración de la reconstrucción iterativa es una área de investigación activa. Stone *et al*. [9] describe el algoritmo acelerado de reconstrucción en GPUs en el área de resonancia magnética avanzada (MRI). Ellos reconstruyen imágenes de 128^3 voxeles en un poco más de un minuto. Johnson y Sofer [10] proponen un método paralelo para las aplicaciones de emisión tomográfica que explota la dispercidad y simetría de un modelo y demuestran que su método es aplicable a la mayoría de los algoritmos iterativos de reconstrucción. El tiempo de reconstrucción necesario para imágenes de 128x128x23 voxeles es mas de 3 minutos. Pratx *et al*. [11] muestran resultados de la reconstrucción iterativa usando GPUs en la tomografía de emisión de positrones (PET). El tiempo de reconstrucción necesario en una sola GPU para imágenes de 160^3 voxeles es 8.8 segundos. La implementación multi GPU acelera la reconstrucción de imágenes de 350x350x9 hasta 67 segundos en una sola GPU y 32 segundos en cuatro GPUs [12].

En nuestro trabajo previo hemos analizado el uso de PETSc (Extensive Toolkit for Scientific Computation) en la reconstrucción paralela de imágenes [13]. Se mostró que PETSc facilita considerablemente el trabajo de programación y proporciona la posibilidad de uso óptimo de un sistema en el proceso de reconstrucción. En este trabajo, nosotros presentamos un algoritmo iterativo de reconstrucción de imágenes TAC basado en GPU.

El resto del trabajo es como sigue. En la sección 2, se presentan en una forma breve los aspectos matemáticos del problema de reconstrucción y la implementación del algoritmo basada en GPUs. Los resultados experimentales se muestran en la sección 3 y finalmente en la sección 4, se resumen las conclusiones.

II. METODOLOGÍA

A. Aspectos matemáticos

Es posible considerar el problema de reconstrucción de imágenes por proyecciones como un sistema de ecuaciones lineales de la siguiente forma:

$$Ax = P, \tag{1}$$

*La investigación fue financiada por el Proyecto ANITRAN PROMETEO/2010/039.

L. A. Flores con el Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022, Valencia, España (e-mail: liuflo@posgrado.upv.es).

V. Vidal con el Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n,46022, Valencia, España (e-mail: vvidal@upv.es).

P. Mayo con Servicios Tecnológicos, Grupo Dominguis, Sorolla Center, local 10 Avda. de las Cortes Valencianas, 46015 Valencia, España (e-mail: p.mayo@titaniast.com).

F. Rodenas con el Departamento de Matemática Aplicada, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022, Valencia, España (e-mail: frodenas@mat.upv.es).

G. Verdú con el Departamento de Ingeniería Química y Nuclear, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022, Valencia, España (e-mail: gverdu@iqn.upv.es).

donde la matriz del sistema $\boldsymbol{A}$ representa el proceso de escaneo de un objeto y sus elementos dependen del número de proyecciones y del ángulo para el cual se toman las proyecciones. $\boldsymbol{x}$ es una matriz-columna cuyos valores representan intensidades de la imagen a reconstruir, y la matriz-columna $\boldsymbol{P}$ representa las proyecciones recolectadas por un escáner.

Para un ángulo dado, asumimos que el número de proyecciones varía de 1 a m. Si consideramos k diferentes ángulos en (1), entonces $\boldsymbol{P}$ es una matriz-columna con mxk elementos, $\boldsymbol{x}$ es una matriz-columna con n^2 elementos y $\boldsymbol{A}$ es una matriz rectangular de dimensión $mkxn^2$. Muchas propiedades de la imagen reconstruida dependen de las aproximaciones cuando se calcula la matriz del sistema. Hemos usado el algoritmo de Siddon para los cálculos de los elementos de esta matriz. Como se muestra, el método de Siddon proporciona buenos resultados al calcular los elementos de la matriz del sistema en un grid rectangular [14].

Hemos implementado el método Least Square QR (LSQR) para la solución del sistema (1) minimizando el residuo $\min\|Ax-P\|_2$. La matriz $\boldsymbol{A}$ es usualmente grande y dispersa, y se usa sólo para calcular productos de la forma $\boldsymbol{Av}$ y $\boldsymbol{A^Tu}$ para vectores $\boldsymbol{v}$ y $\boldsymbol{u}$.

En la práctica, $\boldsymbol{A}$ es una matriz rectangular, dispersa y por esta razón es recomendable para su almacenamiento usar el formato compacto como Compact Sparse Row (CSR) o Compact Sparse Column (CSC), que permite guardar sólo elementos no nulos. Las dimensiones de $\boldsymbol{A}$ crecen proporcionalmente a la resolución de la imagen a reconstruir y al número de proyecciones aumentando de esta forma el costo computacional.

Nosotros desarrollamos el algoritmo paralelo de reconstrucción orientado a la utilización de GPUs.

Los pasos principales del proceso de reconstrucción se presentan en la Fig. 1.

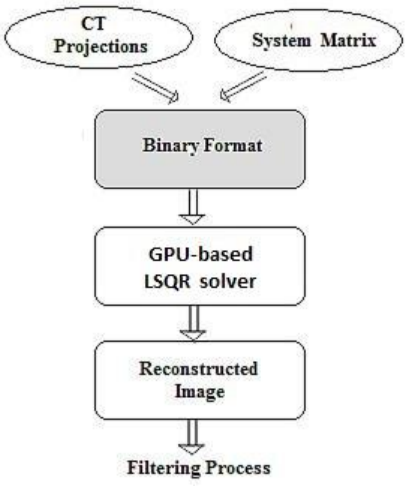

Figura 1. El proceso de reconstrucción con el método LSQR en el modelo de programación CUDA

La matriz del sistema y las proyecciones se generan previamente y se almacenan en formato binario.

Una unidad gráfica NVIDIA Tesla M2050 de procesamiento dedicada a la computación científica se usó en este trabajo para llevar a cabo los experimentos. Esta tarjeta GPU tiene en total 448 cuda cores con 3GB ECC de memoria compartida por todos los procesadores. La utilización de GPU con una gran habilidad de cómputo paralelo eleva considerablemente la eficiencia de nuestro algoritmo.

NVIDIA también introdujo CUDA™ [15], el cual es un modelo de programación paralela de propósito general que facilita la programación en NVIDIA GPUs para la solución de diferentes problemas computacionalmente complejos en una forma más eficiente que usando el CPU. CUDA permite usar C o C++ como lenguajes de programación de alto nivel.

También hemos usado librerías CUBLAS [16] y CUSPARSE [17] que permiten al usuario acceder a los recursos computacionales de NVIDIA GPU. CUBLAS es la implementación de BLAS (Basic Linear Algebra Subprograms) sobre una rutina NVIDIA® CUDA™. Para poder usar la librería CUBLAS en una aplicación, las matrices y vectores deben ser alojados en el espacio de memoria de una GPU, seguido de las llamadas a las funciones deseadas de CUBLAS, y posteriormente los resultados se transfieren de la memoria de GPU al CPU. La librería CUBLAS proporciona funciones para la transferencia de datos entre la CPU y la GPU.

La librería CUSPARSE contiene un conjunto de subrutinas básicas de álgebra lineal para efectuar operaciones con matrices dispersas y está diseñada de tal forma que las funciones pueden ser llamadas desde C o C++. Estas subrutinas incluyen operaciones entre vectores y matrices en formato disperso y denso, y también rutinas de conversión a diferentes formatos.

III. Resultados y Discuciones

Para los propósitos experimentales hemos usado proyecciones reales e imágenes de referencia adquiridos en el Hospital Clínico Universitario en Valencia. Hemos trabajado con las proyecciones fan-beam recolectadas por un escáner con 512 detectores en el rango 0 - 180 con el incremento angular de 0.9 grados. Para la reconstrucción con el método iterativo, el set inicial de proyecciones fue completado hasta 360 grados utilizando la estructura simétrica de la matriz del sistema. Para analizar la capacidad del algoritmo iterativo de reconstruir imágenes por menor número de proyecciones, del conjunto inicial de proyecciones fueron derivados tres subconjuntos con los pasos angulares de 0.9, 1.8 y 3.6 grados.

El algoritmo fue probado en una sola tarjeta GPU en un nodo del cluster Euler que pertenece a la Universidad Alicante en España. El nodo de cómputo contiene 2 x CPU Intel Xeon X5660, cada uno con 6 cores de 2.80 GHz y 3 x

GPU NVIDIA TESLA M2050 con 448 cores y 3GB de memoria.

Para imágenes de 256x256 y 512x512 píxeles, los tiempos de solución del sistema (1), con una sola CPU y una tarjeta GPU, están dados en la Tabla 1. Los tiempos en GPU corresponden al tiempo de ejecución de operaciones sólo en la tarjeta sin tomar en cuenta el tiempo de la espera en la cola. La desviación estándar de los resultados después de ejecutar la aplicación diez veces es de 2.9e-004. En la matriz del sistema, el número de filas se obtiene multiplicando el número de los detectores y ángulos usados para la reconstrucción de la imagen; el número de columnas corresponde al tamaño de la imagen reconstruida (256x256 y 512x512 píxeles).

TABLA 1. EL TIEMPO DE RECONSTRUCCIÓN DE IMÁGENES EN UNA CPU Y UNA GPU EN EL CLUSTER EULER

Matriz del Sistema (filas x columnas)	CPU (segundos)	One GPU card (segundos)
M1 = (256x100) x (256x256)	2.7	0.1569
M2 = (256x200) x (256x256)	5.3	0.3056
M3 = (256x400) x (256x256)	10.5	0.6127
M4 = (512x100) x (512x512)	12.3	0.6584
M5 = (512x200) x (512x512)	24.4	1.2741

Los resultados muestran la eficiencia del algoritmo basado en la habilidad computacional de la GPU. El SpeedUp de 19.2 fue logrado en la reconstrucción de imágenes de 512x512 píxeles. Comparando con los mejores resultados presentados en [12] (reconstrucción de imágenes de 350x350x9 píxeles requiere 67 segundos en una sola GPU), podemos ver que nuestra implementación (considerando el caso de 2D) permite la reconstrucción de imágenes con mayor resolución en menos tiempo.

TABLA II. COMPARACIÓN DE CALIDAD ENTRE IMÁGENES DE REFERENCIA Y RECONSTRUIDOS EN 512X512 PÍXELES

Nº de Ángulos	MSE	PSNR
100	0.0143	66.9300
200	0.0110	67.8019
400	0.0100	68.3378

Después de la reconstrucción se realizó la comparación cuantitativa de calidad entre imágenes de referencia y las reconstruidas por diferente número de ángulos. Los resultados se resumen en la Tabla 2. Para la comparación se usaron las funciones de Mean Square Error (MSE) y Peak Signal-to-Nose Ratio (PSNR). Los resultados muestran que el algoritmo iterativo LSQR permite la reconstrucción de imágenes de buena calidad por un menor número de ángulos y, por lo tanto, por un menor número de proyecciones. Esto puede ser muy útil en situaciones cuando la obtención del conjunto completo de proyecciones no es posible físicamente, por ejemplo en escáneres usados para realizar examinaciones urgentes en cualquier lugar. En este caso, ellos no proporcionan los datos espaciados igualmente, por lo tanto la reconstrucción iterativa es más apropiada para estos dispositivos.

La figura 3 muestra imágenes reconstruidas en paralelo usando diferentes números de proyecciones. Usualmente, las imágenes reconstruidas se someten a post-proceso (como filtrado) para mejorar la calidad. En la figura 3, las imágenes reconstruidas se presentan sin aplicación de algún filtrado.

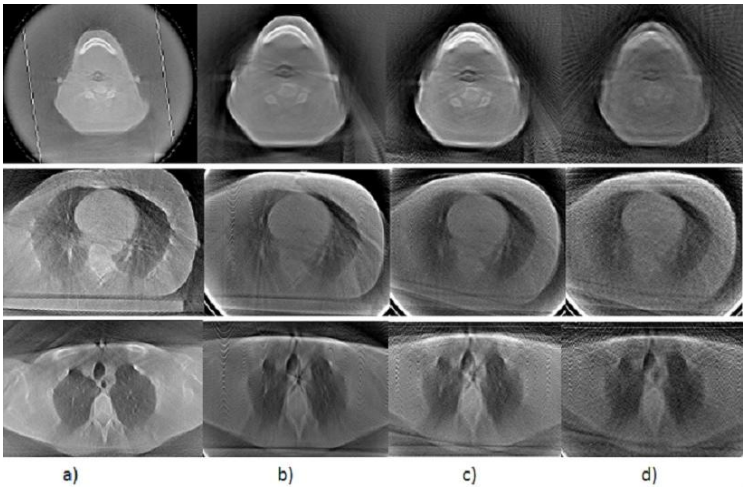

Figura 3. Imágenes reconstruidas (512x512 píxeles): a) imágenes de referencia; b), c), d) reconstrucción iterativa por 400, 200 y 100 ángulos en la iteración 12 cuando se logra la tolerancia indicada.

Finalmente, la figura 4 ilustra la capacidad del algoritmo iterativo para reconstruir imágenes por un conjunto de proyecciones incompleto y no igualmente espaciadas mientras el algoritmo FBP falla al hacer esto.

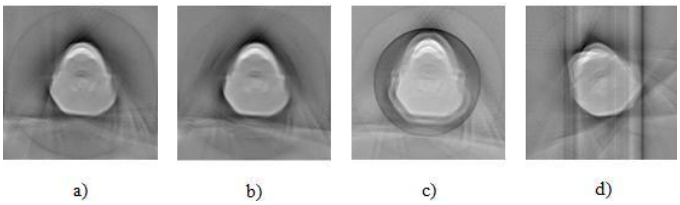

Figura 4. Reconstrucción por un conjunto incompleto de proyecciones: (a) LSQR y (c) FBP- reconstrucción con ángulos removidos (256 detectores x 170 ángulos); (b) LSQR y (d) FBP - reconstrucción con detectores removidos (226 detectores x 200 ángulos). Ángulos y detectores removidos fueron elegidos arbitrariamente.

IV. CONCLUSIONES

El algoritmo iterativo de reconstrucción de imágenes basado en GPU presentado en este trabajo muestra que los métodos iterativos son capaces de reconstruir imágenes con menor costo computacional.

El modelo de programación CUDA con las librerías CUBLAS y CUSPARSE permiten superar las dificultades en la solución de problemas computacionalmente complejos aprovechando los recursos de las unidades graficas de NVIDIA GPUs.

Se esperan resultados más significativos en la reconstrucción de imágenes 3D en donde se presenta una cantidad de cómputo enorme.

AGRADECIMIENTOS

Expresamos nuestra gratitud al Dr. Sergio Díez, Jefe de Protección de Servicios Radiológicos y Radiofísicos del Hospital Clinico Universitario, por la colaboración en este trabajo.

Asimismo a la Universidad de Alicante por permitirnos ejecutar nuestros algoritmos en su cluster Euler.

REFERENCIAS

[1] R. S. Deans, *The Radon transform and some of its applications*. Dover Publications, INC. Mineola, New York, 2007

[2] K. Mueller, F. Xu, and N. Neophytou, "Why do GPUs work so well for acceleration of CT?," in SPIE Electronic Imaging '07 (Keynote, Computational *Imaging V),San Jose, CA*, 2007.

[3] F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," IEEE Transaction of Nuclear Science, 2005

[4] G. Wang, H.Yu, and B. De Man, "An outlook on X-ray CT research and development," Medical Physics, vol. 35(3), pp. 1051-1064, Mar. 2008.

[5] B. M. Crawford and G. T. Herman,"Low-dose, large-angled cone-beam helical CT data reconstruction using algebraic reconstruction techniques," Image and Vision Comp., vol. 25, pp. 78-94, 2007.

[6] J. Nuyts, B. De Man, P. Dupont, M. Defrise, P. Suetens, and L. Mortelmans, "Iterative reconstruction for helical CT : A simulation study," Phys. Med. Biol., vol. 43, pp. 729-737, 1998.

[7] R. G. Wells, M. A. King, P. H. Simkin, P. F. Judy, A. B. Brill, H. C. Gi®ord, R. Licho, P. H. Pretorius, P. B. Schneider, and D. W. Seldin, "Comparing Filtered backprojection and ordered-subsets expectation maximization for small-lesion detection and localization in 67Ga SPECT," J. Nucl. Med, vol. 41, pp. 1391-1399, 2000.

[8] N. Sinha and J. T. W. Yeow, "Carbon nanotubes for biomedical applications," IEEE Trans. Nano., vol. 4(2), pp. 180-196, 2005.

[9] Stone S. S., Haldar J. P., Tsao S.C., Hwu W.-m W., Sutton B. P., Liang Z. P., 2008. Accelerating advanced MRI reconstructions on GPUs. Journal of Parallel and Distributed Computing, vol. 68, issue 10, 1307-1318.

[10] Johnson C.A., Sofer. A., 1999. A data-parallel algorithm for iterative tomographic image reconstruction. Frontiers of Massively Parallel Computation, pp. 126-137.

[11] Pratx G., Chinn G., Olcott P.D., Levin C. S., 2009. Fast, Accurate and Shift-Varying Line Projections for Iterative Reconstruction Using the GPU. IEEE Transactions on Medical Imaging, 28(3), pp. 435-445.

[12] Jang B, Kaeli D., Do S., Pien H., 2009. Multi GPU implementation of iterative tomographic reconstruction algorithms. Biomedical Imaging: From Nano to Macro, pp. 185-188.

[13] L. Flores, V. Vidal, P. Mayo, F. Rodenas, G. Verdú, "Fast Parallel Algorithm for CT Image Reconstruction,'' Proceedings of 34th Annual International Conference of the IEEE Engieneering in Medicine & Biology Society. August 28-September 1, 2012 San Diego, p. 4374-4377. ISBN: 978-1-4244-4120-4

[14] M. T. Cibeles Mora, "Metodos de reconstruccion volumetrica algebraica de imágenes tomograficas." PhD thesis, UPV, Valencia, Spain, 2008.

[15] http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf. Last access 11.2012.

[16] http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUBLAS_Library.pdf. Last access 11.2012.

[17] http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUSPARSE_Library.pdf. Last access 11.2012.