Rev. Int. Mét. Num. Cálc. Dis. Ing. (2010) 26: 39-46

Técnica de visualización científica para datos escalares discretos

Heikel Yervilla Herrera · Yaidel Reyes López · Alcides Viamontes Esquivel · Carlos A. Recarey Morfa

Recibido: Julio 2009, Aceptado: Diciembre 2009

Resumen La visualización científica de datos escalares continuos es actualmente muy utilizada. Nuevas formulaciones basadas en partículas o nubes de puntos, como el Método de Elementos Discretos, Métodos Libres de Mallas y de algunos más tradicionales como el Método de Elementos Finitos de Puntos, Volumen Finito, entre otros, ha permitido obtener un mejor análisis de las estructuras materiales y su comportamiento, específicamente en el área de la ingeniería. El presente trabajo describe una nueva técnica de visualización científica relacionada con la representación de datos escalares discretos que puede ser considerada una modificación a la manera en que se visualizan los datos discretos en la actualidad, conocida en el área de la visualización científica como *tiny polygons*. Se pretende que no solo sea aplicada a datos resultantes de los métodos mencionados, sino que sea extensible a cualquier otro capaz de generar datos escalares discretos o en combinación con datos escalares continuos. En la implementación de la técnica presentada en este trabajo se hace uso de los cuaterniones, con el objetivo de formar las figuras que finalmente serán visualizadas, y de uno de los enfoques del problema del vecino más cercano para mejorar la complejidad computacional. Se discute además la repercusión que tiene dicha técnica a la hora de obtener información a partir de la imagen final.

H. Yervilla Herrera · Y. Reyes López · A. Viamontes Esquivel · C.A. Recarey Morfa
Centro de Investigación de Métodos Computacionales y Numéricos en la Ingeniería (CIMCNI-UCLV-CIMNE)
Universidad Central "Martha Abreu"de las Villas
Carretera a Camajuaní km $5\frac{1}{2}$, Las Villas, Cuba
Tel./Fax: 53 42 224746
e-mail: yervilla@uclv.edu.cu; yaidelr@uclv.edu.cu; alve@uclv.edu.cu; recarey@uclv.edu.cu

SCIENTIFIC VISUALIZATION TECHNIQUES FOR DISCRETE SCALAR DATA

Summary The scientific visualization of continuous scalar data is currently widely used. New formulations based on particles or point clouds, like the Discrete Element Method, Mesh Free Methods and some more traditional methods such as the Finite Elements Method of Points, Finite Volume Method, among others, has allowed a better analysis of material structures and their behavior, specifically in the area of engineering. This paper describes a new scientific visualization technique related to the representation of scalar discrete data that can be considered a modification to the way discrete data are visualized now days, known as tiny polygons. It is pretended that this technique be extended to any other method, rather than the mentioned above, capable of generate discrete scalar data or combination of them with continuous data. In the implementation of the scientific visualization technique presented in this paper cuaternions are used, with the aim of forming the figures that eventually will be displayed and one of the approaches to the nearest neighbor problem is used too, with the purpose of improving the computational complexity. Furthermore it is discussed the impact of this technique to obtain information from the final image.

1. Introducción

El ser humano obtiene una gran cantidad de información a través de la visión. Se estima que un 50 % de las neuronas del celebro humano están dedicadas a la percepción. El sistema visual es un buscador de patrones de extrema fuerza y sutileza gracias al cual, con tan solo observar una imagen durante un corto tiempo,

el cerebro obtiene información detallada del objeto de estudio.

La visualización de información es, hoy, un tema recurrente en innumerables disciplinas tanto en las ciencias sociales como exactas. El término **visualización científica** se refiere al proceso que envuelve la utilización de la computadora con el fin de obtener imágenes a partir de datos. En el caso de la mecánica computacional se refiere a la visualización científica de grandes volúmenes de datos escalares, vectoriales y de otra índole provenientes de los resultados de simulaciones y modelaciones de problemas de la física-matemática. La visualización científica apoyada por ordenadores permite a ingenieros e investigadores obtener representaciones gráficas a partir de estos resultados.

Según [1] el objetivo de la visualización es propiciar un profundo nivel de interpretación de los datos y fomentar nuevas percepciones dentro del proceso de entendimiento dependiendo de la habilidad de visualización del ser humano.

Las técnicas de visualización son el elemento fundamental dentro del proceso de representación de los datos. Técnicas como Interpolación Directa de Colores, Iso-Líneas, Iso-Superficies, Líneas de Fluidos, entre otras son muy utilizadas debido a las características propias de los datos.

Por otra parte, a partir de los resultados del cálculo de modelos numéricos convencionales (diferencias finitas, volumen finito, elementos finitos) es posible obtener de forma aproximada el campo de cierta magnitud sobre un sólido, fluido o interacción entre ambos. En los métodos tradicionales, esta magnitud es continua, o en algunos casos presenta discontinuidades aisladas. Nuevas formulaciones de estos métodos y otros como los métodos basados en partículas o en nubes de puntos, conocidos como Método de Elementos Discretos (MED) y Métodos sin Mallas (MSM) respectivamente, han arrojado relevantes resultados a aquellos problemas que implican grandes cambios geométricos o deformaciones del modelo de análisis, o intentan predecir el comportamiento de medios con un fuerte grado de discontinuidad inherente. Estos permiten reproducir el comportamiento natural discontinuo de ciertas propiedades mecánicas en el medio o estructura. Este tipo de información es referido en este trabajo como datos escalares discretos y son vistos como puntos en el espacio, los cuales representan cada una de las discontinuidades en el cuerpo estudiado. Por ejemplo, los parámetros de movimiento de las partículas aisladas en un gas o el grado de ionización en una macro-molécula con radicales múltiples. En muchos fenómenos se aprecian magnitudes con un comportamiento mixto.

Estas nuevas formulaciones han venido a engrosar el mundo de las investigaciones de problemas de la Física-Matemática y el comportamiento de los materiales. Muchas de las propiedades obtenidas a partir de estos métodos pueden ser visualizadas utilizando las técnicas más comunes de visualización científica, pero cuando se tienen que procesar datos del tipo escalares discretos o mixtos es necesario la utilización de técnicas no convencionales.

2. Antecedentes y estado del arte

La visualización asistida por computadoras es utilizada casi desde el mismo momento en que surge el primer ordenador. El primero en utilizar el término es William Fetter[1] a comienzos de la década de 1960, dando lugar al surgimiento de aplicaciones y lenguajes orientados a la creación de imágenes. La visualización de datos científicos es utilizada por primera vez a finales de la década de 1970. A partir de aquí comienzan a surgir nuevas técnicas de visualización orientadas a obtener mejor calidad en las imágenes resultantes. Hoy en día esta es una disciplina con una gran significación en la sociedad moderna.

En [2,1,3] se describen técnicas y algoritmos relacionados con la visualización científica. Las técnicas mencionadas en estos libros están orientadas hacia datos escalares continuos y vectoriales ya que son estos los más comunes y de mayor utilización. Al aparecer los datos escalares discretos, como resultado de los nuevos estudios científicos, las técnicas de visualización existentes hasta el momento son incapaces de representarlos correctamente por lo que es necesario crear nuevas para obtener una correcta representación.

Lo que en la mayoría de los casos se hace, cuando se tienen datos escalares discretos, es utilizar pequeños símbolos para representar el punto discontinuo. En [4] se hace referencia a la utilización de la técnica *tiny polygon* por la sencillez que ello implica a la hora de visualizar los puntos discontinuos. Esta técnica es simple, poco utilizada y, por consiguiente, pasa casi desapercibida para la bibliografía existente. La propuesta presentada en este trabajo es una variación o modificación buscando que se pueda obtener una mejor correspondencia entre el comportamiento discreto de la variable y su representación gráfica con la finalidad de conseguir mejores resultados a la hora de interpretar la imagen final.

[1] Pionero de la computación gráfica, primero en dibujar una figura humana utilizando una computadora

3. Descripción de los algoritmos de visualización discreta

La idea desarrollada trata de resolver el problema de la visualización científica de datos escalares discretos, pues las técnicas utilizadas con tal finalidad no son lo suficientemente esclarecedoras a la hora de obtener información relevante. La metodología consiste en utilizar figuras para visualizar los puntos que representan la discontinuidad. Por la sencillez que ofrecen los cubos y polígonos regulares a la hora de una representación son las figuras que simbolizan los puntos discretos en el interior y exterior del cuerpo respectivamente, aunque pueden ser utilizadas indistintamente cualquier tipo de figura capaz de simbolizar el punto discreto.

Uno de los algoritmos consiste en representar, con cubos o cualquier figura 3D, las discontinuidades en la misma posición en que se encuentran, aplicándole cierta transparencia al objeto principal. Este procedimiento se corresponde con lo utilizado en la actualidad para representar datos discontinuos, no presenta una gran complejidad pero al ser combinado con la otra técnica desarrollada puede proporcionar información valiosa para el investigador o ingeniero. El otro algoritmo, un poco más complejo, radica en llevar estas discontinuidades a la superficie del objeto principal representándolas con polígonos regulares. Es necesario mencionar que la metodología propuesta en este trabajo sólo es factible en caso de que el cuerpo sea representado por una malla de polígonos, en caso de no contar con ella se hace imprescindible algún método de obtención de la superficie para utilizar esta técnica de visualización.

La representación mediante figuras en el interior del cuerpo, específicamente en la misma posición en que se encuentra el punto discontinuo, resulta sencilla. Entre lo destacable en esta técnica cabe mencionar que el tamaño de la figura puede ser determinante a la hora de interpretar la representación. De acuerdo a determinadas particularidades de los datos puede modificarse el tamaño de la figura para lograr una mejor visualización (Algoritmo 1).

Para colorear la superficie puede ser utilizada cualquier técnica convencional de visualización de datos continuos, interpolación directa de colores o codificación de colores en las caras de la malla, aplicándole cierta transparencia a la superficie para poder ver la imagen y todos las figuras en su interior.

Esta técnica permite un mayor realismo puesto que el especialista o ingeniero observa las discontinuidades en el lugar exacto donde se encuentran. Tiene la desventaja de que en conjuntos de datos con una gran cantidad de discontinuidades no se observe con claridad la información pues se originan solapamientos entre las figuras, lo cual puede ser mejorado de cierta manera modificando su longitud o tamaño.

Algorithm 1 Algoritmo para obtener y visualizar datos escalares discretos utilizando figuras 3D en el interior del cuerpo

```
Entrada : Lista puntos, valor real radio
// puntos::lista de puntos discontinuos;
// radio::radio de la figura que representará las
discontinuidades;
Lista figuras;
For i = 0 to puntos.count − 1 do
    tipo figura figura =
    construir-figura(puntos[i],radio);
    figuras.Add(figura);
end
For i = 0 to figuras.count − 1 do
    draw(figuras[i]);
end
```

El otro algoritmo de visualización de datos escalares discontinuos desarrollado consiste en trasladar las discontinuidades a la superficie del cuerpo. Este método es mucho más costoso computacionalmente que el anterior pero tiene la ventaja de que al representar las discontinuidades en el exterior del cuerpo se tiene una idea de la forma y la posición con respecto a la superficie de cada una de ellas.

La idea consiste en dibujar polígonos regulares en las caras del cuerpo principal, específicamente, entre las caras más cercanas al punto discontinuo, se escogen las que quedan completamente de frente con el objetivo de no crear imágenes difusas. Si se representa en todas las caras más cercanas no se tiene idea de la relación punto-superficie que se quiere expresar. Entiéndase cara frontal o que queda de frente como aquella que contiene el plano que al proyectar el punto discreto, la proyección sea interior a la cara.

El algoritmo consiste en buscar, por cada punto que representa una discontinuidad, el vértice de la malla más próximo. Este algoritmo es bastante costoso por lo que se utilizó una estructura de particionado del espacio, el *kd-tree* [2], para organizar los vértices de la malla. El algoritmo de construcción de esta estructura así como la manera en que se enfrenta la búsqueda a través de ella puede ser encontrada en la Apartado 4.

Una vez encontradas todas las caras que contienen este vértice, las cuales son las candidatas a contener el polígono que simbolice el punto discontinuo, se buscan las caras frontales para dibujar definitivamente los polígonos. Para encontrar estas caras se proyecta el punto en cada uno de los planos que representan, lue-

[2] Árbol de k dimensiones

go se analiza si los puntos proyectados en cada uno de los planos pertenecen a las caras correspondientes. Estas caras serían las que quedan de frente al punto que representa la discontinuidad.

A continuación se presenta parte del trabajo con vectores utilizado en este trabajo.

La ecuación del plano está dada por

$$\overrightarrow{n} \cdot (x - p_0) = 0 \tag{1}$$

donde $\overrightarrow{n}$ es una normal del plano y p_0 un punto que pertenece al plano.

Un plano esta formado por todos los puntos x que cumplen 1.

Sea p_1 un plano definido según 1, construido a partir de los tres puntos que constituyen cada una de las caras de la malla a analizar, donde n es una normal al plano p_1 y v un vector con inicio en el plano p_1 y dirección p.

Entonces la proyección $p^{'}$ del punto p en el plano p_1 está dada por:

$$p^{'} = p - r \cdot n \tag{2}$$

donde

$$r = v \times n$$

Una vez encontradas las caras frontales al punto discontinuo se generan los polígonos utilizando los cuaterniones (Apartado 5). Sobre estas caras se dibujarán los polígonos que van a simbolizar las discontinuidades en el cuerpo. Los polígonos se generan a partir de un radio y un centro y con cantidad de lados se calculan los vértices exteriores utilizando rotaciones vectoriales; el centro del polígono coincide con la proyección del punto en el plano que representa la cara frontal. Para una completa información relacionada con la técnica antes descrita ver Algoritmo 2.

La importancia de esta técnica viene dada en la interpretación que el usuario o ingeniero pueda obtener de la imagen resultante. Como los polígonos son solo dibujados en las caras más cercanas que quedan de frente se obtiene una idea de la posición de las discontinuidades con respecto a la superficie del cuerpo.

Uno de los aspectos a tener en cuenta y que ya se ha mencionado es la utilización del *kd-tree* para la organización espacial de los vértices de la malla puesto que el algoritmo de busqueda de vecindad es computacionalmente muy costoso (Apartado 4).

Algorithm 2 Algoritmo para obtener y visualizar datos escalares discretos utilizando polígonos en el exterior del cuerpo

```
Entrada  : Lista puntos, valor real radio, valor
           entero n, Mesh mesh
// puntos::Lista de puntos discontinuos o
discretos;
// radio::Radio de los polígonos;
// n::Cantidad de lados del polígono;
// mesh::Malla de superficie que representa al
objeto de estudio;
 kdtree kdtree =kdtree(mesh.vértices, 0);
For i = 0 to puntos.count − 1 do
    punto  vértice-más-cercano =
    buscar-vecino-más-cercano(kdtree.raíz, puntos[i]);
    Lista caras =
    buscar-caras-frente(vértice-más-cercano,
    puntos[i]);
    polígono  polígono
    = construir-polígono(puntos[i],radio); (ver Alg. 5)
    polígonos.Add(caras, polígono);
end
For i = 0 to polígonos.count- 1 do
    draw(polígonos[i]);
end
```

4. Problema del vecino más cercano: el *kd-tree*

El problema que se presenta consiste en buscar el vecino más cercano a un punto en el conjunto de vértices de una mallas en el espacio euclidiano de tres dimensiones. La búsqueda del vecino más cercano es un problema habitual en una gran variedad de ramas: descubrimiento de conocimiento, clasificación y reconocimiento de patrones, estadística, compresión de datos, inteligencia artificial, entre otras. El *kd-tree* se presenta como una de las estructuras más utilizadas en la busqueda del vecino y k-vecinos más cercanos [5,6]. Otras estructuras de datos y algoritmos utilizadas con dicho fin son el *vp-tree* [7], *el hB-tree* [8], *quad-tree* [9], ramas y cotas [10] y más recientemente el *cover-tree* [11,12].

En [13,14,15] se presenta el *kd-tree* como una de las estructuras de particionado del espacio más usadas en la búsqueda del vecino más cercano. Según [16], el *kd-tree* para dimensiones bajas es extremadamente versátil. El árbol kd es una estructura de datos de particionado del espacio que organiza los puntos de un espacio euclídeo de k dimensiones. Es un caso especial de los árboles BSP. La problemática que dio lugar a la decisión de la utilización de una estructura de particionado del espacio estuvo dada porque deben hacerse reiteradas búsquedas en una lista de puntos lo cual sería muy costoso. La construcción del árbol estático a partir de n puntos es de orden $\Theta\ n\ \log\ n$ y la búsqueda $\Theta\ \log\ 2n$.

La idea, detrás de esta estructura de datos, es asociar a cada nodo de un árbol una caja, y que la relación de paternidad desde el punto de vista del árbol, sea de inclusión de las cajas asociadas a los nodos. Esta idea es compartida, en particular, con otra estructura, conocida como quadtree/octree. Una de las diferencias entre estas estructuras es el número de hijos que tiene cada nodo. En el caso del *kd-tree* son dos, mientras que en el quadtree/octree son cuatro/ocho respectivamente, dependiendo de la dimensión del espacio. En el *kd-tree*, los hijos de un nodo representan la división de la caja del nodo padre según un eje el cual se va alternando por niveles de profundidad.

Para conseguir que cualquier prototipo (puntos en nuestro caso) tenga la misma probabilidad de estar a un lado o a otro del hiperplano y por tanto construir un árbol lo más equilibrado posible, se suele elegir el hiperplano de forma que se sitúe en la mediana de los valores de la coordenada discriminante. El árbol se construye de la siguiente forma: se van ordenando los puntos por la coordenada discriminante en cada nivel, se obtiene la mediana, se crea el nodo con la información de la mediana y se pasa a construir los dos hijos a partir de los subconjuntos resultantes de dividir la lista principal por la posición de la mediana y así recurrentemente. (Algoritmo 3).

Algorithm 3 Algoritmo recursivo para la construcción de la estructura de datos kd-tree

```
Entrada : Lista lista, valor entero profundidad
// lista::Lista de nodos de la malla;
// profundidad::Variable que controla la coordenada
a analizar;
if lista.Count <> 0 then
    valor entero coordenada = profundidad mod 3;
    lista.sort(coordenada);
    tree node nodo = new nodo(lista.mediana );
    node. hijo-izquierdo =
    kdtree(lista a la izquierda de la mediana, profundidad)+
    1;
    node. hijo-derecho =
    kdtree(lista a la derecha de la mediana, profundidad)+
    1;
end
```

En el algoritmo anterior se considera la lista a la izquierda/derecha de la mediana, luego de ordenar por alguna coordenada, como la sub-lista de todos los puntos menores/mayores a está. La mediana es considerada el valor del nodo en el árbol.

La búsqueda del vecino más cercano, utilizando el árbol kd, también es recursiva y se realiza de la siguiente manera: Dada una muestra x se compara la coordenada que es discriminante para ese nodo con el valor de corte v (la mediana por coordenada discriminante) y se procede en una dirección según esa comparación. Si $x[c] + dnn \leqslant v$ (donde dnn es la distancia al vecino más cercano hasta el momento) el hijo derecho no puede contener al vecino más cercano, de forma similar si $x[c] - dnn \geqslant v$ el hijo izquierdo tampoco lo puede contener, donde c es la coordenada a tener en cuenta según la profundidad (Algoritmo 4).

Algorithm 4 Búsqueda del vecino más cercano utilizando el kd-tree

```
Entrada : tree node raíz, punto p
// p::punto al cual se le busca su vecino;
// x::nodo actual en el árbol kd;
// c::coordenada según profundidad;
// v::valor de corte según profundidad;
punto vecino = raíz;
valor real dnn = distancia(vecino, p);
if x[c] + dnn <= v then
    vecino = buscar-vecino-más-cercano(
    hijo-izquierdo, p)
end
if x[c] - dnn > = v then
    vecino = buscar-vecino-más-cercano(
    hijo-derecho, p)
end
return vecino;
```

Otro aspecto a mencionar es la utilización de los cuateriones y su conveniente notación para ser utilizado en rotaciones vectoriales a la hora de formar las figuras que simbolizarán el punto discontinuo en la superficie del cuerpo.(Apartado 5)

5. Rotaciones vectoriales: los cuaterniones

Los cuaterniones, en matemática, son una extensión no conmutativa de los números complejos. No es un término reciente, fueron descritos por primera vez en 1843 por el irlandés William Rowan Hamilton y aplicados a transformaciones geométricas en tres dimensiones. Aunque en determinados trabajos matemáticos teóricos o prácticos son sustituidos por vectores tienen especial importancia en aplicaciones que involucren rotaciones [17,18].

En álgebra moderna los cuaterniones están formados por cuatro componentes reales y pueden expresarse como:

$$H = \{a + bi + cj + dk \ : \ a,\ b,\ c,\ d \in \mathbb{R}\} \tag{3}$$

Los cuaterniones presentan una notación conveniente para representar orientaciones y rotaciones. Característica que les confiere una importancia especial en

la computación gráfica, robótica, navegación, satélites, entre otras.

Sea $q = xi + yj + zk$ un punto (o un vector) del espacio, u un vector unitario del mismo espacio y θ un número real. La rotación alrededor del eje $(0, u)$ con un ángulo θ envía el punto q sobre el punto $q' = x'i + y'j + z'k$ dado por:

$$q' = h \cdot q \cdot \tilde{h} \text{ donde } h = \cos(\theta/2) + u \cdot \sin(\theta/2) \qquad (4)$$

A continuación se muestran los algoritmos para obtener los polígonos regulares, que simbolizarán las discontinuidades en la superficie del cuerpo, utilizando los cuaterniones y las rotaciones vectoriales. El primero es el algoritmo general para construir un polígono por rotación de un vector en el espacio (Algoritmo 5) y el segundo algoritmo está dedicado a rotar un vector en el espacio utilizando cuaterniones (Algoritmo 6).

Algorithm 5 Algoritmo para la construcción de polígonos a partir de rotaciones vectoriales

```
Entrada : vector v, valor entero n
// v::vector a rotar;
// n::cantidad de veces a rotar el vector v;
// lista::Lista de vértices del polígono;
Lista lista;
quaternión u = Eje-Rotación(v);
alpha = 360/n;
h = cos(alpha/2) + u * sen(alpha/2);
For i = 0 to n do
    vector v' = Rotar-Vector(v,h);
    punto p = Calcular-Vértice(v,v',r);
    lista.Add(p);
end
return lista;
```

Algorithm 6 Algoritmo que permite rotar un vector en el espacio utilizando cuaterniones

```
Entrada : vector v, quaternión h
// v::vector a rotar;
// h::cuaternión según (4);
 quaternión h' = Conjugada(h);
 quaternión q = new Quaternión(v);
 quaternión q' = h * q * h';
return new Vector(q');
```

El eje de rotación es determinado utilizando las ecuaciones 1 y 2, su significado geométrico se corresponde con el vector formado por el punto que representa la discontinuidad y su proyección en la cara de la malla que le queda completamente de frente. Los puntos que forman el polígono se calculan utilizando su centro (proyección del punto discontinuo), el vector en rotación y una distancia que representa el radio de la circunferencia circunscrita al polígono regular.

6. Análisis de resultados

Las técnicas de visualización científica presentadas para visualizar discontinuidades en el interior del cuerpo forman parte de una herramienta de visualización desarrollada para visualizar propiedades físicas y mecánicas en sólidos y fluidos en nuestro centro de investigación. Paralelamente a estas técnicas se desarrollan otras orientadas a datos escalares continuos y datos vectoriales. La geometría del cuerpo puede estar definida utilizando Geometría Sólida Constructiva (CSG) o mallas de superficie.

A continuación se muestran una serie de imágenes que muestran los resultados de la aplicación de las técnicas de visualización para datos discretos.

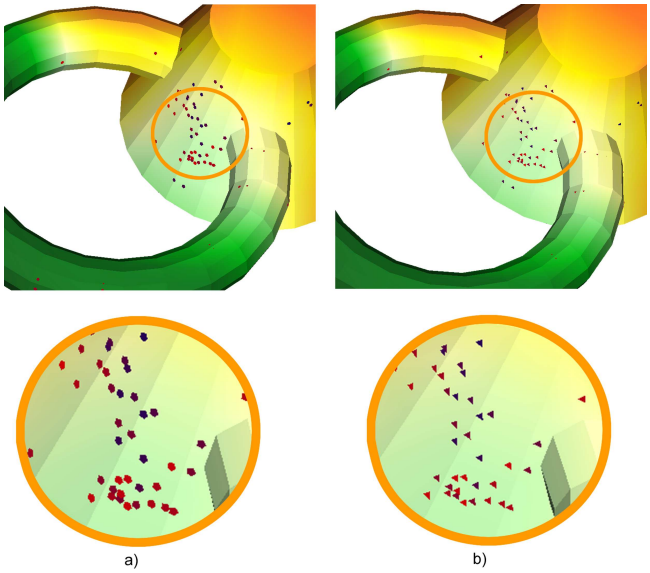

Figura 1. Acercamiento a la estructura de un anillo. a) Imagen generada utilizando pentágonos. b) Imagen generada utilizando triángulos

En la Figura 1 se representa un anillo que contiene datos escalares continuos y discretos. Se combinaron técnicas de visualización científica para ambos tipos de datos: interpolación directa de colores y visualización mediante polígonos en la superficie del objeto. Esta es una muestra simple de una estructura donde se mezclen datos discretos y continuos. Se puede apreciar que se permite utilizar cualquier tipo de polígono regular con el objetivo de buscar una mejor interpretación de

los datos. La diferente coloración de los polígonos depende de la distancia entre punto discreto que simboliza y la superficie o de un valor escalar asociado al punto discontinuo. El color de representación depende de determinadas consideraciones en las propiedades a visualizar por el investigador o ingeniero.

La Figura 2 muestra ambas técnicas de visualización científica: visualización utilizando figuras tanto en el interior como en el exterior del objeto. A la hora de interpretar las Figura 2a y b no es posible conseguir gran cantidad de información pero al combinarlas (Figura 2c) se puede obtener datos acerca de la relación de los puntos discretos y la superficie, pueden cotejarse las figuras en el interior con las representadas en el exterior y obtener una mejor aproximación de los resultados de la modelación.

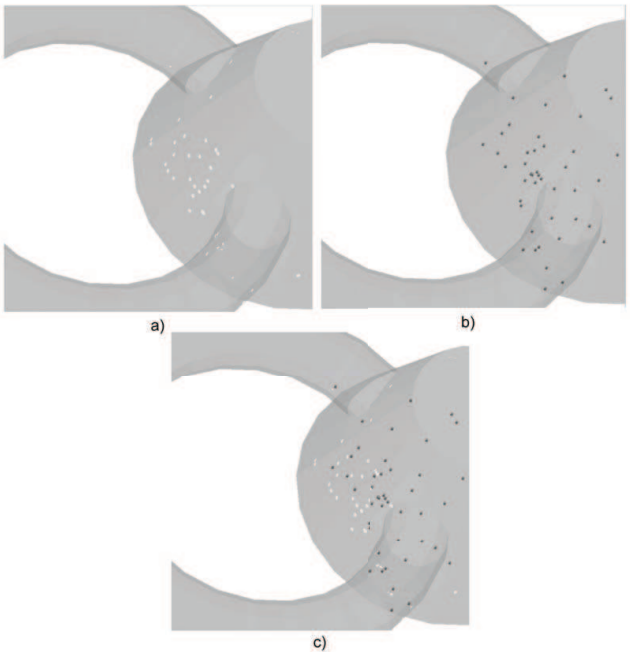

Figura 2. Imagen del anillo donde se observan las discontinuidades en el interior y el exterior. a) Representación de las discontinuidades utilizando polígonos en el exterior del cuerpo. b) Representación utilizando cubos en el interior del objeto. c) representación que combina ambas técnicas, visualización utilizando polígonos en el exterior del cuerpo y cubos en el interior

La Figura 3 combina varias técnicas de visualización científica sobre datos escalares continuos y discretos: líneas de contorno, superficies de contorno y la técnica de visualización para datos escalares discretos desarrollada en este trabajo. La interpretación depende de determinadas particularidades de los datos y de las propiedades que el investigador o ingeniero quiera representar en cada momento y con cada técnica.

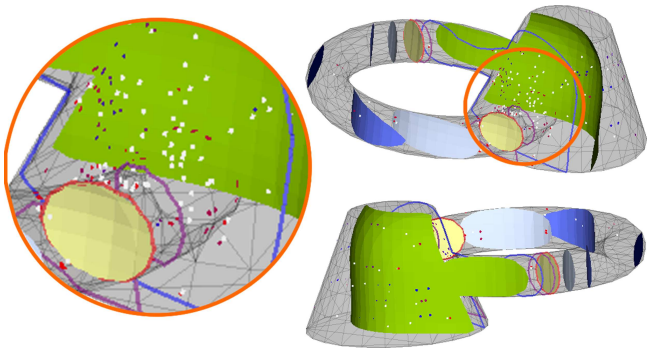

Figura 3. Imagen que combina datos escalares continuos y discretos

En las representaciones anteriores se pueden apreciar cada una de las ventajas que a la hora de interpretar el modelo se pueden obtener de la técnica de visualización desarrollada. Combinando convenientemente cada una de las técnicas de visualización científica convencionales con la propuesta en este trabajo y teniendo en cuenta cada una de las particularidades de los datos se pueden obtener buenos resultados.

7. Conclusiones

El desarrollo de nuevas técnicas de visualización científica permite a ingenieros e investigadores obtener información que puede ser determinante a la hora de tomar decisiones sobre la estructura o el fenómeno estudiado. La nueva técnica de visualización científica orientada a datos escalares discretos propuesta en este trabajo permite obtener información de fenómenos en los cuales intervengan alguna información discontinua. Puede ser aplicada en conjunto con las técnicas de visualización para datos escalares continuos y así lograr una mejor interpretacion del modelo. La aplicación de nuevas formulaciones tanto de los métodos tradicionales como de los Métodos de Elementos Discretos y Métodos sin Mallas ha traído consigo mejoras al análisis sobre determinados fenómenos o estructura material. Muchas de estas mejoras están dadas sobre el análisis que puede hacerse de la interpretación gráfica de la imagen final.

Referencias

1. Brodlie K.W., Carpenter L., Earnshaw R.A., Gallop J.R., Hubbold R.J., Mumford A.M., Osland C.D., Quarendon P., Eds. (1992) *Scientific visualization: techniques and applications*. Springer-Verlag New York, Inc., New York,NY, USA.

2. Johnson C., Hansen C. (2004) *Visualization Handbook*. Orlando, FL, USA: Academic Press, Inc.
3. Gallagher R.S. (1994) *Computer Visualization: Graphics Techniques for Engineering and Scientific Analysis*, 1st Edition. 0849390508. CRC Press, Inc.
4. Alexa M., Rusinkiewicz S., Guan X., Mueller K. Eds. (2004) Point-based surface rendering with motion blur
5. Bentley J. (1979) Multidimensional binary search trees in database applications. *IEEE Transactions on Software Engineering*. 5(4):333-340
6. Bentley J.L. (1975) Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509-517
7. *Data structures and algorithms for nearest neighbor search in general metric spaces* (1993) Society for Industrial and Applied Mathematics. Philadelphia, PA, USA.
8. Lomet D., Salzberg B. (1990) The hb-tree: A multiattribute indexing method with good guaranteed performance. *ACM Transactions on Database Systems* 15:625-658
9. Gargantini I. (1982) An effective way to represent quadtrees. *Commun. ACM* 25(12):905-910
10. Fukunage K., Narendra P. (1975) A branch and bound algorithm for computing knearest neighbors. *Computers IEEE Transactions on* C-24(7):750-753, July 1975
11. Beygelzimer A., Kakade S., Langford J. (2006) Cover trees for nearest neighbor in *ICML '06: Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM:97-104, 2006
12. Kollar T. (2006) Fast nearest neighbors. University of Rochester, Tech. Rep.
13. Arya S., Mount D.M., Netanyahu N.S., Silverman R., Wu A.Y. (1998) An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM* 45(6):891-923. [Online]. Available: citeseer.ist.psu.edu/arya94optimal.html
14. Sagawa R.,Masuda T., Ikeuchi K. (2003) Effective nearest neighbor search for aligning and merging range images. *3dim* 0:79
15. Atramentov A., LaValle S.M. (2002) Efficient nearest neighbor searching for motion planning in *Proceedings IEEE International Conference on Robotics and Automation*: 632-637.[Online].
Available: citeseer.ist.psu.edu/atramentov02efficient.html
16. Nene S.A., Nayar S.K. (1997) A simple algorithm for nearest neighbor search in high dimensions *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19:989-1003
17. Hart J.C., Francis G.K., Kauffman L.H. (1994) Visualizing quaternion rotation. *ACM Trans. Graph.* 13(3):256-276. [Online].
Available: citeseer.ist.psu.edu/article/hart93visualizing.html
18. Dam E.B., Koch M., Lillholm M. (1998) Quaternions, interpolation and animation, July 1998. [Online]. Available: citeseer.ist.psu.edu/dam98quaternions.html