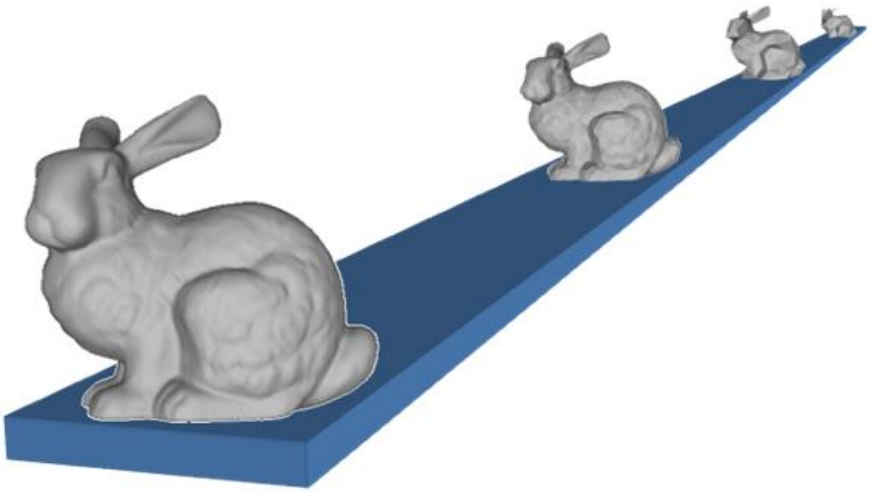
# Algoritmos de Pre y Post Proceso para Métodos Numéricos de Puntos, Métodos de Partículas y Libres de Mallas

H. Yervilla
L. González
A. Recarey



Monografía CIMNE Nº-136, Marzo 2013

# Algoritmos de Pre y Post Proceso para Métodos Numéricos de Puntos, Métodos de Partículas y Libres de Mallas

H. Yervilla [1]
L. González [1]
A. Recarey [1]

[1] Universidad Central “Marta Abreu” de las Villas

Monografía CIMNE Nº-136, Mayo 2013

**CENTRO INTERNACIONAL DE MÉTODOS NUMÉRICOS EN INGENIERÍA**
Gran Capitán s/n, 08034 Barcelona, España

CENTRO INTERNACIONAL DE MÉTODOS NUMÉRICOS EN INGENIERÍA
Edificio C1, Campus Norte UPC
Gran Capitán s/n
08034 Barcelona, España
www.cimne.com

Primera edición: Marzo 2013

**ALGORITMOS DE PRE Y POST PROCESO PARA MÉTODOS NUMÉRICOS DE PUNTOS, MÉTODOS DE PARTÍCULAS Y LIBRES DE MALLAS**
Monografía CIMNE M136


ISBN: 978-84-941004-7-5

Depósito legal: B-12797-2013

*A mi abuelita, ella es la persona más linda del mundo.*

*A mis padres y mi hermano.*

Me gustaría agradecer este trabajo a todos mis compañeros de la Universidad Central “Marta Abreu” de las Villas, a todo el equipo del AULA CIMNE-UCLV, CIMNE y en especial al equipo de GID de CIMNE y al Departamento de Bioingeniería de CIMNE por la por la acogida que me dieron en su grupo de investigadores, lo cual impulsó el desarrollo de este trabajo. Mi estancia en su equipo fue muy productiva para este trabajo y sé que para la siguiente etapa que me he trazado vendrán más logros.

En mecánica computacional, la visualización científica provee a investigadores e ingenieros de herramientas para el estudio de datos numéricos. La base de cada una de estas herramientas son las técnicas de visualización científicas que permiten la extracción de información a partir de los datos.

Este trabajo aborda las transformaciones necesarias a las técnicas de visualización científica convencionales para visualizar los resultados de la aplicación de métodos de partículas y métodos sin mallas. Para ello se tiene en cuenta la gran cantidad de datos resultantes de la aplicación de estos métodos y la presencia o no de información del contorno. Se desarrolla además una técnica de visualización para la representación de micro-fisuras y discontinuidades, las cuales constituyen el comienzo de las cadenas de fallos estructurales. Se escoge un método de generación de mallas por las facilidades que brinda y se adapta para la generación computacional de nubes de puntos para volúmenes y superficies.

Para cada una de las técnicas propuestas se estudian las ventajas de las estructuras de datos utilizadas y se muestran sus aportes a la computación gráfica y al análisis de resultados.

In computational mechanics, scientific visualization provides researchers and engineers with tools for studying numerical data. The basis of each of these tools is comprised by scientific visualization techniques.

This thesis deals with the necessary changes to conventional scientific visualization techniques in order to visualize the results obtained from the application of particle based methods and mesh-less methods. This is done taking into account the large amount of data that results from the application of these methods and the presence or absence of contour information. Moreover, it is developed a visualization technique for representing micro-cracks and discontinuities, which are the beginning of chains of structural failures. A mesh generation method is selected, given its provided facilities, and it is adapted to generate point clouds for representing volumes and surfaces.

For each proposed technique we study the advantages of the data structures used, and show its contributions to computer graphics and to data analysis.

# Índice

## Introducción

En mecánica computacional, cada vez más, las técnicas de modelización y simulación numérica en problemas de ingeniería conducen al desarrollo de nuevos métodos que modelan problemas a escala de la micro-estructura. Los modelos discretos (modelos de partículas) permiten aproximaciones de fenómenos físicos a una escala cada vez menor. En la ingeniería, por ejemplo, permiten realizar estudios de seguridad estructural con un alto grado de precisión y fiabilidad, lo que a su vez posibilita prever fallos estructurales severos.

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 nubes de puntos (*Método de Elementos Discretos*[1] y/o *Métodos Libres de Mallas*[2]) 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.

En la aplicación del *DEM* es necesario llevar a cabo una fase previa conocida como fase de empaquetamiento. En esta etapa se generan partículas por todo el dominio de estudio. Las cantidades de partículas obtenidas para realizar simulaciones a nivel micro-estructural son significativas en cuanto a su cantidad.

Los datos provenientes de la aplicación de estos métodos pueden llegar al orden de millones de partículas y en espacio en disco, gigas de información. La capacidad de cálculo y almacenamiento de los ordenadores cada día es mayor y permite a ingenieros e investigadores realizar simulaciones más fidedignas con respecto a la realidad pero ello conlleva el manejo de más información. La gran cantidad de partículas que forman el empaquetamiento es un problema cuando se quiere obtener una representación visual.

Antes de desarrollar una simulación utilizando el *DEM* es necesario analizar el comportamiento de ciertas propiedades del empaquetamiento tales como densidad local y global o distribución

[1] Por sus siglas en ingles DEM
[2] del inglés *meshless methods*

espacial, radio máximo/mínimo, si la generación se ajusta a la geometría del medio estudiado, entre otras, y tras la simulación se hace necesario el estudio de los resultados.

La gran cantidad de datos de las generaciones obtenidas dificulta en gran medida la visualización de propiedades por lo que es necesario aplicar filtros con el objetivo de eliminar datos que nunca influirán visualmente y con ello mejorar el tiempo requerido para la obtención de las imágenes y lograr interactuar, en tiempo real, con las representaciones de los empaquetamientos. La determinación de la visibilidad y/o simplificación de una escena es un problema fundamental en computación gráfica desde sus inicios.

Los algoritmos de visualización científica actuales están orientadas fundamentalmente a la visualización de datos escalares continuos, vectoriales y tensoriales. Una representación utilizando los software actuales de visualización a partir de resultados de simulaciones numéricas no permitiría obtener imágenes lo suficientemente demostrativas de las propiedades físicas del sistema de partículas pues no se cuenta con herramientas específicas para esta problemática.

Por otra parte, en los métodos libres de mallas se tiene un conjunto de posiciones/puntos independientes, en los cuales se conoce el valor de cierta magnitud, y se desea encontrar una función que permita deducir el valor de la magnitud en nuevos puntos y de esta forma estudiar su comportamiento y a la vez obtener visualizaciones. Es usual la generación de una malla con el objetivo de visualizar resultados, proceso que es complejo temporalmente.

Los métodos basados en puntos, Método Punto Material[3], Hidrodinámica de Partículas Suavizadas[4], Método de *Galerkin* de Elementos Libres, entre otros, necesitan de una discretización del medio. Las nubes de puntos utilizadas en estos métodos son obtenidas, usualmente, a partir de escáneres. En muchos problemas se tiene la forma implícita de determinados modelos y es necesaria la generación de una nube de puntos que la represente.

El *objetivo* de este trabajo es desarrollar algoritmos de pre y post-proceso orientados a la manipulación, visualización y generación de datos para métodos numéricos de puntos, métodos de partículas y libres de mallas.

[3] MPM por sus siglas en ingles.
[4] SPH por sus siglas en ingles

Para dar cumplimiento a este objetivo se desarrollaron las siguientes tareas:

1. Diseñar e implementar algoritmos para determinar visibilidad en conglomerados de partículas aplicando técnicas de aceleración en la visualización.

2. Aplicar las técnicas de visualización científica interpolación de colores y superficies de contorno a conglomerados de partículas para la representación de propiedades físicas.

3. Proponer y desarrollar un método para la visualización de propiedades de datos en Métodos Libres de Mallas.

4. Crear algoritmos para la visualización científica de datos discretos que simbolizan discontinuidades en el modelo de estudio.

5. Aplicar el algoritmo rellenado de iso-superficies a la generación de nubes de puntos de superficies y volúmenes para la simulación en métodos basados en puntos.

En el **capítulo I** se hace un esbozo de la visualización científica y la forma para representar datos escalares, vectoriales y tensoriales. Se estudian las posibles técnicas para acelerar el proceso de visualización en general. Se analiza la interpolación de datos esparcidos y las funciones de base radial con el objetivo de visualizar resultados de métodos sin mallas. Se exponen estructuras de datos de particionado espacial identificando claramente sus mejores usos y ventajas en cada caso y se analizan algunos métodos para la generación de nubes de puntos.

El **capítulo II** contiene cada uno de los algoritmos propuestos tanto para pre como post-proceso. Específicamente, se presentan los algoritmos para el cálculo de la visibilidad en escenas de conglomerados de partículas, las transformaciones prácticas a los algoritmos de visualización científica para la representación de propiedades de conglomerados de partículas. Se propone un nuevo algoritmo para la representación de micro-fisuras y discontinuidades. Se explican los conceptos y ventajas del algoritmo rellenado de iso-superficies utilizado para la generación de nubes de puntos.

El **capítulo III** contiene los resultados de los algoritmos propuestos para la visualización de datos obtenidos mediante la aplicación de métodos de partículas y sin mallas, para modelos con algún tipo de micro-fisuras o discontinuidades y el algoritmo basado en un método de mallado para la generación de nubes de puntos. Se exponen las ventajas, desventajas y algunas recomendaciones de cada uno de ellos.

# Capítulo I: Introducción a la visualización científica de resultados de métodos numéricos.

El ser humano obtiene una gran cantidad de información a través de la visión, se estima que un 50% de las neuronas del cerebro humano están dedicadas a la percepción. El sistema visual es un buscador de patrones de extrema fuerza y sutileza, 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.

## 1.1 Visualización científica.

El término "**visualización científica**" es utilizado cuando se procesan datos usando computadoras con el fin de obtener imágenes que describan determinada propiedad o comportamiento. En el caso de la mecánica computacional se refiere a la representación de información utilizando técnicas bien definidas para datos escalares, vectoriales o 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.

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 del ser humano.

La visualización asistida por computadoras es utilizada desde casi el mismo momento en que surge el primer ordenador. El primero en utilizar el término es *William Fetter*, 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 este momento 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.

El desarrollo acelerado de la computación, y por tanto, el incremento de sus potencialidades, ha motivado en gran medida el auge de técnicas para la simulación, modelación y estudio de fenómenos reales. Las capacidades de cálculo y almacenamiento de las computadoras

permiten el procesamiento de grandes volúmenes de datos, haciendo las técnicas y algoritmos más exactos. Esto ha llevado a que se necesiten metodologías potentes y eficaces a la hora de visualizar datos provenientes de simulaciones numéricas o fenómenos reales.

Las técnicas de visualización son el elemento fundamental dentro del proceso de representación de datos científicos. En [1-3] se describen la mayoría de las técnicas y algoritmos relacionados con la visualización científica. En el caso de datos provenientes de métodos de partículas o métodos sin mallas no se cuenta con algoritmos específicos para su visualización, es necesario modificar las metodologías existentes para obtener representaciones con valor gráfico.

## 1.2 Visualización de propiedades escalares, vectoriales y tensoriales.

Las técnicas de visualización pueden agruparse atendiendo al tipo de datos que visualizan. Existen infinitas maneras de representar un conjunto de datos. Es por ello que resulta necesario escoger, una vez definido el tipo de datos a visualizar, la técnica que sea capaz de extraer la información que se desea. La dimensión del espacio donde se ubican los datos constituye una característica determinante en la selección, atendiendo a que no todos los métodos tienen las mismas potencialidades o factibilidad en una, dos, tres o incluso más dimensiones. También es importante destacar que usualmente no basta con una sola técnica, sino que a un mismo conjunto de datos iniciales se aplican varias técnicas, obteniendo información útil de cada resultado y arribando a conclusiones que pudieran ser definitivas.

Los datos a visualizar son magnitudes de determinadas propiedades –físicas, mecánicas, térmicas, etc. – obtenidas ya sea por mediciones realizadas en el mundo real, o a partir de procesos numéricos. Valores de propiedades como temperatura, profundidad o distancia son representados mediante magnitudes escalares, la fuerza y la velocidad del viento o de las partículas en un fluido utilizando vectores. Atendiendo al tipo de atributos a representar los algoritmos de visualización pueden dividirse en técnicas de visualización científicas para datos escalares, vectoriales y tensoriales. Técnicas de visualización científica como mapeo de colores, iso-líneas, iso-superficies, líneas de fluidos entre otras están entre las más utilizadas y conocidas.

Las nuevas formulaciones del *Método de Elementos Finitos*[5] y otros semejantes y el creciente auge de otros como los métodos basados en partículas o nubes de puntos han venido a engrosar el mundo de las investigaciones de problemas de la Física-Matemática y el comportamiento de los materiales. 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 forman parte de las aplicaciones más comunes de estas formulaciones. Existen fenómenos donde se aprecian magnitudes con un comportamiento mixto.

Muchas de las propiedades obtenidas a partir de estos métodos pueden ser visualizadas utilizando las técnicas más comunes de visualización científica mencionadas con anterioridad, pero cuando se tienen que analizar datos del tipo discreto o mixto es necesaria la utilización de alguna metodología que suministre al investigador o ingeniero la información requerida.

Cuando se tienen datos escalares discretos, lo más común es utilizar pequeños símbolos o figuras para representar el punto discontinuo; esta técnica es conocida como *pequeños polígonos*[6]. Es poco utilizada y por consiguiente pasa casi desapercibida para la bibliografía existente.

## 1.3 Aceleración de la visualización de grandes volúmenes de partículas.

El principal desafío para sistemas de *generación de imágenes*[7] en tiempo real es proporcionar un adecuado *frame rate*[8], incluso para los modelos más complejos en 3D. El desarrollo de hardware se ha incrementado dramáticamente en los últimos años, permitiendo la generación de imágenes de varios millones de primitivas gráficas en segundos. Además, el hardware gráfico cada vez es más programable, permitiendo más realismo en las escenas. Por otra parte, debido al continuo deseo de mayor detalle y realismo, la complejidad de los modelos es cada vez más grande, haciendo las escenas tan complejas que no pueden ser presentadas de manera interactiva o incluso en tiempo real. Por tales motivos, la aceleración de la visualización continúa siendo un tópico muy tratado en computación gráfica. [4]

[5] FEM por sus siglas en ingles
[6] Tiny Poligon
[7] en ingles conocido como render
[8] en inglés **Frames per Second** (**FPS)** o **Frame rate** o **frame frequency** en computación es conocido como ***real-time computing*** (**RTC**)

La única manera de obtener un alto *frame rate* es reduciendo la complejidad de la imagen a representar. Existen muchos algoritmos y todos se basan en algunas de las siguientes estrategias: [4]

1. La optimización de los cuellos de botella en el proceso de visualización, producto de la complejidad del comportamiento de los *drivers* y *hardwares* gráficos, puede agilizar la visualización.
2. El cálculo de visibilidad permite remover porciones de la escena invisibles desde el punto de vista del observador que no tendrán que ser procesadas por el *hardware* gráfico.
3. La simplificación de la geometría cuando partes de la escena se encuentran a una distancia tal que no contribuyen mucho a la imagen final.
4. Las representaciones basadas en imágenes es posible usarlas si la simplificación de la geometría no es factible.

En [5] se presentan y explican los algoritmos para acelerar el proceso de visualización de gráficos en computadoras. Técnicas para eliminar objetos dentro de una escena[9], la utilización de impostores gráficos[10] y las técnicas de nivel de detalle[11] son de las más conocidas y aplicadas con tales objetivos.

### 1.3.1 Técnicas para seleccionar objetos a partir de una escena.

Cuando miramos un objeto desde un determinado ángulo solo podemos observar solo una parte de este. Pongamos como ejemplo un cubo, el cubo tiene seis caras y como máximo solo podemos observar tres: cuando la cámara se sitúa en la línea del horizonte y de frente a una de sus caras (perspectiva cónica) solo podemos observar una cara, mientras que si nos movemos hacia uno de los lados comenzamos a observar otra de las caras (perspectiva caballera), para ver tres caras tenemos que mirar el cubo desde una perspectiva isométrica. Además, de que no es posible observar determinadas caras de un objeto, también existen objetos en la escena

[9] *culling techniques*
[10] *impostor algorithms*
[11] *level-of-detail techniques*

que no son visibles ya que se encuentran ocultos tras otros o sencillamente no se encuentran en nuestro ángulo de visión.

Estos tres sencillos aspectos o características son ampliamente utilizados en computación gráfica con la finalidad de acelerar la generación de las imágenes: determinar qué caras o qué partes de las caras que conforman un sólido no son visibles y qué objetos se encuentran ocultos tras otros más cercanos a la cámara o no se ven desde la posición de esta, dando, en parte, solución al problema de la visibilidad. Técnicas tales como *back-face culling*, *hierarchical view-frustum culling, portal culling, detail culling* y *occlusion culling* son las más utilizadas y conocidas. (Véase Figura 1).

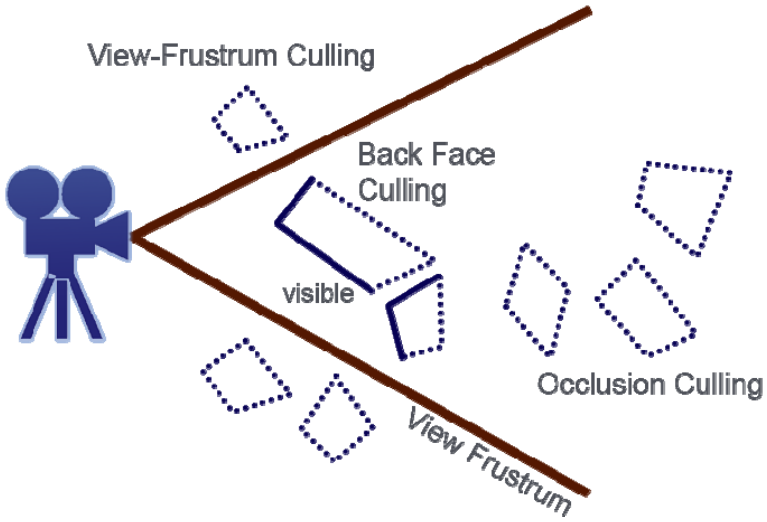



**Figura 1 Algunas de las técnicas para determinar la visibilidad. (i) View-Frustum Culling. (ii) Back-Face Culling. (iii) Occlusion Culling. (Tomado de [6])**

Mediante la técnica *eliminación de polígonos no visibles*[12] es posible determinar cuando una parte de la superficie de un objeto es visible dependiendo de la posición de la cámara, esto reduce considerablemente el número de primitivas a dibujar. Esta forma de seleccionar se aplica a objetos cerrados, y la determinación de la visibilidad o no de una parte del objeto de realiza chequeando la disposición de esa parte con respecto a la cámara, o sea, si está de frente o no, ya que la porción del objeto que está contraria al observador es invisible. Matemáticamente hablando, se traduce a una comparación del ángulo formado por el vector que indica la dirección de la cámara y la normal de la superficie del objeto, si este ángulo es

[12] back face culling

menor de 90 grados el polígono puede ser descartado. En [7-9] podemos encontrar eficientes implementaciones de esta técnica.

La técnica *eliminación de objetos fuera del ángulo de visión*[13] se encarga de obtener qué primitivas se encuentran dentro o parcialmente dentro del *cono de visión* y necesitan ser visualizados. Para ello se utilizan estructuras de datos tales como *AABBs*, *OBBs*, *octree*, *BSP tree*, unos de buen rendimiento en escenas dinámicas y otros en escenas estáticas. [5]

Los algoritmos para la *eliminación de objetos ocluidos por muros*[14] son usados fundamentalmente en modelos de arquitectura considerando las paredes como grandes objetos ocluyentes, los objetos en la escena son explorados en una variedad de formas buscando el *conjunto potencialmente visible* (PVS) para varios puntos de visión. El PVS es formado tomando la escena completa y a partir del conjunto de puntos de visión se van excluyendo aquellas partes de la escena que no van a contribuir a la imagen final.

La técnica *eliminación de detalles*[15] sacrifica calidad en la escena por ganar en velocidad de generación de las imágenes. Se basa en que pequeños detalles en la escena contribuyen poco o nada cuando el observador está en movimiento, con se está en una posición fija usualmente esta técnica se deshabilita.

El problema de la visibilidad puede ser resuelto vía hardware utilizando el *z-buffer* pero aunque puede hacerlo correctamente no es un mecanismo lo suficientemente eficaz, por ejemplo, imaginemos que el observador está mirando a través de una línea donde yacen 10 esferas, una imagen generada a partir de este observador mostraría una sola esfera, todas las esferas son comparadas con el *z-buffer* y potencialmente escritas para el *color buffer* y el *z-buffer*, dando lugar a que nueve esferas sean dibujadas innecesariamente.

La técnica *eliminación de objetos ocultos*[16] se basa en seleccionar los objetos de una escena que no pueden verse desde la posición de la cámara porque se encuentran ocluidos por otros y pueden ser despreciados a la hora de generar la imagen [5, 6]. En [10-12] se encuentran varias implementaciones eficientes de esta técnica. La eficacia de esta técnica se basa en que en la escena existan objetos que sean “buenos” ocluyentes, es decir, objetos relativamente grandes capaces de ocultar tras si varios objetos más pequeños.

[13] View-frustum culling
[14] Portal culling
[15] Detail culling
[16] Occlusion culling

### 1.3.2 Utilización de impostores gráficos.

Los impostores[17] son usualmente referenciados como entidades basadas en imágenes, adquiridas a partir de la generación de partes de la escena y que son usadas como representaciones alternativas de esa misma escena para acelerar el proceso de visualización. Este tipo de primitiva es mucho más eficiente cuando el objeto y el observador se mueven lentamente. Una característica que los debe identificar es que un impostor debe ser más rápido de dibujar que el objeto que representa y debe tener las mismas características visuales. [4, 5]

Un impostor puede ser generado en una etapa de pre-proceso, llamado impostor estático, o dinámicamente en tiempo de ejecución, llamado impostor dinámico. Durante el proceso de visualización el impostor es mostrado en lugar de partes de la escena original. El tiempo de visualización depende básicamente del tamaño de este en pantalla y no de la complejidad de la parte de la escena que va a representar. Permite una representación rápida de objetos de alta complejidad geométrica. [4]

A la hora de generar un impostor son adquiridas tanto la apariencia como la estructura geométrica del objeto real. Esto se hace a partir de un punto de visión (punto del observador), llamado punto de visión de referencia. Existen diferentes tipos de técnicas para obtener impostores caracterizadas por el tipo y el volumen de datos geométricos. El caso más simple de impostor gráfico es cuando parte de una escena es generada a partir una *textura*[18] y se combina con un *cuadrilátero*[19] para colocarla dentro de la escena.

La principal ventaja de los impostores comparado con la simplificación de geometrías es que el proceso de generación no depende de ningún conocimiento de la estructura geométrica del objeto real. Esto le permite ser usados en escenas de cualquier tipo. Si la diferencia entre un impostor y el objeto original no es detectable es llamado *impostor valido*. [4]

Entre las desventajas en cuanto a su uso se puede encontrar con un alto nivel de requerimiento de memoria o de actualización del impostor dependiendo si es estático o dinámico respectivamente.

[17] Conocidos también como sprites
[18] Impostor texture.
[19] Impostor geometry

### 1.3.3 Técnicas de nivel de detalle.

Los algoritmos de nivel de detalle[20] permiten obtener múltiples representaciones de un modelo, algunas de alta resolución (donde se necesitan muchas primitivas) y otras de baja resolución (pocas primitivas), dependiendo de la distancia a que se encuentre el observador. Son a menudo llamadas técnicas de *multi-resolución*.

El modelado *multi-resolución* representa a un objeto con múltiples aproximaciones y permite extraer la más adecuada de todas según criterios dependientes de la aplicación. Un modelo *multi-resolución* está formado por los datos que representan el objeto y los algoritmos de extracción de estos. Los criterios para determinar la mejor aproximación dependen de cada aplicación en particular. [13]

El fin del modelado *multi-resolución* es enviar al sistema gráfico únicamente la información que mejor describe a un objeto en cada momento. Usualmente en un modelo *multi-resolución* tanto las estructuras de datos como el algoritmo de extracción del nivel de detalle se basan en el uso de primitivas básicas. Utilizar primitivas gráficas que aprovechen la información de conectividad entre los triángulos de una malla, como las tiras o los abanicos de triángulos, reduce notablemente la cantidad de información necesaria para dibujar la malla. [13]

[20] Level of Detail (LOD)

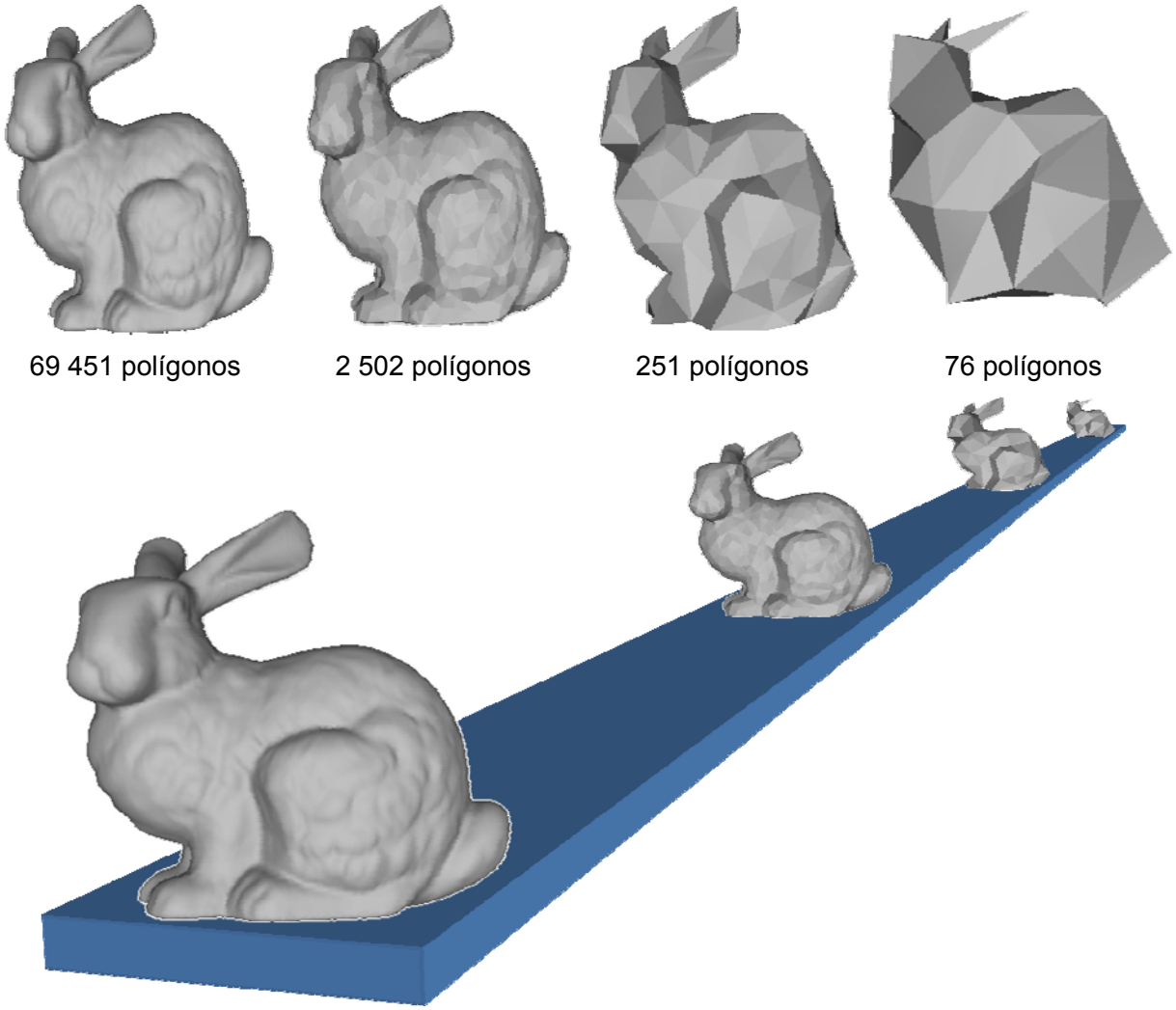



Figura 2 Diferentes niveles de detalle para un mismo modelo.

Existen tres acercamientos básicos para manejar el nivel de detalle, un acercamiento discreto, uno continuo y otro dependiente de la vista.[14]

El acercamiento tradicional propuesto en [15], donde se codifican los principios básicos detrás de esta técnica, conocido como nivel de detalle discreto y que en la actualidad es usado en muchas aplicaciones gráficas sin hacerle modificaciones apenas se basa en crear múltiples versiones de cada objeto, cada una con un nivel de detalle, durante un pre-proceso. En tiempo de ejecución es escogido el nivel apropiado para representar el objeto. A medida que el observador se aleja del objeto el número total de polígonos se reduce y se incrementa la velocidad de generación de las imágenes. Debido a que las versiones de los objetos son calculadas previamente el proceso de simplificación no puede predecir a partir de qué dirección

el objeto será observado y esto provoca una reducción o ampliación uniforme de cada objeto. Es llamado isotrópico o independiente de la vista [14]. (Véase Figura 2)

El nivel de detalle discreto tiene muchas ventajas, la mayoría de los cálculos se realizan en una etapa previa a la generación de la imagen y el algoritmo gráfico simplemente necesita escoger cual nivel es el recomendado para cada objeto. Además muchos hardwares gráficos permiten cargar múltiples versiones de un modelo.

El acercamiento continuo crea niveles de detalles individuales durante un pre-procesamiento, el sistema de simplificación crea una estructura de datos que contiene un rango de colores continuo de detalles (*spectrum*). El nivel de detalle deseado es extraído de esta estructura en tiempo de ejecución. La mayor ventaja es que se logra una mejor granularidad. Una mejor granularidad permite un mejor uso de recursos y una mejor fidelidad en la cantidad de polígonos. Cuando debe ser cargado un modelo grande desde disco o a través de la red, un nivel de detalle continuo provee una generación de la imagen de manera progresiva y carga ininterrumpida, a menudo, propiedades muy útiles.

El nivel de detalle dependiente de la vista extiende el continuo usando un criterio de simplificación dependiendo de la vista para seleccionar dinámicamente el más apropiado. Además, es anisótropo, un objeto sencillo puede abarcar múltiples niveles de simplificación, permite también mucho mejor granularidad: mucha más fidelidad y una mejor optimización de los escasos recursos.

A pesar de las ventajas de los niveles de detalle continuo y dependiente de la vista, el acercamiento discreto es más común en la práctica. El procesamiento extra requerido para evaluaciones, simplificaciones y refinamiento del modelo en tiempo real y el requerimiento de memoria extra para la estructura de datos utilizada hace que muchos investigadores se decanten por el acercamiento tradicional. [14]

Para más detalle de estas técnicas ver [5, 14].

### 1.3.4 Unidad de procesamiento gráfico.

Independientemente de la implementación de técnicas de nivel de detalle para la visualización de resultados de métodos numéricos en la ingeniería la cantidad de información a visualizar puede ser muy grande. El procesado de formas y texturas en estos casos se hace de manera

continua y hacerlo por medio de la *CPU* no suele ser muy buena idea, ya que no está específicamente diseñada para esta tarea.

Para eso está la *unidad de procesamiento gráfico* o *GPU*, un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos y/o aplicaciones 3D interactivas. Este procesador está específicamente diseñado para trabajar con gráficos, con funciones específicas y una arquitectura basada en el procesamiento paralelo.

La *GPU* implementa ciertas operaciones gráficas llamadas primitivas optimizadas para el procesamiento gráfico. Una de las técnicas más comunes para el procesamiento en 3D es el *antialiasing*, que suaviza los bordes de las figuras para darles un aspecto más realista. Adicionalmente existen primitivas para dibujar rectángulos, triángulos, círculos y arcos. Las *GPU* actuales disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos.

También se intenta aprovechar la gran potencia de cálculo de las *GPU* para aplicaciones no relacionadas con los gráficos, lo que recientemente se viene a llamar *GPGPU*, o *GPU* de propósito general[21].

## 1.4 Visualización en métodos sin mallas.

Una vez realizado el proceso de análisis o simulación utilizando métodos libres de mallas los resultados son presentados como un conjunto de valores numéricos asociados a localizaciones discretas esparcidas por todo el dominio de estudio y que no tienen asociación estructural dada la característica que tienen estos métodos de carecer del uso de mallas durante el proceso de cálculo. Un ejemplo de esto son los resultados mostrados en [16] donde se aplica el método de puntos finitos para mecánica de fluidos utilizando conjuntos de más de 500 000 partículas. Un proceso de generación de mallas con esa cantidad de puntos -suponiendo que se considere el centro geométrico de las partículas- sería bastante costoso.

Uno de los aspectos importantes a considerar en el ámbito de la visualización científica es que las imágenes son dibujadas sobre una matriz de dimensiones finitas compuestas de píxeles que representan un único color. Cuando a partir de los puntos esparcidos se realiza la triangulación correspondiente para obtener la malla, normalmente a cada triángulo corresponde

[21] General Purpose GPU, por sus siglas en inglés

un conjunto determinado de píxeles en la imagen resultante. Sin embargo, cuando se cuenta con una cantidad relativamente grande de puntos, al realizar la triangulación varios triángulos corresponden a un mismo píxel. Si se genera una imagen con resolución de 500x500 píxeles a partir de un conjunto de datos conformado por dos millones de puntos, entonces se van a tener 250 mil píxeles que son insuficientes para representar ni siquiera, uno a uno los puntos del conjunto de datos.

Los datos a representar por lo general representan medios continuos, pero lo que en realidad se tiene es un conjunto finito de valores pertenecientes a puntos discretos del dominio de estudio. Estos puntos pueden ser tomados de forma regular o irregular/aleatoria. Cuando los puntos son tomados de manera aleatoria entonces se les da el nombre de "datos esparcidos"[22].

### 1.4.1 Interpolación de datos esparcidos.

Para obtener una representación de datos con estas características es deseable encontrar una función que permita deducir el valor de la magnitud en nuevos puntos y de esta forma estudiar el comportamiento del proceso. Esto es, se quiere encontrar una función $F_a$ que ajuste "adecuadamente" los datos que se tienen. Existen básicamente dos enfoques que abordan este problema, en el primero de ellos $F_a$ mantiene exactamente los valores iniciales en los puntos conocidos, dándosele el nombre de función interpolante, el segundo enfoque se basa en obtener $F_a$ de forma tal que la distancia de los puntos conocidos a ella sea mínima, y en este caso se le da el nombre de función aproximante. En este trabajo se va a abordar únicamente el enfoque de la función interpoladora. En el caso de las funciones aproximantes pueden encontrarse aspectos esenciales relacionados con ellas en [17-20].

Realizando una definición más precisa del problema: Sean $x_i \in R^s; i = 1..n$ los puntos esparcidos de manera aleatoria y $f(x) \in R$ el valor asociado a cada uno de ellos. Entonces la definición formal del problema es la siguiente[23]:

***Problema 1 (Interpolación de datos esparcidos)***: Dada la dupla $(\mathrm{x_i}, \mathrm{f(x_i)})$, $\mathrm{i = 1..n}$ con $\mathrm{X = \{x_1, x_2, \dots, x_n\}, X \subset R^s}$ y $\mathrm{f(x_i) \in R}$ encontrar la función continua $\mathrm{F_a(x): R^s \rightarrow R}$ tal que $\mathrm{F_a(x_i) = f(x_i)}$.

[22] Del inglés "*scattered data*"

[23] Aunque en este artículo se hace alusión únicamente a funciones de tipo escalar, los elementos que se formulan pueden ser fácilmente generalizados a funciones de tipo vectorial.

Asumir que $F_a$ es una combinación lineal de un conjunto de funciones básicas es una manera muy común y conveniente de resolver este problema, esto es,

$$\mathrm{F_a(x)} = \sum_{\mathrm{k=1}}^{\mathrm{n}} \mathrm{c_k F_k(x)}\,, \mathrm{x} \in \mathrm{R^s} \qquad (1)$$

Esto, dada la condición $\mathrm{F_a(x_i)} = \mathrm{f(x_i)}$, se convierte en un sistema de ecuaciones lineales de la forma

$$\mathrm{Ac} = \mathrm{y} \qquad (2)$$

donde cada uno de los elementos de la matriz de interpolación $\mathrm{A}$ tiene la forma

$\mathrm{A_{jk}} = \mathrm{F_k(x_j)}, \mathrm{j,k} = 1..\mathrm{n}; \mathrm{c} = [\mathrm{c_1}, \mathrm{c_2}, \dots, \mathrm{c_n}]^{\mathrm{T}}$ e $\mathrm{y} = [\mathrm{f(x_1)}, \mathrm{f(x_2)}, \dots, \mathrm{f(x_n)}]^{\mathrm{T}}$

De aquí que el ***Problema 1*** va a tener una única solución si y solo si la matriz A es no singular. Lo anteriormente descrito es tratado con mayor detalle en [19], incluyendo un análisis adicional para el caso de la multivariabilidad.

Por la gran frecuencia con la que se encuentra el problema de la interpolación de datos esparcidos, es posible encontrar varios trabajos relacionados con su solución [21-24], ya no solo enfocados al problema de la visualización, sino en un ámbito más general. De aquí que existan un gran número de técnicas que son bien conocidas en la actualidad y que dan solución al ***Problema 1*** y de las que pueden encontrarse varias referencias en la bibliografía.

Existen métodos basados en la inversa de la distancia entre los puntos. La idea original fue dada por *Shepard* [25] y tiene un enfoque global, lo que la hace demasiado ineficiente. Sin embargo varias modificaciones han sido realizadas, tanto para mejorar la eficiencia computacional llevando la idea inicial a un enfoque local, como para incrementar la exactitud de la función interpoladora [26, 27].

Por otra parte existen métodos que trabajan a partir de la vecindad natural entre los datos. Estos se basan de alguna manera en la construcción del diagrama de *Voronoi* correspondiente a los puntos que conforman los datos iniciales [21, 28]. La principal dificultad radica en la lentitud de estos métodos ante la presencia de grandes volúmenes de datos, especialmente por la necesidad de obtener el Diagrama de *Voronoi*. Sin embargo en [24] se obtienen buenos resultados en este sentido.

La interpolación mediante el uso de funciones de bases radial ha venido ganando en importancia dado el auge que han alcanzado los métodos libres de mallas, el *MED* y el *Método de Elementos Finitos para Partículas*[24] ya que no requieren de una estructuración de los datos, como en el caso de las interpolaciones basadas en el *Diagrama de Voronoi*.

### 1.4.2 Interpolación usando Funciones de Base Radial

La técnica de interpolación de datos esparcidos usando *funciones de base radial*[25] consiste en la obtención de la combinación lineal de un conjunto finito de funciones de la forma $\phi(\|.\|)$, donde $\|\cdot\|$ generalmente hace alusión a la distancia euclidiana, y que son radialmente simétricas. Una de sus ventajas radica en la independencia que poseen de la dimensión del espacio, ya que utilizan mayormente la norma euclidiana, lo que las hace fácilmente extensibles a cualquier dimensión.

Elementos relacionados con la teoría de las FBR pueden encontrarse en [19, 22, 29, 30].

Se define entonces de manera formal las *funciones radiales*:

**Definición *(Funciones Radiales):*** Una función $\phi : \mathrm{R}^s \rightarrow \mathrm{R}$ se dice que es radial si existe una función univariable $\varphi : [\mathrm{o}, +\infty] \rightarrow \mathrm{R}$ tal que $\varphi(\mathrm{x}) = \varphi(\|\mathrm{x}\|)$ con $\|\cdot\|$ una norma cualquiera, generalmente la norma euclidiana.

Si se aplica esta definición y se sustituye en (1) se obtiene la formulación de la solución al ***Problema 1*** usando **FBR**:

$$F_a(x) = \sum_{k=1}^{n} c_k \phi(\|\mathrm{x} - \mathrm{x_k}\|), x \in R^s \qquad (3)$$

Existen varios tipos bien conocidos de funciones radiales como son la *Gausiana* $\phi(\mathrm{r}) = \mathrm{e}^{-\mathrm{ar}}$, los *splines de planos delgados*[26] $\phi(\mathrm{r}) = \mathrm{r}^2 \log(\mathrm{r})$ y las *multi-cuádricas* $\phi(\mathrm{r}) = \sqrt{\mathrm{r}^2 + \mathrm{c}^2}$. Si se analizan las dos últimas funciones, a medida que el radio aumenta, aumenta también el valor que ellas devuelven, por tanto, son funciones de influencia global en el conjunto de datos esparcidos. Este comportamiento se traduce en que la matriz $A$ definida en (2) sea una matriz muy densa, lo cual implica que ante la adición de un nuevo punto sea necesario recalcular todos los coeficientes. Por su parte, la forma *Gausiana* tiende a cero cuando se incrementa el

[24] por sus siglas en ingles PFEM
[25] por sus siglas en ingles FBR
[26] thin-plate splines

radio, haciendo que la influencia en un punto $x$ de los elementos $(x_i, f(x_i))$ sea cero si $\|x - x_i\| < r_L$ donde $r_L$ es tal que $e^{-ar_L} < \varepsilon$, para $\varepsilon$ suficientemente pequeño. Esta característica de "localidad" da como resultado una matriz $A$ de banda y dispersa, por lo que adicionar nuevos puntos solo afectaría un pequeño conjunto de coeficientes. Por estas razones, el enfoque local es computacionalmente menos costoso, ya que solo es necesario considerar los puntos pertenecientes a una vecindad y no la totalidad de ellos.

Algunos ejemplos muy interesantes de funciones radiales con influencia local son las presentadas por *Wendland* en [31]. Estas funciones tienen la forma

$$\phi(r) = \begin{cases} p(r) & 0 \le r \le 1 \\ 0 & r > 1 \end{cases}$$

donde $p(r)$ es uno de los polinomios dados por *Wendland*, y son fácilmente escalables.

Analizando de modo general las funciones radiales de alcance local: sea $\phi(r)$ la función radial de alcance local que se va a utilizar para construir la función de interpolación, entonces $(r_i, f(x_i))$ influye únicamente en los puntos que se encuentren ubicados a una distancia $L \le r_L$, donde $\phi(r > r_L)$ es cero o despreciable. Esto es equivalente a decir: sea $x \notin X$ entonces $I_x = (x_i;\ x_i \in X,\ \|x - x_i\| \le r_L)$ es el conjunto de puntos que ejercen determinada influencia sobre $x$. Localmente, el problema sería resuelto hallando la solución a (1), pero ahora únicamente considerando los puntos pertenecientes a $I_x$. La obtención de $I_x$ es un proceso computacionalmente costoso valorando que se trata de un volumen considerable de datos esparcidos sin ninguna relación previa, más allá de su ubicación geométrica. La utilización de estructuras de datos que garanticen un almacenamiento adecuado y que faciliten las consultas de vecindad se hace necesaria para agilizar computacionalmente los cálculos necesarios. En este trabajo se hace uso de los árboles de cubrimiento con ese objetivo.

## 1.5 Estructuras de datos para búsqueda de vecindad y particionado espacial.

En la mayoría de los trabajos relacionados con la computación gráfica y la geometría computacional hay problemas que son reiterativos. Mejoras a algoritmos y a estructuras de datos relacionados con la búsqueda de vecindad y el particionado del espacio aparecen cada día. A continuación se exponen algunas de las estructuras de datos utilizadas.

### 1.5.1 Árbol octal.

Un árbol octal u *octree* es una extensión de los arboles binarios de búsqueda donde cada nodo puede tener hasta ocho hijos. Son una estructura jerárquica de consulta espacial, que se utilizan para representar la relación espacial de objetos geométricos en 3D y permite búsquedas rápidas reduciendo la cantidad de datos a un octavo. Cada uno de sus hijos representa uno de los ocho octantes en que es subdividido el espacio. Es usado fundamentalmente para particionado espacial, como guía en algoritmos de mallado, como estructura de búsqueda espacial y en algoritmos de nivel de detalle y obtención de visibilidad en escenas.

Un árbol octal es obtenido subdividiendo jerárquicamente el espacio en ocho partes [32, 33]. Esta subdivisión continúa recursivamente hasta lograr el tamaño de celda deseado. Una característica importante es que relacionado a cada nodo se encuentra un color, el cual indica la posición del cubo con respecto al objeto. Los colores utilizados son el blanco, el negro y el gris, el blanco indica que el cubo se encuentra completamente dentro de la figura, el negro indica lo contrario y el gris que el cubo se encuentra en la superficie del cuerpo. Si un nodo es de color blanco o negro indica que todos sus hijos son del mismo color del padre. Este uso de colores permite para casos deseados no continuar subdividiendo el espacio innecesariamente, ejemplo, un octante completamente en el exterior, de color negro, es posible que no sea deseable subdividirlo y así ganar en espacio de almacenamiento. (Véase Figura 3)

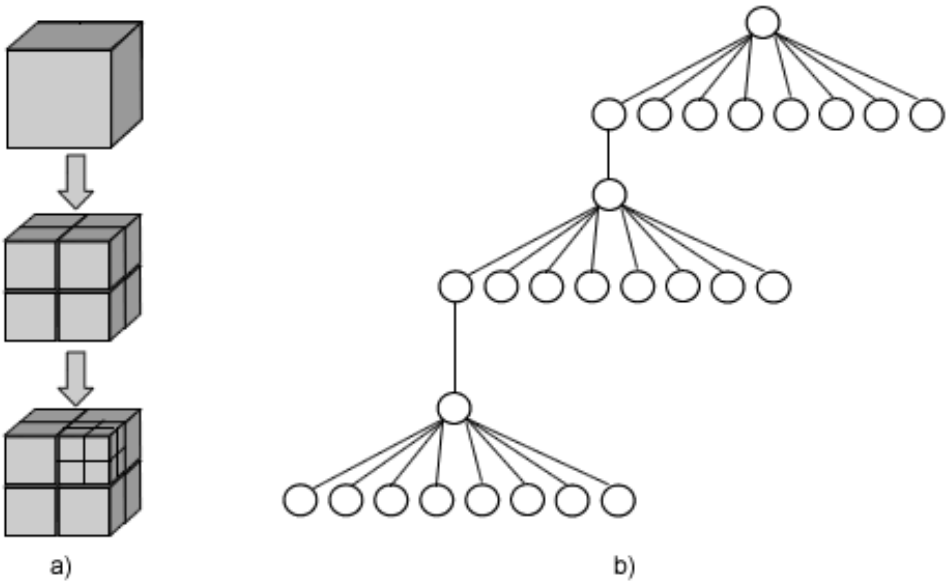



Figura 3 Árbol octal

El árbol octal es una estructura muy utilizada y de aplicaciones muy variadas. Es usada en el cálculo de visibilidad en escenas, si un nodo del árbol es invisible entonces todos los objetos en ese nodo no necesitan ser chequeados individualmente, en técnicas de nivel de detalle, ya que es sencillo el cálculo de la distancia de un objeto a partir de la cámara y consecuentemente asignar un apropiado *LOD*, en intercepción de rayos, si un nodo del árbol no es interceptado entonces los objetos que pertenecen a ese nodo no son interceptados de ninguna manera, también en detección de colisiones, entre otras aplicaciones.

### 1.5.2 Árbol de cubrimiento.

El árbol de cubrimiento[27] es una estructura jerárquica utilizada fundamentalmente en consultas de rango y en el problema del vecino más cercano. Es un árbol nivelado donde cada nivel es "cubierto" por el nivel inferior. Cada nivel es indexado con un número entero el cual decrece a medida que se desciende por el árbol. Cada punto en el árbol puede estar asociado con múltiples nodos, pero se requiere que cada punto aparezca a lo sumo una vez en cada nivel.

Sea $C_i$ el conjunto de puntos en $S$ que pertenecen a los nodos de nivel $i$, entonces, el árbol de cubrimiento debe cumplir las siguientes propiedades: [34]

**1.** $\boldsymbol{C_i} \subset C_{i-1}$. Esto implica que una vez que el punto $p \in S$ aparezca por vez primera en $C_i$ entonces cada nivel inferior del árbol contiene al nodo asociado con $p$ (Anidación). (Figura 4 a)

**2.** Para todo $p \in C_{i-1}$ existe $q \in C_i$ tal que $d(p,q) \leq 2^i$ y el nodo en el nivel $i$ asociado con $q$ es el padre del nodo en el nivel $i-1$ asociado con $p$. (Cubrimiento). (Figura 4 b)

**3.** Para todo punto $p, q \in C_i$, tal que $p \neq q,\ d(p,q) > 2^i$. (Separación). (Figura 4 c)

Con $d(p,q)$ la distancia euclidiana entre $p$ y $q$.

[27] Del inglés covertree

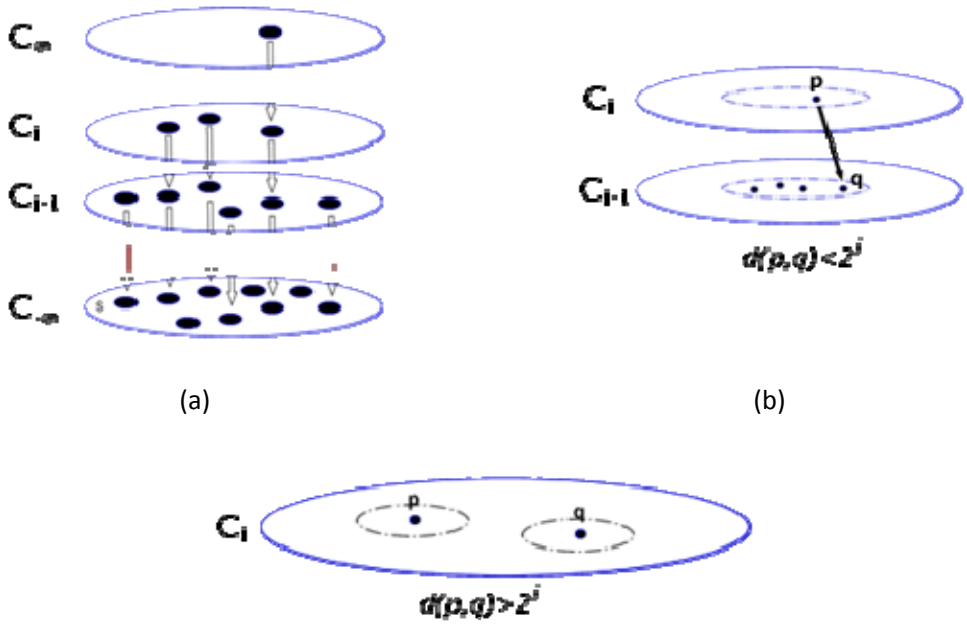



Figura 4 Propiedades del árbol de cubrimiento. (a) Anidación. (b) Cubrimiento. (c) Separación.

Es conceptualmente fácil describir el algoritmo en términos de una representación implícita del árbol de cubrimiento considerando un infinito número de niveles, con conteniendo los puntos en S asociados con el nodo raíz y con , sin embargo en un análisis de la representación explicita el espacio requerido es un . Esto es posible debido a que si un punto aparece en el nivel entonces este está en todos los niveles por debajo de y por consiguiente es hijo de si mismo en todos estos niveles. La representación explicita del árbol combina todos los nodos en los cuales el único hijo es el mismo, esto implica que cualquier nodo implícitamente tenga un padre aparte de su propio padre o un hijo aparte de su propio hijo, lo cual nos da inmediatamente un independientemente de la constante de crecimiento y de la dimensión del espacio.

La constante de expansión es definida como un pequeño valor tal que por todo y , siendo una bola cerrada de radio alrededor de en ⊂ . Si es organizado uniformemente en alguna superficie de dimencion , entonces .

Si $c$ es la constante de expansión de $S$, según [34] tenemos que:

| ***Árbol de cubrimiento*** | |
|---|---|
| Espacio utilizado | $O(n)$ |
| Tiempo de construcción | $O(c^6 n \ln n)$ |
| Inserción o eliminación | $O(c^6 \ln n)$ |
| Consulta | $O(c^{12} \ln n)$ |

Tabla 1 Costo computacional de la estructura de datos árbol de cubrimiento

### 1.5.3 Árbol de k-dimensiones.

El árbol kd[28] es una estructura de datos de particionado del espacio que organiza los puntos de un espacio euclidiano de k dimensiones. Es un caso especial de los árboles BSP. Aunque existen muchas aplicaciones para esta estructura su propósito es siempre descomponer jerárquicamente el espacio en un número relativamente pequeño de celdas tal que estas no contengan muchos objetos de entrada. Entre las ventajas claramente apreciables es que provee una forma rápida de acceso por posición hasta cualquier objeto buscado. [35]

En [36-38] se presenta el árbol kd como una de las estructuras de particionado del espacio más usadas en la búsqueda del vecino más cercano. Según [35, 39], el árbol kd para dimensiones bajas es extremadamente versátil pero en la medida en que aumenta la dimensión del espacio no es tan bueno como otras estructuras ya que en la búsqueda el algoritmo visita prácticamente todos los nodos del árbol; se dice que baja el rendimiento con el aumento de la dimensión.

La idea principal del algoritmo para la construcción del árbol kd a partir de un conjunto de prototipos *P* es la siguiente: encontrar un hiperplano que divida el conjunto *P* en dos subconjuntos y proceder recursivamente con los subconjuntos. El principal aspecto a resolver es la elección del hiperplano y de la coordenada que va a servir para dirigir la búsqueda a un lado u otro del hiperplano, la *coordenada discriminante*. (Véase Figura 5 para un ejemplo en 2D)

Para intentar conseguir que cualquier prototipo tenga la misma probabilidad de estar a un lado o a otro del hiperplano y por tanto que el árbol resulte lo más equilibrado posible, se suele

[28] Del ingles kd-tree, (k dimensional tree)

elegir el hiperplano de forma que se situé en la mediana de los valores de la *coordenada discriminante*. Además la *coordenada discriminante* debe ser aquella que tenga una mayor amplitud, es decir, aquella para que la diferencia entre la coordenada mínima y máxima sea la mayor en valor absoluto, aunque esto no es totalmente necesario, al alternar la coordenada discriminante por niveles de profundidad se logra el equilibrio deseado.

El árbol se construye de la siguiente forma: en cada nodo, que representa un conjunto de prototipos (el nodo raíz representa todo el conjunto), se elige la *coordenada discriminante* y se obtiene la mediana de los valores de dicha coordenada para ese conjunto; a continuación, se divide dicho conjunto en dos subconjuntos utilizando la mediana, situando en cada uno de ellos los valores a un lado y a otro de la mediana respectivamente. A continuación se crean recursivamente los arboles asociados a cada uno de los subconjuntos. El proceso termina cuando el tamaño de subconjuntos es menor o igual que un tamaño fijado con anterioridad o cuando la cantidad de niveles es $O(\log n)$.

Comparándolo con el árbol octal, este nunca es considerado un *árbol kd*, ya que estos dividen en una dimensión mientras que las estructuras octales dividen alrededor de un punto. Los árboles kd además son siempre binarios, lo cual no se cumple para los árboles octales.

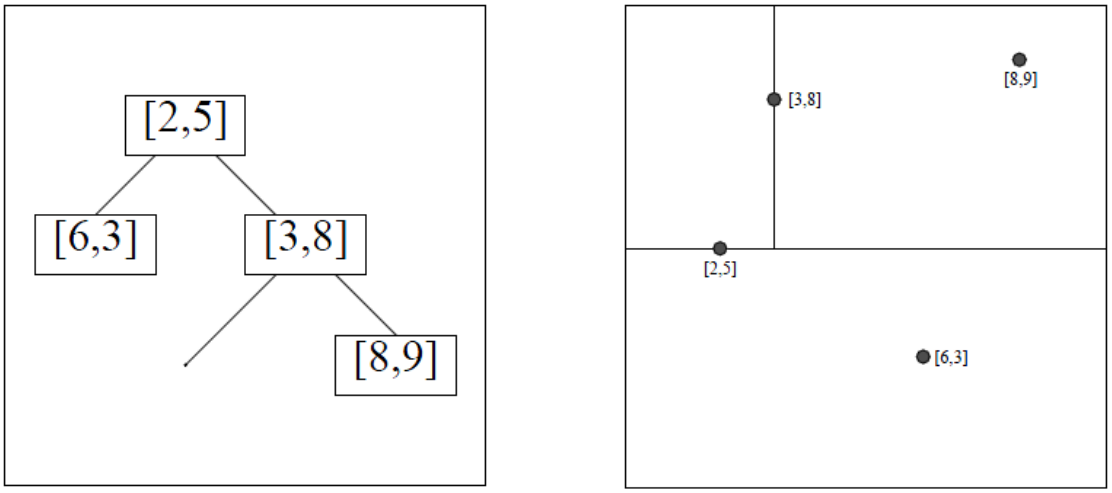



Figura 5 2d-tree de cuatro elementos, el nodo [2,5] se dividió por el plano y = 5 y el nodo [3,8] por el plano x = 3.

## 1.6 Algoritmos de generación de mallas y nubes de puntos.

Un problema físico se simula mediante un modelo matemático, generalmente se trata de un sistema de ecuaciones diferenciales que describen la variación de las incógnitas entre distintos

puntos del espacio y en el tiempo. En el *FEM*, las soluciones se calculan en forma aproximada, reemplazando el espacio continuo por un conjunto discreto de polígonos en el caso 2D o tetraedros o hexaedros en 3D; el tiempo también se discretiza en intervalos finitos y se calcula la solución en cada paso. En el interior de un polígono o poliedro elemental, las funciones se aproximan mediante un valor interpolado a partir de los que se calcularon en los vértices. Para este método la malla constituye el conjunto de elementos finitos. La forma y el tamaño de los elementos se relacionan con el error admisible de la solución aproximada. [40]

En muchas otras aplicaciones se han desarrollado métodos que no requieren de una malla. En estos la discretización del espacio consiste en un conjunto discontinuo y finito de puntos o nodos y las soluciones, en un punto cualquiera del espacio, se interpolan de los valores nodales según distintas técnicas. Siempre se requiere que los nodos más cercanos tengan mayor influencia sobre el valor en el punto. Los métodos sin mallas tienen algunos inconvenientes, son computacionalmente costosos y las condiciones de frontera son difíciles de establecer. Irónicamente una malla es a menudo usada en uno de los pasos del método, por ejemplo, para realizar la integración numérica de funciones básicas. [41]

Muchos de los cuerpos a partir de los cuales se realizan dichas simulaciones tienen complicadas formas. Esta característica, entre otras, hace que sea un problema bastante tratado y a la vez complejo, en parte, porque es difícil formar cuerpos “feos” sin sacrificar la calidad de los objetos que la componen. Los tetraedros son los elementos más usados en las mallas de volumen en la ingeniería.

Los algoritmos de generación se basan fundamentalmente en divisiones espaciales. Este proceso envuelve la descomposición del dominio en pequeños elementos, usualmente tetraedros o hexaedros. En [42] se propone un algoritmo utilizando el octree. En [43, 44] se propone otro basado en una estructura en forma de *celosía cúbica centrada*[29] mediante la cual se obtienen buenos resultados en cuanto a la calidad de los tetraedros generados.

Los primeros algoritmos de generación que garantizan limites en los ángulos de los triángulos en 2D y que utilizan una celosía como estructura base son los propuesto en [45], en [46] basado en el algoritmo de *Delaunay* y en [47], que incluye la idea de movimientos de los vértices de los nodos de los tetraedros. Seguidamente aparecieron estos en 3D pero se hizo difícil garantizar que los ángulos diedros no estuvieran muy cerca de 0 o 180 grados.

[29] BCC Lattice, body centered cubic lattice

Hay varios algoritmos, basados en octree, que garantizan teóricamente pequeños ángulos diedros en los tetraedros [48-52]. Desafortunadamente estos algoritmos comparten la característica que los ángulos diedros son muy pequeños, menores que 0.1 grado. Los algoritmos basados en triangulaciones de *Delaunay* ofrecen mejores resultados en la práctica pero la calidad de sus tetraedros no está totalmente garantizada (ángulos diedros mayores que 1 grado). El único algoritmo que constituye una acepción es el propuesto en [53] y basado en una *celosía cúbica centrada* que garantiza ángulos diedros entre 30 y 135 grados pero tiene la desventaja que no es robusto.

*Francois Labelle* y *Jonathan Richard Shewchuk*, en 2007, proponen un algoritmo que garantiza buenos ángulos diedros. Es numéricamente robusto y fácil de implementar debido a que utiliza, al igual que *Marching Cubes*, un conjunto de plantillas pre-computadas para la generación de los tetraedros, característica esta que lo hace más rápido que las generaciones basadas en *Delaunay* y *Frente de Avance*. [41, 54]

***Calidad de los elementos de la malla.***

Hay muchas medidas "geométricas" de la calidad de los elementos, muchas se basan en longitudes de aristas, radios de esfera circunscripta o inscripta o en invariantes del *Jacobiano*. En [55] se presentan 17 tipos de parámetros para medir la calidad de un tetraedro. En [56] se hace una recopilación de los principales indicadores de calidad utilizados en la generación de mallas de tetraedros, ellos consideran medidas sobre los elementos básicos de los tetraedros tales como lados, caras y ángulos interiores.

En el *FEM* una buena medida de la calidad de las simulaciones físicas en 3D relacionada con la calidad de la malla y sus elementos es la amplitud de los ángulos diedros, en particular no deben existir ángulos diedros muy cercanos a 0 o 180 grados; ángulos diedros cercanos a 180 grados causan errores de interpolación [41, 57, 58] y los ángulos diedros cercanos a ambos extremos hacen que las matrices de rigidez, asociadas al método, estén mal condicionadas [41, 59, 60]. Otro error común en muchas aplicaciones, incluyendo el *FEM*, es el relacionado con el gradiente, ángulos diedros cercanos a 0 no causan este error mientras que cercanos a 180 si lo hacen.

Este parámetro de calidad asociado a las mallas de tetraedros es considerado en la mayoría de los algoritmos de generación. [47-53, 61, 62]

Un ángulo diedro es aquel delimitado por dos semiplanos que parten de una arista común. Es un concepto geométrico ideal y sólo es posible representarlo parcialmente como dos paralelogramos con un lado común, que simbolizan dos semiplanos.

El valor de un ángulo diedro es la amplitud del menor ángulo posible que conforman dos semirrectas pertenecientes cada una a un semiplano. Un tetraedro tiene 6 ángulos diedros, uno por cada eje. El ángulo diedro de uno de sus ejes es el ángulo entre las dos caras triangulares que inciden en él. (Véase Figura 6)

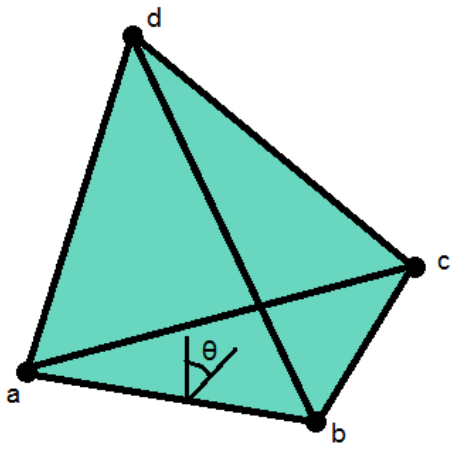



Figura 6 Ángulo diedro formado entre dos cara de un tetraedro.

Suponiendo que *a*, *b*, *c* y *d* sean los puntos que forman un tetraedro y sea , v y w entonces las normales asociadas a las caras triangulares y están dadas por y respectivamente. El ángulo diedro θ del eje *ab* es el ángulo entre estas dos normales y puede ser calculado usando la fórmula del producto escalar y el valor de la función coseno. Ver [41] para más detalle.

## 1.7 Conclusiones parciales.

Las nuevas formulaciones de métodos numéricos han traído consigo nuevos resultados a problemas de ingeniería y a otras actividades humanas que pueden mejorar las condiciones de vida a partir de estudios más fidedignos y precisos.

La visualización científica se ha convertido en una práctica de mucho interés y aplicación. El desarrollo de herramientas de visualización no se ha quedado al margen, los ingenieros

y científicos cuentan con softwares de muchas capacidades pero sin embargo son de propósito general y en otros casos requieren del pago de licencias para su uso.

La gran cantidad de datos en los conglomerados de partículas constituye un problema para la visualización de propiedades y del sistema en sí. Es necesario el empleo de mecanismos eficaces para lidiar con esta problemática tales como técnicas para el manejo de datos en la memoria principal y para el cálculo de visibilidad. Esta es considerada una escena de alta complejidad y filtrar parte de la información mejora la cantidad de imágenes por segundo.

Las estructuras de búsqueda espacial árbol de cubrimiento y árbol de k-dimensiones son de las más eficientes para las búsquedas de rango y la búsqueda del vecino más cercano en 3D, respectivamente.

# Capítulo II: Algoritmos de pre y post-proceso para la generación de nubes de puntos y la visualización de datos de métodos de partículas, libres de mallas y de puntos

La visualización científica de datos provenientes de la aplicación de métodos numéricos es hoy un tema con mucha atención para investigadores y desarrolladores. La gran cantidad de datos a procesar constituye un reto tanto a nivel de hardware como a nivel de software. Librerías para manejo masivo de información, técnicas para el cálculo de visibilidad en escenas, entre otros, sirven como apoyo a la visualización.

## 2.1 Visualización de conglomerados de partículas.

El *DEM* en una primera fase necesita la generación de una gran cantidad de partículas. Manejar tantos datos, a pesar de los adelantos en el hardware con que se cuenta en la actualidad, continua siendo un problema. Cargar en memoria principal toda esa información puede no ser factible ya que en la mayoría de los casos es insuficiente. Es factible además el uso de técnicas para el cálculo de la visibilidad ya que eliminan partes de los datos que no influyen en la imagen final y no tienen que ser procesados por las técnicas de visualización científicas posibles a aplicar.

### 2.1.1 Uso de la librería estándar de plantillas para memoria externa.

Muchos de los algoritmos utilizados en el pre-proceso de la información proveniente de los métodos de partículas necesitan cargar todos los datos en memoria principal[30]. La gran cantidad de información a manejar contrasta con la capacidad de almacenamiento de esta memoria. Una solución a este problema consiste en mantener parte de la información en disco, moviéndola a la memoria principal cuando sea necesario. El uso de algoritmos capaces de manejar desde el mismo software las especificidades de cada conjunto de datos es más eficiente que los implementados por el sistema operativo los cuales son de propósito general.

La *STXXL*[31] es una librería desarrollada por el *Instituto Max-Planck*[32]. Es una implementación de la librería estándar de plantillas para memoria externa[33], implementa contenedores y

[30] También conocida como memoria física o interna.
[31] Standard template library STL

algoritmos para el procesamiento de grandes volúmenes de datos. La cercanía con la *STL*, forma parte del estándar de C++, hacen que el uso y la compatibilidad con aplicaciones existentes sean casi transparentes. [63]

Entre las ventajas que conlleva el uso de la *STXXL* podemos encontrar: [63]

1. Provee soporte transparente para el uso de discos paralelos, es la única librería de manejo de memoria externa que implementa este tipo de algoritmos.
2. Es capaz de manejar problemas de una gran cantidad de datos (probada para docenas de terabytes).
3. Mejora de la utilización de los recursos de la computadora.
4. El tiempo de desarrollo utilizando la librería es mínimo debido a su compatibilidad con la *STL*. Los algoritmos de la *STL* pueden ser aplicados directamente. La complejidad de los algoritmos de entrada/salida permanece constante.

El uso de la librería es bastante sencillo y conlleva pocos cambios en el código.

### 2.1.2 Aceleración de la visualización.

Los empaquetamientos de partículas generados en la primera fase del *MED* presentan una serie de características y/o particularidades que pueden ser tenidas en cuenta a la hora de hacer una representación visual. A continuación se presentan las más significativas para el cálculo de la visibilidad.

1. Están formados por grandes cantidades de pequeñas partículas (muy pequeñas en relación al objeto que pertenece la generación) y que, generalmente, no difieren significativamente en tamaño.
2. Generalmente, los empaquetamientos son aglomeraciones bastante densas de partículas. La densidad puede alcanzar valores entre 40 y 55 porciento.
3. Los empaquetamientos representan volúmenes (en el caso de tres dimensiones), esto hace que el empaquetamiento en su conjunto pueda considerarse como un cuerpo cerrado ya que se generan partículas por todo el medio de estudio.

[32] Max-Planck-Institute
[33] Out-of-core

Si se tiene en cuenta la *1ra característica*, la utilización de una técnica de oclusión sería sumamente costosa e ineficiente. No obstante, existen algunos casos donde se ha aplicado el *DEM* para realizar simulaciones utilizando partículas que difieren considerablemente en sus dimensiones, como en [64] y se pudiera analizar, en estos casos, la utilización de las partículas más grandes como ocluyentes potenciales.

Sin embargo, si se analiza la *2da propiedad*, se puede notar que un por ciento bastante alto de partículas no influye de ninguna manera en la imagen final. Dada la alta densidad del empaquetamiento solo serán visibles aquellas partículas que se encuentren más cercanas a la superficie del objeto.

Esta es una característica que puede ser utilizada a favor de reducir la cantidad de partículas a procesar en cada paso de la visualización. En la Figura 7 se muestra un ejemplo en dos dimensiones de las partículas en un empaquetamiento. A partir de este momento, este conjunto de partículas señalado en rojo va a ser llamado corteza del empaquetamiento.

La idea se basa en no considerar, para la generación de las imágenes, aquellas partículas que no pertenecen a la corteza. Sin embargo, estas partículas no pueden ser del todo desechadas ya que en caso de aplicar cortes volumétricos a los empaquetamientos sería necesario reconstruir parcialmente la corteza. La obtención de las partículas de la corteza se realiza en una etapa previa a la visualización pues en los empaquetamientos estáticos las partículas nunca cambian de posición y por tanto la corteza siempre es la misma. En otras situaciones, algunas etapas de post-proceso, si es necesario reencontrar/recalcular las partículas que pertenecen a la corteza.

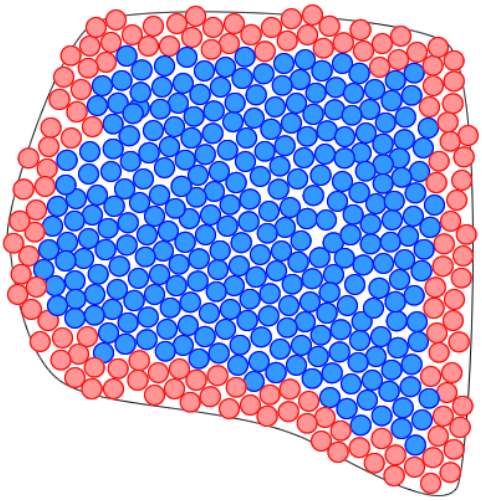


Figura 7 Representación figurada de un empaquetamiento donde las partículas interiores (color azul) son ocluidas por las partículas más cercanas a la superficie (color rojo)

Lo ideal sería tener una corteza constituida únicamente por aquellas partículas que influyen en la imagen final desde alguna posición de la cámara. Sin embargo, teniendo en cuenta la gran cantidad de partículas que forman los empaquetamientos sería sumamente costoso determinar de forma exacta la corteza del empaquetamiento.

La idea práctica se basa en obtener una aproximación de la corteza mediante un método tan costoso. En este caso se consideran solo aquellas partículas que se encuentren a la distancia de la superficie. La selección de esta distancia, depende de varios elementos: fracción de volumen local, radio mínimo y radio máximo. El éxito fundamental de este método radica en realizar una aproximación bastante buena de la corteza. Una distancia demasiado pequeña provocaría agujeros en la representación del empaquetamiento desde algunas posiciones de la cámara y por tanto se pierde la integridad de la imagen. Por otra parte, la selección de una distancia demasiado grande implica una corteza demasiado gruesa y se estarían teniendo en cuenta demasiadas partículas que no van a influir en la imagen final.

#### 2.1.2.1 Algoritmos para el cálculo de visibilidad en escenas de conglomerados de partículas con información del contorno.

Cuando se cuenta con información del contorno del conglomerado de partículas puede ser más fácil el cálculo de las partículas que serán visibles desde la posición de la cámara. Contar con la malla de superficie permite obtener la corteza del conglomerado de partículas y también aquellas partes de esta que serán visibles desde el punto de visión del observador de manera sencilla.

Considerando la *3ra propiedad*, pudieran no dibujarse aquellas partículas de la corteza que se encuentren próximas a las partes de la superficie que no están de frente a la cámara. Considerando la corteza del empaquetamiento como una forma rústica de superficie y asociando normales en dependencia de la parte de la superficie real a la que corresponda es posible obtener las partículas potencialmente visibles (véase Figura 8). A las partículas correspondientes a cada rectángulo (caso 2D) se les asocia la normal correspondiente a la región de la corteza a la que pertenecen. Nótese que una partícula puede pertenecer a más de una región.

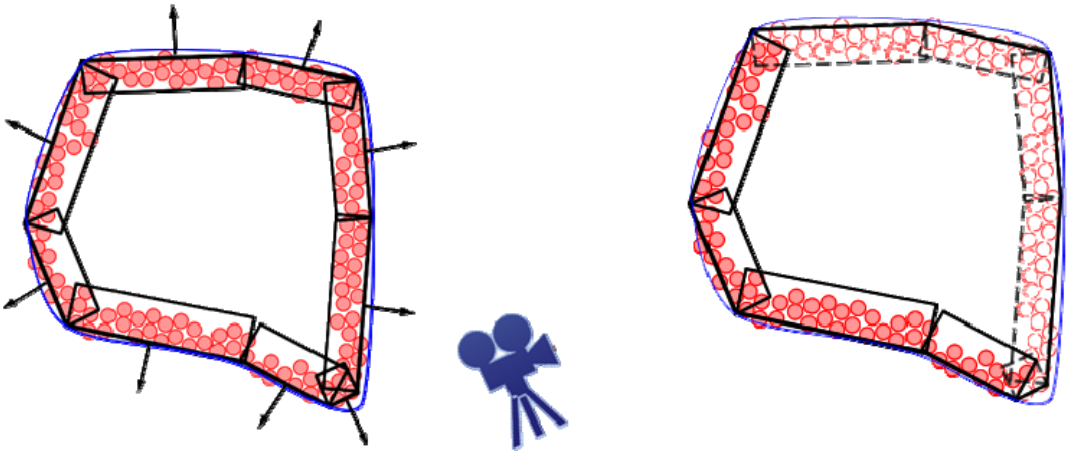


Figura 8 Orientación de las regiones en la corteza del empaquetamiento en 2D. (a) Asociación de normales a sectores de la corteza de acuerdo a la correspondencia de esta con la superficie. (b) Ejemplo de selección de las esferas visible a partir de la posición de la cámara.

**Algoritmo: Obtención de partículas potencialmente visibles en conglomerados de partículas.**

```
Entrada: dirección camara vector, contorno mesh, lista de partículas list

  cells cell = new cells(); //estructura para organizar las partículas
  Foreach partícula in list do
    cells_index index = is_in(particle); //índices (i,j,k) de la celda a la que pertenece la partícula
    cells.add(index,i); //i – índice de la partícula en la lista.
  end
  cluster clusters = new cluster(); //obtención de los clusters que estan en la corteza.
  Foreach face in mesh do
    figura fig = new figura(face,lenght); //figura regular recta con base en la cara de la malla.
    computar_valores_bordes(minI, minJ,minK, maxI, maxJ, maxK, fig);
    For u = minI to maxI do
      For v = minJ to maxJ do
        For w = minK to maxK do
          Foreach partícula in cells[u,v,w] do
            if fig.is_in(partícula) then //calculando interioridad de las partículas en la figura.
              cluster.add(partícula) ;
            endif
          end
        end
      end
    end
    clusters.add(cluster) ;
  end
  computar_cluster_visibles(clusters,vector);

  Return clusters.partículas ;
  //A la hora de dibujar solo se pintan los cluster que estan de frente a la posición de la camara.
  //la estructura de datos "clusters" es estática y solo se actualiza en caso de ser necesario.
```

Algoritmo 1. Cálculo de la visibilidad cuando se tiene información del contorno del conglomerado de partículas.

Una vez asociadas las normales a cada una de las regiones de corteza, se puede verificar si estas regiones se encuentran de frente o no a la cámara, y por tanto si sus partículas son visibles (véase Figura 8). La característica de que una partícula pueda pertenecer a más de una región de la corteza es deseable ya que su visibilidad va a estar determinada por la disyunción entre las visibilidades de las regiones a las que pertenece. Esto favorece que no surjan agujeros cuando no se dibuje una región que no se encuentre de frente y que sea adyacente a otra que si es visible (véase Algoritmo 1).

El cálculo de la normal asociada a cada una de las caras de la corteza se realiza utilizando la ecuación del plano. La ecuación del plano está dada por

$$\vec{n} \cdot (x - p_0) = 0 \qquad (4)$$

donde $\vec{n}$ es la normal al plano y $p_0$ un punto que pertenece a este. El plano se obtiene a partir de los tres puntos de cada una de las caras y de la ecuación del plano se obtiene la normal por despeje. Para encontrar la corteza se forman figuras geométricas con base en las caras del cuerpo que cubre el conglomerado de partículas, orientándolas hacia el interior y tomando una altura $d$, esto es, en el caso de mallas de superficies de triángulos utilizadas para definir la corteza, un prisma triangular. Se obtienen entonces las partículas interiores a la figura geométrica formada. Las partículas que se van a dibujar se obtienen comparando la normal de la corteza con el vector formado por la posición de la cámara y su dirección.

#### 2.1.2.2 Algoritmos para el cálculo de visibilidad en escenas de conglomerados de partículas sin información de contorno.

Los métodos libres de mallas y de partículas traen el inconveniente que no cuentan con información del contorno. La aplicación del algoritmo tratado anteriormente para eliminar todas las partículas que no son visibles desde cualquier punto de mira no es posible en este caso. No contar con una forma, aunque sea rústica, del contorno dificulta la obtención de una corteza suave ya que no es posible proyectar hacia el interior y escoger como visibles las partículas que quedan en el rango de proyección.

El algoritmo propuesto en este caso utiliza una división por celdas. No solo como contenedoras de la información de las partículas sino también en el cálculo de la visibilidad en la escena.

El algoritmo queda de la siguiente manera:

1. Construir una caja uniforme contenedora de partículas (*PUCB*)[34]. En cada celda se tiene información de la posición en el espacio de cada una de las partículas y a su vez información relacionada con la cantidad de partículas que contiene.
2. A partir del *PUCB* construir un mapa de numeración por distancia (celosía) donde cada celda guarda la distancia discreta de las esferas que contiene con el exterior, esto es, una celda que no contenga partículas tiene distancia 0 pues claramente está en el exterior del conglomerado; las celdas con partículas en su interior tienen una distancia con el exterior siguiendo los siguientes pasos:
    i. El mapa se construye por niveles, en cada nivel solo se asignan valores hasta dos veces mayor que el nivel estudiado, esto es, en el nivel 0 solo se asignan valores de distancia 1 y 2, y así sucesivamente; las demás

[34] Particles Uniform Contained Box

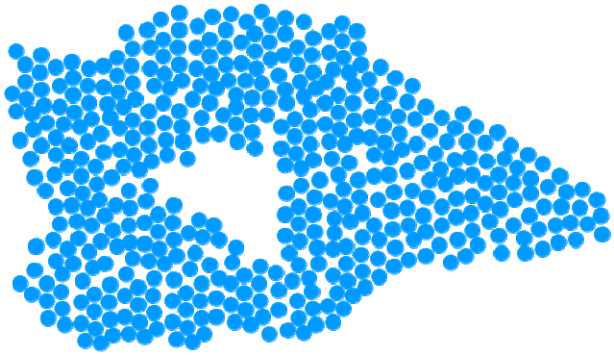
distancias son imposibles de determinar por falta de información. Una celda tiene valor n si el máximo de sus vecinos es de valor n-1.

ii. Inicialmente todas las celdas tienen valor 0.

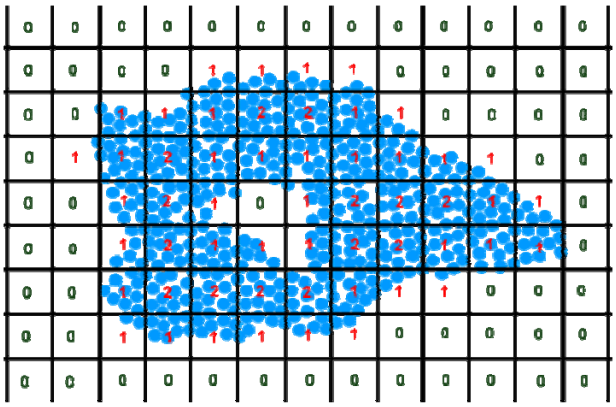
iii. Una vez asignadas todas las distancias ¨calculables¨ en un nivel se pasa al siguiente hasta que no queden celdas sin asignar.

3. Con el mapa de distancias completo es posible obtener las partículas visibles para una distancia , la cual determina hasta que nivel de profundidad son visibles.

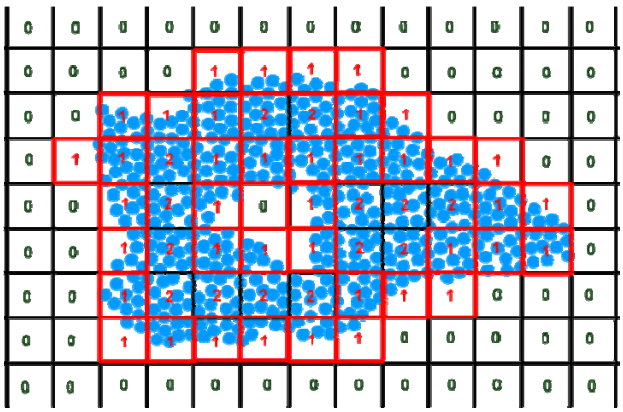
Veámoslo gráficamente en 2D.

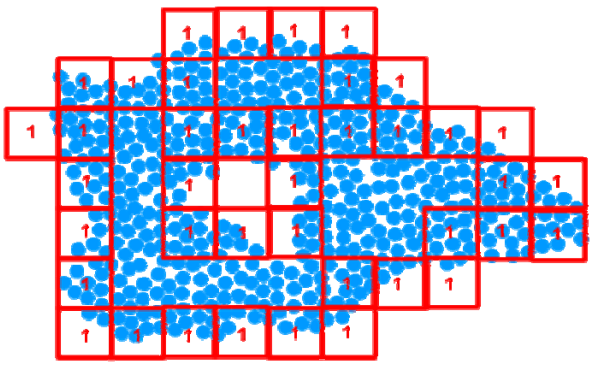


**Sistema o conjunto de partículas a partir del cual se calculará la visibilidad.**

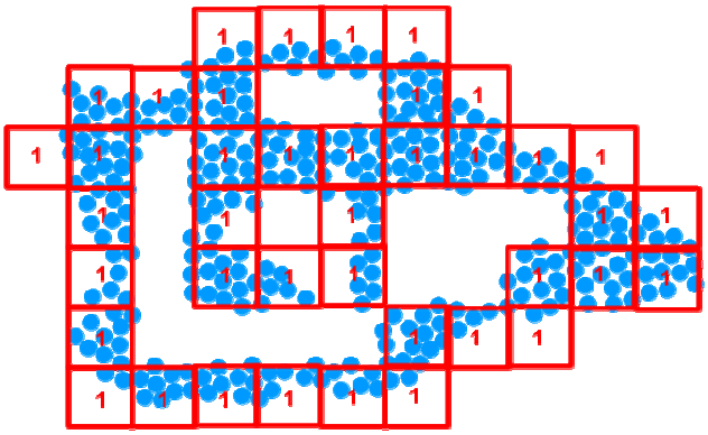


**División por celdas (*celosía*) y mapa de distancias "discreta" a partir del sistema de partículas.**





**Escogiendo las celdas de la *celosía* a partir del mapa de distancia y una profundidad dada. En este caso solo fueron tomadas las partículas con un nivel de profundidad o distancia "1".**



**Partículas que forman parte de la escena hasta un nivel de profundidad o distancia "1" y que son las visibles desde cualquier punto de visión.**

Figura 9 Algoritmo gráfico para la obtención de las partículas potencialmente visibles en un conglomerado.

El cálculo de la visibilidad dado un punto de visión específico también es posible asignando normales a las celdas y comparándolas con la dirección en que mira la cámara u observador. El valor de las normales en la estructura de datos puede ser calculado considerando la dirección de los niveles de profundidad hacia el interior del conglomerado de partículas. (Véase Figura 10)

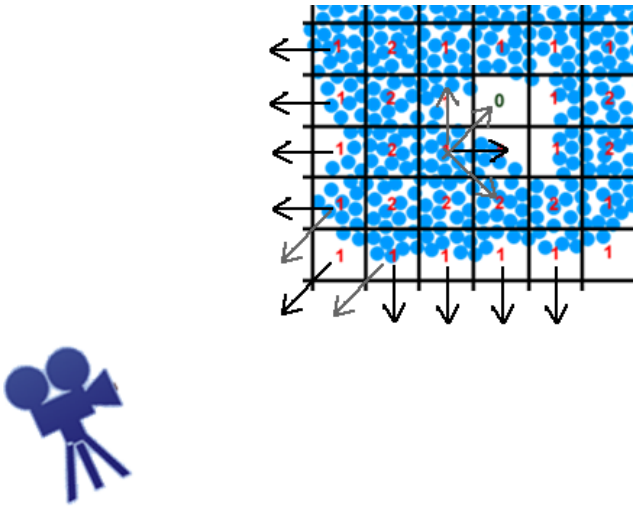


Figura 10 Cálculo de las normales asociadas a cada una de las celdas de la grilla considerando la distancia discreta como indicador de dirección.

Para una completa explicación véase Algoritmo 2.

**Algoritmo: Obtención de partículas potencialmente visibles en conglomerados de partículas sin contorno.**

```
Entrada: dirección camara vector, lista de partículas list, distancia d

   cells cell = new cells(); //estructura para organizar las partículas
   Foreach partícula in list do
      cells_index index = is_in(particle); //índices (i,j,k) de la celda a la que pertenece la partícula
      cells.add(index, i); //i – índice de la partícula en la lista.
   end
   mapa_distancia mapa = new mapa_distacias(cell);
   int profundidad = calcular_profundidad(cell.longitud, d);
   partículas_visibles = computar_visibilidad(cell, mapa, profundidad);

Return partículas_visibles;
```

Algoritmo 2 Cálculo de la visibilidad cuando no se tiene información del contorno del conglomerado de partículas.

### 2.1.3 Técnicas de visualización científicas para datos resultantes de la aplicación del método de partículas.

La visualización científica de datos, cualquiera sea la fuente, se basa fundamentalmente en una buena utilización de las herramientas con que se cuentan y con ello nos referimos a la amplia gama de técnicas de visualización que existen. Los métodos de partículas presentan determinadas peculiaridades que hacen que la aplicación directa de alguna de los algoritmos de visualización no sea factible o en otros casos posible por las características propias de los datos.

Lo más conveniente es hacer una adaptación de los algoritmos existentes y así poder ser aplicadas a conjuntos de partículas. De las técnicas estudiadas hasta el momento se escogieron el mapeo de colores, específicamente la *interpolación directa de colores*, y las *superficies de contorno*, ya que son estas las que más información pueden mostrar al investigador o ingeniero. Es posible, además, el uso de *cortes volumétricos* al conjunto de partículas pero este es mostrado recalculando la visibilidad para el subconjunto obtenido utilizando los algoritmos expuestos en los epígrafes 2.1.2.1 o 2.1.2.2.

En ambas técnicas, *interpolación directa de colores* y *superficies de contorno*, es posible dos acercamientos, uno discreto y otro continuo. En el discreto observamos las partículas tal y lo que son, aplicando alguna técnica de visualización científica, mientras que en el acercamiento

continuo es necesaria una interpolación de colores por todo el medio, no sobre las partículas. También es viable una combinación de ambos acercamientos.

#### 2.1.3.1 Técnica de visualización científica Interpolación de Colores para conglomerados de partículas.

El mapeo de colores es una técnica muy común y simple de usar. Se basa en la asignación de un color a cada valor escalar [2]. La asignación del color se realiza de acuerdo a una tabla de colores[35] o, de una forma más general, por medio de una función de transferencia. El punto más importante de esta técnica es la selección correcta de los colores de forma tal que se resalten las magnitudes de una manera en que se pueda percibir su significado. Un ejemplo es la representación de la temperatura que para valores fríos se representa con coloraciones azules y para cálidos con colores rojos por la asociación natural que hacen muchas personas de estos colores y la temperatura. Esta es una técnica muy básica y es empleada por otras que utilizan los colores para extraer información de los datos.

La tabla de colores contiene un arreglo de colores y dos valores asociados: mínimo (*min*) y máximo (*max*). El índice del arreglo de colores correspondiente a la magnitud escalar v se calcula de la siguiente forma [8]:

$$i = \begin{cases} 0, & v \leq min \\ n-1, & v \geq max \\ n\left(\dfrac{v-min}{max-min}\right), & eoc \end{cases}$$

donde $n$ es la cantidad de colores de la tabla. Mediante este método, y dependiendo de la cantidad de colores que formen la tabla, es posible obtener representaciones que muestran cambios bruscos de coloración. Esto se debe precisamente a que se está realizando una discretización de valores, que no son necesariamente discretos.

Una forma más general y que evita este problema es mediante una función de transferencia (véase Figura 11). En este método existe una función para cada una de las componentes del color, digamos rojo, verde y azul según la codificación *RGB*. Se evalúa el valor para el que se desea obtener el color en cada una de estas funciones y se obtiene el valor de cada una de las componentes. Como puede apreciarse, una tabla de colores no es más que una discretización de una función de transferencia.

[35] Color lookup table

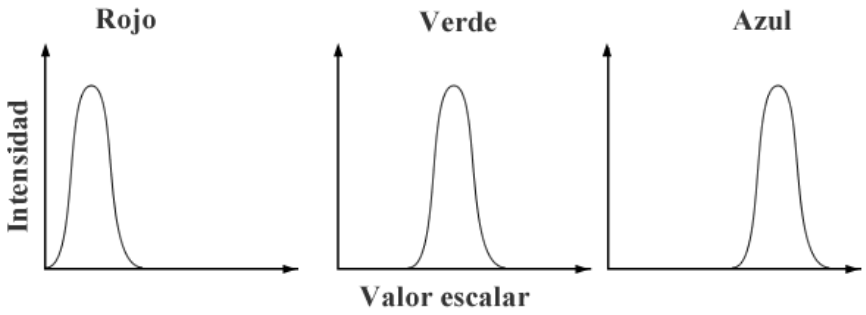



Figura 11 Función de transferencia de colores.

Para la obtención de una función de transferencia no es necesaria conocerla de forma explícita. En muchas ocasiones y dependiendo de la característica que representan los datos se conoce colores sólo para valores específicos. Por ejemplo para una temperatura de 40°C sería rojo, -30°C sería azul y 0°C sería azul claro. Partiendo de esto y utilizando la interpolación de colores se puede obtener una función de transferencia como la que se muestra en la Figura 12. Este es el mecanismo empleado para la obtención de colores según los valores de la función.

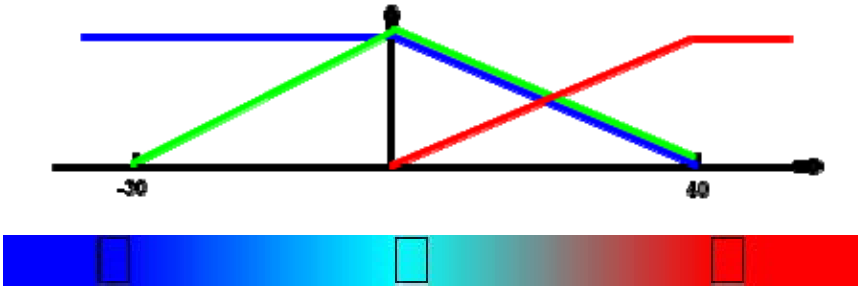



Figura 12 Obtención de la función de transferencia utilizando la interpolación lineal de colores.

Esta técnica es empleada por muchas otras que utilizan los tonos de colores para visualizar resultados.

La interpolación de colores se basa en la asignación de estos a los vértices de un polígono, que pudiera ser una cara de una malla, y posteriormente interpolar estos colores a través de toda la figura. Como resultado se obtiene una representación de los valores de los datos a través de una variación continua de tonos [3].

Existen dos posibles acercamientos para la interpolación de colores aplicada a conjuntos de partículas. Se pueden considerar las partículas como un espacio discreto, esto es, representarla como tal, considerando sus propiedades físicas y geométricas. En este caso se

visualizan, cada una, asignándoles su correspondiente color dependiendo de la función de transferencia utilizada. Otra forma de ver los empaquetamientos es considerándolos como un espacio continuo, esto llevado a la visualización, consiste en hacer una interpolación directa de colores según los valores de las propiedades de las partículas cercanas a la superficie. Es conveniente el uso de alguna estructura de datos para el cálculo de vecindad.

#### 2.1.3.2 Técnica de visualización científica superficies de contorno para conglomerados de partículas.

Otra extensión de la técnica mapeo de colores son las *líneas de contorno*[36] y las *superficies de contorno*[37]. Estas son utilizadas para representar regiones con determinados valores constantes en la superficie y el interior de un objeto respectivamente. Representan la frontera entre las regiones $F(x) < c$ y $F(x) > c$, o sea, donde $F(x) = c$, siendo $c$ un valor de contorno y $x$ un punto del espacio n-dimensional al que pertenece el conjunto de datos. Las *líneas de contorno* son aplicadas a espacios bidimensionales y las *superficies de contorno* a espacios tridimensionales [2].

Específicamente la *superficie de contorno* queda definida como sigue: Dado un campo escalar $F(P)$, $F$ una función escalar en $R^3$, la superficie que satisface $F(P) = \propto$, donde $\propto$ es una constante, es llamada iso-superficie o superficie de contorno definida por $\propto$. El valor $\propto$ es el llamado iso-value o valor de contorno [58]. Por las características propias de los datos solo es posible la utilización de la técnica *superficie de contorno*.

La idea empleada para la obtención de una *superficie de contorno* es similar a la utilizada para la obtención de las *líneas de contorno*, pero con la complejidad adicional que trae consigo el trabajo en tres dimensiones. En este caso se procede a la discretización del espacio mediante pequeñas figuras geométricas tridimensionales. El tipo de figura utilizada con mayor frecuencia y que consta de una aceptación amplia, entre otras cuestiones por las facilidades que brinda para la descomposición espacial, es el cubo; la técnica se conoce como *Marching Cubes* y no es más que la extensión a tres dimensiones de *Marching Squares*. El resultado de este algoritmo es una malla triangular que aproxima la *superficie de contorno*.

Dada la función de corte $F$, se calculan todos los valores en los vértices del cubo. *Marching cubes* usa la técnica divide y vencerás para localizar la superficie sobre los cubos analizados.

[36] Iso-lines
[37] Iso-surface

El resultado de este análisis es un conjunto de triángulos que aproximan la intercepción de la iso-superficie con ese cubo. Los triángulos de salida, que a la postre forman la malla de superficie, son generados a partir de plantillas pre-computadas. Los vértices de los triángulos dependen solamente de los valores de la función $F$ en los vértices del cubo y escoger cada una de las plantillas que los generan depende solamente de los signos en estos [58, 65]. La cantidad de combinaciones posibles para este algoritmo es de $2^8$, esto es, 256 casos, pero por simetría y rotaciones pueden ser reducidos a solo 15 [2, 66, 67]. (Véase Figura 13)

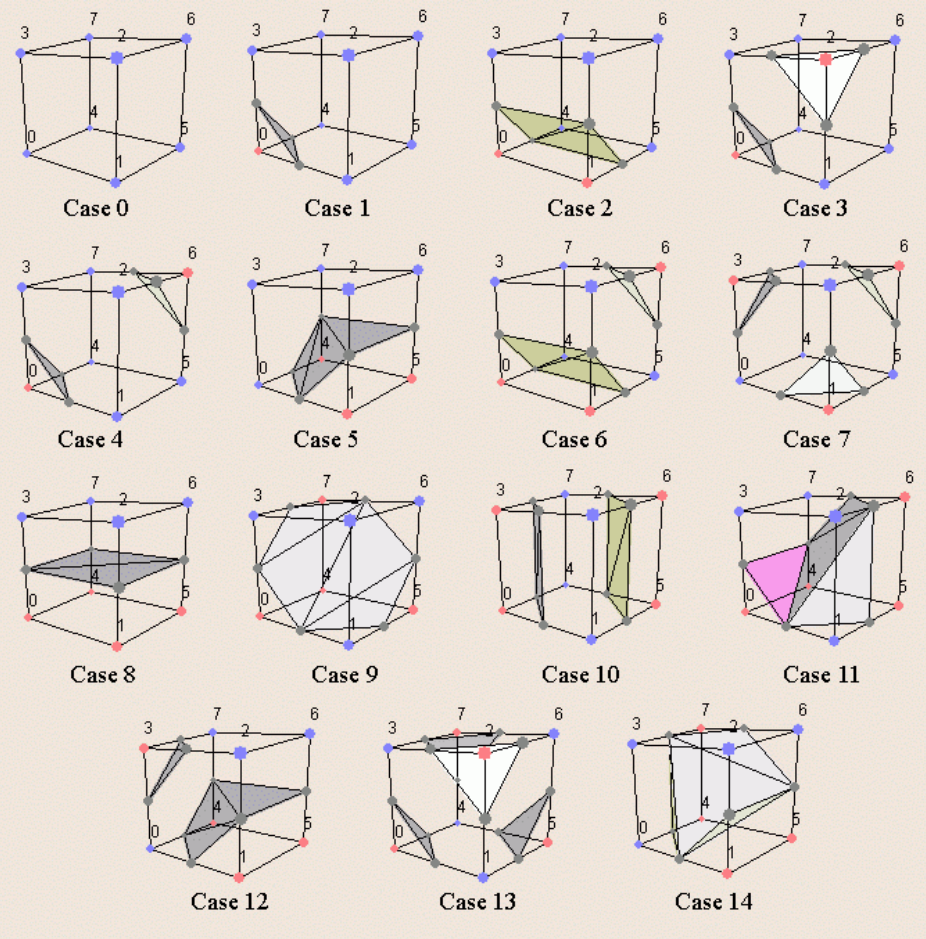



Figura 13 Plantillas utilizadas por el algoritmo *Marching Cubes*

Un aspecto importante a señalar es la presencia de ambigüedades en la generación del fragmento de malla para cada cubo. Si bien en las líneas de contorno este detalle carece de mucha importancia, para la generación de superficies de contorno constituye un verdadero problema que ha sido ampliamente analizado (véase [66, 67]). Los casos que se consideran ambiguos son toda la familia (dígase los equivalentes que pueden obtenerse mediante

rotaciones y complemento) de 3, 6, 7, 10, 12 y 13 en la Figura 13, en los cuales pueden presentarse situaciones en que la superficie dentro de un cubo se pueda generar de más de una forma, trayendo como resultado la aparición de hoyos en la malla resultante.

Una muy popular estructura de datos usada como guía para el mallado es el octree [57]. La técnica *Marching Cubes* y el *octree* son de gran importancia para la obtención de iso-superficies para conjuntos de partículas. Para información del *octree* véase epígrafe 1.5.1.

Como en la anterior técnica aplicada a este tipo de datos se consideran los mismos acercamientos, uno discreto y otro continuo.

En el acercamiento continuo la técnica no varía mucho en cuanto a forma de obtención con respecto a la original. Se utiliza el *octree* y *Marching Cubes*. La interioridad de los nodos del árbol es obtenida a partir de la propiedad de las partículas en ese nodo. La coloración de la iso-superficie está dada a partir de las partículas cercanas a los vértices de la malla. Para la vecindad de las partículas con la iso-superficie es utilizado el mismo *octree* por conveniencia ya que en un paso anterior fueron organizadas dentro de esta estructura. La complejidad algorítmica de la búsqueda en el *octree* es $O(\log n)$. Véase Algoritmo 3.

**Algoritmo Obtención de superficies de contorno en conglomerados de partículas (acercamiento continuo)**

**Entrada**: **valor de contorno** ***iso-value***, **lista de particulas** ***l***

```
octree octree = new octree(l, iso − value);
iso_superficie surface = marching_cubes(l, value);
funcion_color fcolor = new funcion_color(lista_colores, lista_valores);
interpolar_colores(surface, fcolor);
```

Algoritmo 3 Cálculo de una iso-superficie en un conglomerado de partículas considerando estas como un conjunto continuo de datos.

El acercamiento discreto nos lleva a visualizar las partículas que les correspondan exactamente el valor de contorno con el color correspondiente según una función de transferencia. Esto es casi imposible considerando que las propiedades de las partículas son continuas lo que va a acarrear que no se encuentren valores coincidentes entre el valor de contorno y la propiedad estudiada. Una manera de resolver esta dificultad es generar la iso-superficie y a partir de esta se visualizan las partículas que la interceptan con su color correspondiente según su valor.

(Véase Algoritmo 4). Otra forma es utilizar el octree como estructura de datos guía para encontrar las partículas que se encuentren alrededor del valor de contorno. Las dos formas son validas, aunque la segunda mucho más eficiente.

**Algoritmo: Obtención de superficies de contorno en conglomerados de partículas (acercamiento discreto)**

```
Entrada: valor de contorno iso-value, lista de particulas l

   octree octree = new octree(l, iso − value);
   iso_superficie surface = marching_cubes(l, value);
   lista particles_list = intercepción_iso_superficie_partículass(lista, value);
   funcion_color fcolor = new funcion_color(lista_colores, lista_valores);
   interpolar_colores(particles_list, fcolor);
```

Algoritmo 4 Cálculo de una *iso-superficie* en un conglomerado de partículas considerando estas como un conjunto discreto de datos.

## 2.2 Visualización de datos esparcidos.

El método de *Shepard* [19], como se mencionó anteriormente, realiza la interpolación utilizando la inversa de las distancias entre los puntos. Originalmente, este es un método de alcance global, ya que utiliza todos los puntos comprendidos en los datos, sin embargo, existe una modificación que considera únicamente los puntos que se encuentren, como máximo, a una distancia $L$ determinada del punto donde se desea interpolar, esto es, $L \leq r_L$ donde $\phi(r > r_L)$ es cero o despreciable. A continuación se dan detalles de los mismos basado en lo expuesto en [5, 19].

### 2.2.1 Interpolación usando el método de *Shepard* de alcance local.

En el caso de la interpolación mediante el método de *Shepard* la función interpolante tiene la forma:

$$F(x) = \sum_{i=1}^{n} w_i(x) \cdot f(x_i)$$

donde $w_i(x)$ es la función que determina el peso o grado de influencia que tiene $x_i$ sobre $x$. Esta función se obtiene como:

$$w_i = \frac{\psi_i(x)}{\sum_{j=1}^{n} \psi_j(x)}$$

con

$$\psi_i(x) = \frac{1}{{d_i}^2(x)}\,, \qquad d_i = \|x - x_i\|$$

La modificación necesaria para convertir esta función, en una de alcance local es sencilla. Para ello basta con modificar la función de peso, haciendo 0 el peso de aquellos puntos que se encuentran más allá de una distancia $r_w$, esto es,

$$\psi_i(x) = \frac{1}{{d_i}^2(x)} \left(1 - \frac{{d_i}^2(x)}{r_w}\right)^2.$$

Modificación que convierte el problema en la búsqueda de los vecinos que se encuentran dentro de un radio $r_w$ del punto de interpolación.

Para determinar los vecinos que se encuentran a una distancia máxima $r_w$ del punto donde se desea interpolar se utilizó el árbol de cubrimiento.

### 2.2.2 El árbol de cubrimiento y la búsqueda de vecindad en un radio.

Dada la conveniente notación para la búsqueda de vecindades en un radio la utilización del árbol de cubrimiento es muy útil. El algoritmo de creación del árbol es sencillo en lo que se refiere a una representación explícita ya que se tendrían infinitos niveles donde $\mathrm{C}_\infty$ contiene el punto de $\mathrm{S}$ asociado a la raíz del árbol y $\mathrm{C}_{-\infty}$ contiene todos los puntos de $\mathrm{S}$.

Para insertar un nuevo nodo se recorre el árbol a partir de la raíz hasta encontrar la posición que cumpla con las propiedades del árbol de cubrimiento (anidación, cubrimiento y separación). La complejidad de este procedimiento es $\mathrm{O}(\log \mathrm{n})$ [13] (Algoritmo 5).

El algoritmo de búsqueda del vecino más cercano utilizando el árbol de cubrimiento es también sencillo. La búsqueda comienza en la raíz del árbol y utilizando el principio de ramas y cotas se van obteniendo los posibles nodos a elegir. Un nodo es elegible si cumple que $d(p,q) \leq d(p,Q) + 2^i$ siendo $p$ el punto base de la consulta, $q$ un posible vecino o nodo a elegir y $Q$ el conjunto de todos los posibles vecinos. Se define $d(p,Q)$ como la menor de las distancias de $p$ a todos los puntos de $\mathrm{Q}$. (Algoritmo 6).

**Algoritmo: Inserción de un punto en el árbol de cubrimiento**

```
Entrada: conjunto de nodos Q_i, point p, nivel i

  Q = {hijos(q) : q ∈ Q_i};
  if d(p,Q) > 2^i then
    return true;
  else
    Q_{i-1} = {q ∈ Q_i : d(p,q) ≤ 2^i};
    found = insert(Q_i, p, i − 1);
    if found and d(p,q) < 2^i then
        escoger q ∈ Q_i : d(p,q) ≤ 2^i;
        insertar p como hijo de q;
        return false;
    else
        return found;
    endif
  endif
```

Algoritmo 5 Mecanismo utilizado en la creación de un árbol de cubrimiento

Una de las principales ramificaciones del problema del vecino más cercano es la búsqueda de los vecinos dentro de un radio $r$. El árbol de cubrimiento presenta una conveniente notación a la hora de enfrentar dicho problema. Con modificar ligeramente el Algoritmo 6 pueden ser determinados todos los puntos que se encuentran a una distancia máxima $r$ de otro punto cualquiera. Esta propiedad lo hace muy útil y permite su utilización en la interpolación mediante el uso de funciones de base radial de alcance local.

**Algoritmo: Búsqueda de vecindad usando el árbol de cubrimiento**

```
Entrada: árbol de cubrimiento covertree, point p

  Q_∞ = C_∞; // C_∞ es la raiz del covertree
  For i = ∞ downto −∞ do
    Q = hijos(q) : q ∈ Q_i;
    Q_{i-1} = {q ∈ Q_i : d(p,q) ≤ d(p,Q) + 2^i};
  end

Return argmin_{q∈Q_{-∞}} d(p,q);
```

Algoritmo 6 Búsqueda de vecindad con el *cover tree*

La creación del árbol se hace insertando uno a uno cada uno de los puntos, es costosa computacionalmente ( $O(n \log n)$ ), pero solo es necesario construirlo una vez. (Algoritmo 7)

**Algoritmo: Creación del árbol de cubrimiento**

```
Entrada: lista de puntos lista

  covertree ctree = new covertree(); //arbol vacio
  Foreach p in lista do
    ctree.add(p, ∞);
  end

Return ctree;
```

Algoritmo 7 Creación del árbol de cubrimiento

### 2.2.3 Algoritmo empleado para la interpolación de datos esparcidos usando Funciones de Base Radial y el Método de Shepard.

Ya descritos los elementos para la implementación de este método, a continuación se describen los pasos necesarios para su aplicación:

1. Creación del árbol de cubrimiento. La creación del árbol de cubrimiento se realiza mediante la sucesiva inserción de cada uno de los puntos pertenecientes a los datos en un árbol inicialmente vacío. La construcción del árbol se realiza al inicio del proceso y no es necesario repetirla ya que independientemente del punto en que se desee interpolar el árbol de cubrimiento es el mismo. (Algoritmo 7).
2. Obtención de los puntos localizados en el radio de influencia. Una vez creado el árbol de cubrimiento, se utilizan las facilidades que este brinda para obtener los vecinos que se encuentran a una distancia máxima determinada de un punto. (Algoritmo 8).
3. Evaluación con la función de interpolación. Se utilizan los puntos encontrados para evaluar la función de interpolación seleccionada y obtener un valor aproximado de la magnitud que se estudia en el nuevo punto. (Algoritmo 8).

**Algoritmo: Obtención del valor de un nuevo punto usando funciones de base radial**

```
Entrada: árbol de cubrimiento covertree, point p, radio r

  list vecinos = buscar_vecindad_radial(p, covertree, r);
  value = funcion_interpoladora(p, vecinos);

Return: value;
```

Algoritmo 8 Interpolación usando funciones de base radial

## 2.3 Visualización de datos discretos.

La aparición de nuevas formulaciones a métodos numéricos permite un mejor análisis de fenómenos y estructuras. Con ello se logran simular y detectar micro-fisuras y discontinuidades. A continuación se propone una técnica de visualización científica que viene a completar visualmente la modelación y simulación de fenómenos de este tipo.

### 2.3.1 Técnica de visualización científica para datos discretos o continuos a intervalos.

La idea desarrollada trata de resolver el problema de la visualización científica de datos escalares discretos que convencionalmente no se encuentra resuelto. 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 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.

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 a mencionar es que el tamaño de la figura puede ser determinante a la hora de interpretar la visualización. De acuerdo a determinadas particularidades encontradas en los datos, modificando su radio se puede obtener mejores resultados. (Algoritmo 9)

Para colorear la superficie puede ser utilizada cualquier técnica estándar 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 todas las figuras en su interior.

Esta técnica permite mayor realismo pues 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 pues pueden originarse solapamientos entre las figuras, lo cual puede ser mejorado de cierta manera modificando su tamaño.

**Algoritmo: Obtención y visualización de discontinuidades utilizando figuras 3D en el interior del objeto**

```
Entrada: malla mesh, lista de puntos l, radio r,

  For i = 0 to l.count − 1 do
    figura fig = new figura(l[i], r) ;
    lista_figuras.add(fig);
  end
  For i = 0 to lista_figuras.count − 1 do
    draw(lista_figuras[i]);
  end
```

Algoritmo 9 Visualización en el interior del objeto de micro-fisuras y discontinuidades.

El algoritmo de visualización de datos escalares discontinuos propuesto consiste en trasladar las discontinuidades a la superficie del cuerpo. Este método es mucho más costoso 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 de la fisura y la posición con respecto a la superficie.

La idea reside 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 para el investigador o ingeniero. 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 esté dentro de 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 búsqueda de rango, el árbol kd, para organizar los vértices de la malla. Esta estructura es seleccionada por las ventajas que ofrece en bajas dimensiones. El algoritmo de construcción

de esta estructura así como la manera en que se enfrenta la búsqueda a través de ella puede ser encontrado en la sección 1.5.3.

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, luego se busca 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.

La ecuación del plano está dada en (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 triangular, donde $n$ es la normal al plano $p_1$ y $v$ un vector con inicio en el plano $p_1$ y final en el punto $p$.

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

$$p' = p - r \cdot n \text{ donde } r = v \times n \qquad (5)$$

Una vez encontradas todas las caras en las cuales se dibujará se generan los polígonos que van a simbolizar las discontinuidades en el cuerpo utilizando los quaterniones. Dada la cantidad de lados, el radio y el centro del polígono se generan todos los vértices exteriores utilizando rotaciones vectoriales; el centro del polígono coincide con la proyección del punto en el plano. (Algoritmo 10).

**Algoritmo: Visualización de discontinuidades utilizando polígonos regulares en la superficie del objeto**

```
Entrada: malla mesh, lista de puntos l, radio poligono r, cantidad lados n

  kdtree kdtree = new kdtree(mesh.vertices, 0);
  For i = 0 to l.count − 1 do
    point vnn = buscar_vecindad(kdtree.raiz, l[i]) ;
    list f = buscar_caras_frente(vnn, l[i]) ;
    poligono poli = new poligono(l[i], r) ;
    lista_poligonos.add(f, poli);
  end
  For i = 0 to lista_poligonos.count − 1 do
    draw(lista_poligonos[i]);
  end
```

Algoritmo 10 Visualización en el exterior del objeto las micro-fisuras y discontinuidades.

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.

La búsqueda del vecino más cercano es un problema muy abordado y necesario en una amplia cantidad de aplicaciones. Según las características de los datos en este trabajo la estructura de datos más propicia para su uso es el árbol kd.

### 2.3.2 Árbol kd y el problema del vecino más cercano.

Como se menciono en 1.5.3 el árbol se construye 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 recursivamente. (Algoritmo 11).

**Algoritmo: Construcción de la estructura de datos árbol kd**

**Entrada:** lista de puntos ***lista***, nivel de profundidad ***nivel***

$coordenada = nivel \ \mathbf{mod}\ 3$ ;
$lista.\mathbf{sort}(coordenada)$;
$nodo = \mathbf{new}\ \mathbf{nodo}(lista.\mathbf{mediana})$;
$nodo.\mathbf{hijo\_izquierdo} = \mathbf{kdtree}(lista\ a\ la\ izquierda\ de\ la\ mediana,\ nivel + 1)$;
$nodo.\mathbf{hijo\_derecho} = \mathbf{kdtree}(lista\ a\ la\ derecha\ de\ la\ mediana,\ nivel + 1)$;

**Return:** $nodo\_raiz$;

Algoritmo 11 Proceso de construcción del *kd-tree*

La búsqueda del vecino más cercano utilizando el árbol kd es recursiva. Se realiza de la siguiente forma: 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 $x[c] + dnn \leq v$, 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 $x[c] - dnn \geq v$ el hijo izquierdo tampoco lo puede contener. (Algoritmo 12)

**Algoritmo: Búsqueda del vecino más cercano utilizando el árbol kd.**

```
Entrada: nodo raiz, punto p

  Punto vecino = raiz;
  dnn = distancia(vecino, p) ;
  c = cordenada_discriminante() ;
  if  x[c] + dnn ≤ v  then
        vecino = buscar_vecindad(hijo_izquierdo, p) ;
  endif
  if  x[c] − dnn ≥ v  then
        vecino = buscar_vecindad(hijo_derecho, p) ;
  endif

Return: vecino;
```

Algoritmo 12 Búsqueda de vecindad con el *kd-tree*

Otro aspecto a mencionar es la utilización de los quaterniones y su conveniente notación para ser utilizado en rotaciones vectoriales a la hora de formar las figuras que simbolizarán las discontinuidades en la superficie.

### 2.3.3 Rotaciones vectoriales: Quaterniones.

Los quaterniones, en matemática, son una extensión no conmutativa de los números complejos. No es un término reciente, fueron descritos por primera vez en 1844 por el irlandés *William Rowan Hamilton* y aplicados a transformaciones geométricas en tres dimensiones [68]. Aunque en determinados trabajos matemáticos teóricos o prácticos son sustituidos por vectores tienen especial importancia en aplicaciones que involucren rotaciones. [69, 70]

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

$$H = \{a + bi + cj + dk : a, b, c, d \in R\} \qquad (8)$$

Los quaterniones presentan una notación conveniente para representar orientaciones y rotaciones. Característica que les confiere una especial importancia 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 $\theta$ un número real. La rotación alrededor del eje con un ángulo $\theta$ envía el punto $q$ sobre el punto $q' = x'i + y'j + z'k$ dado por:

$$q' = h \cdot q \cdot h' \quad \text{donde} \quad h = \cos(\theta/2) + u \cdot \sin(\theta/2) \qquad (9)$$

A continuación se muestran los algoritmos para obtener los polígonos regulares, que simbolizarán las discontinuidades en la superficie del cuerpo, utilizando los quaterniones y las rotaciones vectoriales. El primero es el algoritmo general para construir un polígono por rotación de un vector en el espacio (Algoritmo 13) y el segundo algoritmo está dedicado a rotar un vector en el espacio utilizando quaterniones (Algoritmo 14).

**Algoritmo: Construcción de polígonos regulares a partir de rotaciones vectoriales.**

```
Entrada: vector v, valor entero n

  Quaternion u = Eje_Rotacion(v);
  alpha = 360/n ;
  h = cos(alpha/2) + u * sen(alpha/2);
  For i = 0 to n do
     point p = Rotar_Vector(v, h);
     lista.add(p);
  end

Return: lista;
```

Algoritmo 13 Construcción de un polígono regular rotando un vector en el espacio.

**Algoritmo: Rotación de un vector en el espacio utilizando quaterniones**

```
Entrada: vector v, quaternion h

  quaternion h' = Conjugada(h);
  quaternion q = new Quaternion(v);
  quaternion q' = h · q · h';

Return: new Vector(q');
```

Algoritmo 14 Rotación de un vector en el espacio

## 2.4 Generación de nubes de puntos para superficies y volúmenes, algoritmo de rellenado de superficies de contorno.

Una nube de puntos[38] es un conjunto de vértices en un sistema de coordenadas tridimensional que típicamente son obtenidas con la intención de representar la superficie o el volumen de un objeto. La generación de nubes de puntos tanto para superficies como volúmenes es un tema que requiere atención por la acogida que han tenido los métodos basados en puntos. Estos métodos, como el *MPM* usado en la simulación de fluidos o procesos o estructuras que contengan grandes deformaciones del material, o en *SPH*, método computacional usado actualmente para la simulación de fluidos y otros campos de investigación como astrofísica, balística, volcanología y oceanografía, necesitan un sistema de puntos o partículas diseminados por todo el medio. [71-74]

Una de las características deseables es la homogeneidad. Mediante esta técnica se verifica que los puntos estén distribuidos de forma homogénea dentro de la geometría contenedora. Una de las formas de comprobarla es particionando la geometría en celdas y verificando que en cada celda haya aproximadamente la misma cantidad de puntos generados [75]. En *SPH* es necesario un sistema simétrico que garantice inicialmente la estabilidad del método.

La modelación basada en puntos es la raíz de los sistemas de partículas. Un sistema de partículas consiste en puntos que son movidos de acuerdo a diferentes reglas. Típicamente esos sistemas son usados para modelar fenómenos borrosos naturales tales como fuego, fluidos, humo o nieve [76].

En la actualidad equipos tecnológicos altamente sofisticados se encargan de generar nubes de puntos con calidad y que aproximan de manera casi exacta objetos y estructuras de la vida real. Un láser escáner tridimensional puede digitalizar no solo la forma de un objeto, midiendo de forma automática un gran número de puntos en la superficie, sino también obtener otros tipos de información como el color de la superficie asociando ese valor a cada uno de los puntos [76].

Por otra parte, en muchos casos contamos con información geométrica de objetos y los escáneres y demás tecnologías no pueden ser utilizados en la generación de nubes de puntos.

[38] Del ingles Point Cloud

En este caso es necesario el uso de una herramienta basada en algoritmos computacionales, capaz de generar con bastante exactitud objetos de alta complejidad geométrica.

El método de *rellenado de iso-superficies*[39] propuesto por *Francois Labelle* y *Jonathan Richard Shewchuk* es posible utilizarlo en la generación de nubes de puntos tanto para volúmenes como para superficies. Este método rellena una superficie de contorno con tetraedros de tamaño uniforme. Se garantizan ángulos diedros de muy buena calidad; ángulos entre $10.78^\circ$ y $164.74^\circ$ son obtenidos o entre $8.9^\circ$ y $158.8^\circ$ con cambios en los parámetros del algoritmo. Además se aseguran ángulos planos también de calidad deseable que aseguran la estabilidad en los métodos numéricos, se garantizan ángulos entre $9.04^\circ$ y $154.78^\circ$ [41, 54].

Existen dos variantes del algoritmo una con todos los elementos con la misma resolución y otra en que solo los elementos de los bordes son finos y uniformes y en el interior, donde no es necesario ser tan detallado, son de tamaño variable. Esto permite que sea utilizado como una herramienta muy útil en diferentes tipos de simulaciones numéricas. Otra de las ventajas claramente visible es que se garantiza la calidad de los tetraedros para el método de elementos finitos utilizando una *estructura cúbica centrada* o *celosía de puntos centrada*.

En [41, 54] se asume que la entrada del algoritmo es una función de corte continua $f : R^3 \to R$ que implícitamente representa el dominio geométrico que será "rellenado" con tetraedros, y el conjunto de puntos $\{p : f(p) \geq 0\}$ los puntos considerados interiores. Los puntos donde $f$ es negativa son considerados fuera del dominio y usualmente no son mallados, aunque por las características del algoritmo pudiera mallarse a ambos lados de la geometría, a partir de $f(x) = 0$, y obtener mallas compatibles. De esta característica se puede generalizar que cualquier juego de datos del que se pueda obtener información de la interioridad de un punto es posible la aplicación del algoritmo. Los mapas de distancias y también las funciones de distancia, los cuales almacenan la información en celosías, permiten el uso del algoritmo de manera casi transparente.

El uso de las *celosías de puntos* es común en mineralogía para representar la estructura de muchos cristales. Esos conjuntos regulares de puntos componen grillas o mallas simples que pueden ser utilizadas directamente en algunos métodos numéricos. Sin embargo, las mismas pueden ser usadas como base para algoritmos de mallado mucho más sofisticados.

[39] Iso-surface stuffing

Como se viene mencionando el algoritmo empleado, al igual que *Marching Cubes* usa una *celosía de puntos centrada* como estructura espacial de fondo para guiar la creación de la malla. Esta estructura es descrita de la siguiente forma:

$\mathrm{BCC} = \mathrm{Z}^3 \cup \left(\mathrm{Z}^3 + \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)\right)$ donde $\mathrm{Z}^3$ es una celosía de puntos y $\mathrm{Z}^3 + \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ es una copia de este escalando cada punto al centro del cubo original. Para mayor información ver [41, 54].

El algoritmo puede ser separado en pasos:

1. Creación de una estructura de fondo para dirigir el mallado: se escoge el subconjunto $P$ a partir de la celosía cúbica centrada, $P$ debe incluir cada punto donde la función de corte $f$ es no negativa y cada punto conectado por un eje de la estructura con un punto donde $f$ es positiva. Calcular y almacenar el valor de $f$ de cada punto de $P$.
2. Cálculo de los puntos de corte: por cada eje de la estructura de fondo con ambos extremos en $P$, si uno de los extremos es positivo (punto interior) y el otro es negativo (punto exterior) se calcula, utilizando algún método de interpolación, el punto de corte aproximado por donde la *cero-superficie* corta el eje.
3. Deformar la estructura de fondo: por cada punto $q \in P$ chequear la presencia de puntos de corte en cada uno de los 14 eje unidos a $q$. Si uno de estos puntos de corte $c$ está muy cerca de $q$ se dice que $q$ es violado por $c$. Si $q$ es violado por algún punto de corte, se deforma la estructura moviendo $q$ al punto de corte que lo viola. Por conveniencia se usa el punto de corte más cercano a $q$. El efecto de este movimiento es mover a $q$ hacia la *cero-superficie*. Se cambia el valor de $f(q)$ a cero. Descartar todos los puntos de corte en los ejes que contienen a $q$ pues estos ejes ya no tendrán un extremo positivo y otro negativo.
4. Mallar a partir de la estructura de fondo: por cada tetraedro, obtenido a partir de la estructura guía, que tenga al menos un vértice con valor positivo mallar utilizando las plantillas pre-computadas. Escoger la plantilla adecuada en cada caso depende del signo de cada uno de los vértices del tetraedro. (Véase Figura 14)

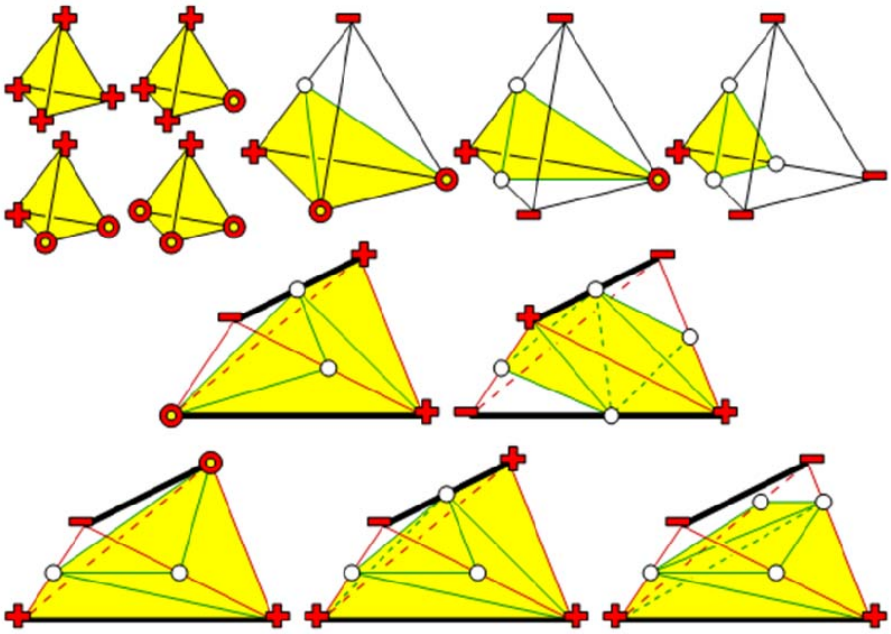


Figura 14 Plantillas pre-computadas para dirigir el proceso de mallado, tomado de [41]

En algunas plantillas es necesario aplicar determinada norma para no caer en inconsistencias en el mallado. Este criterio es nombrado "***regla de paridad***". Está dada según el siguiente precepto:

Sean y los extremos de un eje negro y sean y los puntos de corte donde los ejes rojos son truncados, y las diagonales del cuadrilátero formado. Debido a la geometría de la estructura de fondo cualquier tiene un número par de coordenadas mayor que las correspondientes coordenadas de y tiene un número impar de coordenadas mayor que las coordenadas de o viceversa. Entonces tenemos que:

- Si es un eje negro se escoge si tiene un número impar de coordenadas mayor que las coordenadas de , se escoge si el numero es par. Esta regla nos permite el uso de la plantilla inferior derecha, la cual tiene mejor calidad que otras alternativas (Figura 14). Obsérvese que la plantilla tiene no una sino dos de estas cara-cuadrilátero, frontal y trasera, y dos diagonales que no comparten un punto final.
- Si y están en un eje rojo se invierte la regla y se escoge si tiene un numero par de coordenadas mayores que las coordenadas de y si el numero es impar.

El algoritmo de *rellenado de iso-superficies*, entre otras cosas, permite controlar el tamaño de los elementos, lo que para el caso en que se quiere usar ayuda a mantener controlada la distancia entre los puntos en la nube.

Las modificaciones necesarias para la generación de nubes de puntos tanto para volúmenes como superficies están basadas en las plantillas del algoritmo original. La generación para volúmenes en el caso simétrico-homogéneo es transparente, para el aleatorio-homogéneo es necesario solo el movimiento de vértices interiores y para la generación en superficies modificando las plantillas para solo obtener los vértices que se encuentren sobre esta.

## 2.5 Conclusiones parciales.

Las técnicas de eliminación de polígonos y objetos ocultos para determinar la visibilidad permiten reducir la complejidad de escenas de conglomerados de partículas en métodos con y sin información del contorno.

La visualización de propiedades en conglomerados de partículas es necesaria y las técnicas de visualización *Interpolación de Colores* y *Superficies de Contorno* son las más adecuadas para mostrar información de las partículas por las características de los datos y de las técnicas en sí.

La visualización de información proveniente de los métodos sin mallas es un tema muy tratado por la dificultad de no contar con información de contorno. El uso del método propuesto resuelve efectivamente el problema, se presenta una solución para 2D. Se propone una solución a la gran cantidad de información en las búsquedas a la hora de la interpolación utilizando una estructura de datos, árbol de cubrimiento, muy eficiente en la búsqueda de vecindades por distancias.

Se creó un algoritmo para la representación de datos discretos y continuos a intervalos, auxiliado por la estructura de consulta espacial árbol de k dimensiones, permitiendo visualizar con más claridad las micro-fisuras, deformaciones, grietas o cualquier otro tipo de discontinuidad.

El algoritmo *rellenado de iso-superficies,* seleccionado con el objetivo de generar nubes de puntos para superficies y volúmenes, ofrece garantías en cuanto a la calidad de la distribución de los puntos para su aplicación en simulaciones de fluidos.

# Capítulo III. Resultados de los métodos propuestos para pre y post-proceso en métodos de partículas, sin mallas y de puntos

La visualización de información proveniente de métodos numéricos, tales como los conocidos *Elementos Finitos*, *Diferencias Finitas* y otros más recientes como *Elementos Discretos*, es un tema de mucha importancia en el área de la ingeniería y de investigaciones relacionadas con la modelación y simulación. La gran cantidad de datos y la diversidad en cuanto a tipo hace complejo el proceso de visualización.

Las soluciones propuestas en este trabajo a determinados problemas relacionados con la representación visual de datos son el punto de partida al desarrollo de técnicas bien definidas y que respondan adecuadamente a las necesidades de investigadores e ingenieros. Se presentan resultados para *métodos de partículas*, *métodos libres de mallas* o cualquiera que implique grandes cambios geométricos o deformaciones del modelo de análisis o con un fuerte grado de discontinuidad inherente.

## 3.1 Resultados obtenidos aplicando los algoritmos propuestos para datos provenientes del método de partículas.

La problemática de visualizar conglomerados de partículas tiene dos aristas, la gran cantidad de información y la aplicación de técnicas de visualización científicas capaces de representar correctamente la información. El desarrollo de técnicas de aceleración para la visualización, en concreto técnicas de obtención de visibilidad, permite disminuir, en este caso, gran cantidad de partículas. Esto pudiera formar parte de un pre-preprocesamiento de los datos para luego aplicar técnicas más afines a la visualización u otras relacionadas con la aceleración en sí.

Las metodologías propuestas, en forma de algoritmos, para la visualización científica forman parte del inicio del camino a la adaptación de muchas más técnicas afines a los resultados del MED.

### 3.1.1 Eliminación de información no visible en conglomerados de partículas.

Como se puede deducir la visualización de grandes volúmenes de datos es un problema hasta para las computadoras más sofisticadas y potentes. La cantidad de partículas manejadas en el *DEM* puede llegar a varios millones lo que puede implicar, además, gigas en cuanto a

almacenamiento. Analizando las características vistas en el epígrafe 2.1.2 se puede derivar que no es necesaria la visualización completa de todo el sistema de partículas ya que una gran mayoría no son visibles desde ningún punto de vista.

Se implementaron dos técnicas para eliminar elementos potencialmente no visibles desde la posición de la cámara, *eliminación de polígonos y objetos ocultos*. Se analizaron cinco sistemas de partículas diferentes para ver las diferencias en cuanto a su comportamiento: un cubo, una cuchilla perteneciente a una herramienta de corte del terreno, dos estructuras ósea (fémur y cráneo) y una estructura de ingeniería (estructura de un edificio). Para la generación del medio se usó el procedimiento de empaquetamiento de partículas genéricas para el *MED* propuesto en [77].

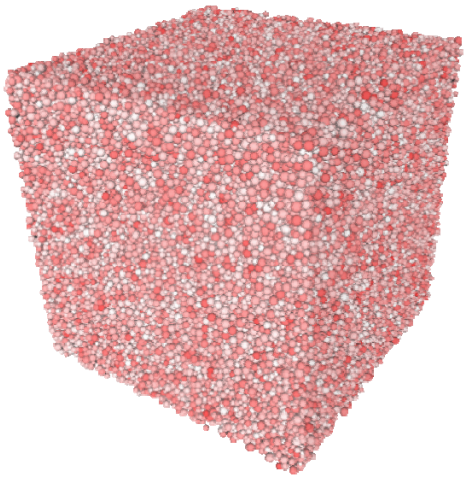
En la Tabla 2 se muestran los resultados obtenidos tras la aplicación de este método. Estos resultados están expresados en función de la cantidad de partículas que se tuvieron en cuenta a la hora de generar las imágenes tras el uso de las técnicas de visibilidad mencionadas. La Figura 15 muestra los empaquetamientos a los que se hace alusión en la Tabla 2.

| Objeto de estudio. | Cantidad de Partículas | Partículas en la Corteza | Partículas Dibujadas |
|---|---|---|---|
| Cubo | 132 681 | 68 591 | 14 381 |
| | | | 38 459 |
| | | | 27 012 |
| Cuchilla | 352 203 | 159 236 | 47 490 |
| | | | 92 421 |
| | | | 107 727 |
| Fémur | 211 599 | 153 453 | 76 930 |
| | | | 78 065 |
| | | | 86 128 |
| Estructura | 3 368 687 | 2 771 440 | 1 342 632 |
| | | | 1 523 494 |
| | | | 1 921 528 |
| Cráneo Humano | 1 101 920 | 942 777 | 660 133 |
| | | | 664 042 |
| | | | 66 9097 |

Tabla 2 Cantidades de partículas utilizadas en cada etapa de la aplicación del método propuesto.

La columna “Partículas en la Corteza” muestra la cantidad de partículas que pueden ser visibles desde alguna posición de la cámara. Esta constituye una simplificación inicial del conjunto de partículas a dibujar. Por su parte la columna “Partículas dibujadas” muestra la cantidad de partículas que finalmente fueron dibujadas en la obtención de la imagen desde tres

posiciones cualquieras del observador. En este caso el criterio de selección fue eliminar aquellas partículas de la corteza que no se encontraran de frente a la cámara. Se muestran tres valores tomados desde diferentes posiciones de la cámara.



(a)

(b)

(c)

(d)

(f)

**Figura 15 Ejemplos de empaquetamientos. (a) Cubo, 132 681 partículas. (b) Cuchilla de corte en el terreno, 352 203 partículas. (c) Estructura ósea (Fémur), 12 605 partículas. (d) Estructura ingenieril (Edificio), 3 368 687 partículas. Cráneo Humano, 1 101 920 par**

La cantidad de partículas eliminadas, utilizando los algoritmos propuestos, para desarrollar el proceso de visualización varía de acuerdo a la forma geométrica de los cuerpos y a la relación que existe entre el tamaño de la partícula y el volumen total del cuerpo. Como puede apreciarse, los mejores resultados se obtienen en el caso del cubo y la cuchilla, que son objetos que no tienen partes estrechas o delgadas, como en el caso del fémur y la estructura. El cráneo, aunque visualmente no lo sea, es también un “objeto” delgado pues solo se tomó una capa fina, la capa que constituye el hueso, para generar el sistema de partículas. A pesar de esto, con la utilización del método propuesto, se logra disminuir la cantidad de partículas considerablemente, acelerando, con ello, el proceso de visualización.

La variación al algoritmo, cuando no se cuenta con datos del contorno, también mostró resultados relevantes. En este caso es más complejo obtener una corteza más suave y uniforme por las características mencionadas en el capítulo anterior. A continuación se muestran los resultados asociados a la aplicación de esta metodología. (Véase Tabla 3)

| Objeto de estudio | Cantidad de partículas. | Tamaño de celda (veces el radio máximo) | Cantidad de celdas. | Profundidad de corteza. | Cantidad de partículas en la corteza. | Tanto por ciento |
|---|---|---|---|---|---|---|
| Cubo | 111 495 | 5 | 5 832 | 1 | 26 421 | 23.69 |
| | | 10 | 1 000 | 1 | 57 640 | 51.69 |
| | | 3 | 27 000 | 1 | 19 184 | 17.20 |
| | | 3 | 27 000 | 2 | 40 456 | 36.28 |
| | | 6 | 4 913 | 1 | 40 077 | 35.94 |
| Fémur | 211 599 | 6 | 46 460 | 1 | 101 080 | 47.76 |
| | | 3 | 284 130 | 1 | 69 015 | 32.61 |
| | | 3 | 284 130 | 2 | 137 216 | 64.84 |
| | | 5 | 72 358 | 1 | 89 694 | 42.38 |
| | | 10 | 13 230 | 1 | 136 588 | 64.54 |
| Cuchilla | 352 203 | 5 | 72 576 | 1 | 80 104 | 22.74 |
| | | 10 | 12 160 | 1 | 123 607 | 35.09 |
| | | 3 | 284 316 | 1 | 78 591 | 22.31 |
| | | 3 | 284 316 | 2 | 172 378 | 48.93 |
| | | 6 | 44 640 | 1 | 88 000 | 24.98 |

Tabla 3 Resultados asociados al cálculo de las partículas visibles cuando no se tiene información del contorno

La tabla muestra en este orden:

- La cantidad de partículas del conglomerado de partículas.
- El tamaño de celda: en un principio puede ser cualquier valor real positivo pero tomar un valor pequeño puede acarrear que aparezcan celdas interiores vacías lo que no es bueno para la visibilidad y un factor muy grande puede también traer resultados no deseados pues estaríamos tomando partículas para la corteza que no afectarían la imagen final; en este caso está calculado con respecto al radio máximo de las partículas. La cantidad de celdas se corresponde con la cantidad de celdas a recorrer en la celosía y en el mapa de distancia.
- Cantidad de celdas a recorrer.
- Profundidad de la corteza es la profundidad tomada en el mapa de distancia para obtener las partículas visibles, esto es, “2” significa tomar todas las celdas de profundidad “1” y “2”.
- La cantidad de partículas en la corteza.
- Tanto por ciento de partículas potencialmente visibles con respecto al total.

Atendiendo a los datos mostrados en la Tabla 3 se puede llegar a conclusiones parciales con respecto al grosor más conveniente de la corteza, además de que quizás sea más ventajoso a

la hora de obtener una buena imagen fijar una corteza pequeña y aumentar el factor de ancho de las celdas, lo cual sería más costoso en tiempo pero se evitarían huecos en la imagen. Comparándolos con los resultados del método donde se tiene información del contorno (Tabla 2) se puede ver que con un ancho 10 (este fue el valor tomado para la Tabla 2) se mejora en la cantidad de esferas en la corteza. (Ver por ciento asociado).

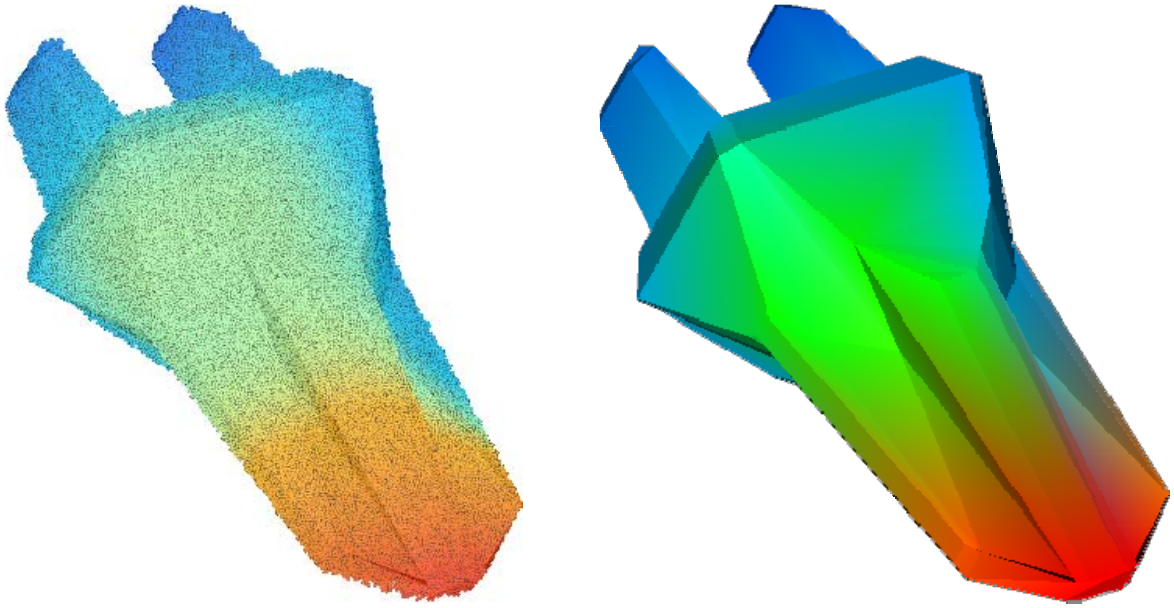
El algoritmo permite a la vez representar partes interiores de la nube de partículas basado en los niveles de profundidad calculados.

El valor asociado al tamaño de celda es más conveniente tomarlo a partir de un estudio del sistema de partículas con respecto no solo al radio máximo sino a la densidad, porosidad, radio mínimo y otras propiedades de la nube de partículas.

Solo fueron calculadas las partículas potencialmente visibles pues la idea es aplicar posteriormente alguna técnica, como impostores o nivel de detalle, para la visualización. El cálculo de las partículas que quedan de frente al observador y que realmente son visibles es posible hacerlo comparando la dirección de los vectores asociados a la dirección en que observa la cámara y la dirección de la profundidad en el mapa de distancias discretas calculado. (Figura 10)

### 3.1.2 Visualización científica de propiedades asociadas a conglomerados de partículas.

Se implementaron las técnicas de visualización científicas interpolación directa de colores y superficies de contorno, considerando, en ambos casos, dos acercamientos. En el discreto se considera cada partícula como un objeto con propiedades independientes mientras que en el acercamiento continuo la visualización se realiza considerando el sistema de partículas como una entidad continua.



**(a)** **(b)**

Figura 16 Interpolación de colores para conglomerado de partículas. Acercamiento Discreto (a), Acercamiento Continuo (b).

En la Figura 16 se aprecia una interpolación de colores sobre la cuchilla de corte del terreno. En Figura 16a se interpolación se realiza visualizando todo el conglomerado de partículas, se visualizan de manera independiente cada una de las partículas con su color correspondiente. En este caso, Figura 16a, como muchos otros, la propiedad estudiada permite que aparentemente quede una interpolación suave de los colores por todo el medio.

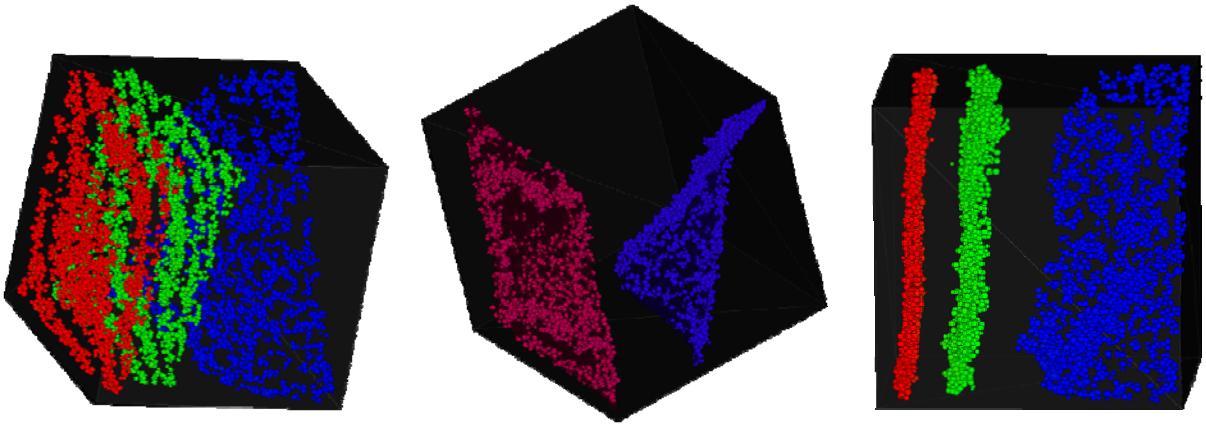
La Figura 16b muestra una representación continua de la acción del terreno sobre la cuchilla. En el contorno del objeto se interpolan los colores a partir de la vecindad de las partículas. Esta metodología permite ver siempre una interpolación suave de los colores sobre el objeto estudiado. Cuando contamos con información del contorno es posible interpolar los colores correspondientes a la propiedad estudiada sobre este y lograr el efecto deseado gracias a las facilidades que proveen las librerías gráficas en cuanto a la interpolación de colores [78]. En la implementación de las técnicas propuestas en este trabajo se utilizó *OpenGL*[40].

La técnica de visualización científica, *superficie de contorno*, implementada para conglomerados de partículas también hace uso de la *interpolación de colores*. Como se mencionó, se pueden considerar las dos aproximaciones anteriores, discreta y continua, además que, en este caso es posible una combinación de ambas. La utilización indirecta de la

[40] Librería de funciones que proveen una interfaz de software al hardware gráfico (software interface to graphics hardware)

interpolación de colores está dada porque, tanto en el acercamiento continuo como el discreto, el cálculo de los valores para la coloración de la iso-superficie está dado por un conjunto de partículas puede arrojar valores diferentes en cada uno de los vértices donde se hace el muestreo y las partículas a visualizar en la iso-superficie discreta pueden tener diferencias en cuanto al valor de la propiedad estudiada, aunque apenas puede ser perceptible.

En las Figura 17 y Figura 18 se muestran imágenes de la aplicación del algoritmo propuesto para la obtención de iso-superficies. Obsérvese que es posible combinar los acercamientos discreto y continuo y la técnica interpolación directa de colores.



**a)** **b)** **c)**

Figura 17 Superficies de contorno obtenidas de un conglomerado de partículas en un cubo. a) y c) acercamiento discreto, b) combinación de ambas perspectivas.

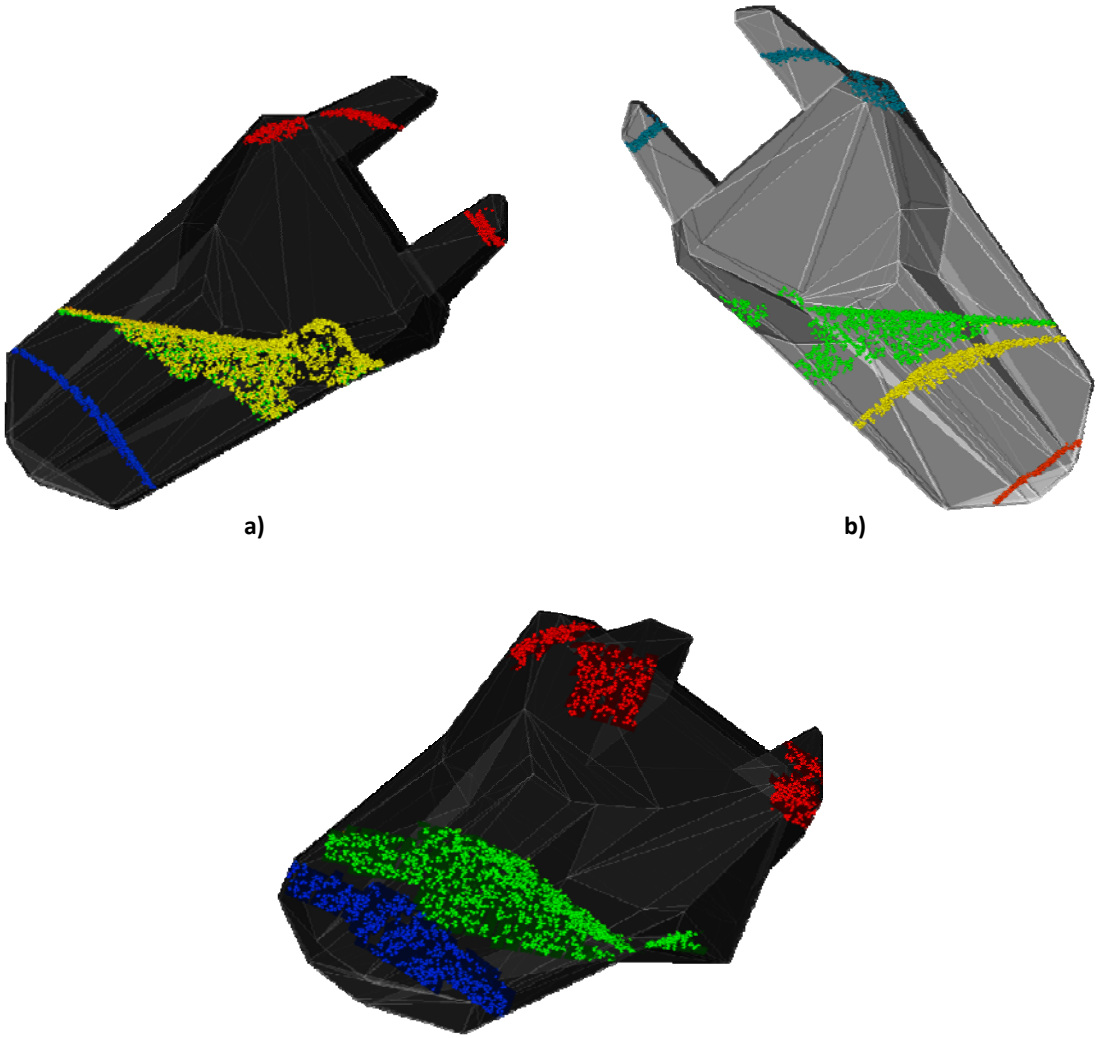



Figura 18 Superficies de contorno en la cuchilla de corte del terreno. a) y b) acercamiento discreto, c) combinación del continuo y el discreto.

## 3.2 Visualización científica de datos discretos y continuos a intervalos.

Las técnicas de visualización científicas utilizadas para la representación de datos discretos y continuos a intervalos se basan en el uso de pequeñas figuras geométricas que simbolizan la discontinuidad. La metodología presentada en este trabajo con el fin de visualizar datos discretos puede ser muy factible en el caso de grietas en estructuras ya que muestra la relación existente entre los datos en sí y la superficie o contorno del cuerpo que los contiene.

En la Figura 19 se muestra una discontinuidad en una estructura de un anillo, en a) y b) observamos la relación de los datos con la superficie, en c) y d) también observamos esta característica incluyendo también la información en el lugar real donde se encuentra. En b) y d)

se observan diferentes coloraciones de la información en el contorno del cuerpo que se visualiza de acuerdo a la distancia entre los datos y la superficie.

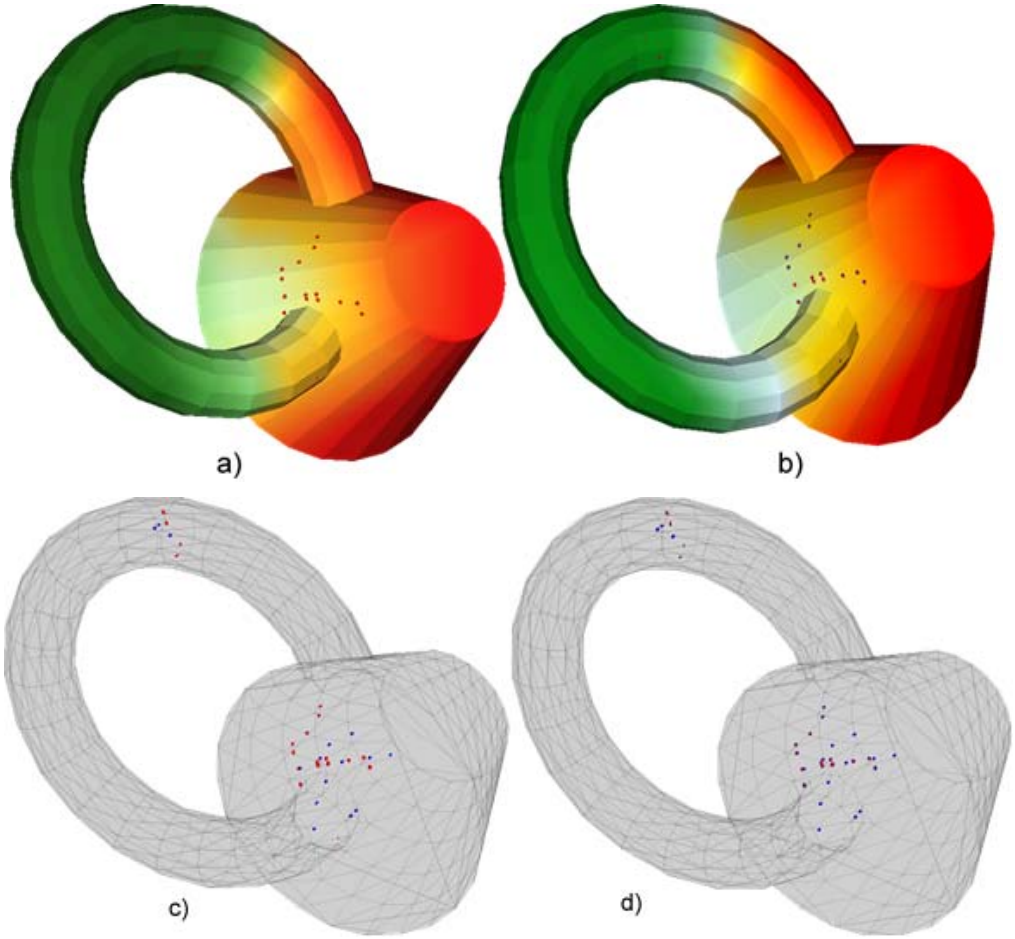



Figura 19 Discontinuidad en el interior de un anillo

Otra característica que permite la implementación hecha de la técnica es la generación de las discontinuidades utilizando distintas figuras geométricas, esto gracias a los quaterniones. Esto puede ayudar visualmente al investigador ya que puede elegir dependiendo de las características de los datos y de la cantidad. (Véase Figura 20) Con respecto a la cantidad de información a visualizar es necesario un estudio previo ya que un gran volumen puede ocasionar malas interpretaciones de la imagen resultante. En este caso es necesario un filtrado de los datos para evitar solapamientos en la imagen final.

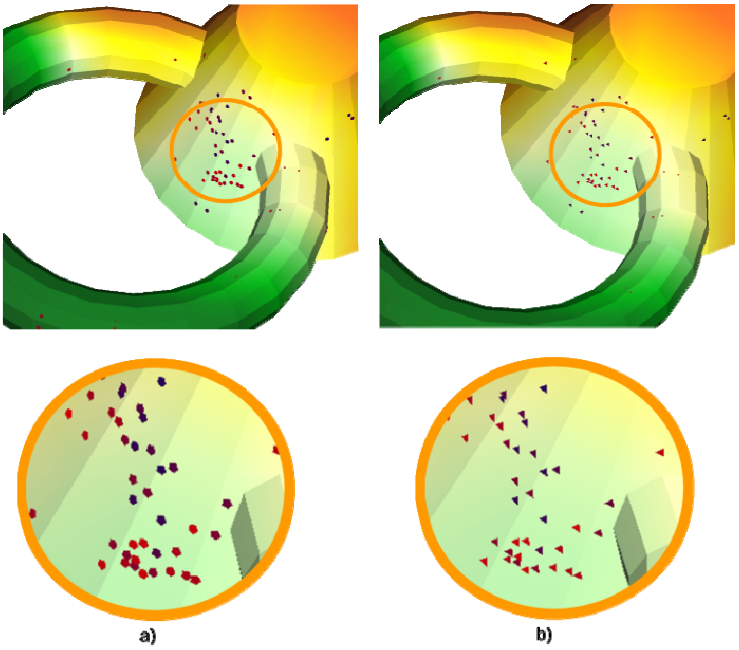



Figura 20 Acercamiento a la estructura de un anillo. a) imagen generada utilizando pentágonos. b) imagen generada utilizando triángulos

Cuando contamos con información continua y discontinuidades en su interior una combinación de varias técnicas relacionadas con la visualización de este tipo de dato permite obtener un mejor análisis de resultados. (Véase Figura 21)

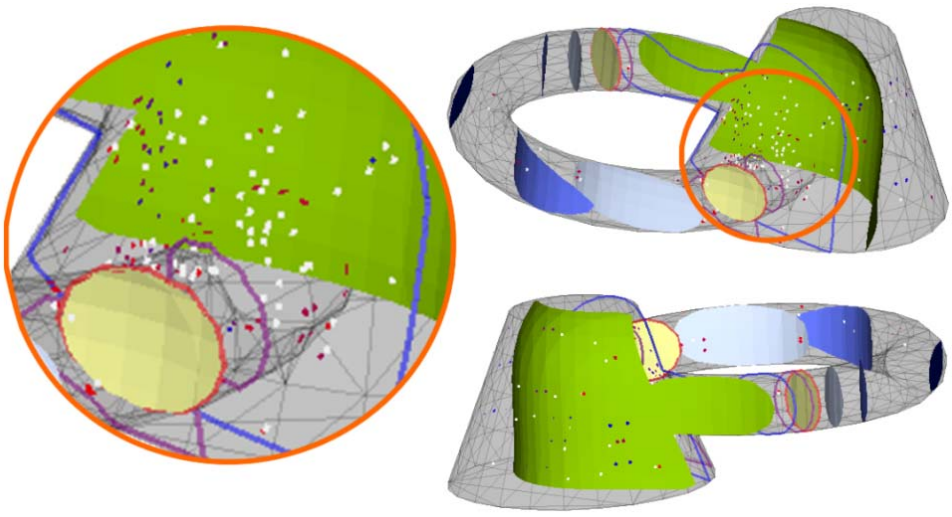


Figura 21 Combinación de técnicas de visualización científicas para datos escalares continuos y discretos.

## 3.3 Resultados obtenidos aplicando los algoritmos de visualización propuestos para información métodos sin mallas.

El algoritmo fue empleado para visualizar nubes con distintas cantidades de puntos. El Gráfico 1 muestra el comportamiento de los tiempos de interpolación de acuerdo a la cantidad de puntos empleados. En el eje de las ordenadas se tiene el tiempo promedio necesario para interpolar el valor en un punto del dominio, y por las abscisas se tiene la cantidad promedio de puntos que influyeron en una interpolación.

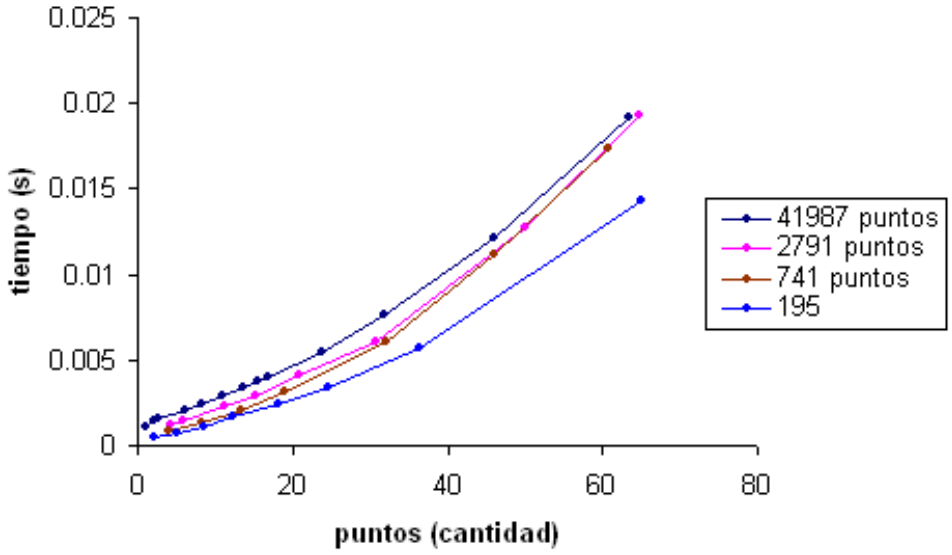



Gráfico 1 Tiempos promedios de interpolación para un punto.

En un análisis del Gráfico 1 es perceptible que a pesar del incremento notable de la cantidad de puntos utilizados en cada uno de los casos, no crece de modo acelerado la distancia entre las funciones, y por tanto, tampoco crece en gran medida el tiempo necesario para interpolar un punto, esto, bajo condiciones similares, dígase una cantidad similar del promedio de vecinos que se considera debido al radio de influencia. Por tanto, se puede considerar el uso de árboles de cubrimiento como una estructura adecuada para acelerar el problema de la búsqueda de los vecinos en un radio determinado, enfocado a la interpolación de datos esparcidos cuando se utilizan *FBR* de alcance local.

La utilización de este algoritmo ante resultados compuestos por grandes volúmenes de datos provenientes del *MED*, *PFEM* u otros métodos es factible, ya que aligera el costo

computacional y el tratamiento de estas grandes cantidades de puntos. Este método agiliza el post procesamiento de resultados donde las nubes de puntos obtenidas son muy densas y los valores presentan cambios significativos en vecindades no lejanas, haciendo de la interpolación local la mejor opción para la eficiencia y la exactitud de la imagen resultante.

El radio de influencia debe ser seleccionado de manera correcta de acuerdo a la densidad de la nube de puntos. La selección de un radio demasiado pequeño puede traer como resultado la aparición de discontinuidades no deseadas en la imagen final (Figura 22). Por otra parte, mientras mayor sea el radio seleccionado mayor será la cantidad de vecinos que influyen sobre un punto, y por tanto mayores los tiempos de interpolación, sin embargo, en nubes de puntos suficientemente densas, la diferencia entre las imágenes generadas para un radio mayor o menor que otro no son generalmente significativas.

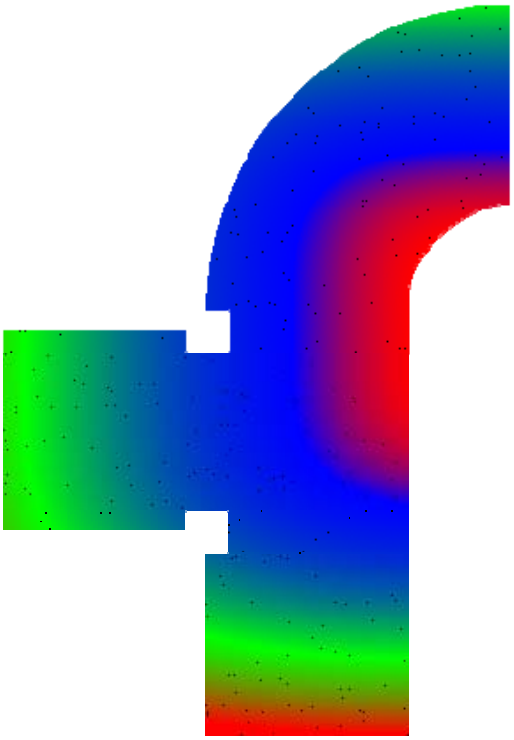


**Figura 22 Aparición de discontinuidades -puntos negros- en la generación de una imagen debido a la selección de un radio demasiado pequeño.**

En la Figura 23 se puede apreciar la diferencia entre dos imágenes generadas para la misma nube de puntos, en este caso, de 41 987 puntos. El radio considerado para la Figura 23a fue de 0.03u, lo cual trajo consigo que el promedio de los puntos que se consideraron y el tiempo promedio para cada interpolación fueran 6.21 y 0.0021s respectivamente. Para la Figura 23b el

radio fue de 0.1u, y el promedio de los puntos que se consideraron y el tiempo promedio para cada interpolación fueran 63.45 y 0.0192s respectivamente.

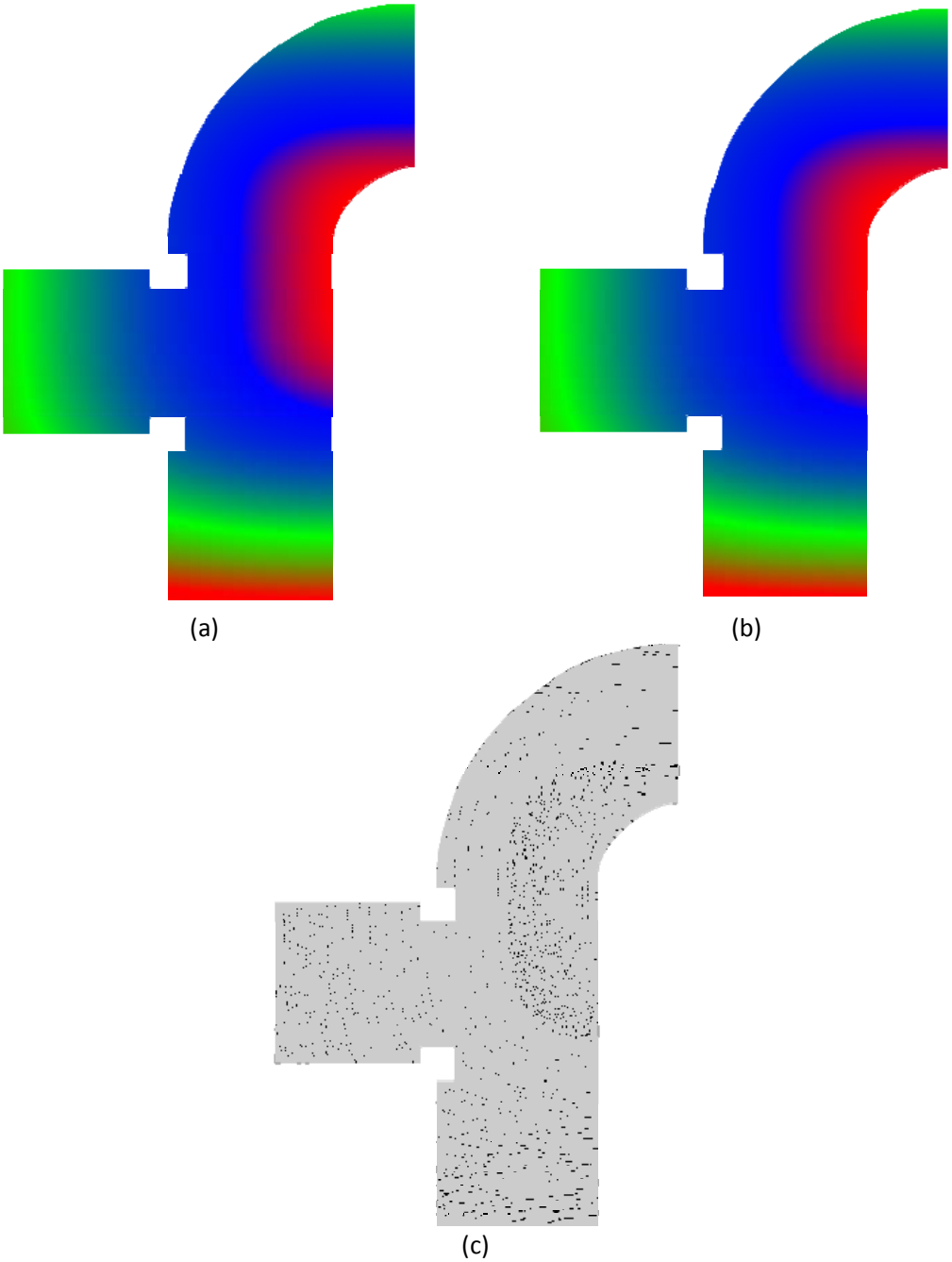



**Figura 23 Diferencia entre dos imágenes generadas interpolando a partir de una nube de 41987 puntos. (a) Con un radio de 0.03u. (b) Con un radio de 0.1u. (c) Diferencia entre ambas imágenes; los puntos negros indican los píxeles donde existía diferencia de colores.**

Puede notarse, además, en la Figura 23c que las diferencias entre ambas imágenes -puntos negros- no son considerables, además de ser prácticamente imperceptibles a simple vista. De

cualquier forma, hasta el momento, esta es una decisión del usuario, dependiendo de los resultados que se deseen resaltar y de la suavidad en los cambios de valores en puntos correspondientes a vecindades cercanas. Si se trata de eficiencia computacional, un radio pequeño sería lo ideal, pero el comportamiento de los valores en la nube de puntos es muy importante también para tener resultados más exactos.

## 3.4 Obtención de nubes de puntos para superficies y volumen aplicando el algoritmo “rellenado de iso-superficies”.

Los métodos numéricos basados en puntos o partículas han ganado auge en los últimos tiempos, aunque son tan antiguos como los tradicionales. Las nubes de puntos se crean habitualmente con un láser escáner tridimensional, el cual mide de forma automática un gran número de puntos en la superficie de un objeto. Es necesario para estos métodos también obtener información volumétrica en forma de puntos.

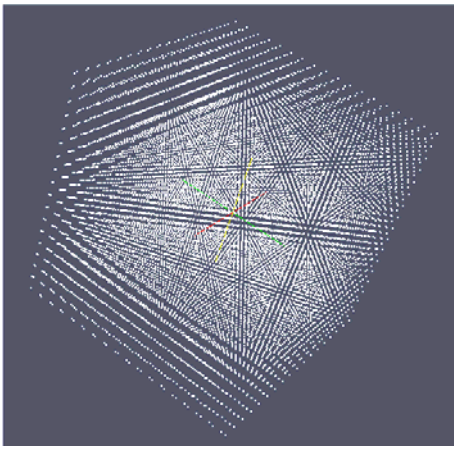


Figura 24 Generación de una nube de puntos en una estructura cúbica con 13662 puntos.

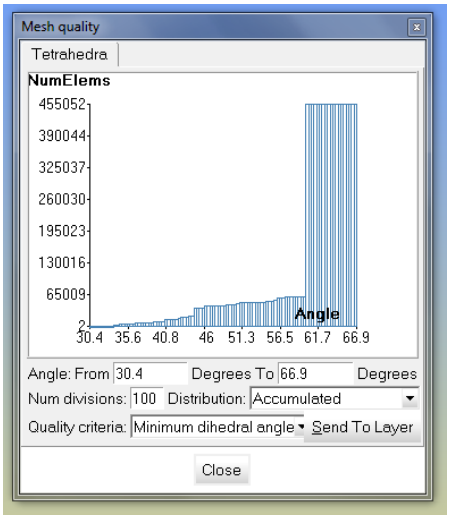
Como se puede apreciar en la Figura 24 los puntos son simétricos, un test de uniformidad daría negativo. En caso de ser necesaria la aleatoriedad en la nube de puntos volumétrica es posible lograrla moviendo cada punto en la esfera que tiene su centro en el punto y radio como la distancia mínima entre los puntos, que también es conocida. Esto solo en los puntos interiores, la superficie no es uniforme, salvo algunos casos específicos, por el movimiento de puntos que se produce en el propio algoritmo de rellenado de iso-superficies.

Por otra parte en otros métodos, como SPH, es deseable la simetría entre los puntos o partículas del sistema ya que se necesita garantizar la estabilidad en la etapa inicial, esto es, en un inicio es necesario un equilibrio entre todas las fuerzas presentes en el modelo.

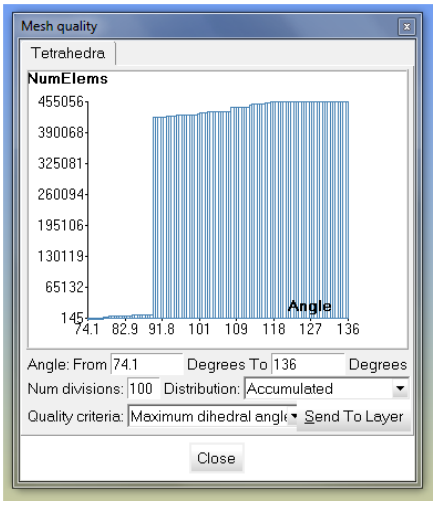
La homogeneidad según [75] se puede medir comparando la cantidad de puntos en una celda, pero como el algoritmo se basa en una división simétrica del espacio, esta característica es evidentemente alcanzable según dicha prueba.

Haciendo alguna pequeña variación al método de generación de mallas volumétricas propuesto en [41, 54] es posible la generación de mallas de superficies con muy buena calidad. Esta facilidad nos permite crear nubes de puntos también para superficies. El método permite en ambos caso, mallas de volumen y de superficies, refinamiento adaptativo dependiendo de las características deseadas para el sistema de puntos, esto es, en la superficie donde se necesita mayor nivel de detalle los elementos/puntos estén más cercanos entre sí, mientras que en el interior donde no es necesario esta característica estén más separados. El refinamiento se hace utilizando un *árbol octal* no convencional donde en el interior del árbol los nodos tengas mayor tamaño que los nodos que intercepten la superficie.

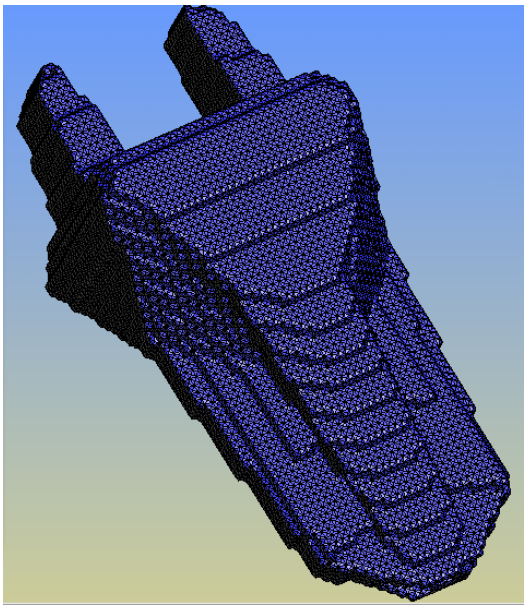
Como se mencionó, anteriormente, una buena medida de la calidad de los elementos en los métodos de mallados es el ángulo diedro. Se probaron en una generación de la cuchilla de corte del terreno y los resultados fueron muy buenos, obteniéndose ángulos diedros mínimo entre $30.4^{\circ}$ y $66.9^{\circ}$ (véase Figura 25a) y ángulos máximo entre $74.1^{\circ}$ y $136^{\circ}$ (véase Figura 25b). Esto nos da una buena medida de lo eficiente que puede ser este algoritmo tanto para la generación de mallas de tetraedros como en las aplicaciones que se proponen en este trabajo.




**(a)**




**(b)**



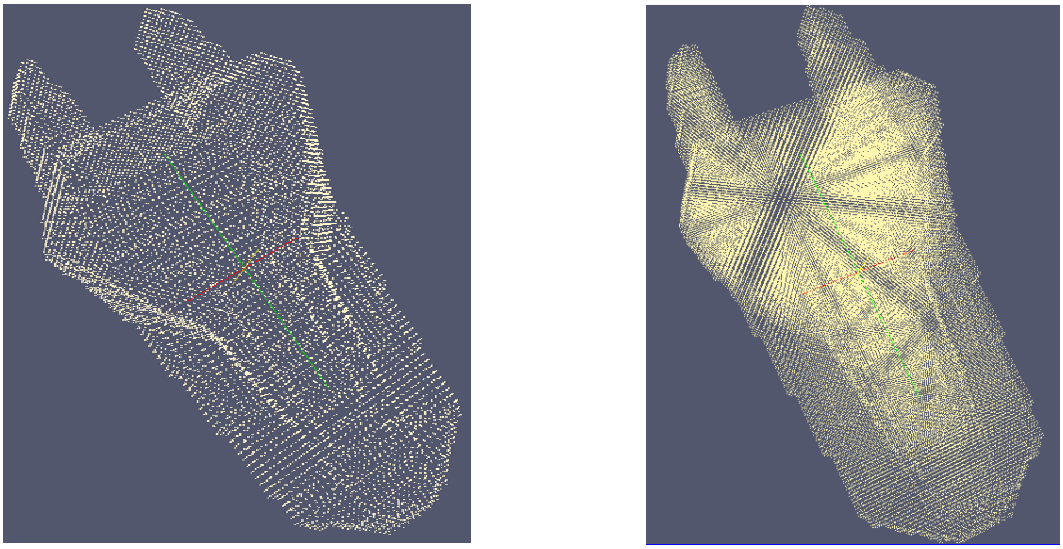
**(c)**

**Figura 25 Comprobación de la calidad de los elementos de la malla generada utilizando el algoritmo rellenado de iso-superficies.**

En la Figura 26 se muestra la generación de nubes de puntos para la cuchilla de corte del terreno. Se utilizó el algoritmo para el rellenado de superficies de contorno propuesto en [41, 54]. Esta aplicación, así como la generación de mallas de volumen es posible gracias a la creación de mapas de distancias a partir de la imagen real. Para su creación solo es necesario poder calcular la distancia desde cualquier punto al objeto, si el punto es interior la distancia debe ser negativa, o tener algún criterio de interioridad, esto es, conocer si un punto cualquiera esta dentro o fuera del objeto.

Esta última forma de obtención de la malla o las nubes de puntos puede ser costosa ya que este criterio siempre lo es en cuanto a tiempo y, además que, la cantidad de puntos a

muestrear puede ser grande en la mayoría de los casos, y esto habría de computarse cada vez que se fuera a generar. El mapa de distancia es más factible ya que es calculado una sola vez y reutilizado cada vez que se quiera generar. El cálculo de la distancia de un punto a la malla es menos costoso que obtener si es interior o exterior o en el peor de los casos de igual complejidad.



**a)** **b)**

**Figura 26 Nube de puntos, generadas utilizando el algoritmo propuesto en [41, 54], para la cuchilla de corte del terreno. a) nube de puntos para superficie de 10208 elementos. b) nube de puntos para volumen de 118994 elementos.**

## 3.5 Conclusiones parciales.

Se implementaron los algoritmos propuestos para el filtrado de datos en escenas de conglomerados de partículas arrojando resultados favorables.

Las adaptaciones de las técnicas de visualización científicas propuestas para la representación de propiedades en conglomerados de partículas mantienen su significado y permiten resolver adecuadamente la problemática del análisis y verificación de propiedades.

La técnica propuesta para la visualización de datos esparcidos permite representar propiedades de datos provenientes de métodos libres de mallas.

La generación de nubes de puntos utilizando el algoritmo propuesto por *Francois Labelle* y *Jonathan Richard Shewchuk* permite obtener nubes para superficies y volúmenes, incluso de geometrías complejas.

## Conclusiones y líneas de trabajo.

El desarrollo de nuevos algoritmos de visualización científica permite a ingenieros e investigadores obtener información que puede ser determinante a la hora de tomar decisiones sobre la estructura o fenómeno estudiado. La aplicación de nuevas formulaciones tanto de los métodos tradicionales como de los métodos libres de mallas ha traído consigo mejoras al análisis sobre determinados fenómenos o estructuras materiales. Muchas de estas mejoras están dadas sobre el análisis que puede hacerse de la representación visual obtenida a partir de los datos.

El método expuesto en este trabajo para el cálculo de la visibilidad en escenas de conglomerados de partículas contribuye a aligerar el costo computacional de la visualización de los empaquetamientos mediante la selección de los elementos que potencialmente van a influir en la imagen final.

Las adaptación e implementación de los algoritmos de visualización científicas interpolación de colores e iso-superficies propuestas para la representación de propiedades en conglomerados de partículas permiten resolver adecuadamente la problemática del análisis y verificación de propiedades.

Para la visualización y estudio de propiedades de datos esparcidos se crea un nuevo método que utiliza las FBR de alcance local unido a la estructura de datos árbol de cubrimiento. El método se aplicó usando la función interpolante de Shepard en su modificación a función local. Es independiente de la dimensión del modelo ya que tanto las FBR como los árboles de cubrimiento dependen únicamente de la distancia entre dos puntos. Se resuelve el problema de la manipulación de datos en estos métodos a través del árbol de cubrimiento.

Se realizó un estudio sobre la influencia en el tiempo de interpolación de la cantidad de puntos dentro del radio de influencia indicado. Una correcta selección de este permite generar imágenes capaces de representar correctamente las propiedades de los datos y con bajos tiempos de generación.

El algoritmo propuesto con el objetivo de visualizar datos escalares discretos permite obtener una mejor interpretación de deformaciones, grietas u otros tipos de discontinuidades en el medio. Puede ser aplicado en conjunto con las técnicas de visualización para datos escalares continuos para ampliar el campo de análisis en métodos numéricos que combinan datos continuos y discretos.

Las nubes de puntos generadas, utilizando el algoritmo rellenado de iso-superficies, cumplen con determinadas características y propiedades esperadas por algunos métodos numéricos como homogeneidad y simetría. La facilidad de su implementación y la gran variedad de adaptaciones permite generar datos en geometrías complejas.

En cuanto a las líneas futura de desarrollo pretendemos lo siguiente:

El ancho de la corteza en conglomerados debe ser estudiado adecuadamente ya que pueden aparecer agujeros en la escena. Se propone el uso de técnicas estadísticas con esta finalidad.

Se deben aplicar técnicas, tales como, nivel de detalle, impostores gráficos, entre otras existentes, para disminuir más aún el costo computacional y lograr interacciones en tiempo real con las representaciones de los empaquetamientos de partículas.

En la visualización de datos esparcidos se deja abierta nuevas investigaciones sobre la selección adecuada de un radio de influencia que dependa de la densidad de la nube de puntos y del comportamiento de los valores en la visualización de datos esparcidos, así como su aplicación en modelos 3D.

Se pretende probar, usando técnicas estadística, la homogeneidad en nubes de puntos para volúmenes generadas con el algoritmo de rellenado de iso-superficies cuando se requiere aleatoriedad en los puntos.

Se pretende desarrollar, además, algoritmos para la aceleración por *hardware* que hagan un uso eficiente de la *GPU* para el proceso de visualización en general.

# Bibliografía


1. Brodlie, K.W., *Scientific Visualization: Techniques and Applications*, L.A.C. K.W. Brodlie, R.A. Earnshaw, J.R. Gallop, R.J. Hubbold, A.M. Mumford, C.D. Osland, P. Quarendon, Editor. 1992, Springer; 1 edition. p. 284.
2. Charles D. Hansen, C.R.J., *Visualization Handbook*. 2004: Academic Press, Inc. 1 edition. 984.
3. Richard S. Gallagher, S.P., *Computer Visualization: Graphics Techniques for Engineering and Scientific Analysis*. 1995: CRC-Press, 1 edition.
4. Jeschke, S., *Accelerating the Rendering Process Using Impostors*, 2005, Institute of Computer Graphics and Algorithms, Vienna, University of Technology.
5. Tomas Akenine-Moller, E.H., Naty Hoffman, *Real-Time Rendering*. 2008: AK Peters, Third Edition.
6. Daniel Cohen-Or, Y.L.C., Claudio T. Silva,Frodo Durand, *A Survey of Visibility for Walkthrough Applications.* IEEE Transactions on Visualization and Computer Graphics, 2003. 9: p. 412-431.
7. Assarsson, U., et al., *Optimized view frustum culling algorithms for bounding boxes.* J. Graph. Tools, 2000. 5(1): p. 9-22.
8. Kumar, S., et al., *Hierarchical back-face computation*, in *Proceedings of the eurographics workshop on Rendering techniques '96*1996, Springer-Verlag: Porto, Portugal. p. 235-ff.
9. Mel Slater, Y.C. *View volume culling using a probabilistic cashing scheme.* in *in Proceedings of Framework for Immersive Virtual Enviroments FIVE.* 1996.
10. Batagelo, H.C. and S.-T. Wu, *Dynamic Scene Occlusion Culling Using a Regular Grid*, in *Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing*2002, IEEE Computer Society. p. 43-50.
11. Georgios Papaioannou, A.G., Dimitrios Christopoulos, *Efficient Occlusion Culling using Solid Occluders.* Proceedings of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), 2008.
12. Pantazopoulos, I. and S. Tzafestas, *Occlusion Culling Algorithms: A Comprehensive Survey.* J. Intell. Robotics Syst., 2002. 35(2): p. 123-156.
13. Garland, P.S.H.a.M., *Multiresolution Modeling for Fast Rendering.* 1994: p. 43-50.
14. David Luebke, M.R., Jonathan D. Cohen, Amitabh Varshney, Benjamin Watson, Robert Huebner *Level of Detail for 3D Graphics*. 2002: Morgan Kaufmann; 1 edition.
15. Clark, J.H., *Hierarchical geometric models for visible-surface algorithms.* SIGGRAPH Comput. Graph., 1976. 10(2): p. 267-267.
16. Ortega, E.O., Eugenio; Idelsohn, Sergio, *A finite point method for adaptive three-dimensional compressible flow calculations.* International Journal for Numerical Methods in Fluids, 2009. 60(9): p. 937-971.
17. Alexa, M., et al., *Computing and Rendering Point Set Surfaces.* IEEE Transactions on Visualization and Computer Graphics, 2003. 9(1): p. 3-15.
18. Dey, T.K. and J. Sun, *An adaptive MLS surface for reconstruction with guarantees*, in *Proceedings of the third Eurographics symposium on Geometry processing*2005, Eurographics Association: Vienna, Austria. p. 43.
19. Fasshauer, G.F., *Meshfree Approximation Methods with MATLAB*. Vol. volume 6 of Interdisciplinary Mathematical Sciences. . 2007: World Scientific Publishing Co., Inc. .
20. Levin, D., *The approximation power of moving least-squares.* Math. Comput., 1998. 67(224): p. 1517-1531.
21. Bobach, T., Bertram, M., and Umlauf, G. *Comparison of Voronoi Based Scattered Data Interpolation Schemes.* . in *International Conference on Visualization, Imaging and Image Processing*. 2006.
22. Liu, G.R., *Mesh Free Methods: Moving Beyond the Finite Element Method*. 2003: CRC Press; 1 edition. 712.
23. Nielson, G.M., *Scattered Data Modeling.* IEEE Comput. Graph. Appl., 1993. 13(1): p. 60-70.
24. Park, S.W., et al., *Discrete Sibson Interpolation.* IEEE Transactions on Visualization and Computer Graphics, 2006. 12(2): p. 243-253.
25. Shepard, D., *A two-dimensional interpolation function for irregularly-spaced data*, in *Proceedings of the 1968 23rd ACM national conference*1968, ACM. p. 517-524.
26. Ken W. Brodlie, M.R.A., and Keith Unsworth. , *Constrained Visualization Using the Shepard Interpolation Family.* Computer Graphics Forum, 2005. 24(4): p. 809-820.
27. Nielson, R.F.a.G., *Smooth interpolation of large sets of scattered data.* International Journal for Numerical Methods in Engineering, 1980. 15: p. 1691-1704.
28. Sibson, R., *A vector identity for the Dirichlet tessellation.* Mathematical Proceedings of the Cambridge Philosophical Society, 1980. 87(01): p. 151-155.
29. Iske, A., *Multiresolution Methods in Scattered Data Modelling*. Lecture Notes in Computational Science and Engineering. Vol. 37. 2004. 182.
30. Floater, M.S. and A. Iske, *Multistep scattered data interpolation using compactly supported radial basis functions.* J. Comput. Appl. Math., 1996. 73(1-2): p. 65-78.

31. Wendland, H., *Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree.* Advances in Computational Mathematics, 1995. 4(1): p. 389-396.
32. Jackins, C.L.a.T., S.L., *Oct-trees and their use in representing three dimensional objects.* Computer Graphics and Image Processing, 1980. 14: p. 249-270.
33. Wilhelms, J. and A.V. Gelder, *Octrees for faster isosurface generation*, in *Proceedings of the 1990 workshop on Volume visualization*1990, ACM: San Diego, California, United States. p. 57-62.
34. A. Beygelzimer, S.K., and J. Langford. *Cover Trees for Nearest Neighbor*. in *23rd International Conference on Machine Learning*. 2006. Pittsburgh, PA.
35. Panigrahy, R., *Nearest Neighbor Search using Kd-trees.* Symposium A Quarterly Journal In Modern Foreign Literatures, 2006.
36. Anna Atramentov , S.M.L., *Efficient Nearest Neighbor Searching for Motion Planning.* 2002.
37. Arya, S., et al., *An optimal algorithm for approximate nearest neighbor searching fixed dimensions.* J. ACM, 1998. 45(6): p. 891-923.
38. Sagawa, R. *Effective nearest neighbor search for aligning and merging range images* in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003.* . 2003.
39. Lomet, D.B. and B. Salzberg, *The hB-tree: a multiattribute indexing method with good guaranteed performance.* ACM Trans. Database Syst., 1990. 15(4): p. 625-658.
40. Eppstein, M.B.a.D., *Mesh Generation and Optimal Triangulation*, in *Computing in Euclidean Geometry*, D.-Z.D.a.F. Hwang, Editor 1992, World Scientific: Singapore. p. 23-90.
41. Labelle, F., *Tetrahedral Mesh Generation with Good Dihedral Angles Using Point Lattices*, in *Computer Science*2007, University of California, Berkeley.
42. Yerry, M.A.a.S., Mark S, *Automatic three-dimensional mesh generation by the modified-octree technique.* International Journal for Numerical Methods in Engineering, 1984. 20: p. 1965--1990.
43. Fuchs, A. *Automatic Grid Generation with Almost regular Delaunay Tetrahedra*. in *7th International Meshing Roundtable*. 1998. Dearborn, Michigan, USA.
44. Naylor, D.J., *Filling space with tetrahedra.* International Journal for Numerical Methods in Engineering, 1999. 44(10): p. 1383-1395.
45. Baker, B.S., E. Grosse, and C.S. Rafferty, *Nonobtuse triangulation of polygons.* Discrete Comput. Geom., 1988. 3(2): p. 147-168.
46. Chew, L.P., *Guaranteed-Quality Triangular Meshes*, 1989.
47. Bern, M., D. Eppstein, and J. Gilbert, *Provably good mesh generation*, in *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*1990, IEEE Computer Society. p. 231-241 vol.1.
48. Li, X.-Y. and S.-H. Teng, *Generating well-shaped Delaunay meshed in 3D*, in *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*2001, Society for Industrial and Applied Mathematics: Washington, D.C., United States. p. 28-37.
49. Chew, L.P., *Guaranteed-quality Delaunay meshing in 3D (short version)*, in *Proceedings of the thirteenth annual symposium on Computational geometry*1997, ACM: Nice, France. p. 391-393.
50. Mitchell, S.A. and S.A. Vavasis, *Quality Mesh Generation in Higher Dimensions.* SIAM J. Comput., 2000. 29(4): p. 1334-1370.
51. Cheng, S.-W. and T.K. Dey, *Quality Meshing with Weighted Delaunay Refinement.* SIAM J. Comput., 2004. 33(1): p. 69-93.
52. Cheng, S.-W., et al., *Silver exudation.* J. ACM, 2000. 47(5): p. 883-904.
53. Labelle, F., *Sliver removal by lattice refinement*, in *Proceedings of the twenty-second annual symposium on Computational geometry*2006, ACM: Sedona, Arizona, USA. p. 347-356.
54. Francois Labelle, J.R.S., *Isosurface stuffing: fast tetrahedral meshes with good dihedral angles.* ACM Trans. Graph., 2007. 26(3): p. 57.
55. Robert, P.a.R., A. *Influence of the Shape of the Tetrahedron on the Accuracy of the Estimate of the Current Density*. in *ESA `START' Conference*. 1993. Aussois, France.
56. Burkardt, J. *Computational Geometry Lab: TETRAHEDRONS*. 2010.
57. Knoll, A., *A Survey of Octree Volume Rendering Methods.*
58. Timothy S. Newman. Ã. , H.Y., *A survey of the marching cubes algorithm.* Computers & Graphics, 2006. 30: p. 854-879.
59. Bank, R.E. and L.R. Scott, *On the conditioning of finite element equations with highly refined meshes.* SIAM J. Numer. Anal., 1989. 26(6): p. 1383-1394.
60. Shewchuk, J.R. *What is a Good Linear Element? - Interpolation, Conditioning, and Quality Measures* in *11th International Meshing Roundtable*. 2002.
61. Yu, J.W.a.Z., *Adaptive and Quality Tetrahedral Mesh. Generation.* 2011.

62. Freitag, L.A.a.C.O.-G. *A Comparison of Tetrahedral Mesh Improvement Techniques*. in *5th International Meshing Roundtable*. 1996. Sandia National Laboratories.
63. Dementiev, R., L. Kettner, and P. Sanders, *STXXL: standard template library for XXL data sets.* Softw. Pract. Exper., 2008. 38(6): p. 589-637.
64. Piet Stroeven, M.S.L., Johannes Sluys, Huan He, ZhanqiGuo, *Self-healing capacity of concrete-computer simulation study of unhydrated cement structure.* Image Analysis and Stereology, 2007. 26: p. 137-143.
65. Lorensen, W.E. and H.E. Cline, *Marching cubes: A high resolution 3D surface construction algorithm.* SIGGRAPH Comput. Graph., 1987. 21(4): p. 163-169.
66. Nielson, G.M. and B. Hamann, *The asymptotic decider: resolving the ambiguity in marching cubes*, in *Proceedings of the 2nd conference on Visualization '91*1991, IEEE Computer Society Press: San Diego, California. p. 83-91.
67. Thomas Lewiner, H.L., Antônio Wilson Vieira, Geovan Tavares, *Efficient implementation of marching cubes cases with topological guarantees.* Journal of Graphics Tools, 2003. 8(2): p. 1-15.
68. Hamilton, W.R., *On quaternions, or on a new system of imaginaries in algebra.* Philosophical Magazine, 1844. 25(3): p. 489-495.
69. Vince, J., *Geometric Algebra for Computer Graphics*. 1st Edition ed. 2008: Springer.
70. Hanson, A.J., *Visualizing Quaternions*. 2006, San Francisco: The Morgan Kaufmann Series in Interactive 3D Technology. 498.
71. Zhen Chen, R.B., *An Evaluation of the Material Point Method*, S.N. Laboratories, Editor 2002: Albuquerque, New Mexico, California. p. 1-40.
72. Monaghan, J.J., *An introduction to SPH.* Computer Physics Communications, , 1988. 48(1): p. 89-96.
73. Lucy, L.B., *A numerical approach to the testing of the fission hypothesis.* Astronomical Journal, 1977. 82: p. 1013-1024.
74. Gingold, R.A.M., J. J., *Smoothed particle hydrodynamics - Theory and application to non-spherical stars.* Monthly Notices of the Royal Astronomical Society, 1977. 181: p. 375-389.
75. D. He, N.E., L. Cai, *New statistic techniques for structure evaluation of particle packing.* Materials Science and Engineering A, 2001. 298(1-2): p. 209-215.
76. Reunanen, M., *Point-Based Modeling.* Seminar on Computer Graphics. Spring 2004.
77. Irvin Pérez Morales, R.R.V., Yordanis Pérez Brito, Harold Díaz-Guzmán, Carlos A. Recarey Morfa, *Procedure for packing generic particles for the discrete element method.* Rev. Int. Métod. Numér. Cálc. Diseño Ing., 2009. 25(2): p. 95-110.
78. Board, O.A.R., et al., *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2.1*. 2007: Addison-Wesley Professional. 928.