Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 6229-6242 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

3I: UNA HERRAMIENTA PARA LA VISUALIZACIÓN Y PROCESAMIENTO DE IMÁGENES 2D & 3D EN PARALELO

Juan P.D'Amato[a,b], Maria V.Cifuentes[a,c], Pablo Lotito[a,b]

[a] *PLADEMA, Fac. Ciencias Exactas, Universidad Nacional del Centro, Tandil, Argentina*

[b] *CONICET, Argentina*

[c] *CIC, Provincia de Buenos Aires, Argentina*

{jpdamato, cifuente, plotito}@exa.unicen.edu.ar, *http://www.plademа.net/*

Keywords: Computación gráfica, Procesamiento de Imágenes, GPU.

Abstract. Se presenta una herramienta para el procesamiento intensivo de imágenes digitales basada en unidades gráficas de procesamiento (GPU) y CPU de múltiple núcleo. La herramienta incorpora novedosos filtros para la eliminación de ruido y estimación de información faltante en imágenes digitales tridimensionales. Ambos procesamientos se integran dentro de un pipeline de evaluación reiterada de la imagen hasta alcanzar una dada convergencia. Finalmente las imágenes 3D se visualizan utilizando un algoritmo basado en ray-casting.

3I: A TOOL FOR VISUALIZING AND PROCESSING IN PARALLEL 2D & 3D IMAGES

Keywords: Computer Graphics, Image Processing, GPU.

Abstract. A tool based on graphics processing units (GPUs) and multi-core CPU for intensive processing of images is presented. Innovative filters for noise removal and estimation of missing information in three-dimensional digital images are included and integrated into a pipeline which evaluates iteratively the image reaching a given convergence. Finally, 3D images are displayed using an algorithm based on ray-casting.

1 INTRODUCCIÓN

El procesamiento de imágenes, y en especial de imágenes 3D, es una tarea que demanda importantes recursos si se desea mantener cierta interactividad con el usuario. Esta necesidad ha incentivado el uso de tecnologías en paralelo que permiten reducir los tiempos de espera, tales como la unidad de procesamiento gráfico (GPU) equipada en computadoras personales, dispositivos móviles, videoconsolas, etc. vienen equipados con unidades de procesamiento gráfico (GPU). Su arquitectura altamente paralela brinda capacidades de cálculo eficientes comparadas con las implementaciones en CPU, razón que motiva a investigadores y empresarios a la reformulación de algoritmos de procesamiento de grupos de pixeles independientes, hasta ahora reservados a las grandes computadoras.

En el caso particular de recuperación y depuración de información en imágenes existen estrategias más complejas que los filtros usuales basados en información espacial (Van Loan, 1992), que hacen indispensables el uso de las nuevas tecnologías (Fung et al., 2009; Buck, 2004; Nukada et al., 2008). Asimismo, se logran soluciones expeditivas en la segmentación de imágenes 3D (Sherbondy et al., 2003; Shenke et al., 2009; Yang et al., 2002), la simulación y visualización de imágenes de ultrasonido (kuttera et al., 2009), el análisis de video para discriminación de objetos móviles (Zivkovic et al., 2006) y el tratamiento de imágenes satelitales de múltiples bandas para la detección de zonas (D'Amato et al., 2009) .

Existen métodos aproximados que permiten integrar la información espacial-intensidad como el filtro Bilateral propuesto por (Tomasi et al., 1998) y extendido en (Elad, 2002) que pueden migrarse a GPU con ciertas variantes. Si el problema es la pérdida de información, se suele recurrir a técnicas de interpolación para intentar estimarla. En Aragone et al. (Aragone et al., 2009) se propone un filtro más robusto aplicado a la reconstrucción de mallas de superficie, con el objetivo de minimizar el máximo cambio de gradiente.

Nuestra propuesta presenta una herramienta de procesamiento intensivo de imágenes 2D y 3D basada en placas gráficas y CPU de múltiple núcleo desarrollada con el fin de incorporar filtros novedosos utilizados en la eliminación de ruido y en el cálculo de datos faltantes o incluso en la estimación del fondo en video. Para soportar estos procesamientos, se propone un pipeline de trabajo basado en la evaluación repetida de los mismos hasta lograr una convergencia deseada y una visualización basada en ray-casting. De entre los algoritmos nombrados, se seleccionaron aquellos que demostraban resultados más interesantes y que requerían una mayor capacidad de cálculo (Elad, 2009; Aragone et al.,2009). También se detallan algunas características adicionales, tales como visualización de volúmenes y discretización de mallas de superficie.

Por otra parte, el nuevo estándar de Kronos Group (Kronos Group, 2010) se espera simplifique la interacción de las tecnologías de GPU con distintos lenguajes de desarrollo, y será utilizado en este trabajo.

En la siguiente sección se describen los algoritmos principales implementados. A continuación se presenta la arquitectura de la herramienta y se describen los módulos que la integran. Finalmente se muestran algunos resultados visuales y comparativos de perfomance con respecto a la implementación en CPU.

1.1 Pipeline del GPU

Incorporar la GPU en el procesamiento de imágenes en general es una tarea natural, ya que operaciones iguales se ejecutan sobre datos con la misma estructura (arquitectura SIMD). De esta forma, el procesamiento de imágenes puede descomponerse en muchos subprocesos que trabajan con partes del dominio de datos alojados en memoria global. Con el advenimiento de CUDA y más recientemente OpenCL, el concepto y la arquitectura se replanteó, permitiendo

ahora procesar datos de distinto tipo no atados a estructuras convencionales como matrices de pixeles (ver Fig. 1). Cada procesamiento es denominado *kernel*, el cual es un pequeño programa compilado y ejecutado por la biblioteca incorporada en el driver de la GPU.

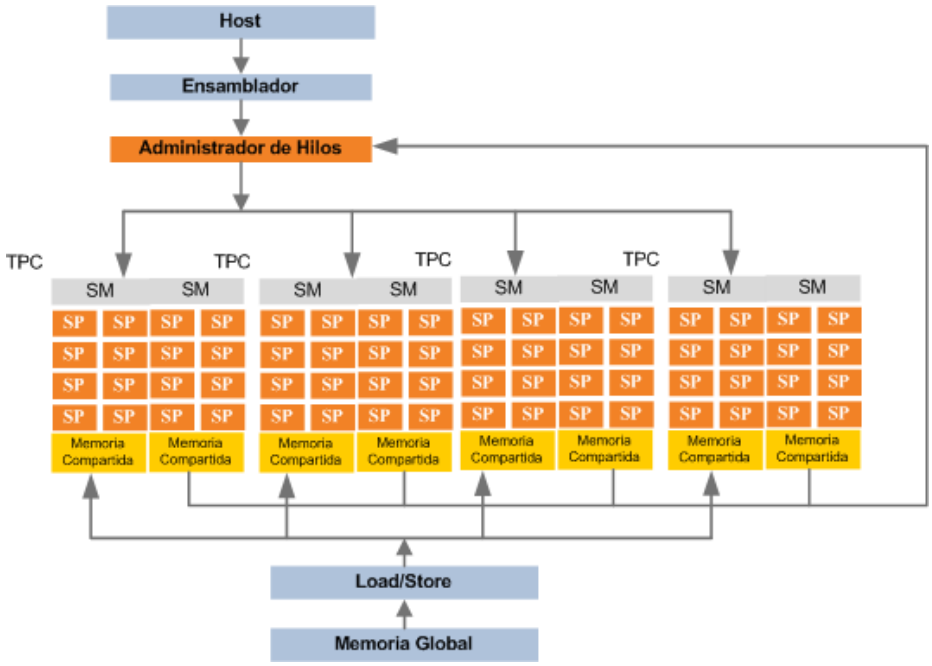

Fig. 1 Esquema de memoria y ejecución propuesto en CUDA

Con este concepto ya implantado en el medio, denominado GPGPU, herramientas de uso cotidiano como un procesador de imágenes o video en tiempo real se han visto claramente beneficiadas, facilitando a su vez la tarea a diseñadores y editores gráficos. Por otro lado, esta capacidad de cálculo incrementada lleva a demandar algoritmos más sofisticados de análisis y mejora de la información.

2 PROCESAMIENTO Y RESTAURACIÓN DE LAS IMÁGENES

A las imágenes adquiridas a través de cámaras públicas se le incorporan ruidos a consecuencia de la baja calidad en las conexiones o del movimiento propio de la cámara. Lo mismo sucede en imágenes médicas obtenidas mediante ultrasonido donde la naturaleza compleja de la señal suele traer problemas al digitalizarse. O bien cuando se provocan cambios en la resolución que circunstancialmente agregan nuevos patrones en la imagen digitalizada. Tales infiltraciones ruidosas se reducen mediante la aplicación de algoritmos precisos.

Nuestra propuesta radica en la utilización de algoritmos complejos que resuelven problemas ocasionados a partir de información espuria o hasta faltante. Básicamente, se extrajo una estructura común de ciertos algoritmos la cual definirá el pipeline propuesto a continuación.

2.1 Eliminación del ruido mediante filtros de Espacio-Temporales

En imágenes generadas a partir de tomografías computadas o similares suelen observarse patrones característicos de superficie "escalonada" (ver Fig. 2). El escalonado desaparece incorporando estrategias de suavizado como los filtros bilaterales.

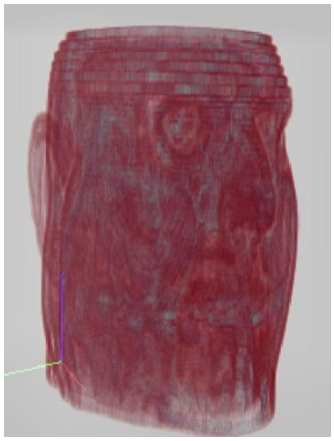

Fig. 2 Presencia de patrones escalonados en imágenes tomográficas

La eficacia del filtrado bilateral aplicado al procesamiento digital de imágenes es debida a la conservación de bordes a pesar de las importantes atenuaciones efectuadas sobre las regiones uniformes (homogéneas) de la imagen, resultantes de considerar tanto el rango de intensidades como su distancia espacial. Elad sugiere un promedio ponderado de los pixeles para remover el ruido y restaurar la señal *X*, dada la señal degradada *Y* formulado en (Tomasi et al., 1998; Elad, 2002) como

$$\hat{X}[k] = \frac{\sum_{n=-N}^{N} W[k,n]Y[k-n]}{\sum_{n=-N}^{N} W[k,n]} \qquad \text{(ec.1)}$$

Esta ecuación es un promedio ponderado normalizado de una vecindad de *[2N+1]* pixeles alrededor del pixel *k*. Los pesos *W[k,n]* se calculan incluyendo dos factores temporal (espacial en el caso de imágenes) y pesos radiométricos.

$$W[k,n] = W_S[k,n].W_R[k,n] \qquad \text{(ec.2)}$$

El primero de los factores mide la distancia geométrica entre el centro de la muestra *[k]* y el *[k,n]*, y aplica la métrica euclídea. El segundo factor mide la distancia radiométrica entre los valores del centro de la muestra *Y[k]* y *[k-n]* pixeles, y nuevamente, la métrica euclídea es elegida.

$$W_s[k,n] = \exp\left\{-\frac{d^2\{[k],[k-n]\}}{2\sigma_S^2}\right\} = \exp\left\{-\frac{n^2}{2\sigma_S^2}\right\}$$
$$W_R[k,n] = \exp\left\{-\frac{d^2\{Y[k],Y[k-n]\}}{2\sigma_R^2}\right\} = \exp\left\{-\frac{[Y[k]-Y[k-n]]^2}{2\sigma_R^2}\right\} \qquad \text{(ec.3)}$$

Aunque menos eficiente, la solución total, conocida como *brute-force*, propone resolver la (ec.1) analizando cada pixel contra el resto, lo que finalmente resulta en un costo computacional cuadrático. Considerando que los puntos alejados de un pixel no aportan demasiado, se planteó la propuesta *better brute-force* que realiza un procesamiento de pixeles

cercanos definidos dentro de una ventana de análisis, tal como se aprecia en la Fig. 3. El esquema basado en ventanas no sólo permite reducir drásticamente el tiempo de procesamiento sino que también es sencillo de implementar en la GPU.

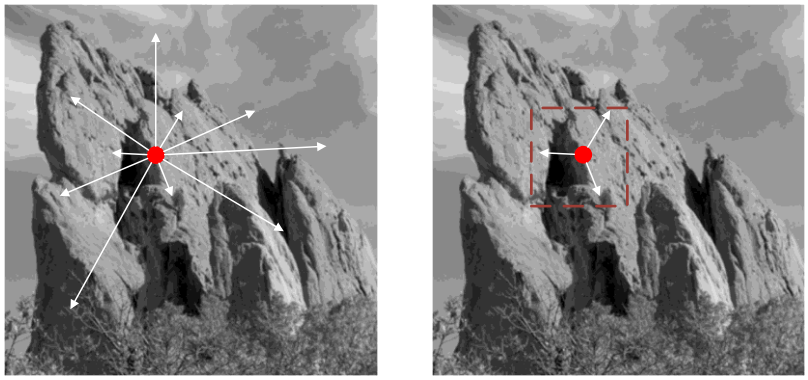

Fig. 3 Implementación de "Brute Force" (izquierda) y "Better" (derecha).

En general, las sucesivas aplicaciones del algoritmo que culminan cuando ya no son necesarias más transformaciones en los pixeles, logran mejoras en los realces de bordes como se demuestra en la Fig. 4 donde se comparan con otros métodos de filtrado.

Fig. 4 Comparación de los métodos de filtrado: Imagen original (a), filtro mediana (b) y filtro bilateral (c).

2.2 Estimación de la información faltante

Algunos casos típicos de la estimación de la información faltante se presentan luego de aplicar un cambio de escala en una imagen o introducir los frames perdidos de una secuencia de video. En la literatura se han propuesto distintos enfoques que completan la información faltante (Starck et al., 2001). Una alternativa diferente para la reconstrucción de información consiste en plantearlo como un problema de optimización con restricciones. Por ejemplo se puede buscar la superficie que pasa por los puntos dato con la mínima curvatura, o con menor gradiente.

En particular, se incluye la experiencia basada en un problema de extensión Lipchitziana optimal. Esto es, dado un conjunto de celdas con valores fijos (obtenidos a partir de los datos medidos en nuestro problema) se busca una extensión a un dominio más amplio, que minimice la constante de Lipchitz.

Una función *f* definida en un subconjunto de R^n, se dice que es lipschitciana en *x* si las variaciones locales de la función están acotadas por una constante multiplicada por la variación de la variable, más precisamente, si existe una constante *K* tal que para todo valor de *y* se tiene que

$$\left| f(x) - f(y) \right| \leq K \left| x - y \right| \qquad \text{(ec.4)}$$

La menor cota *K* que verifique la desigualdad anterior se denomina constante de Lipschitz. En el caso de funciones derivables, la constante de Lipschitz es una cota de la norma gradiente, y buscar la función de menor cota de Lipschitz equivale a buscar una función de variación mínima en el sentido del gradiente.

Este problema es la versión discreta del problema general de obtener la función lipchitciana de mínima constante de Lipchitz entre todas las funciones que con valores fijos en un subconjunto de su dominio. Dicho de otra manera, si, por ejemplo, se busca una función definida en [0, 2] y se conocen los valores que toma en 0, 1, y 2, entonces la función buscada es una función cuya constante de Lipschitz no supera la que ponen los valores prefijados, en este caso resulta seccionalmente lineal, ver Fig. 5. Para funciones definidas en la recta es casi trivial, ya que basta definir la función como lineal (afín) en cada subintervalo. Para el caso de funciones definidas en el plano, el problema es más interesante ya que por cuestiones topológicas no se puede definir un orden en el dominio de modo de construir la extensión como en el caso anterior (unidimensional).

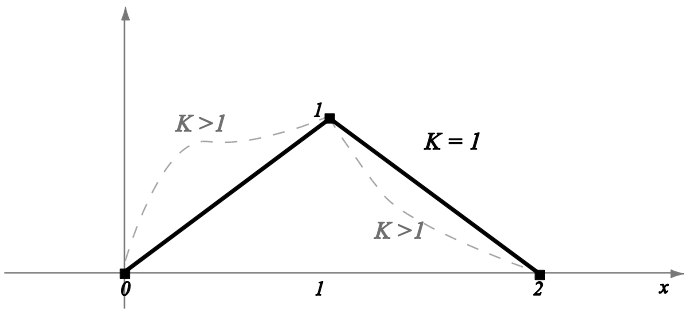

Fig. 5 Ejemplo de extensión lipschitziana unidimensional.

A su vez el problema de extensión Lipschitciana minimal es una caso particular del problema de minimizar un funcional como se explica (Garcia Bauza et al. 2008). El interés de encuadrar nuestro problema en este marco radica en poder utilizar los métodos

desarrollados en (Parente et al., 2007) para su resolución. Para ello, y siguiendo el esquema presentado en (Parente et al., 2007), utilizamos un procedimiento de discretización basado en diferencias finitas. Para el problema de estimación de información faltante, como el que se presenta en un cambio de escala, se discrimina celdas que incluyen información de las que no. Las derivadas parciales discretizadas se definen en la forma usual y utilizamos combinaciones convexas de estas diferencias parciales para definir el gradiente en puntos vecinos al borde. Definimos $F^{i,j}$ como la discretización de la función f en los pixeles k. Se puede ver en (Aragone et al., 2004) que la solución del problema discretizado converge a la del problema continuo (imagen de definición infinita). El problema a ser resuelto numéricamente resulta

$$\min_{k \in R^p} \max_{i,j=0,..,p} F^{i,j}(k) \qquad \text{(ec.5)}$$

donde *p* es el tamaño del vecindario a un pixel. Para tratar este problema de optimización convexa no diferenciable se puede considerar algún algoritmo de tipo "haces" como los presentados en (Bonnans et al., 2003). Dentro de dicha clase utilizaremos el Método de Punto Proximal Inexacto Híbrido con Métrica Variable, presentado en (Parente et al., 2008).

Nuestro aporte implementa una versión de este filtro Min-Max que parte de un espacio discretizado definido por vecindarios de un pixel que pueden variar en tamaño y forma como puede observarse en la Fig. 6. Este problema de optimización no-lineal puede ser aproximado mediante un algoritmo iterativo de aproximación local. De acuerdo al tamaño de vecindario definido, la solución puede converger más rápidamente, aunque el cálculo se torne más costoso. La convergencia está dada por un criterio definido en la sección siguiente. La salida del filtro Min-Max consiste en un campo escalar de intensidades que se ajustan a la condición definida en la (ec.4).

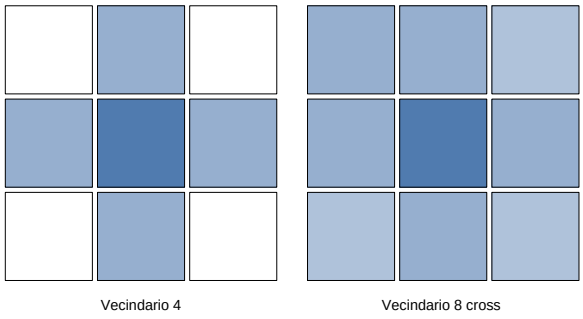

Fig. 6 Dos posibles vecindarios para ser utilizados en 2D

3 NÚCLEO DE LA HERRAMIENTA

Atendiendo a las características de estos algoritmos, se observa que requieren la evaluación reiterativa de la imagen hasta cumplir un dado criterio. Dicho criterio representa la tasa de cambio entre dos estados consecutivos de procesamiento, cuando este valor cumple la condición de ser menor a una dada cota, se lo considera alcanzado. La forma más usual para el procesamiento de una imagen *Y* en la iteración i con una tolerancia ε se presenta como

$$\sum_{\forall k \in \Omega} \left(Y^{i+1}[k] - Y^{i}[k] \right)^2 \leq \varepsilon \qquad \text{(ec.6)}$$

La otra característica destacable de los procesamientos mostrados, es el cálculo por pixel en un entorno acotado. Esto asegura que cada píxel procesado conserva cierta independencia de los restantes, facilitando la implementación de estos algoritmos en una plataforma en paralelo con accesos rápidos en memoria local, tal como también propusieron (Sherbondy, 2003; Schenque, 2005). Los resultados son visualizados en 3D con cierta interactividad con el usuario, por lo que se propone también se ejecute en paralelo.

Con estos requerimientos, la idea es generar una secuencia de procesos con interacción CPU-GPU fácilmente extensible a otros tipos de filtros. El resultado es un pipeline de procesos con evaluación iterativa, donde cada proceso se comporta de la forma del *Algoritmo 1*. El control del pipeline requiere del uso de mecanismos de sincronización y espera, tarea administrada por la CPU.

Algoritmo 1 : IterativeImageProc (Imagen Y_{CPU}, neighborhood v)
Local { }

1. $Y_{GPU} <<$ map(Y_{CPU})
2. $Y'_{GPU} <<$ allocate(Y_{GPU})
3. **while** not convergence (Y_{GPU}, Y'_{GPU})
4. $Y_{GPU} <<$ copy(Y'_{GPU})
5. $Y'_{GPU} <<$ Proc(Y_{GPU}, v)
6. **end while**
7. $Y_{CPU} <<$ copy (Y'_{GPU})
8. Return Y_{CPU}

Finalmente, el usuario deberá instanciar el pipeline con el procesamiento correspondiente y la distribución de pixeles vecinos.

3.1 Invocación de los algoritmos

Para la ejecución de filtros, se definió un módulo capaz de ejecutar filtros u operaciones tanto en GPU como en CPU. Para el mismo, se definieron clases que abstraen la información de las operaciones utilizadas, de forma similar a como se invocan en OpenCL. Cada filtro es representado por un kernel encargado de procesar píxeles, el cual se genera internamente por dos formas, con un Script compilado para la GPU o por una referencia a su implementación en CPU.

Para la ejecución en CPU, se implementó un algoritmo multi-thread sencillo que recorre cada dato de la entrada e invoca el kernel asociado. Se requirió definir los mismos métodos que incluye OpenCL, (tales como lecturas de imágenes, funciones matemáticas implementadas y los tipos correspondientes) para mantener su compatibilidad, ya sea a nivel de eficiencia como para simplificar el trabajo de debugging en una u otra plataforma.

3.2 Visualización de imágenes 3D

La técnica más utilizada para el renderizado de imágenes volumétricas es mediante ray-casting (Frey et al., 2009). Por cada posición de la pantalla, se tiran rayos en dirección perpendicular y se computan la acumulación de transparencia de los vóxeles de la imagen que son intersectados por dicho rayo.

Como las imágenes usadas son monocromáticas o mono-valuadas, a cada valor leído en la imagen de entrada se le asocia una función de coloreo o función de transferencia. Los valores de cada píxel se encuentran normalizados en el rango [0..1], valor que se utiliza como índice en una tabla de coloreo que describe tanto el color RGB como la transparencia. En la Fig. 7 se muestran dos funciones de transferencia utilizadas en la visualización de una tomografía.

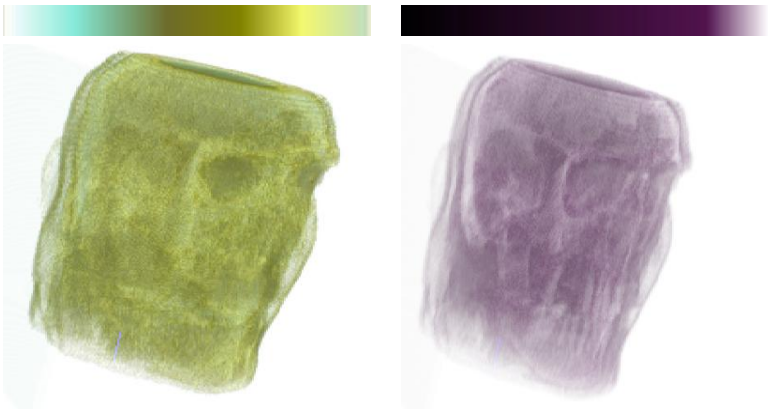

Fig. 7 Dos funciones de transferencia utilizadas para distinguir diferentes regiones de la imagen

También es posible cambiar el punto de vista del observador inmerso en un espacio tridimensional, con operaciones básicas como rotación, traslación y escala. Como estas operaciones son más frecuentes, y en particular en los casos que la aplicación ejecuta en una CPU de menor capacidad de procesamiento, la visualización alterna a un modo denominado "Motion" en la cual se utiliza un buffer de la vista de menor resolución para lograr mantener la fluidez visual (ver Fig. 8).

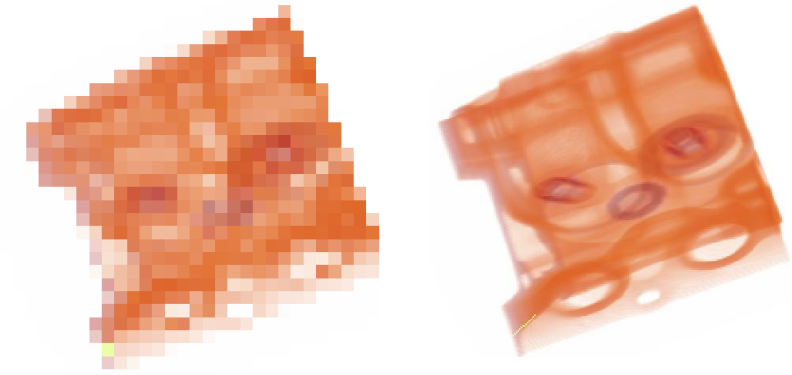

Fig. 8 Visual de un volumen de una pieza de motor en Modo "Motion" (izq) y "Normal" (der).

4 RESULTADOS

Para la evaluación de la herramienta se analizaron imágenes obtenidas desde tomográfos computados donde el filtrado bilateral produce una mejora en la discriminación de los distintos componentes de la imagen. En la Fig. 9, se observa que la superficie representada por el cráneo se discriminaba mejor con respecto al interior y al exterior.

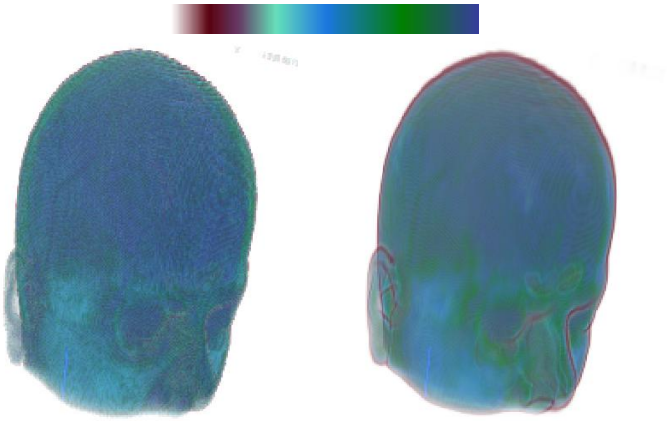

Fig. 9 Arriba, función transferencia utilizada. Tomografía original (izq) y aplicado un filtro bilateral (der).

Al mismo tiempo, tal como se detalló en su implementación, el filtro logra eliminar el efecto "escalonado", sin tener pérdidas en los bordes ni en la definición de los detalles como se observa en la Fig. 10.

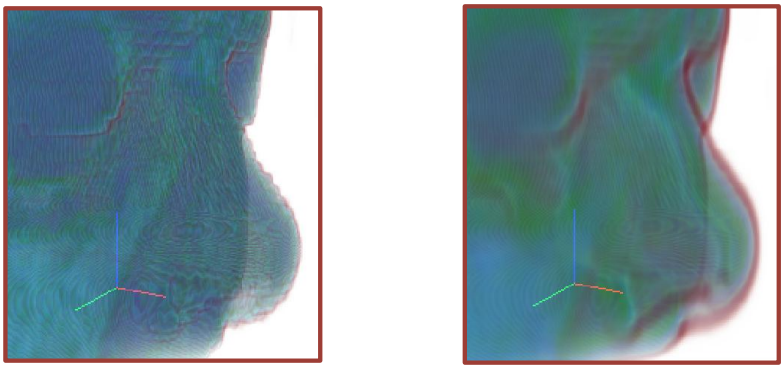

Fig. 10 Zoom en una zona de interés. Tomografía original (izq) y aplicado un filtro bilateral (der).

La herramienta también fue evaluada para la generación de imágenes 3D a partir de mallas de superficie, luego procesadas a fin de mejorar su calidad respecto a criterios basados en la conservación del volumen. La Fig. 11 visualiza aplicaciones de discretización. Las imágenes obtenidas luego de aplicar los filtros de cambio de escala sirven de entrada a un procesamiento de mallas de elemento finito y como métrica de densificación.

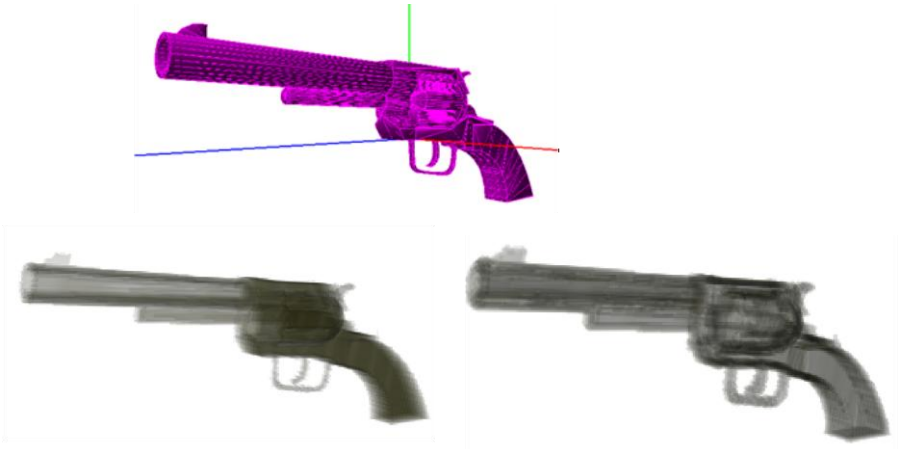

Fig. 11 Malla de triángulos original (arriba), Imagen obtenida a partir de su discretización (izq) aplicando un filtro de gradiente (der).

El método MinMax con convergencia global aplicado a un cambio de escala de orden 4, requirió en general de cerca de 20 iteraciones para la convergencia. Luego de aplicado el filtro Min-Max, el campo escalar de intensidades respeta el criterio impuesto definido en la (ec. 4). Puede verificarse que, los bordes se han conservado, los artefactos debidos a la interpolación común se han eliminado y se ha producido un cierto suavizado de la imagen concluyendo en que, el gradiente direccional se asemeja más a su versión original.

En la Fig. 12 se muestra el caso de una imagen 2D donde es posible observar mejor los resultados. Estos análisis son muy preliminares y se desea continuar esta línea, utilizando al máximo el potencial de la herramienta de forma de llegar a definir ciertos parámetros del método (como son los vecindarios de pixeles) hoy definidos ad-hoc.

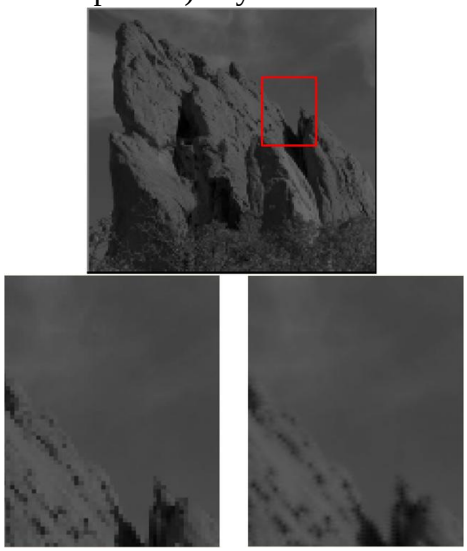

Fig. 12 Cambio de escala de la imagen. Interpolación por vecinos más cercanos (izq) y con el método MinMax (der).

Finalmente se midieron, tal como se aprecia en la Fig. 13 los tiempos de ejecución para un conjunto de imágenes, en los distintos filtros presentados. Los tiempos fueron ponderados de acuerdo a la resolución de imágenes. En general, la implementación en GPU (GTX Nvidia 285 la cual cuenta con 216 shaders) en comparación a la ejecución en una CPU de 4 núcleos mostraba un SpeedUp promedio mayor a 30x. Para el caso particular del rendering, donde la mayoría de la operaciones son de sólo lectura el SpeedUp es mayor a 155x.

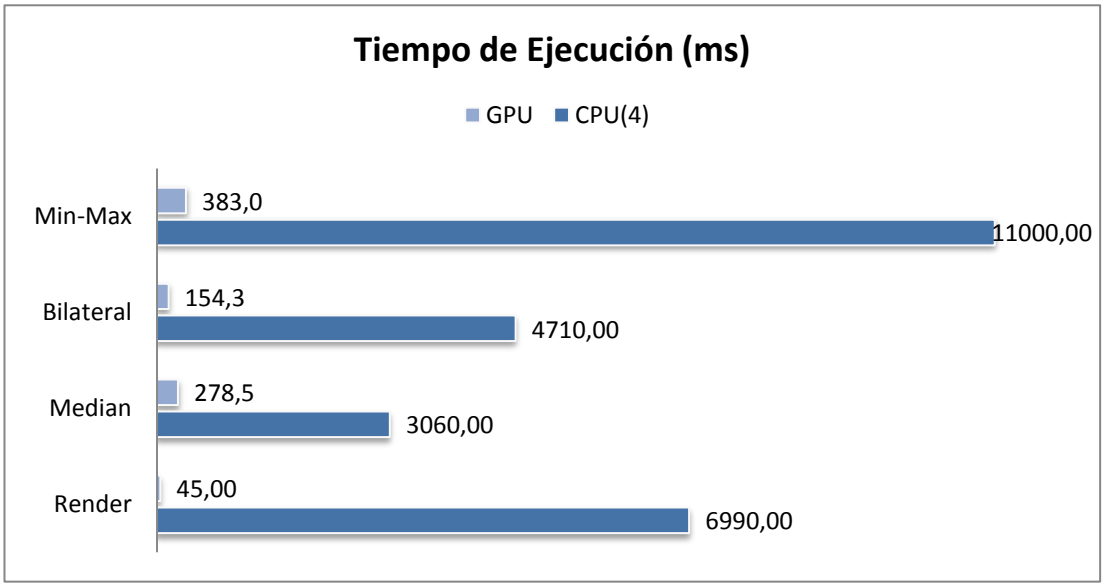

Fig. 13 Tiempos de cálculo de GPU y una CPU de 4 núcleos para diferentes procesamientos

5 CONCLUSIONES

Se ha desarrollado exitosamente una herramienta de procesamiento 2D/3D con GPU aplicado a procesamientos iterativos con convergencia, la cual se está utilizando para el procesamiento de imágenes médicas e industriales. Al mismo tiempo forma parte del tratamiento de mallas de elementos finitos y de la visualización de simulaciones basadas en grillas tridimensionales, como son el método de Lattice Boltzmann.

La herramienta ha sido concebida como modular, lo que permite añadir fácilmente nueva funcionalidad, por lo que se espera poder incluir algoritmos de imágenes con alta demanda de procesamiento, como son los algoritmos de segmentación de objetos basados en crecimiento por regiones. Se desea seguir trabajando en esta línea, utilizando la herramienta para estudiar y validar los algoritmos propuestos, comparando finalmente con otros existentes.

Los módulos implementados en GPU han mostrado una clara mejora de rendimiento con respecto a los CPU. La utilización del estándar OpenCL también ha sido un desafío por encontrarse aún en proceso de definición, pero se espera que continúe su evolución.

La aplicación es de uso libre y se encuentra en proceso de desarrollo. Se puede descargar del sitio: http://www.pladema.net/~jpdamato/index.html.

REFERENCIAS

Aragone, L., González, R.L.V. y Reyero, G., Numerical approximation of optimization problems with L^{∞} functionals. *Investigación Operacional*, vol. 25 (1-04): 14-24, ISSN 0257-4306, Cuba, 2004.

Aragone, L., D'Amato, J.P., Lotito, P. y Parente L., MIN-MAX gradient surface reconstruction via absolute minimization. *Mecánica Computacional*, vol. 28, p.p. 2563-2572, 2009.

Bonnans, J. F., Gilbert, J. C., Lemarchal, C. y Sagastizbal, C. A., Numerical Optimization. Springer, 2003.

Buck, I., GPGPU: General-purpose computation on graphics hardware-GPU computation strategies & tricks. *ACM SIGGRAPH*, Course Notes, nro. 8, 2004.

D'Amato, J.P., Mayorano, F., Rubiales, A., Massa, J. y Tristan, P., Migrating multispectral image processing to the GPU. 38º *Jornadas Argentinas de Informática e Investigación Operativa, High-Performance Computing Symposium,* vol. 38, pp. 1-11, 2009.

Elad, M., On the Origin of the Bilateral Filter and Ways to Improve It. *IEEE transactions on image processing*, vol. 11 (10): 1141-1151, October, 2002.

Frey, S. y Ertl, T., Accelerating raycasting utilizing volume segmentation of industrial ct data. *EG UK Theory and Practice of Computer Graphics*,pp.1Ð8, 2009.

Fung, S., Mann, C. y Aimone, OpenVIDIA: Parallel GPU Computer Vision. http://openvidia.sourceforge.net/index.php/OpenVIDIA disponible desde 2009.

Khronos Group, OpenCL - The open standard for parallelprogramming of heterogeneous systems. http://www.khronos.org/opencl/, disponible en 2010.

Kuttera, O., Shamsb, R. y Navaba, N., Visualization and GPU-accelerated simulation of medical ultrasound from CT images. *Computer methods and programs in biomedicine,* 94: 250–266, 2009.

Nukada, A., Ogata, Y., Endo, T. y Matsuoka, S., Bandwidth intensive 3-D FFT kernel for GPUs using CUDA. *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, pp. 1–11, Piscataway, NJ, USA, 2008.

Parente, L. A., Aragone, L. S., Lotito, P. A. y Reyero, G. F., Numerical solution of a variational problem with L∞ functional. *Proceedings in Applied Mathematics and Mechanics*, v.7(1): 1060401-1060402, ISSN: 1617-7061, 2007.

Parente, L. A., Lotito, P. A. y Solodov, M. V., A Class of Inexact Variable Metric Proximal Point Algorithms. *SIOPT, SIAM Journal on Optimization*, v.19: 240-260, ISSN: 1052-6234, 2008.

Rockafellar, R. T., Monotone Operators and the Proximal Point Algorithm. *SIAM Journal of Control and Optimization*, v.14: 877-898, 1976.

Sherbondy, A., Houston, M. y Napel, S., Fast volume segmentation with simultaneous visualization using programmable graphics hardware. *Proceedings of IEEE Visualization*, pp. 171–176, October, 2003.

Schenke, S., Wunsche, B. y Denzler, J., GPU-Based Volume Segmentation. *Proceedings of IVCNZ '05*, , pp. 171-176, Dunedin, Nueva Zelanda, 28-29 Noviembre, 2005.

Starck, Jean-Luck., Candes, E.J., y Donoho, D.L., Very high quality image restoration by combining wavelets and curvelets. *Wavelet Applications in Signal and Image Processing IX*, vol. 4478: 9-19, 2001.

Tomasi, C. y Manduchi, R., Bilateral Filtering for Gray and Color Images. *Proceedings of the 1998 IEEE International Conference on Computer Vision*, 839-846, Bombay, India, 1998.

Van Loan, C., Computational Frameworks for the Fast Fourier Transform. *SIAM Press*, Philadelphia, PA, 1992.

Yang, R. y Welch, G., Fast image segmentation and smoothing using commodity graphics hardware. *Journal of Graphics Tools, Hardware-Accelerated Rendering Techniques*, vol. 7 (4): 91–100, 2002.

Zivkovic, Z. y van der Heijden, F., Eficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, vol. 27 (7): 773-780, 2006.