Generación directa de mallas tridimensionales segmentadas a partir de imágenes

Alex J. Cuadros-Vargas

Alex J. Cuadros-Vargas es doctor y magíster en Ciencias Matemáticas y de Computación por la Universidade de São Paulo (Brasil). Cuenta con un postdoctorado por la misma universidad con una investigación sobre generación de mallas a partir de imágenes. Ha sido investigador invitado en la University of Pennsilvania, así como en el Scientific Computing and Imaging Institute (SCI) de la Utah State University (EE.UU.). Es miembro fundador de la Sociedad Peruana de Computación. Actualmente se desempeña como profesor investigador en la Universidad Católica San Pablo (Arequipa).

Generación directa de mallas tridimensionales segmentadas a partir de imágenes

El problema de generar mallas directamente partir de imágenes ha atraído la atención de muchos investigadores en los últimos años. Tradicionalmente la tarea de construir mallas apropiadas para simulaciones numéricas a partir de imágenes se realiza en varias etapas, como preprocesamiento (eliminación de ruido, suavización, segmentación, binarización, etcétera), extracción de contornos (Adams y Bischof, 1994; Comer y Delp, 2000; Haralick y Shapiro, 1992; Sonka, 1999), generación de mallas superficiales a partir de contornos (Nonato, Cuadros-Vargas, Minghim, y Oliveira, 2005) y generación de mallas con criterios de calidad a partir de superficies —o de contornos en el caso bidimensional— (Alliez, Cohen-Steiner, Yvinec, y Desbrun, 2005; Shewchuk, 1997).

En los últimos años se ha hecho un gran esfuerzo para crear algoritmos capaces de actuar directamente sobre imágenes, con el objetivo de evitar algunas etapas. Podemos agrupar tales técnicas en dos clases. La primera clase de algoritmos busca la generación de la malla directamente a partir de las imágenes de entrada, con lo cual se evita el preprocesamiento. Estas técnicas, por lo general, consiguen resultados simples y producen mallas de baja calidad, es decir, no apropiadas para simulaciones numéricas. Uno de los primeros trabajos de generación de mallas a partir de imágenes fue propuesto por Terzopoulos y Vasilescu, quienes trataron el problema usando mallas adaptativas acopladas a la imagen como un sistema de resortes y masas (Terzopoulos y Vasilescu, 1992). En este método los nodos de la malla se movían utilizando información de gradiente y curvatura, y los nodos se concentraban en regiones donde existía una variación brusca de datos. Una técnica similar fue propuesta por Hale (2001). En este caso, se utilizaba una función potencial para alinear las características de la imagen con un conjunto de puntos distribuidos originalmente sobre un grilla regular. Hale utilizaba un paso de preprocesamiento que re-

ducía el ruido para luego generar la malla por medio de una triangulación de Delaunay. Para minimizar el error de interpolación entre la malla generada y la imagen original, Garland y Heckbert propusieron un método interactivo que insertaba puntos en una triangulación de Delaunay en la que el error de interpolación era máximo (Garland y Heckbert, 1995). Grosso y sus colaboradores, por su parte, utilizaron una medida de error derivada de la aproximación de mínimos cuadrados para crear un conjunto de reglas para refinar tetraedros y octaedros (Grosso, Lürig, y Ertl, 1997). Roxborough y Nielson utilizaron también la aproximación de mínimos cuadrados para estimar el error de interpolación pero, a diferencia de Grosso, aplicaron una única regla que dividía la arista más grande de un tetraedro (Roxborough y Nielson, 2000). Takahashi y sus colaboradores propusieron luego un método de tetraedralización adaptativa para calcular esqueletos topológicos de volumen: usando iterativamente criterios topológicos y geométricos, la malla era creada y refinada (Takahashi, Takeshima, Nielson, y Fujishiro, 2004). El método iterativo propuesto por García y sus colaboradores, por su parte, controlaba el error medio cuadrático escogiendo los vértices de la malla a partir de la curvatura de la imagen; en otras palabras, se disponía la mayor cantidad de vértices en áreas con curvaturas elevadas; el modelo de la malla se construía insertando los vértices escogidos en una triangulación de Delaunay (García, Sappa, y Vintimilla, 1999). Un algoritmo interesante y no iterativo fue presentado por Yang y sus colaboradores, quienes utilizaban *zero-crossing* juntamente con un algoritmo llamado Floyd-Steinberg Error-Diffusion para escoger un conjunto de puntos a partir de los cuales se calculaba una triangulación de Delaunay. El error de aproximación se reducía gracias a este método, y los autores argumentaban que el algoritmo producía mallas de buena calidad (Yang, Wernick, y Brankov, 2003).

Una característica común de las estrategias descritas es que la malla es generada directamente a partir de la imagen de entrada sin realizar una presegmentación.

La segunda clase de técnicas produce resultados más apropiados para fines de simulaciones numéricas, pero no pueden evitar una etapa previa de segmentación para definir regiones de interés a partir de las cuales se construye la malla. Este grupo de algoritmos, sin embargo, no garantiza totalmente la calidad de los elementos de las mallas generadas. Cebral y Lohner propusieron una técnica que realiza un preproceso que comprende binarización, filtros estadísticos, llenado de huecos e intervención del usuario. Para generar la malla se creaba una isosuperficie que se utilizaría como entrada para una técnica de avance de frontera (Cebral y Lohner, 1999). Zhang y sus colaboradores (2003), por un lado, y Berti (2004), por el otro, presentaron técnicas basadas en *octrees*. Los primeros aplicaron primero un paso de eliminación de ruido en el volumen de entrada; después, se seleccionaban dos isosuperficies y el espacio existente entre ellas era considerado

como la región donde sería creada la malla; a partir de este intervalo se construía un *octree* para guiar la creación de una malla adaptativa de tetraedros u octaedros; finalmente, se mejoraba la calidad de la malla con operaciones de contracción de arista (Zhang, Bajaj, y Sohn, 2003). La técnica de Berti, por su parte, comenzaba realizando un proceso de segmentación en la imagen de entrada, en la que se restringía a generar solamente dos regiones; luego creaba un *octree* a partir de la frontera definida por la segmentación, una malla tetraedral utilizando el *octree* y una variación volumétrica del algoritmo Marching Cubes (Lorensen y Cline, 1987); finalmente, el modelo se mejoraba aplicando operaciones de eliminación, movimiento y suavización en los vértices de la malla (Berti, 2004).

En investigaciones anteriores —de doctorado y postdoctorado— desarrollé una técnica, llamada Imesh, que evita el paso previo de preprocesamiento en la imagen de entrada y, al mismo tiempo, produce mallas segmentadas y con propiedades geométricas apropiadas para realizar simulaciones numéricas. El algoritmo Imesh fue desarrollado en C++ utilizando la biblioteca de geometría computacional CGAL (Fabri, Giezeman, Kettner, Schirra, y Schonherr, 2000)[1]. La técnica posee tres etapas principales, denominadas (a) generación de malla; (b) segmentación de malla; y (c) generación de malla con calidad.

En la primera etapa, generación de malla, se crea una malla directamente a partir de una imagen de entrada sin preprocesar. Al final de este proceso la imagen de entrada termina representada por la malla gracias a patrones de color calculados en cada elemento de esta última.

El objetivo de la segunda etapa, segmentación de malla, es separar la malla obtenida en un número deseado de segmentación de mallas. Además de los patrones de color obtenidos de la imagen, esta etapa utiliza propiedades topológicas y geométricas de la malla como información adicional para realizar el proceso de segmentación.

Finalmente, la etapa de generación de malla con calidad refina las segmentación de mallas generadas respetando las fronteras y garantizando la calidad para la malla como un todo.

El resultado final de este proceso es una malla en la que los elementos respetan un criterio geométrico de ángulo mínimo, característica muy deseable para realizar simulaciones numéricas tales como deformaciones elásticas o colisiones físicas.

1. Las características y la evolución de la técnica Imesh se pueden revisar con más detalle en Cuadros-Vargas, Gerhardinger, De Castro, Neto, y Nonato, 2006; Cuadros-Vargas, Lizier, Minghim, y Nonato, 2009; Cuadros-Vargas y Nonato, 2006; Cuadros-Vargas, Nonato, Minghim, y Etiene, 2005; Cuadros-Vargas, Nonato, Tejada, y Ertl, 2007; Lizier, Jr., Cuadros-Vargas, Jr., y Nonato, 2009.

El potencial de aplicabilidad del algoritmo Imesh en el ámbito de nuestra comunidad es vasto. En medicina, por ejemplo, a partir de una tomografía se podría generar un modelo en el cual sería posible estudiar con detalle y tridimensionalmente cada uno de los huesos implicados en la implantación de una prótesis de rodilla; esto permitiría al cirujano tener una idea bastante clara de la zona de la operación y sus características sin necesidad de tocar al paciente. Otro campo podría ser el estudio y diagnostico de la osteoporosis: por ejemplo, a partir de una resonancia magnética de alta resolución del interior de un hueso humano se podría generar un modelo detallado de su porosidad; este modelo podría presentar características apropiadas para realizar una simulación física de presión en el hueso afectado por esta enfermedad hasta que —computacionalmente— se rompa, lo cual permitiría cuantificar su resistencia. Otro ejemplo de aplicación en el campo de la medicina es el diagnóstico de anomalías en venas o arterias, tales como oclusiones vasculares, arteriales o aneurismas cerebrales. En este caso, el algoritmo Imesh podría generar una malla superficial (de triángulos) que detalle el interior de la zona en cuestión. Con este modelo podría realizarse una simulación de fluidos utilizando partículas, y podría determinarse visualmente los lugares con mayor riesgo de obstrucción. En el campo de la geografía, por su parte, hay posibilidades interesantes. Por ejemplo, a partir de la imagen satelital de un lago, el algoritmo Imesh podría generar una malla de triángulos y separar la tierra y el agua en diferentes segmentaciones de mallas. De este modo, con un mínimo margen de error, podría calcularse el área total del lago a partir del cálculo y sumatoria de las áreas de todos los triángulos de la segmentación de malla que representa la parte del agua. De la misma forma, a partir de la imagen satelital de un río se podría generar un modelo que separe agua y tierra, y realizar una simulación numérica inyectando computacionalmente fluido compuesto por partículas para estimar cuál sería el volumen máximo de agua que podría soportar el río antes de desbordarse. Naturalmente, estos ejemplos no agotan el potencial de aplicabilidad de la técnica Imesh en nuestra comunidad.

Objetivos

Es importante notar que la técnica Imesh no es un solo algoritmo; en realidad, se trata de un conjunto de algoritmos destinados a resolver diferentes partes del problema de generación de mallas simpliciales a partir de imágenes 2D y 3D.

Si bien el desarrollo del algoritmo ha abierto un gran campo de posibilidades y hasta el momento ha generado resultados importantes en el ámbito bidimensional (Cuadros-Vargas, et al., 2009; Lizier, et al., 2009) —dado que no sigue el camino tradicional de los algoritmos disponibles en la literatura—, existen en la actualidad diversas partes del al-

goritmo que necesitan ser investigadas con más profundidad, especialmente en el caso tridimensional. Por ejemplo, el paso de segmentación de mallas de simplejos con patrones de color —tema muy poco estudiado en la literatura— es un punto fuerte para el algoritmo Imesh, pues éste no tiene restricción en cuanto al número de submallas generadas en el modelo (se han llegado a generar modelos de hasta 19 submallas), algo que lo distingue de las técnicas disponibles en la literatura, que normalmente pueden generar sólo dos submallas.

Otro ejemplo es el proceso final de refinamiento de calidad de malla en el caso tridimensional, problema que aún no dispone de una solución consolidada en la literatura para cualquier tipo de superficie. Por esta razón es de nuestro interés encontrar algún tipo de solución para esta etapa del algoritmo.

Con miras a una futura publicación, en este informe presentamos algunos afinamientos y reformas orientados a mejorar los modelos generados por el algoritmo Imesh, especialmente en el ámbito tridimensional.

Los principales aspectos que aborda la presente investigación son los siguientes: (1) dependiendo de la imagen de entrada, las mallas generadas por el algoritmo no poseen fronteras suaves. Este hecho dificulta algunos procesos de la parte final del algoritmo (generación de malla con calidad), por lo que ciertos criterios geométricos de calidad no se pueden incluir en los elementos de las mallas generadas. En el presente trabajo se presenta una propuesta para mejorar este aspecto; (2) la rapidez del algoritmo Imesh fue un aspecto pensado y optimizado desde su concepción; sin embargo, puede ser acelerado todavía más a partir de dos tareas: (a) colocar el código de Imesh en un contexto multitarea, lo que permitiría aprovechar al máximo la totalidad de los núcleos disponibles en una computadora; y (b) migrar el código para una versión superior de la biblioteca CGAL. Cabe resaltar que el hecho de incluir ideas de paralelismo en el algoritmo Imesh no implica una mejora de los modelos resultantes en términos geométricos o topológicos; no obstante, hacerlo implica mayor agilidad en el desarrollo de nuevas ideas y en el refinamiento de las existentes, al tiempo que abre la posibilidad de tratar entradas de datos mucho más grandes, lo que antes no se intentaba por el tiempo que llevaría procesarlas y depurarlas. De la misma forma —aunque con menor relevancia—, la migración del código hacia una biblioteca más avanzada abre la posibilidad de utilizar nuevos recursos geométricos que pueden hacer la diferencia en los procesos del algoritmo.

Las mejoras en el código se incluyeron en la investigación por las razones citadas, pues sin ellas difícilmente nos sería posible mejorar, agilizar y diversificar nuestros resultados

actuales. Sin estas optimizaciones de código la investigación quedaría estancada en procesos lentos y en el tratamiento de un conjunto restringido de ejemplos. En este informe presentamos también la implementación de una propuesta para acelerar la ejecución del algoritmo utilizando procesos multitarea.

El algoritmo Imesh persigue objetivos muy ambiciosos, pues intenta generalizar en una única técnica características de varios tipos de técnicas presentes en la literatura. En concreto, el algoritmo intenta generar mallas Delaunay segmentadas, y que tales mallas posean criterios de calidad en sus elementos, todo ello directamente a partir de imágenes de entrada no preprocesadas; más aún, se pretende que las operaciones utilizadas durante los procesos de la técnica Imesh sean independientes de la dimensión de la imagen, es decir, los cálculos utilizados deben funcionar, en la medida de lo posible, de la misma forma para tratar imágenes bidimensionales y tridimensionales.

Para cumplir con estos objetivos, en investigaciones anteriores (Cuadros-Vargas, 2007) se establecieron tres etapas principales, que pueden ser revisadas en la propuesta de este proyecto: (1) construcción de malla, para representar la imagen de entrada con una malla Delaunay; (2) segmentación de malla, para separar los objetos importantes contenidos en la malla que representa a la imagen; y (3) refinamiento de calidad de malla, para incluir criterios de calidad dentro de ésta, de tal forma que el modelo final sea apropiado para ser utilizado por procesos de simulaciones numéricas.

En este informe se describen los resultados obtenidos en función de los objetivos del proyecto, así como las actividades realizadas para mejorar el algoritmo Imesh; finalmente, se discutirán nuevas propuestas para futuras mejoras.

Resultados

La adaptación del código del algoritmo Imesh para la última versión de la biblioteca CGAL se realizó con éxito. Se practicaron adaptaciones de muchas de las clases que conforman el algoritmo Imesh, según los nuevos requisitos y la nueva sintaxis de la versión 3,5 de CGAL. El resultado de esta etapa no puede ser fácilmente mostrado, ya que se trata de modificaciones del código de Imesh en lenguaje C++. Sin embargo, estas modificaciones están presentes en todos los pasos y resultados que describiré a continuación.

La idea principal detrás del proyecto fue añadir, al ya existente criterio gradiente + límite, una nueva alternativa llamada *criterio de isovalores,* que busca mejorar la suavidad de los

bordes obtenidos por el proceso de construcción de malla. Esta nueva alternativa fue desarrollada dentro de Imesh.

Es interesante que, además de que se puede aplicar el criterio de isovalores para detectar bordes en el proceso de generación de malla, también se puede extender y considerar como un criterio de isofronteras. Este criterio de isofrontera se puede emplear para establecer elementos de borde (aristas en 2D o *faces* en 3D) entre las submallas generadas, de modo que se ayude a mejorar el proceso de segmentación.

La combinación de estas dos ideas rindió muy buenos resultados, como se puede ver en las siguientes figuras. En las primeras seis se presentan comparaciones entre los resultados obtenidos con el criterio gradiente + límite y los obtenidos con el criterio isovalores + isofronteras.

En la figura 1 se aprecia el modelo de un torso humano calculado con el primer criterio (gradiente + límite); aquí fue posible hacer una separación en cuatro submallas. En la figura 2 se aprecia la misma figura, esta vez calculada con el segundo criterio (isovalores + isofronteras), aunque también con una segmentación de cuatro submallas: se observa que en esta última aparecen mejor definidas las formas y fronteras de los objetos contenidos en el volumen.

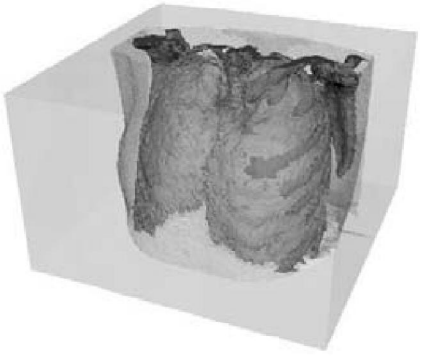

Figura 1
Torso humano
(gradiente + límite).

Figura 2
Torso humano
(isovalores + isofronteras).

Las figuras 3 y 4 muestran un volumen que representa rodillas humanas, donde con el primer criterio fue posible separar únicamente cuatro objetos (figura 3), mientras que con el segundo (figura 4) se consiguió separar los seis huesos contenidos en el modelo más la piel del exterior, lo que resulta en un total de ocho submallas; junto con ello, la suavidad entre fronteras generadas en este último caso también es notablemente mejor.

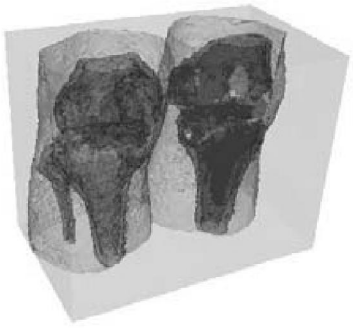

Figura 3
Rodillas
(gradiente + límite).

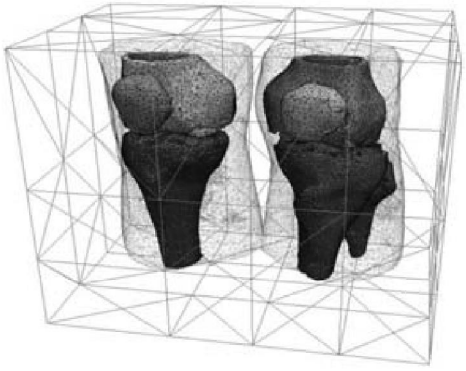

Figura 4
Rodillas
(isovalores + isofronteras).

Las figuras 5 y 6 muestran la misma comparación para el volumen de un pie humano. Este volumen en particular contiene mucho ruido, lo cual impide que el criterio gradiente + límite consiga definir claramente las formas y objetos en la malla (figura 5); en esta figura, además, la suavidad entre fronteras se encuentra muy desorganizada. En la figura 6, por el contrario, el criterio de isovalores + isofronteras consiguió separar hasta 19 objetos diferentes en el modelo (17 huesos, la piel y el exterior); así mismo, la definición de los objetos y la suavidad son mejores.

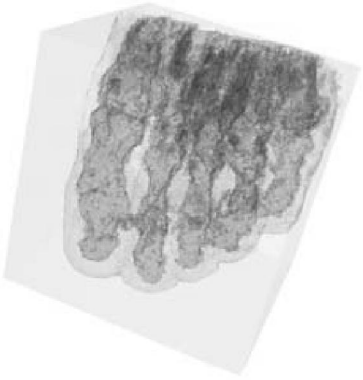

Figura 5
Pie humano
(gradiente + límite).

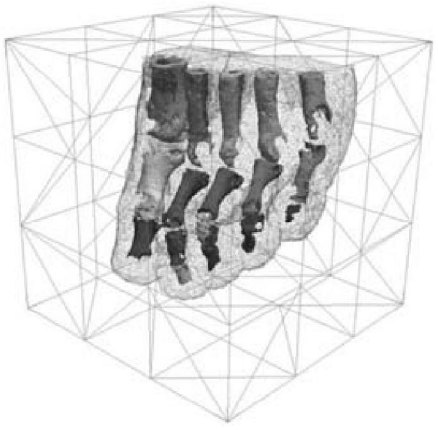

Figura 6
Pie humano
(isovalores + isofronteras).

Las siguientes cuatro figuras muestran otros resultados trabajando en dimensiones diferentes. Las figuras 7 y 8 muestran los resultados obtenidos en 3D.

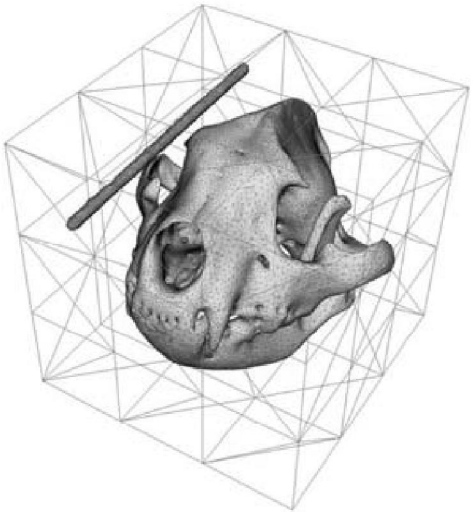

Figura 7
Cabeza de roedor en 3D
(gradiente + límite).

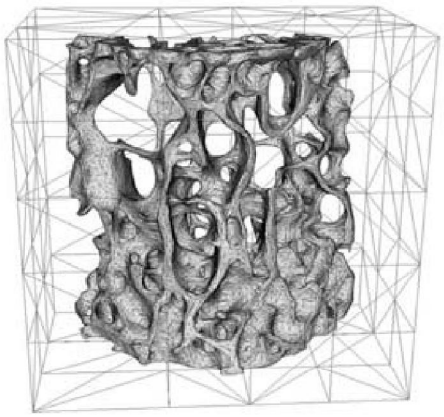

Figura 8
Interior del hueso de un ratón en 3D
(isovalores + isofronteras).

Las figuras 9 y 10 presentan resultados obtenidos en 2D.

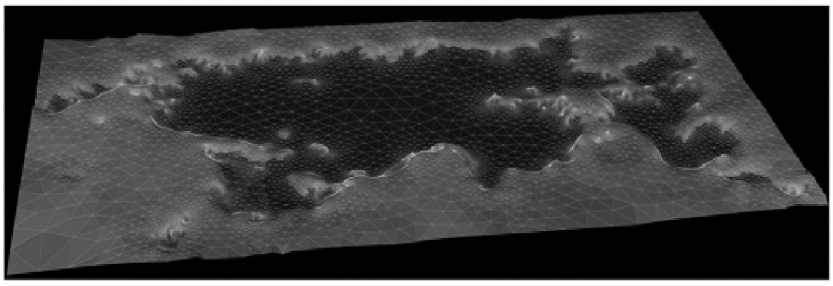

Figura 9
Lago Titicaca (Perú) en 2D
(isovalores + isofronteras).

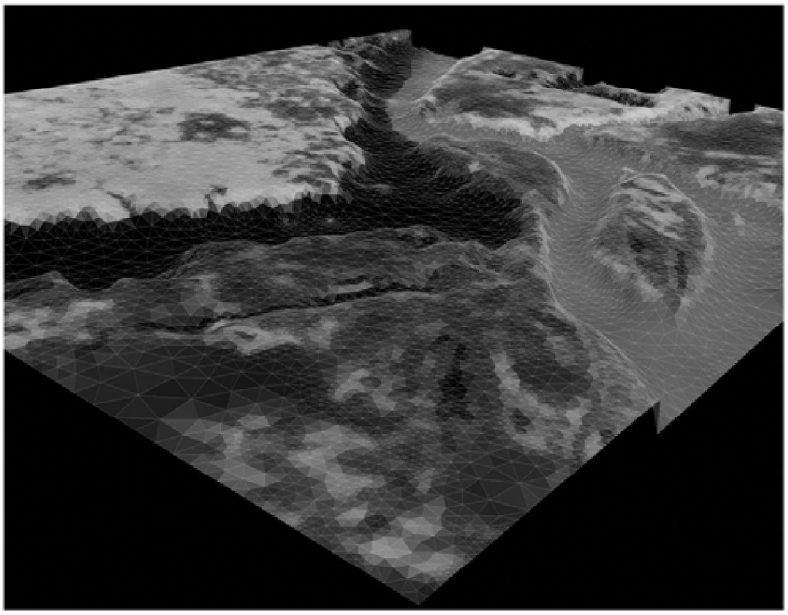

Figura 10
Río Negro y río Solimões (Brasil) en 2D
(isovalores + isofronteras).

Las siguientes figuras presentan una comparación más detallada de la suavidad de fronteras obtenida mediante los dos criterios considerados. La figura 11 muestra la malla generada de un cráneo humano utilizando el primer criterio, lo que de hecho constituyó, en el proyecto inicial, el problema que debería ser solucionado. Las figuras 12 y 13 muestran los detalles marcados en dicha figura. En la imagen se pueden observar todas las imperfecciones geométricas que no podrán ser controladas por el criterio de gradiente + límite.

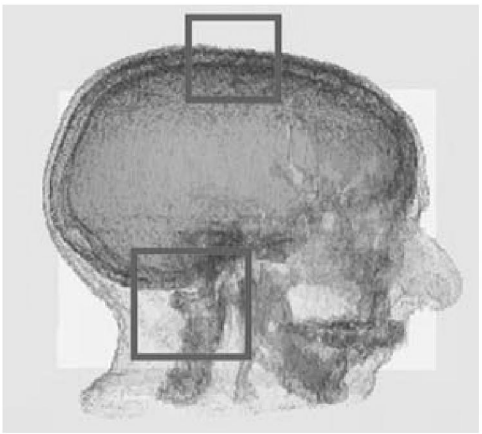

Figura 11
Cráneo humano
(gradiente + límite).

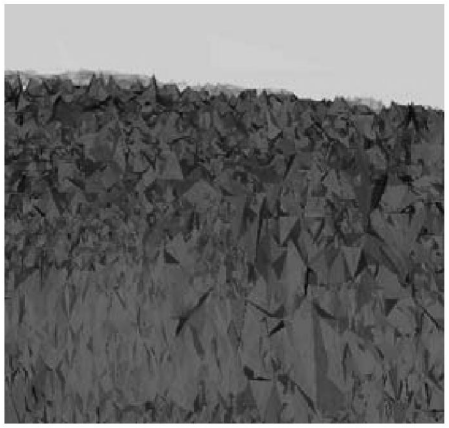

Figura 12
Detalle de la figura 11
(gradiente + límite).

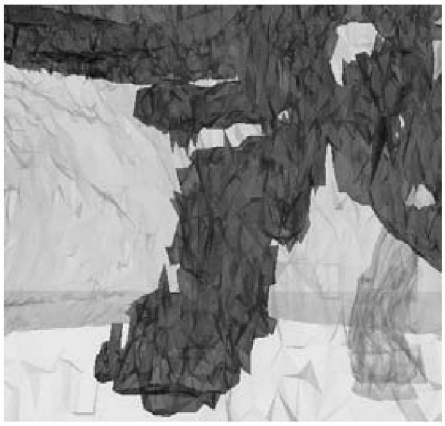

Figura 13
Detalle de la figura 11
(gradiente + límite).

La figura 14, por su parte, muestra un modelo generado a partir del mismo volumen pero utilizando isovalores + isofronteras. Las figuras 15 y 16 presentan los detalles para este caso: como se puede notar, la suavidad entre fronteras mejoró mucho; sin embargo, existen todavía imperfecciones geométricas que deben ser tratadas.

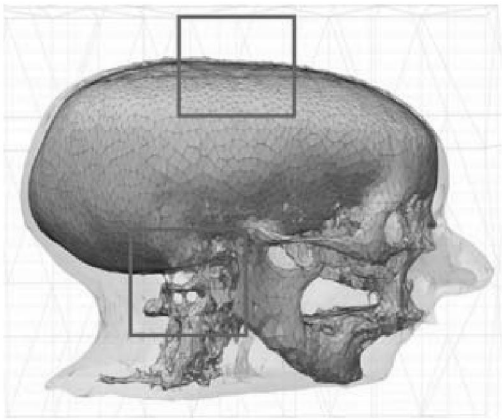

Figura 14
Cráneo humano
(isovalores + isofronteras).

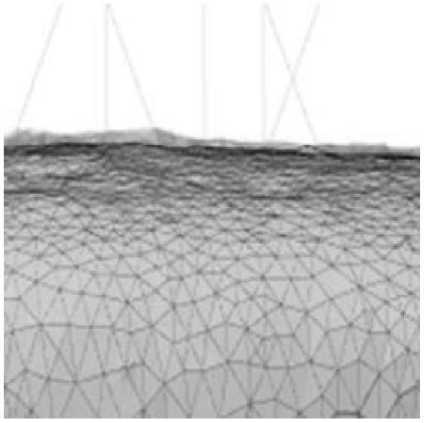

Figura 15
Detalle de figura 14
(gradiente + límite).

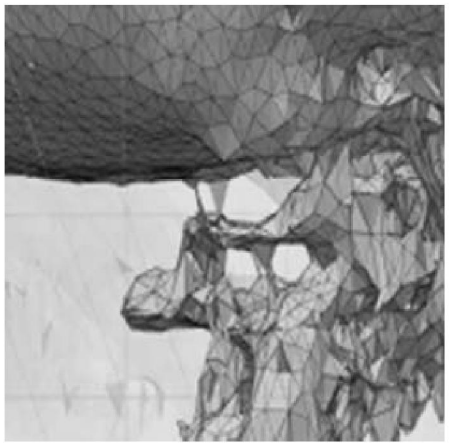

Figura 16
Detalle de figura 13
(isovalores + isofronteras).

Con respecto al tiempo de ejecución es interesante mostrar una comparación entre los tiempos de ejecución utilizando los criterios presentados. Para ello, no obstante, es necesario notar dos aspectos importantes: (1) al aplicar los dos nuevos criterios a una misma imagen, se genera una cantidad diferente de elementos, por lo que la medición del tiempo, al no darse en iguales condiciones, no sería exacta, si bien es posible trabajar en modificaciones del código para garantizar que ambos criterios generen la misma cantidad de elementos; (2) cuando se escriba el algoritmo para presentar un informe de investigación final de la versión tridimensional, sólo se planea mencionar la idea de isovalores + isofronteras, pues generó resultados mucho más precisos que el primer criterio, quedando esta última sin sustento para ser presentada. Siendo así, podemos afirmar que los modelos generados por isovalores + isofronteras son ejecutados hasta quince veces más rápido que los generados por gradiente + límite.

El cuadro 1 muestra la diferencia de tiempos entre los procesos secuencial (1 *thread*) y de multiprocesamiento (2 o más *threads*) para el caso de la etapa de construcción de malla de algunos volúmenes de datos. Las pruebas fueron realizadas en un computador Intel Xeon™, dual CPU 3,2 GHz, dual CPU 3,2 GHz (4 núcleos).

Cuadro 1
Minutos de procesamiento en la construcción de mallas usando threads para volúmenes de datos

Índices	Una *thread*	Dos *threads*	Cuatro *threads*
Rodillas[a]	11,41	6,07	5,11
Pie[b]	8,51	4,72	2,81
Tórax[c]	13,86	9,27	6,64

[a] Figura 4. Dimensión = 379 x 229 x 305 píxeles.
[b] Figura 6. Dimensión = 256 x 256 x 256 píxeles.
[c] Figura 2. Dimensión = 384 x 384 x 241 píxeles.

Con el objetivo de tornar viable a la técnica Imesh para entradas de datos más complejas y de mayor tamaño se realizó un estudio de cómo transformar el código del algoritmo Imesh —concebido inicialmente de forma secuencial— a una versión multitarea. Como se anotó anteriormente, la técnica Imesh no es un sólo algoritmo; en realidad se trata de un conjunto de algoritmos destinados a resolver diferentes partes del problema de generación de mallas simpliciales a partir de imágenes (2D y 3D). Siendo así, la transformación de su versión secuencial a una versión multitarea implica modificar varios algoritmos en 2D y 3D,: creación de malla basada en diferentes operadores, cálculo de patrones de color, varios algoritmos de segmentación de malla, suavización de bordes, algunas técnicas de refinamiento de calidad de simplejos, lectura de imágenes en varios formatos y escritura de mallas en varios formatos, entre otros. La idea de incluir criterios multitarea surgió porque los procesos secuenciales del algoritmo anterior no aprovechaban el total del poder de cálculo del computador utilizado. Si pensamos en un computador con un procesador de cuatro núcleos, veríamos que los procesos secuenciales del algoritmo utilizan, en promedio, sólo uno de ellos. El porcentaje de poder de procesamiento desperdiciado crecería aún más si consideráramos un computador con ocho o más núcleos, pues independientemente de la cantidad de éstos, un proceso secuencial posee básicamente el mismo comportamiento, por lo que se usaría básicamente un núcleo. La necesidad de utilizar la capacidad máxima de un computador se tornaría aún más crítica cuando se trabajara con volúmenes grandes o más complejos (por ejemplo, volúmenes de datos tensoriales), pues el tiempo de procesamiento crecería cada vez más sin utilizar la capacidad total del computador. Por otro lado, en momentos de crea-

ción y prueba de nuevas ideas —especialmente en el caso tridimensional— la detección y depuración de errores llegaría a ser un proceso demasiado lento y tedioso.

Para conseguir el objetivo del multiprocesamiento se estudió el algoritmo y se percibió que existía la repetición del siguiente contexto en diferentes escenarios: (1) un conjunto de objetos que deben ser procesados; (2) una operación que debe ser ejecutada sobre los objetos del conjunto. Siendo así, se diseñó una clase genérica *(template),* que a partir de un conjunto de objetos asocia a cada uno de ellos una operación deseada y encamina los cálculos para que se realicen en una *thread* separada. De esta forma es posible aprovechar el 100 % del procesador (con todos sus núcleos) durante los cálculos realizados sobre el conjunto. La clase implementada se comporta como un clásico bucle *for,* y se puede generalizar —mediante programación genérica— a diferentes contextos dentro del código del algoritmo Imesh, también para las versiones bidimensional y tridimensional. La clase implementada incluso se puede utilizar para otros algoritmos, pues básicamente crea un bucle paralelo que no depende de tipos específicos. El desarrollo de la idea fue probada en la primera etapa del algoritmo Imesh (construcción de malla), en 2D y 3D, y mostró buenos resultados.

Discusión

Según el proyecto inicial, se esperaba que la combinación de los criterios de isovalores + isofronteras fuese suficiente para ofrecer una entrada apropiada para el proceso de garantía de calidad de malla, aspecto éste especialmente importante para el caso tridimensional. Sin embargo, aunque la utilización de tales criterios haya mejorado mucho el desempeño del algoritmo Imesh, los modelos generados aún contienen ciertas imperfecciones geométricas, como en el caso de ángulos pequeños y de otros elementos aislados. Esto dificulta la implementación de algoritmos de refinamiento Delaunay, última etapa del algoritmo Imesh. Para el caso de algoritmos de refinamiento Delaunay bidimensionales existen técnicas robustas que pueden garantizar la calidad de los elementos de la malla aunque tengan imperfecciones geométricas complejas (Chew, 1989; Pav y Walkington, 2004; Shewchuk, 2002a, 2002b). De hecho, esta etapa se encuentra funcionando completamente para cualquier modelo de entrada en la versión bidimensional del algoritmo: Imesh-2. Por otro lado, los algoritmos de refinamiento de calidad actuales de la literatura Delaunay tridimensionales (Alliez, et al., 2005; Cheng, Dey, y Edelsbrunner, 2000; Chew, 1997; Edelsbrunner y Guoy, 2001; Shewchuk, 1998; Tournois, Wormser, Alliez, y Desbrun, 2009) imponen, en la práctica, restricciones al modelo de entrada. Tales restricciones pueden considerar, por ejemplo, modelos simples de entrada, mode-

los libres de singularidades y modelos libres de fronteras con ángulos pequeños. Por el momento tales restricciones aún no pueden ser satisfechas en los modelos generados por el algoritmo Imesh-3.

Debido a los problemas mencionados, se viene estudiando la posibilidad de crear —en una investigación ulterior— una etapa dentro del algoritmo Imesh, después del proceso de segmentación, con la finalidad de mejorar el modelo generado y hacerlo apropiado para comenzar el último proceso de garantía de calidad de malla. Esta nueva etapa podría llamarse *mejoramiento de malla,* y se podría dividir en dos pasos principales. El primero, suavización de fronteras, intentaría optimizar la rotulación de células$_d$[2] próximas a las fronteras entre submallas generadas en el proceso de segmentación. Para realizar esta optimización este paso puede basarse en conceptos empleados en un método de segmentación de imágenes con textura llamado EM/MPM (Comer y Delp, 2000), y una extensión para el contexto de mallas Delaunay (Cuadros-Vargas, et al., 2006). Se espera que tal optimización, concentrada en las células$_d$ de frontera, elimine un gran porcentaje de las puntas, presentes en los modelos generados actualmente. El segundo paso, simplificación de puntos, buscaría dejar el modelo más ligero, en términos de cantidad de puntos y células$_d$, utilizando, a su vez, dos operaciones de eliminación: (a) simplificación de puntos internos de las submallas; y (b) simplificación de puntos coplanares (o colineales) sobre las fronteras entre submallas. Ambas operaciones pueden basarse en propiedades de la triangulación de Delaunay para realizar simplificaciones en malla procurando no modificar la forma de las fronteras previamente definidas por el proceso de segmentación.

Otra idea que puede ser incluida y viene siendo estudiada puede estar basada en proyección de puntos en superficies (Alexa et al., 2001, 2003; Amenta y Kil, 2004; Fleishman, Cohen-Or, y Silva, 2005; Scheidegger, Fleishman, y Silva, 2005). Esta técnica se puede aplicar también para mejorar el modelo de entrada para el proceso de garantía de calidad de malla. Este mecanismo de proyección puede ser incluido juntamente con el criterio de isovalores en el proceso de generación de malla, esto es, antes de insertar un nuevo punto en malla es posible calcular una nube de puntos local —a partir de isovalores— y proyectar el nuevo punto en la superficie formada por esta nube. Hay que tener en cuenta que el hecho de colocar puntos desde el inicio (generación de malla) sobre superficies suaves puede ocasionar que el proceso de particionamento de malla genere menos imperfecciones geométricas; de esta forma se espera que este cálculo mejore aún más la suavidad de los modelos generados.

2. El término *célula$_d$* se refiere a triangulo en dos dimensiones y tetraedros en tres dimensiones.

En resumen, por el momento estamos apostando por la siguiente combinación de factores para mejorar el modelo de entrada necesario para la última etapa de garantía de calidad de malla: criterio de isovalores (implementado en 2D y 3D) + cálculo de proyección de puntos en superficies + suavización de fronteras + simplificación de puntos internos y coplanares o colineales. Se espera que en una investigación futura las propuestas que contemplen estas ideas mejoren mucho el camino para la implementación de algoritmos de refinamiento de calidad Delaunay, principalmente para el caso tridimensional.

Referencias

Adams, R., y Bischof, L. (1994). Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell., 16*(6), 641-647.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., y Silva, C. T. (2001). Point set surfaces *VIS '01: Proceedings of the Conference on Visualization '01.* (pp. 21-28). Washington D. C.: IEEE Computer Society, 21-28.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., y Silva, C. T. (2003). Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics, 9*(1), 3-15.

Alliez, P., Cohen-Steiner, D., Yvinec, M., y Desbrun, M. (2005). Variational tetrahedral meshing. *ACM Trans. Graph., 24*(3), 617-625.

Amenta, N., y Kil, Y. J. (2004). Defining point-set surfaces. *ACM Trans. Graph., 23*(3), 264-270.

Berti, G. (2004). Image-based unstructured 3d mesh generation for medical applications. Ponencia presentada en el European Congress on Computational Methods in Applied Sciences and Engeneering (ECCOMAS).

Cebral, J., y Lohner, R. (1999). From medical images to cfd meshes. *Proceedings of the 8th International Meshing Roundtable.*

Comer, M. L., y Delp, E. J. (2000). The EM/MPM algorithm for segmentation of textured images: analysis and further experimental results. *IEEE Transactions on Image Processing, 9*(10), 1731-1744.

Cuadros-Vargas, A. J. (2007). *Volumetric mesh generation from images.* PhD, Universidade de São Paulo, São Paulo (Brasil).

Cuadros-Vargas, A. J., Gerhardinger, L. C., De Castro, M., Neto, J. B., y Nonato, L. G. (2006). Improving 2nd mesh image segmentation with markovian random fields. *Proceedings of the XIX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI '06),* 1530-1834.

Cuadros-Vargas, A. J., Lizier, M., Minghim, R., y Nonato, L. (2009). Generating segmented quality meshes from images. *Journal of Mathematical Imaging and Vision, 33*, 11-23.

Cuadros-Vargas, A. J., y Nonato, L. G. (2006). Imesh: Quality mesh generation from images. *Proceedings of the European Congress on Computational Fluid Dynamics (ECCOMAS '06),* 1-13.

Cuadros-Vargas, A. J., Nonato, L. G., Minghim, R., y Etiene, T. (2005). Imesh: An image based quality mesh generation technique. *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI '05),* Washington D. C.

Cuadros-Vargas, A. J., Nonato, L. G., Tejada, E., y Ertl, T. (2007). Generating segmented tetrahedral meshes from regular volume data for simulation and visualization applications. *Proceedings of the International Symposium on Computational Modelling of Objects Represented in Images–Fundamentals, Methods and Applications (compIMAGE 2006).*

Cheng, S., Dey, T., y Edelsbrunner, H. (2000). Sliver exudation. *Journal of the ACM, 47*, 883-904.

Chew, L. P. (1989). *Guaranteed-quality triangular meshes.* Technical report. Department of Computer Science. Cornell University. Ithaca (Nueva York).

Chew, L. P. (1997). Guaranteed-quality Delaunay meshing in 3D [short version]. *SCG '97 Proceedings of the 13th Annual Symposium on Computational Geometry,* Nueva York.

Edelsbrunner, H., y Guoy, D. (2001). An experimental study of sliver exudation.

Fabri, A., Giezeman, G., Kettner, L., Schirra, S., y Schonherr, S. (2000). The design of CGAL, a computational geometry algorithms library.

Fleishman, S., Cohen-Or, D., y Silva, C. T. (2005). Robust moving least-squares fitting with sharp features. *ACM Trans. Graph., 24*(3), 544-552.

García, M., Sappa, A., y Vintimilla, B. (1999). Efficient approximation of gray-scale images through bounded error triangular meshes. *Proceedings of the 1999 International Conference on Image Processing (ICIP '99).*

Garland, M., y Heckbert, P. (1995). *Fast polygonal approximation of terrains and height fields.* Technical Report (CMU-CS-95-181). Carnegie Mellon University. Pittsburgh (Pennsilvania).

Grosso, R., Lürig, C., y Ertl, T. (1997). The multilevel finite element method for adaptative mesh optimization and visualization of volume data. En R. Yagel y H. Hagen (eds.), *Visualization '97*: IEEE Computer Society Press.

Hale, D. (2001). Atomic images: a method for meshing digital images *10th International Roundtable,* 185-196.

Haralick, R., y Shapiro, L. (1992). *Computer and robot vision* (Vol. 1). Massachusetts: Addison-Wesley Publishing Company.

Lizier, M. A., Jr., D. C. M., Cuadros-Vargas, A. J., Jr., R. M. C., y Nonato, L. G. (2009). Generating segmented meshes from textured color images. *Journal of Visual Communication and Image Representation*, en prensa.

Lorensen, W. E., y Cline, H. E. (1987). Marching cubes: a high resolution 3D surface construction algorithms. *ACM SIG. Comp. Graph., 21*, 163-169.

Nonato, L. G., Cuadros-Vargas, A. J., Minghim, R., y Oliveira, M. C. F. D. (2005). Beta-connection: generating a family of models from planar cross sections. *ACM Trans. on Graph., 24*(4), 1239-1258.

Pav, S., y Walkington, N. (2004). Robust three dimensional Delaunay. *13th International Meshing Roundtable.*

Roxborough, T., y Nielson, G. M. (2000). Tetrahedron based, least squares, progressive volume models with application to freehand ultrasound data. *Visualization '00: Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000),* Washington D. C.

Scheidegger, C. E., Fleishman, S., y Silva, C. T. (2005). Triangulating point set surfaces with bounded error. *SGP '05: Proceedings of the 3rd Eurographics Symposium on Geometry Processing,* Aire-le Ville (Suiza).

Shewchuk, J. R. (1997). *Delaunay refinement mesh generation.* PhD, Carnegie Mellon University, Massachusetts.

Shewchuk, J. R. (1998). *Tetrahedral mesh generation by Delaunay refinement.* Ponencia presentada en el Symposium on Computational Geometry.

Shewchuk, J. R. (2002a). *Delaunay refinement agorithms for triangular mesh generation.*

Shewchuk, J. R. (2002b). *What is a good linear element? Interpolation, conditioning and quality measures.*

Sonka, M. (1999). *Image processing, analysis and machine vision.* Boston (Massachusetts): PWS Publishing.

Takahashi, S., Takeshima, Y., Nielson, G. M., y Fujishiro, I. (2004). Topological volume skeletonization using adaptative tetrahedralization. *Proceedings of Geometric Modeling and Processing,* 227-236.

Terzopoulos, D., y Vasilescu, M. (1992). Sampling and reconstruction with adaptative meshes. *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference (CVPR'91),* Lahaina (Hawai), 829-831.

Tournois, J., Wormser, C., Alliez, P., y Desbrun, M. (2009). Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph., 28*(3), 1-9.

Yang, Y., Wernick, M., y Brankov, J. (2003). A fast approach for accurate content-adaptative mesh generation. *IEEE Trans. on Image Processing, 12*(8), 866-881.

Zhang, Y., Bajaj, C., y Sohn, B.-S. (2003). Adaptative and quality 3D meshing from imaging data. *Proceedings of the 8th ACM Symposium on Solid Modeling and applications.*