UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE

"MÉTODO RÁPIDO DE REPRESENTACIÓN DE CARACTERÍSTICAS FINAS EN UNA MALLA OCTREE"

IGNACIO ALBERTO FIGUEROA RAMÍREZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Claudio Lobos Yáñez
Profesor Correferente: Cristopher Arenas Fuentes

Diciembre - 2023

RESUMEN

Resumen—Este trabajo presenta una modificación a un generador de mallas de elementos mixtos con el objetivo de mejorar la representación de características finas de los dominios de entrada. Para lograr esto, se diseñó e implementó una heurística basada en el filtro de elementos según su calidad, la cual se introdujo en el algoritmo como un paso intermedio para evitar el reemplazo innecesario de hexaedros. Los resultados muestran una mejora en la representación de características finas manteniendo la calidad de las mallas generadas y sin afectar el tiempo de ejecución del algoritmo.

Palabras Clave— mallas de elementos mixtos; heurísticas; calidad

ABSTRACT

Abstract—This work presents a modification to a mixed-element mesh generator with the goal of improving the representation of sharp features of the input domains. To achieve this, a heuristic based on the filtering of elements according to their quality was designed and implemented, which was introduced in the algorithm as an intermediate step to avoid the unnecessary replacement of hexahedra. The results show an improvement in the representation of sharp features while keeping the quality of the generated meshes and without reducing the execution time of the algorithm.

Keywords— mixed element meshes; heuristics; quality

GLOSARIO

J_{ENS} = element normalized scaled Jacobian

J_S = scaled Jacobian

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

La generación de mallas computacionales para representar dominios del mundo real es un área de la computación gráfica que está en constante evolución. Estas mallas resultantes son posteriormente usadas en análisis o simulaciones que buscan replicar comportamientos de los dominios que pueden suceder en el mundo real. Hay distintas formas de categorizar una malla geométrica, como puede ser según su tipo de elemento, dimensión, algoritmo de generación, entre otras.

Un programa capaz de generar mallas funciona aplicando un algoritmo generador, y por lo general hay 2 criterios que se utilizan al momento de evaluar un generador de mallas geométricas: el grado de representación (precisión) y el tiempo de generación (rapidez). Estos criterios son opuestos entre sí, ya que generar una malla que represente en forma precisa cierto dominio de entrada puede requerir una gran cantidad de tiempo, lo que podría disminuir la utilidad práctica de la malla. Asimismo, una malla generada en muy poco tiempo pero que no represente adecuadamente el dominio de entrada, o posea elementos inválidos, no podría utilizarse.

El generador de mallas de elementos mixtos creado por Lobos y González [1], tiene como prioridad la velocidad de generación de las mallas, ya que está pensado para simulaciones biomédicas en tiempo real. Este generador utiliza como base el método Octree para refinar el dominio de entrada, por lo que genera mallas de tipo Octree, y fue diseñado con un énfasis en la representación de dominios curvos. Debido a esto, el generador de mallas no es capaz de representar adecuadamente los dominios rectos, los que son suavizados de manera excesiva.

El presente trabajo tiene como objetivo diseñar e implementar una heurística en el generador de elementos mixtos de Lobos y González que permita representar de mejor manera las características finas de los dominios de entrada, sin afectar el tiempo de generación de las mallas e intentando mantener la calidad.

En el primer capítulo se define formalmente el problema y se establece el objetivo principal y los objetivos secundarios. En el segundo capítulo se explican conceptos claves de las mallas geométricas y que son necesarios para entender el trabajo. En el tercer capítulo se presentan los fundamentos de la propuesta de solución, los algoritmos involucrados y el diseño e implementación de la propuesta en el código del generador. En el cuarto capítulo se describen las pruebas realizadas para evaluar la modificación implementada, y se compara con el algoritmo original con datos tanto cuantitativos como cualitativos. Por último, en el quinto capítulo se concluye acerca del trabajo realizado y se discuten los

resultados obtenidos, los alcances y limitaciones de la propuesta de solución y posibles trabajos a futuro que se pueden desarrollar para abordar las limitaciones.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1. Descripción

Las mallas geométricas son discretizaciones de dominios geométricos definidos por distintos polígonos. En particular, una malla volumétrica es aquella que a partir de un dominio superficial representa un volumen, el cual está compuesto de distintos cuerpos geométricos. Las mallas geométricas y la generación de éstas es una de las áreas más estudiadas de la computación gráfica, y son utilizadas en distintos ámbitos tales como ingeniería, física, geología y biomedicina.

Para el caso de la biomedicina, las mallas geométricas cobran gran relevancia gracias a su uso en simulaciones físicas, ya que es posible generar mallas volumétricas que representen partes del cuerpo humano como órganos, para luego someterlas a simulaciones, como por ejemplo lo que pasaría cuando se le aplica fuerza a un órgano durante una cirugía. Se requiere de mallas volumétricas generadas rápidamente, pues la medicina en muchos casos necesita simulaciones en tiempo real, lo cual presenta un desafío, ya que se debe realizar un balance óptimo entre qué tan precisa es la malla, el tiempo de generación de ésta y qué tan costosas son las simulaciones.

Una malla formada por una gran cantidad de figuras geométricas y bien definida es capaz de representar simulaciones físicas con un alto nivel de precisión, lo que implica un gran nivel de confianza al momento de saltar desde simulaciones a situaciones reales, muchas veces de gran impacto. Sin embargo, la generación y simulación en mallas es un proceso computacional costoso, por lo que una malla que demore un tiempo excesivo en ser generada; o que sus simulaciones tarden días ejecutarse, puede no ser del todo útil para el contexto que se necesita. Y es cuando se llega al límite en el poder de procesamiento de un computador que resulta indispensable diseñar algoritmos y técnicas más eficientes.

Este trabajo se basa en el generador de mallas desarrollado por Lobos y González [1], el cual se enfoca en simulaciones en tiempo real para aplicaciones biomédicas. Toma como entrada una superficie compuesta por polígonos triangulares y genera una malla de volumen compuesta por un conjunto de cuerpos, los cuales son hexaedros, prismas (cuñas), pirámides y tetraedros, por lo que la malla resultante es conocida como malla mixta. Las mallas poseen una buena representación de dominios curvos, lo que se logra aplicando una variedad de patrones de superficie previamente definidos. Al estar diseñado pensando en simulaciones en tiempo real, el generador tiene como objetivo principal producir mallas de buena calidad en poco tiempo.

Sin embargo, algunos dominios pueden ser extremadamente difíciles de representar dada la forma que poseen. Específicamente, aquellas regiones donde se forman ángulos rectos presentan un problema mayor, ya que no es posible aplicar los métodos de representación que se utilizan para superficies suaves. Esto conlleva a una representación poco precisa del dominio, lo que implica simulaciones poco realistas y por lo tanto, disminuye la utilidad de la malla generada. De la misma manera, solucionar esta problemática implicaría corregir una de las deficiencias del generador de mallas, lo que traería consigo representaciones más precisas y de mejor calidad.

Teniendo en cuenta lo anterior, es necesario diseñar e implementar una heurística para el generador de mallas, con el fin de extender su funcionalidad y que sea factible representar las características finas en superficies triangulares a través de la generación de elementos de calidad aceptable, manteniendo el tiempo de generación de mallas lo más acotado posible.

1.2. Objetivos

1.2.1 Objetivo general

- Generar mallas que sean capaces de representar características finas sin producir elementos inválidos, en un tiempo acotado.

1.2.2 Objetivos específicos

- Definir y detectar características finas en superficies triangulares.
- Diseñar un algoritmo rápido capaz de representar características finas en mallas geométricas.
- Modificar el código del generador de mallas actual con la propuesta de solución.
- Comparar los resultados de la propuesta de solución con el generador de mallas actual.

CAPÍTULO 2: MARCO CONCEPTUAL

2.1. Mallas geométricas

Las mallas geométricas son representaciones de dominios discretos formadas a partir de unidades primitivas, las que juntas forman una malla. Por lo general, se utilizan para simular situaciones en donde actúan fuerzas físicas, en las cuales la malla geométrica sufre distintos tipos de deformaciones, cambiando así su estado. Los generadores de mallas usan en su mayoría la CPU como unidad de procesamiento (incluyendo el generador de este trabajo), sin embargo en los últimos años ha habido avances para utilizar la GPU en los métodos de generación de mallas [2], los que buscan aprovechar la gran capacidad de procesamiento en paralelo que poseen.

Existen varios tipos de mallas geométricas según el criterio que se utilice, por ejemplo, si representan superficies o volúmenes (mallas superficiales, mallas volumétricas), si se utiliza un solo tipo de polígono (triangulares, cuadrilaterales), o en el caso de las mallas de volumen un solo tipo de cuerpo (hexaedros, tetraedros, pirámides). Una malla formada por una combinación de elementos se denomina malla de elementos mixtos.

Además, una malla en la que sus elementos primitivos tienen distinto tamaño se denomina malla irregular. Un ejemplo de este tipo se muestra en la Figura 1, el cual es un resultado del generador de mallas sobre el que se va a trabajar.

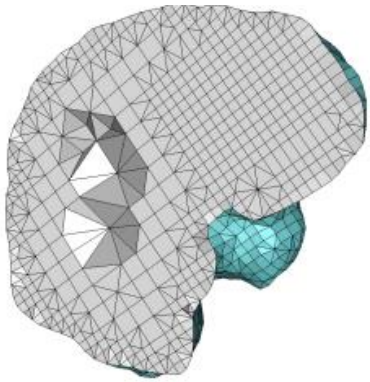


Figura 1: Malla de volumen formada por elementos irregulares [Lobos y González, 2015]

2.2. Método Octree

El método Octree es una técnica de refinamiento introducida por primera vez por Meagher en 1982 [3], en la cual el dominio se va dividiendo recursivamente en octantes, los cuales comparten una estructura similar a la de un árbol. La recursión va a depender del nivel de refinamiento deseado: un mayor nivel implica dividir una región en partes más pequeñas, aumentando la cantidad de elementos. El nivel de refinamiento es representado por la profundidad del árbol, y cada vértice de un octante se denomina nodo. El objetivo del método Octree es lograr la discretización de un cierto dominio, el que puede ser una malla superficial o un objeto geométrico. En la Figura 2 se observa un diagrama del proceso, en el que un octante se subdivide progresivamente.

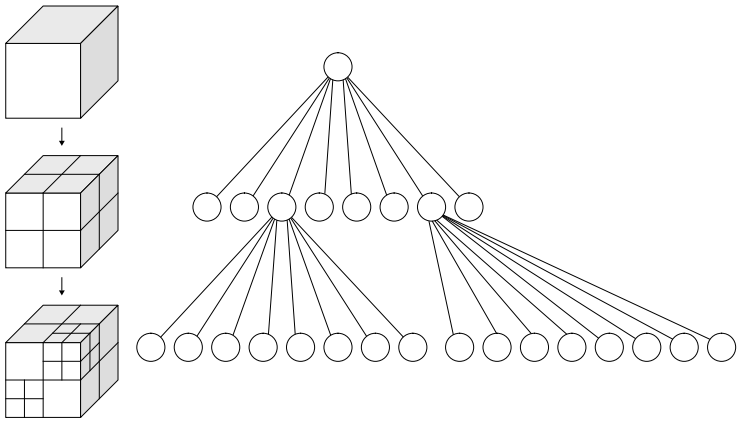


Figura 2: Malla geométrica refinada usando el método Octree con hexaedros, junto a su respectivo diagrama de árbol

Por lo general se utilizan hexaedros como los elementos de división, ya que poseen una buena cantidad de nodos y no generan tantos elementos como ocurriría si se utilizara otro elemento (tetraedro, pirámide). Los parámetros de control de la malla (como el tamaño de los octantes o el número de divisiones) van a depender de la implementación, y pueden estar basados en características como la topología del dominio, puntos de control definidos previamente, o atributos de la geometría de la malla [4]. Durante el proceso de subdivisión, los octantes que no encierran partes del dominio son eliminados, y es común agregar restricciones con respecto al tamaño de los octantes vecinos, por ejemplo en el valor máximo de la diferencia de sus niveles de refinamiento. Un valor máximo de 1 significa que octantes vecinos no pueden tener más de 1 nivel de refinamiento de diferencia, y en este caso la malla sería de tipo 1-regular.

El método Octree tiene varias ventajas, entre ellas el hecho de que es posible generar mallas sobre dominios irregulares con niveles de complejidad altos, ya que el nivel de refinamiento objetivo depende del usuario, lo que en teoría significa que es posible tener una malla altamente precisa con una gran cantidad de elementos. Asimismo, la estructura de árbol facilita el recorrido a través de los nodos de la malla Octree.

Por otro lado, con respecto a las debilidades del método, puede ser difícil controlar con precisión la cantidad de elementos que se generan, ya que para alcanzar altos niveles de refinamiento se requiere generar un número importante de elementos, lo que implica que en muchos casos las mallas resultantes poseen elementos innecesarios, creados debido a las reglas de refinamiento en el algoritmo. De la misma forma, un alto número de elementos significa un alto tiempo de generación, por lo que no es factible intentar generar una malla con niveles de refinamiento muy altos.

Considerando esto, es posible usar el método Octree como un primer paso para generar una malla inicial, a la que luego se le puede aplicar otro tipo de algoritmos hasta llegar al resultado que el usuario requiere, que es la forma en que opera el generador de mallas mixtas de Lobos. En la Figura 3 se observa un ejemplo de una malla geométrica inicial, la cual posee solo hexaedros y se usará como base para desarrollar una malla más compleja.

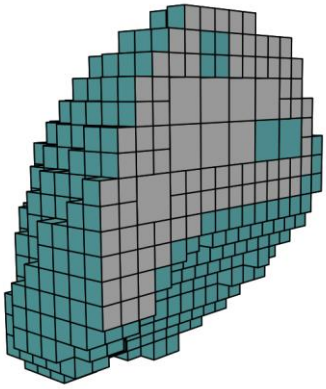


Figura 3: Malla resultante luego de aplicar el método Octree en el paso inicial [Lobos y González, 2015]

2.3. Generador de mallas mixtas

El generador de mallas mixtas sobre el que se va a trabajar en esta memoria fue introducido en la publicación de Claudio Lobos y Eugenio González titulada como "*Mixed-element Octree: a meshing technique toward fast and real-time simulations in biomedical applications*" en el año 2015, y está orientado a simulaciones rápidas en tiempo real en el área de la biomedicina, por lo que el objetivo es generar mallas que no tengan una cantidad excesiva de elementos en poco tiempo.

Para lograr este objetivo, el generador implementa varios algoritmos para ahorrar tiempo, como la posibilidad de definir regiones de interés (*RoI*) en el dominio de la entrada, en las que el nivel de refinamiento resultante será mayor que en otras regiones, como se muestran en la Figura 4. Esto está pensado para situaciones en las que no todas las regiones del dominio tienen la misma importancia, ya que permite al usuario priorizar regiones específicas que podrían ser de especial interés durante las simulaciones, y al mismo tiempo ahorrar en cuanto a costo computacional, pues el nivel de refinamiento será menor en regiones menos importantes.

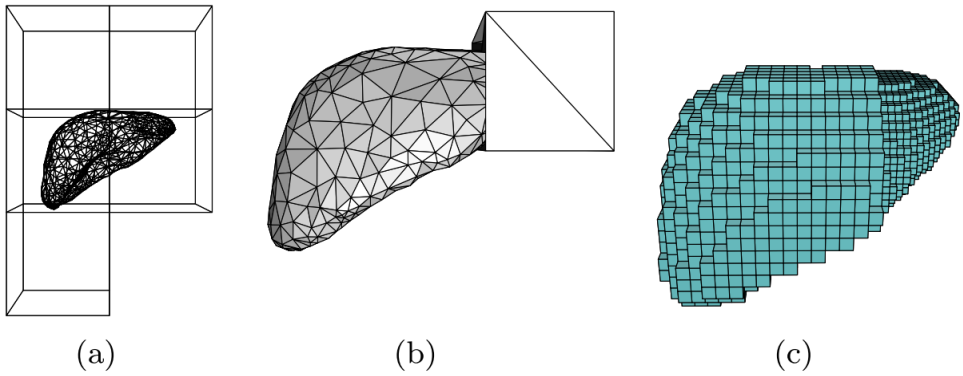


Figura 4: (a) Octantes iniciales en el dominio de entrada, (b) región de interés determinada por el cubo, (c) malla resultante después de aplicar el método Octree, donde la región de interés posee un mayor nivel de refinamiento que el resto de la malla [Lobos y González, 2015]

Con respecto a la calidad de los elementos generados, que corresponde a la métrica que evalúa qué tan bien está formado un elemento, se define una métrica para los elementos básicos basada en el Jacobiano escalado de un nodo. Para el caso de los hexaedros y tetraedros, la definición de un elemento de alta calidad está basada en la literatura actual, sin embargo para el resto de los elementos (pirámides y cuñas) fue necesario extender las métricas de los dos primeros elementos, y considerar que todos estos pueden coexistir en

una misma malla geométrica. Por lo mismo, el Jacobiano escalado se debe normalizar para cada uno de los elementos, y se establecieron categorías para estos según el valor de calidad: elementos inválidos (< 0), elementos de buena calidad ($0.2 <$) y elementos perfectos ($= 1$). En la Figura 5 se observan los elementos básicos que pueden existir en las mallas resultantes. Notar que estos poseen proporciones ideales, lo que no necesariamente ocurre en la práctica.

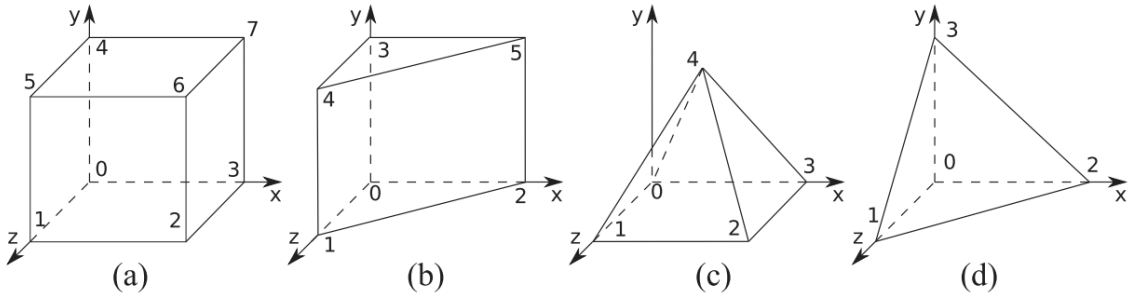


Figura 5: Elementos básicos (a) hexaedro, (b) cuña, (c) pirámide, (d) tetraedro [Lobos y González, 2015]

El generador parte con una malla superficial formada por triángulos, la cual corresponde al dominio de entrada, para luego continuar definiendo la raíz inicial del árbol Octree, que puede ser un solo hexaedro que cubre todo el dominio o un grupo de hexaedros pequeños, donde la elección va a depender de la topología de la malla de entrada. Después, se deben definir las regiones de interés con sus respectivos niveles de refinamiento deseados, además de un nivel de refinamiento global asociado al resto de las regiones. Los hexaedros que no posean elementos del dominio se descartan, mientras que aquellos triángulos de la malla que intercepten algún hexaedro se evalúan utilizando el algoritmo de *clipping Cohen-Sutherland* en su versión 3D. Los octantes que no son 1-regular se dividen rápidamente aprovechando la velocidad de recorrido de la estructura de árbol, quedando como resultado una malla de hexaedros 1-regular que representa una primera aproximación del dominio de entrada.

Posterior a esto, se aplican los patrones de transición a los elementos, lo que resulta en una malla topológicamente congruente. Es decir, existe una distinción definida entre los nodos internos y externos de los elementos. Es en este punto donde comienza la etapa para eliminar los elementos que estén fuera del borde, para lo cual primero se proyectan hacia el borde los nodos internos que se encuentran a una distancia pequeña de éste. Esta distancia es arbitraria, y se estableció como un valor menor o igual al 30% del valor promedio de las aristas incidentes a un nodo. Luego de la proyección, aquellos elementos que tengan todos sus nodos en el exterior del borde son eliminados.

El siguiente paso del algoritmo es aplicar los patrones de superficie a los hexaedros que intercepten la superficie del dominio, lo que significa que algunos de estos hexaedros serán reemplazados por los elementos básicos del generador ya descritos, los cuales son capaces de representar de forma más precisa los dominios curvos que los hexaedros. Al igual que en el paso anterior, aquellos elementos que reemplazaron a los hexaedros y que estén fuera del borde son eliminados.

Después, aquellos nodos que hayan quedado fuera del borde se encojen a través de una proyección hacia el borde del dominio, lo cual produce una mejor representación de las partes suaves de la superficie. El último paso del algoritmo consiste en disminuir el "efecto escalera" en algunas regiones de la malla resultante, lo cual ocurre cuando hay cambios bruscos en las normales de las caras, es decir características finas del dominio. Esto se logra introduciendo algunos elementos extra en estas regiones, que mejoren la transición entre las caras con normales muy distintas. En la Figura 6 se observan algunos de los estados de la malla geométrica durante el proceso completo, siendo el resultado final una malla de elementos mixtos.

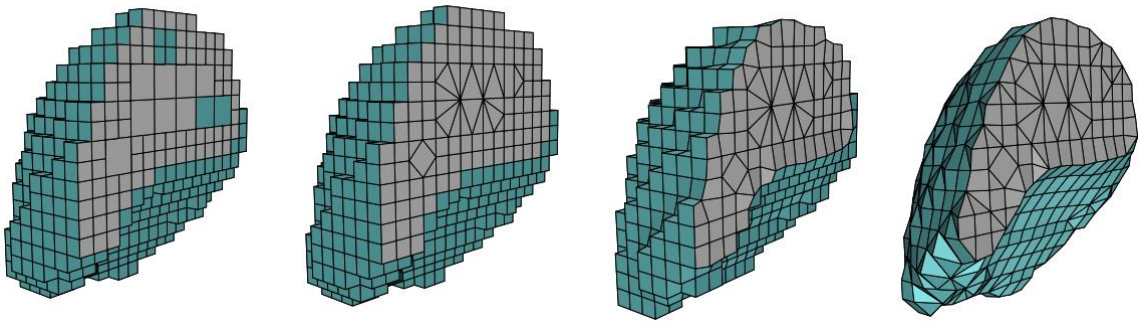


Figura 6: Etapas del generador de malla de izquierda a derecha: creación de la malla de hexaedros inicial; aplicación de patrones de transición; proyección de nodos internos hacia afuera; aplicación de patrones de superficie [Lobos y González, 2015]

2.4. Características finas

Las características finas en un dominio son aquellas regiones en las que suceden cambios bruscos, como por ejemplo en áreas donde se forman ángulos rectos. En el caso específico de este generador de mallas mixtas, al estar diseñado para representar dominios con características suaves, no es posible representar de forma correcta estas características.

Maréchal [5] se refiere a las secciones finas en una malla como un cierto tipo específico de arcos que denomina *crestas*¸ y éstas aparecen cuando el ángulo que se forma entre las dos caras de figuras vecinas es mayor a cierto valor, el cual es definido por el usuario. Entonces, el concepto clave para determinar si es que una sección es o no una característica fina es el ángulo que se forma entre las figuras.

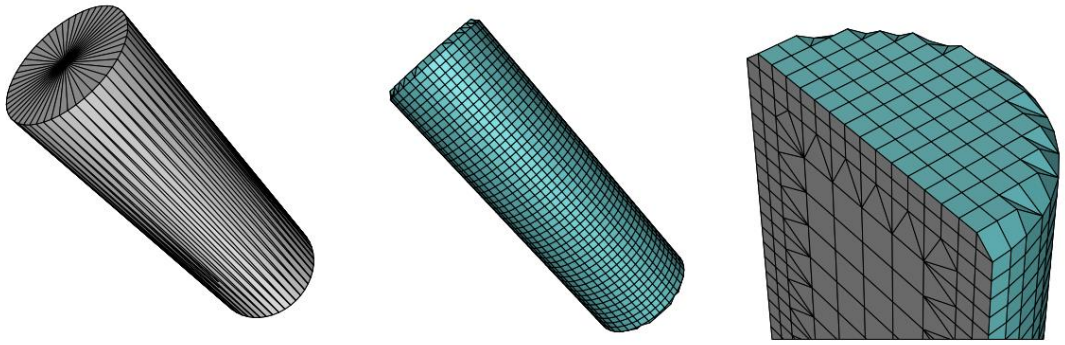


Figura 7: Problema de representación de características finas en una malla de elementos mixtos [Lobos y González, 2015]

Esta problemática fue abordada en la tesis de Cristopher Arenas titulada “Maximizando el uso de hexaedros de buena calidad para representar características finas en mallas de volumen tipo Octree” [6], la cual como su nombre indica planteó la hipótesis de que era posible utilizar hexaedros para poder representar de forma adecuada los cambios bruscos en las regiones del dominio.

Se propuso un conjunto de patrones característicos capaces de representar características finas utilizando elementos distintos al hexaedro (cuña, pirámide y tetraedro), los cuales se pueden ver en la Figura 8. Esta modificación en el algoritmo implica que la malla generada va a estar sujeta a la aplicación de patrones de superficie (como es originalmente) y patrones característicos, los cuales van a ser aplicados en regiones con características finas.

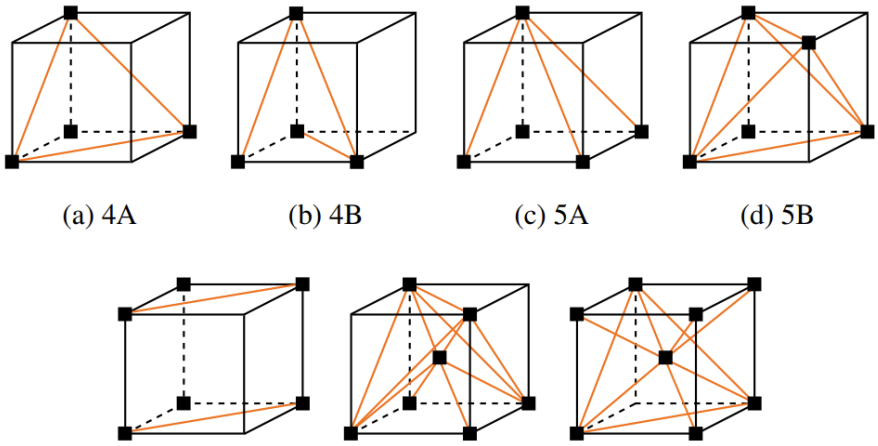


Figura 8: Patrones característicos propuestos por Arenas [6]

La aplicación de los patrones característicos considera un cálculo en la calidad de la malla resultante, dando como resultado mallas de calidad superior a las del algoritmo original, lo que significa que efectivamente hubo una mejora en cuanto a la representación de características finas.

Sin embargo, la gran desventaja de esta propuesta es el alto costo de ejecución del nuevo algoritmo, el cual necesita "al menos 4 veces más tiempo" que el algoritmo original. El autor señala que el motivo de esto es que cada vez que se evalúa si proyectar un nodo o no, se debe computar el costo de la proyección, y dado que mayores niveles de refinamiento implican un aumento considerable en el número de nodos, la diferencia en cuanto a tiempo según el nivel de refinamiento puede alcanzar valores excesivamente altos.

2.5. Métrica para medir la calidad de los elementos

No todos los elementos resultantes de la malla de volumen son iguales, por lo que es imprescindible utilizar algún método para poder medir la calidad de estos e introducir modificaciones al algoritmo generador si la calidad de algún elemento lo amerita, así como también comparar las mallas generadas. Evidentemente, una malla con elementos de mala calidad implicaría simulaciones con resultados erróneos.

Lobos y González [1] proponen un rango de valores estándar para los 4 elementos fundamentales del generador de mallas, métrica que se denominó como *element normalized scaled Jacobian* (J_{ENS} de ahora en adelante) y que se basa en el *scaled Jacobian*

(J_S), una métrica común en la literatura para medir la calidad de un hexaedro. Este método utiliza el determinante de la matriz formada por los vectores normalizados resultantes entre un nodo y sus 3 nodos adyacentes, cálculo que también puede escribirse matemáticamente como una serie de productos punto y cruz:

$$J_S^{\mathrm{p}} = \hat{v}_{\mathrm{pa}} \cdot \left(\hat{v}_{\mathrm{pc}} \times \hat{v}_{\mathrm{pb}}\right) = \hat{v}_{\mathrm{pb}} \cdot \left(\hat{v}_{\mathrm{pa}} \times \hat{v}_{\mathrm{pc}}\right) = \hat{v}_{\mathrm{pc}} \cdot \left(\hat{v}_{\mathrm{pb}} \times \hat{v}_{\mathrm{pa}}\right)$$

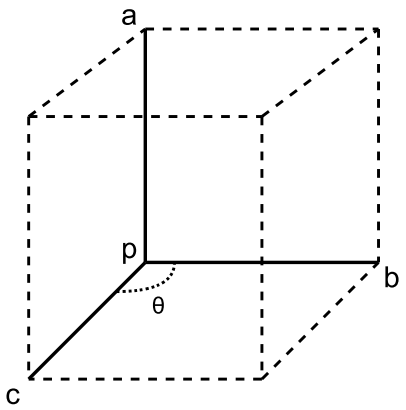


Figura 9: Las aristas dibujadas representan los nodos utilizados en el cálculo de $\boldsymbol{J_S^{\mathrm{p}}}$, notar que el producto cruz implica que el máximo valor se alcanza cuando el ángulo es 90°.

Como los vectores son normalizados, este método puede usarse sin importar el tamaño del hexaedro. El máximo valor de J_S^{p} es 1 y se debe realizar para cada uno de los nodos que forman el elemento. Finalmente, el valor J_S del elemento corresponde al menor valor entre los J_S^{x} de cada punto.

Para los elementos restantes (cuña, pirámide y tetraedro), es posible aplicar una lógica similar si se definen rigurosamente las configuraciones ideales de cada uno de estos elementos, donde el máximo valor también es 1. Este método es el denominado valor J_{ENS}, el cual según su rango puede indicar la existencia elementos inválidos (< 0), elementos de buena calidad ($0.2 <$) o elementos perfectos ($= 1$).

CAPÍTULO 3: PROPUESTA DE SOLUCION

3.1. Fundamentos de la propuesta

La esencia de la propuesta de solución consiste en determinar la calidad individual de cada elemento en la superficie de la malla utilizando la métrica de calidad J_{ENS}, y en base a este valor aplicar los patrones de superficie. Originalmente, el algoritmo aplicaba patrones de superficie a cada elemento en la superficie de la malla de volumen como paso intermedio, lo cual si bien ayudaba a representar las características suaves del dominio, traía consigo una pérdida de la representación de ángulos rectos. Entonces, se implementó una modificación en el algoritmo que consiste en aplicar los patrones de superficie solo a aquellos elementos que estén debajo de cierto umbral de calidad.

Para poder diseñar una solución, es necesario en primer lugar analizar el algoritmo y específicamente, el paso donde se aplican los patrones de superficie, ya que aquí es donde se pierde la mayor cantidad de características finas de las mallas de volumen. La razón de esta pérdida se encuentra en el código mismo del generador de mallas, debido a que el algoritmo actual está diseñado para aplicar los patrones de superficie a todos los elementos superficiales sin distinción alguna.

Entonces, una solución lógica a este problema es diseñar un filtro para que las transformaciones se apliquen en forma selectiva. Adicionalmente, teniendo en cuenta que uno de los objetivos de esta memoria es asegurar la calidad de los elementos, se propone utilizar la métrica de calidad J_{ENS} descrita en la sección anterior, resultando como base el siguiente filtro “si cierto elemento está por encima del umbral de calidad establecido, no aplicar patrones de superficie”. En pocas palabras, lo que se busca es evitar aplicar patrones de superficie a elementos de buena calidad.

3.2. Estado actual del generador de mallas

Originalmente, el generador de mallas consta de ciertos pasos que se aplican en forma secuencial hasta llegar a la malla de volumen final. Esta propuesta de solución considera una modificación en el orden de ciertos pasos y la implementación de un paso adicional a la secuencia, antes de aplicar los patrones de superficie. A continuación, se muestra un diagrama del algoritmo actual y el algoritmo propuesto. Este nuevo algoritmo busca evitar

la eliminación de hexaedros de buena calidad que no transgreden las restricciones del dominio.

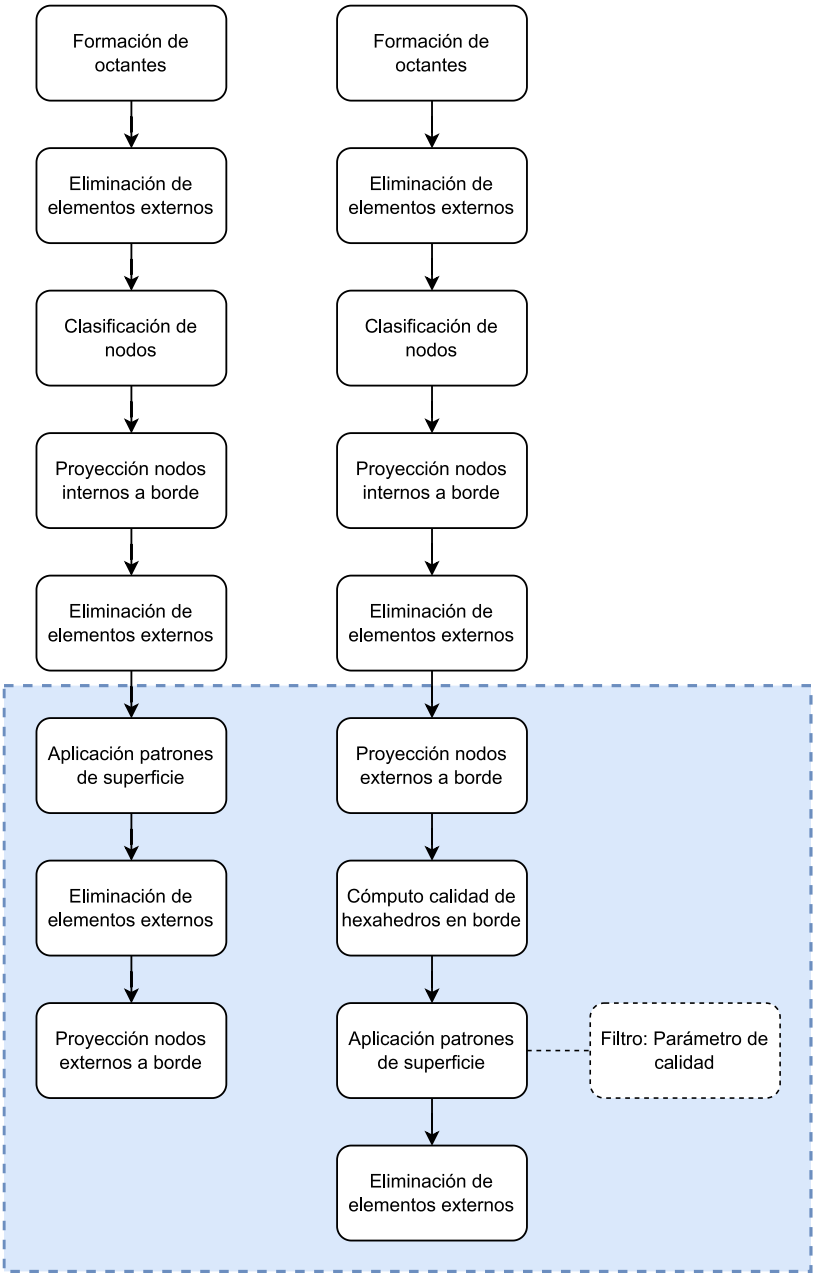


Figura 10: Algoritmo actual (izquierda) y algoritmo propuesto (derecha), en el recuadro azul se encuentran los pasos modificados. (Fuente: Elaboración Propia)

Como se observa, en el algoritmo propuesto la proyección de nodos externos al borde de la malla sigue inmediatamente después de la proyección de nodos internos, y se agregó el cómputo de la calidad de hexaedros superficiales, valores que se usarán como filtro al momento de aplicar los patrones de superficie.

3.3. Código J_{ENS} para medir la calidad de los elementos de una malla

El código J_{ENS} fue proporcionado como un programa aparte por el profesor Claudio Lobos, y funciona tomando como entrada un archivo que representa una malla geométrica, para luego retornar información y estadísticas acerca de la calidad de los elementos que la componen. De este código se extrajo y modificó un fragmento que entrega el valor J_{ENS} de un solo elemento, lo que hizo posible que se pudieran analizar elementos directamente dentro del generador de mallas, así como también trabajar con los valores retornados. El algoritmo usado se muestra a continuación:

Algoritmo 1: Cómputo J_{ENS} de hexaedros externos

```
Input: VE vector de elementos
for each E in VE do
    resultadosJENS ← calcularJENS(E)
    for each p in resultadosJENS do
        if resultadosJENS(p) < tol then
            p.qualityTol ← false
        end if
    end for
end for
```

Para esta propuesta de solución, el vector de elementos se encuentra compuesto en su totalidad por hexaedros, y a cada uno de los nodos p del elemento se le calcula su respectivo J_S^p, valor que es comparado con el parámetro de tolerancia predefinido por el usuario. Si este es inferior, el atributo booleano del punto se cambia a un valor falso. Este proceso se aplica a todos los elementos del vector.

3.4. Estructuras utilizadas y algoritmo

La propuesta de solución utiliza distintas estructuras, algunas que forman parte del programa original y otras que se agregaron como parte de la implementación. A continuación, se presenta un listado de aquellas consideradas fundamentales, así como también una explicación con la utilidad de cada una.

- **Octante (*Octant*)**: Clase original del generador de mallas usada para representar los octantes generados durante el proceso. Estos poseen información de los elementos que pertenecen al octante, y también si el octante es externo o no, lo cual es

importante para la propuesta ya que solo se busca modificar los elementos que se encuentran en la superficie de la malla geométrica.

- **Elemento (*Element*)**: Clase usada por el programa J_{ENS} e integrada en la propuesta, representa uno de los 4 elementos del generador de mallas. Posee información ordenada de la composición y posición de los puntos, y es el *input* para calcular el valor J_{ENS}.
- **Punto (*Point*)**: Clase original del generador de mallas usada para representar los distintos puntos que componen la malla geométrica. A esta clase se le agregó un simple booleano llamado *qualityTol*, el cual posee un valor verdadero por defecto y servirá como un filtro rápido a la hora de aplicar o no los patrones de superficie.
- **Tolerancia (*sq_tol*)**: Parámetro que establece el valor mínimo de calidad J_{ENS} que debe tener un elemento para que los patrones de superficie no se apliquen. Por ejemplo, un valor igual 1 significa que solo los elementos perfectos van a ser omitidos.

A grandes rasgos, el algoritmo de la propuesta de solución se ve de la siguiente forma, donde el núcleo del trabajo es el filtro que se aplica. Destacar que, si bien se evalúa cada elemento de forma individual, la decisión de omitir se aplica al octante en su totalidad, lo que implica que todos los elementos del octante deben cumplir con la restricción de calidad.

Algoritmo 2: Método rápido de representación de características finas

Input: SM malla de superficie, q parámetro de calidad
Output: VM malla de volumen

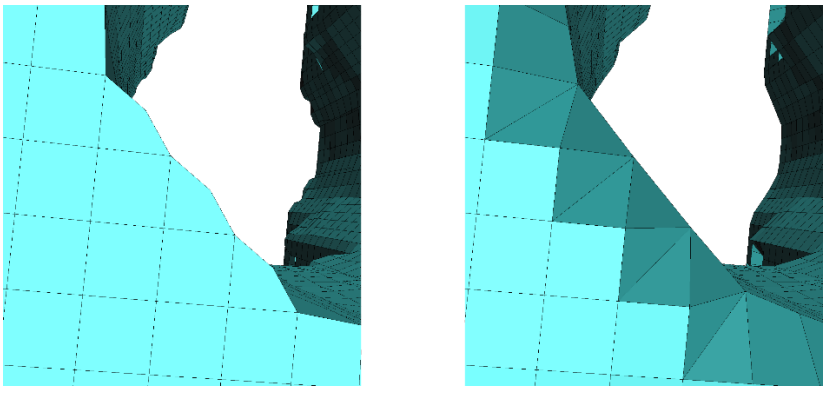
```
1  Octantes ← generarOctantes(SM)
2  Octantes.removerOctantesExternos
3  clasificarNodos(SM)
4  proyectarNodosInternos(SM)
5  eliminarElementosExternos(SM)
6  proyectarNodosExternos(SM)
7  calcularCalidadHexaedrosSuperficiales()
8  for each Elemento in Octantes do
9      if Elemento.calidad ≥ q then
10         continue
11     else
12         aplicarPatronesSuperficie()
           break
13     end if
14 end for
15 eliminarElementosExternos(SM)
16 VM ← guardarMallaResultante(SM)
```

3.5. Implementación de la propuesta de solución

Para implementar esta propuesta de solución fue necesario un estudio exhaustivo y detallado del programa original, pues al ser tan complejo se debe tener cuidado de no introducir errores o efectos no deseados. Por lo tanto, la implementación se realizó en forma modular y ordenada. La propuesta considera un parámetro de entrada -t (flotante), el cual es el valor de tolerancia mínimo que debe tener J_{ENS} para que un elemento sea omitido.

Luego, se introdujo a la secuencia de pasos un método nuevo (*elementsJENS*), donde se preprocesan los elementos que van a ser usados en el cómputo de los valores J_{ENS}, y se guardan en un vector de elementos. En este método también se guardan los puntos ordenados en un vector de puntos, los que necesariamente tienen que estar ordenados para no perder la consistencia topológica entre puntos y elementos. Ambos vectores juntos con el parámetro de tolerancia son los valores *input* para el método (*allJENS_sharp_points*) que calcula los valores J_{ENS} de los hexaedros en la superficie, y también modifica el atributo booleano de los puntos en cuestión si corresponde.

Por último, se intervino el método *applySurfacePatters* con la evaluación del booleano para cada elemento en los octantes superficiales. El algoritmo propuesto compara el valor del booleano en cada uno de los elementos dentro del octante, y posee un criterio conservador, ya que basta con el octante posea un solo elemento con su valor booleano falso para que los patrones de superficie se apliquen a todos los elementos del octante, como ocurre en el algoritmo original. Un ejemplo real de este caso se puede ver en la Figura 11.



(a) (b)

Figura 11: Malla geométrica antes de ser modificada por los patrones de superficie (a) y después (b). La presencia de hexaedros de mala calidad en el borde tiene como consecuencia la aplicación de patrones de superficie a todos los elementos del octante, independiente de la calidad y produciendo una peor representación del borde.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

4.1. Pruebas realizadas y ambiente

Para evaluar el algoritmo propuesto, se realizaron comparaciones tanto cuantitativas como cualitativas con el algoritmo original. Se utilizaron 4 dominios de entrada que presentan características más bien distintas entre sí, los cuales se muestran en la Figura 12 junto a sus composiciones en la Tabla 1.

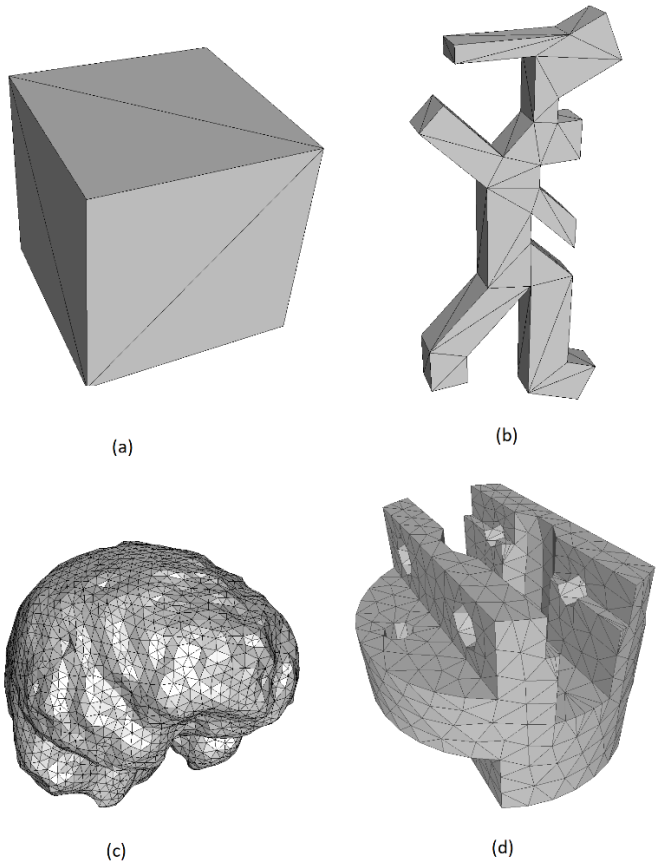


Figura 12: Mallas de entrada usadas en las pruebas, de izquierda-derecha arriba-abajo: (a) *hex*; (b) *fiducials1*; (c) *cortex*; (d) *socket* [7]

Tabla 1: Cantidad de nodos y triángulos en los dominios de entrada.

Figural	Nodos	Triángulos
Hex	8	12
Fiducials1	64	124
Cortex	3152	6300
Socket	836	1696

Las pruebas realizadas consideran 4 niveles distintos de refinamiento para cada figura (3, 4, 5, 6) y 3 valores distintos para el parámetro q para la propuesta de solución (0.2, 0.5, 0.8), dando un total de 16 pruebas distintas para cada figura tomando en cuenta el algoritmo original. Por simplicidad, se referirá en los resultados al algoritmo original como *Algoritmo 1* y a la propuesta de solución como *Algoritmo 2*. Además de medir el tiempo de cada ejecución, se tomaron datos estadísticos de las mallas de volumen resultantes, tanto de la cantidad de elementos como la calidad de cada uno de ellos (J_{ENS}). En específico, las métricas que se consideraron como parte del análisis cuantitativo son:

1. Tiempo de ejecución en segundos [s]
2. Cantidad de elementos en cada malla y porcentaje de hexaedros del total
3. Calidad del peor elemento [-1, 1]
4. Calidad promedio de los elementos de la malla [-1, 1]

Por otro lado, el análisis cualitativo se va a centrar en comparar la representación visual de ángulos rectos en las mallas generadas por el algoritmo original y el algoritmo propuesto, para un valor $q = 0.5$. Las pruebas se realizaron en un notebook con las siguientes características:

- Sistema Operativo: Ubuntu 23.04
- Procesador: 11th Gen Intel® Core™ i7-11800H (4.60 GHz)
- Memoria RAM: 31,1 GiB de RAM

4.2. Resultados cuantitativos

4.2.1 Tiempo de ejecución

Tabla 2: Tiempo de ejecución de cada algoritmo según el nivel de refinamiento en segundos.

		Tiempo [s]			
	Algoritmo	Ref. 3	Ref. 4	Ref. 5	Ref. 6
hex	Alg. 1	0.024	0.098	0.488	2.342
	Alg. 2 (q = 0.2)	0.025	0.099	0.486	2.410
	Alg. 2 (q = 0.5)	0.020	0.099	0.486	2.420
	Alg. 2 (q = 0.8)	0.031	0.101	0.491	2.419
fiducials1	Alg. 1	0.674	3.593	15.974	69.242
	Alg. 2 (q = 0.2)	0.701	3.612	16.311	69.049
	Alg. 2 (q = 0.5)	0.701	3.610	16.304	70.047
	Alg. 2 (q = 0.8)	0.704	3.568	16.057	69.165
cortex	Alg. 1	0.293	0.588	1.626	5.690
	Alg. 2 (q = 0.2)	0.278	0.587	1.621	5.868
	Alg. 2 (q = 0.5)	0.279	0.578	1.623	5.844
	Alg. 2 (q = 0.8)	0.280	0.585	1.621	5.788
socket	Alg. 1	0.124	0.342	1.305	5.542
	Alg. 2 (q = 0.2)	0.119	0.351	1.319	5.697
	Alg. 2 (q = 0.5)	0.130	0.353	1.322	5.594
	Alg. 2 (q = 0.8)	0.120	0.359	1.304	5.662

Como se observa en la Tabla 2, los tiempos de ejecución más altos fueron por lejos los de la figura *fiducials1*, con un tiempo de ejecución superior al minuto y con un valor 10 veces mayor al de la segunda figura que más tiempo tardó (*cortex*). Por otro lado, la figura con el menor tiempo de ejecución fue *hex*.

El tiempo de ejecución del algoritmo aumenta en mayor medida según el nivel de refinamiento que se utilice y la escala de la figura, más que de la cantidad de nodos y caras del dominio de entrada, lo que explica el por qué *fiducials1* demoró más que *cortex*, a pesar de que esta última figura posee una cantidad superior de nodos y caras.

Tabla 3: Diferencia entre el tiempo de ejecución del algoritmo propuesto y el algoritmo original en segundos.

		Diferencia con Alg. 1 [s]			
	Algoritmo	Ref. 3	Ref. 4	Ref. 5	Ref. 6
hex	q = 0.2	0.001	0.001	-0.002	0.068
	q = 0.5	-0.004	0.001	-0.002	0.078
	q = 0.8	0.007	0.003	0.003	0.077
fiducials1	q = 0.2	0.027	0.019	0.337	-0.193
	q = 0.5	0.027	0.017	0.330	0.805
	q = 0.8	0.030	-0.025	0.083	-0.077
cortex	q = 0.2	-0.015	-0.001	-0.005	0.178
	q = 0.5	-0.014	-0.010	-0.003	0.154
	q = 0.8	-0.013	-0.003	-0.005	0.098
socket	q = 0.2	-0.005	0.009	0.014	0.155
	q = 0.5	0.006	0.011	0.017	0.052
	q = 0.8	-0.004	0.017	-0.001	0.120

Comparando los tiempos de ejecución del algoritmo original y el algoritmo propuesto en la Tabla 3, se observa que la diferencia significativa más alta entre ambos es inferior a 1 segundo (0.8), equivalente a aproximadamente un 1% más lento que el algoritmo original. En la mayoría de los casos, la diferencia es inferior a la décima de segundo.

4.2.2 Cantidad de elementos

Tabla 4: Cantidad de elementos totales en cada malla generada y porcentaje de hexaedros respecto al total de elementos (la cantidad desglosada de cada figura se encuentra en el Anexo)

		Cantidad de elementos / Porcentaje de hexaedros							
	Algoritmo	Ref. 3		Ref. 4		Ref. 5		Ref. 6	
hex	Alg. 1	640	58%	4144	53%	22144	47%	103696	44%
	Alg. 2 (q = 0.2)	640	70%	4144	57%	22144	49%	103696	44%
	Alg. 2 (q = 0.5)	640	70%	4144	57%	22144	49%	103696	44%
	Alg. 2 (q = 0.8)	640	70%	4144	57%	22144	49%	103696	44%
fiducials1	Alg. 1	21855	37%	118296	34%	557412	33%	2441999	32%
	Alg. 2 (q = 0.2)	20771	47%	114146	40%	542337	37%	2373263	36%
	Alg. 2 (q = 0.5)	20775	47%	114150	40%	542425	37%	2373349	36%
	Alg. 2 (q = 0.8)	21529	41%	117516	36%	552801	35%	2421923	33%
cortex	Alg. 1	798	11%	5111	9%	26209	9%	119910	10%
	Alg. 2 (q = 0.2)	633	30%	4485	17%	23825	15%	111248	15%
	Alg. 2 (q = 0.5)	752	15%	4895	12%	25414	12%	116944	12%
	Alg. 2 (q = 0.8)	792	11%	5077	9%	26182	9%	119843	10%
socket	Alg. 1	500	15%	4347	24%	25510	28%	134496	26%
	Alg. 2 (q = 0.2)	332	80%	3928	44%	23955	39%	129858	31%
	Alg. 2 (q = 0.5)	332	78%	3992	39%	24118	38%	130222	31%
	Alg. 2 (q = 0.8)	479	32%	4194	32%	25215	31%	133640	27%

La diferencia entre la cantidad de elementos generados por cada algoritmo va a depender de la configuración del dominio de entrada. Analizando la Tabla 4, en el caso de *hex* el algoritmo original y el algoritmo propuesto general la misma cantidad de elementos para todos los parámetros de calidad y niveles de refinamiento, mientras que en el resto de las figuras se observa un valor distinto para todos los niveles de calidad. En general, la cantidad de elementos en los algoritmos tiende a un valor similar entre ellos a medida que aumenta el nivel de refinamiento, y se observa que un valor de calidad menor en el algoritmo propuesto se traduce en una menor cantidad de elementos generados.

Con respecto al porcentaje de hexaedros en las mallas generadas, este es superior en la gran mayoría de los casos en el algoritmo propuesto comparado con el algoritmo original, con una proporción mayor mientras menor es el parámetro de calidad. Se observa que la proporción de hexaedros se acerca a los valores del algoritmo original a medida que aumenta el nivel de refinamiento.

4.2.3 Calidad del peor elemento

Tabla 5: Calidad del peor elemento en las mallas generadas

		Calidad del peor elemento			
	Algoritmo	Ref. 3	Ref. 4	Ref. 5	Ref. 6
hex	Alg. 1	0.3651	0.3000	0.3000	0.3000
	Alg. 2 (q = 0.2)	0.3651	0.3000	0.3000	0.3000
	Alg. 2 (q = 0.5)	0.3651	0.3000	0.3000	0.3000
	Alg. 2 (q = 0.8)	0.3651	0.3000	0.3000	0.3000
fiducials1	Alg. 1	0.2272	0.1035	0.1815	0.1544
	Alg. 2 (q = 0.2)	0.2117	0.1035	0.1815	0.1544
	Alg. 2 (q = 0.5)	0.2272	0.1035	0.1815	0.1544
	Alg. 2 (q = 0.8)	0.2272	0.1035	0.1815	0.1544
cortex	Alg. 1	0.1403	0.1721	0.1379	0.1105
	Alg. 2 (q = 0.2)	0.2023	0.1721	0.1379	0.1105
	Alg. 2 (q = 0.5)	0.1403	0.1721	0.1379	0.1105
	Alg. 2 (q = 0.8)	0.1403	0.1721	0.1379	0.1105
socket	Alg. 1	0.3274	0.1582	0.1619	0.0295
	Alg. 2 (q = 0.2)	0.2280	0.1582	0.1619	0.0295
	Alg. 2 (q = 0.5)	0.3548	0.1582	0.1619	0.0295
	Alg. 2 (q = 0.8)	0.3274	0.1582	0.1619	0.0295

Como muestra la Tabla 5, tanto el algoritmo original como el algoritmo propuesto obtuvieron un peor elemento de igual calidad para todos los parámetros de calidad en todos los niveles de refinamiento, a excepción del nivel 3, donde se observa que el algoritmo propuesto con parámetro q = 0.2 obtuvo un elemento de peor calidad inferior para las figuras *fiducials1* y *socket*, y un peor elemento de calidad superior para la figura *cortex*.

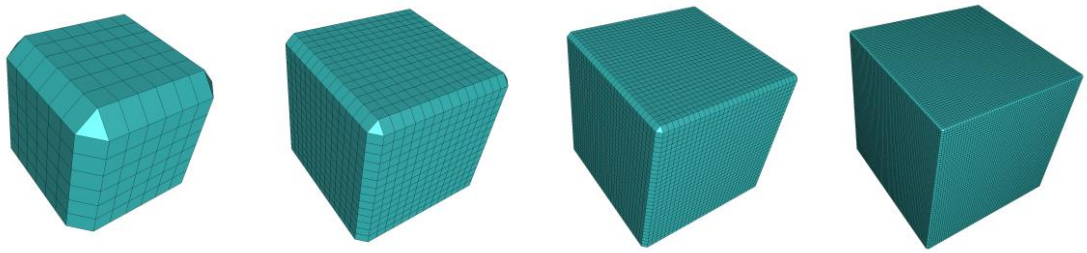
4.2.4 Calidad promedio de los elementos

Tabla 6: Calidad promedio de todos los elementos en las mallas generadas (la calidad promedio por elemento se encuentra en el Anexo)

		Calidad promedio			
	Algoritmo	Ref. 3	Ref. 4	Ref. 5	Ref. 6
hex	Alg. 1	0.8163	0.7682	0.7292	0.7076
	Alg. 2 (q = 0.2)	0.8406	0.7762	0.7323	0.7089
	Alg. 2 (q = 0.5)	0.8406	0.7762	0.7323	0.7089
	Alg. 2 (q = 0.8)	0.8406	0.7762	0.7323	0.7089
fiducials1	Alg. 1	0.7018	0.6803	0.6747	0.6675
	Alg. 2 (q = 0.2)	0.7303	0.6936	0.6821	0.6749
	Alg. 2 (q = 0.5)	0.7308	0.6936	0.6820	0.6749
	Alg. 2 (q = 0.8)	0.7137	0.6848	0.6784	0.6709
cortex	Alg. 1	0.5790	0.5555	0.5535	0.5526
	Alg. 2 (q = 0.2)	0.6160	0.5830	0.5766	0.5732
	Alg. 2 (q = 0.5)	0.5951	0.5658	0.5611	0.5582
	Alg. 2 (q = 0.8)	0.5808	0.5568	0.5536	0.5526
socket	Alg. 1	0.6252	0.6324	0.6591	0.6361
	Alg. 2 (q = 0.2)	0.8492	0.6935	0.6877	0.6486
	Alg. 2 (q = 0.5)	0.8559	0.6755	0.6849	0.6474
	Alg. 2 (q = 0.8)	0.6722	0.6540	0.6668	0.6398

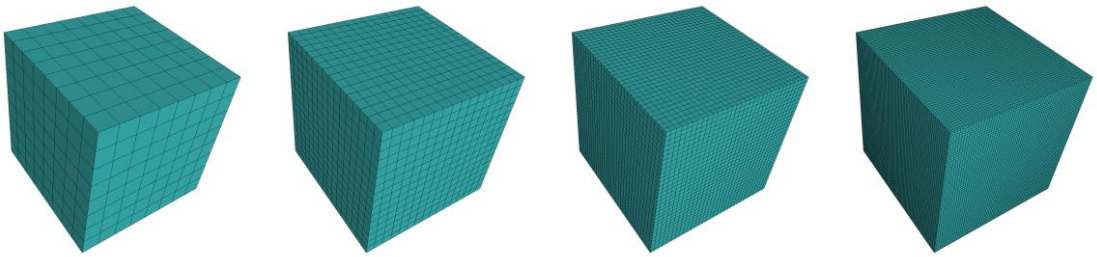
En cuanto a la calidad promedio de los elementos generados, en todos los casos que se encuentran en la Tabla 6 se observa un valor superior o igual del algoritmo propuesto con respecto al algoritmo original, donde la mayor diferencia se produce en la figura *socket* en los primeros niveles de refinamientos. En general, los algoritmos con parámetros q = 0.2 y q = 0.5 obtienen los mejores valores promedios, mientras que el algoritmo con parámetro q = 0.8 obtuvo valores más cercanos al algoritmo original. El valor promedio tiende a acercarse al valor del algoritmo original a medida que aumenta el nivel de refinamiento.

4.3. Resultados cualitativos



Ref. 3 Ref. 4 Ref. 5 Ref. 6

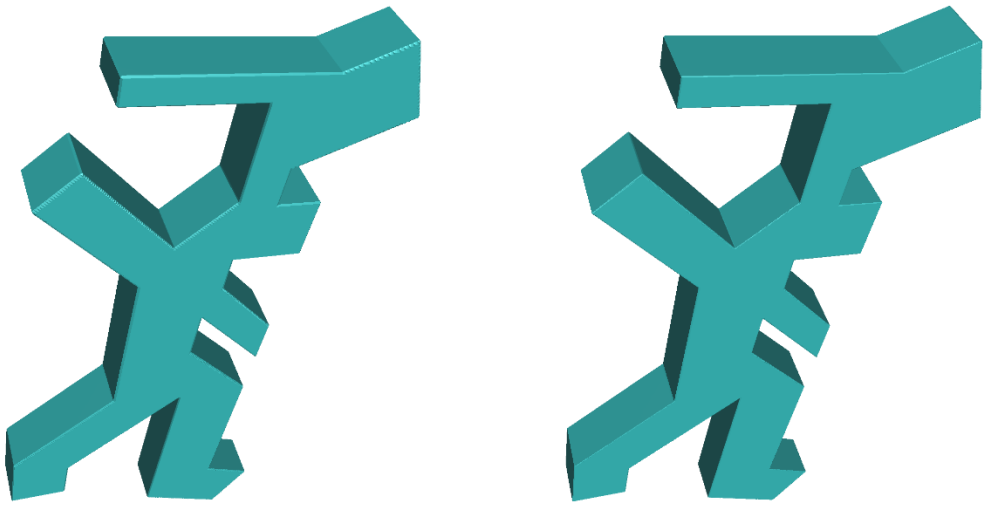
Figura 13: Mallas generadas usando la superficie *hex* como dominio de entrada en el algoritmo original.



Ref. 3 Ref. 4 Ref. 5 Ref. 6

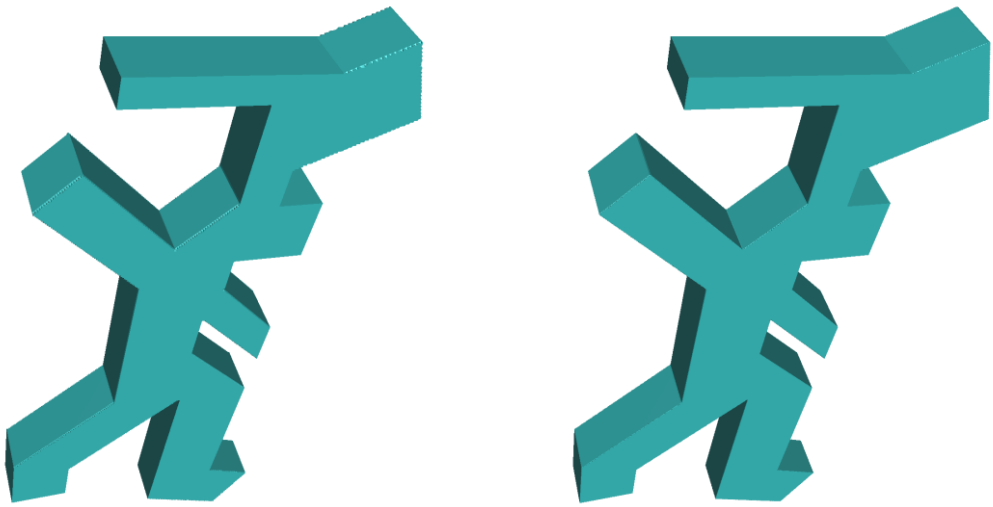
Figura 14: Mallas generadas usando la superficie *hex* como dominio de entrada en el algoritmo propuesto con un parámetro de calidad q = 0.5.

En el caso de una figura simple como *hex*, se observa que hay una mejora importante en la representación del dominio de entrada desde el nivel más bajo de refinamiento. El algoritmo original en la Figura 13 no puede generar de forma correcta los ángulos rectos de la superficie de entrada que existen en las esquinas de la figura, situación que se repite en todos los niveles de refinamiento. Por otro lado, el algoritmo propuesto en la Figura 14 logra una representación correcta del dominio en todos los niveles de refinamiento analizados, logrando generar los ángulos rectos donde corresponden.



Ref. 5 Ref. 6

Figura 15: Mallas generadas usando la superficie *fiducials1* como dominio de entrada en el algoritmo original (los refinamientos 3 y 4 se encuentra en el Anexo).

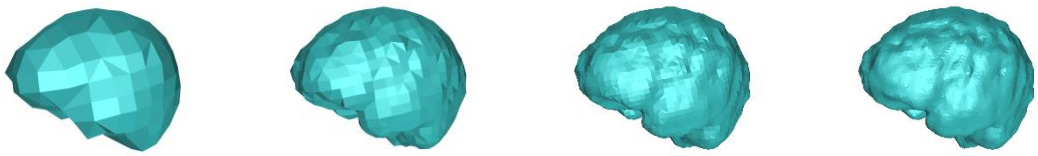


Ref. 5 Ref. 6

Figura 16: Mallas generadas usando la superficie *fiducials1* como dominio de entrada en el algoritmo propuesto con un parámetro de calidad q = 0.5 (los refinamientos 3 y 4 se encuentra en el Anexo).

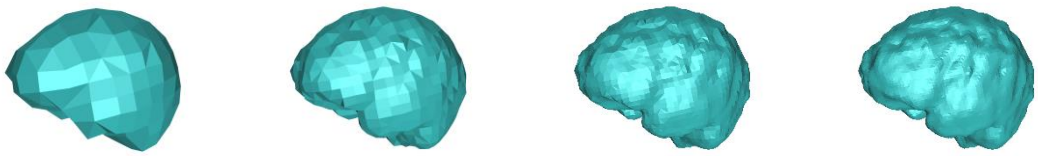
Comparando las mallas generadas por ambos algoritmos, en la Figura 16 se observa una mejora en la representación de ángulos rectos con respecto a la Figura 15 correspondiente al algoritmo original, en el área superior de la figura y en general en todos los ángulos rectos. Las secciones que poseen un tono más brillante y que se encuentran en los bordes de la malla son señal de representaciones que deberían ser rectas, las que disminuyen en las mallas generadas por la propuesta.

Sin embargo, el algoritmo propuesto falla en representar de forma totalmente correcta los ángulos que poseen cierto grado de inclinación (con una configuración similar a la de un rombo), los que se observa en el área media de las mallas generadas.



Ref. 3 Ref. 4 Ref. 5 Ref. 6

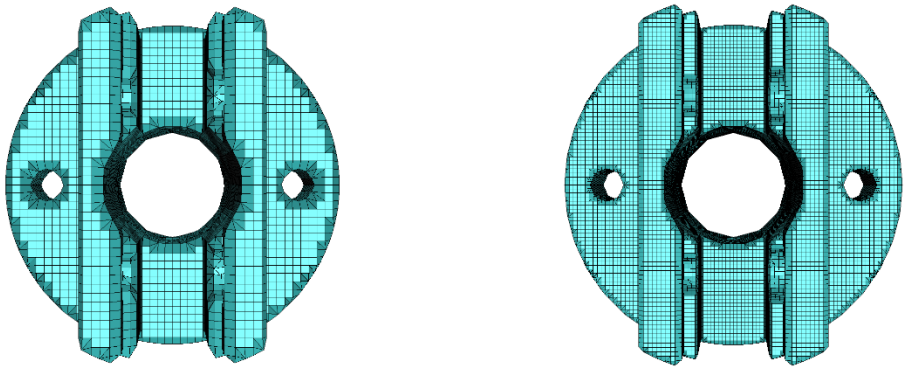
Figura 17: Mallas generadas usando la superficie *cortex* como dominio de entrada en el algoritmo original.



Ref. 3 Ref. 4 Ref. 5 Ref. 6

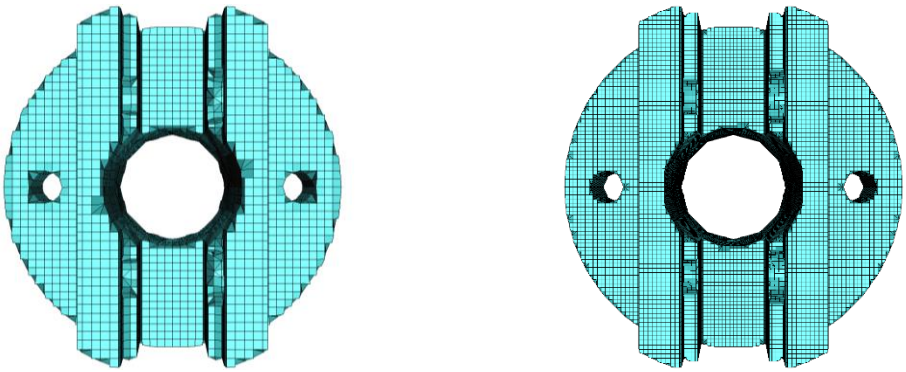
Figura 18: Mallas generadas usando la superficie *cortex* como dominio de entrada en el algoritmo propuesto con un parámetro de calidad q = 0.5.

Para el caso de *cortex*, este dominio de entrada se probó con el objetivo de observar los efectos de la modificación realizada en una malla que posee muy pocos ángulos rectos, y donde el algoritmo original en la Figura 17 tiene una ventaja innata. Si bien hay diferencias entre las mallas generadas por ambos algoritmos, estas son más bien sutiles, y se presentan en áreas donde las mallas generadas en la Figura 18 por el algoritmo propuesto poseen una mayor concentración de hexaedros.



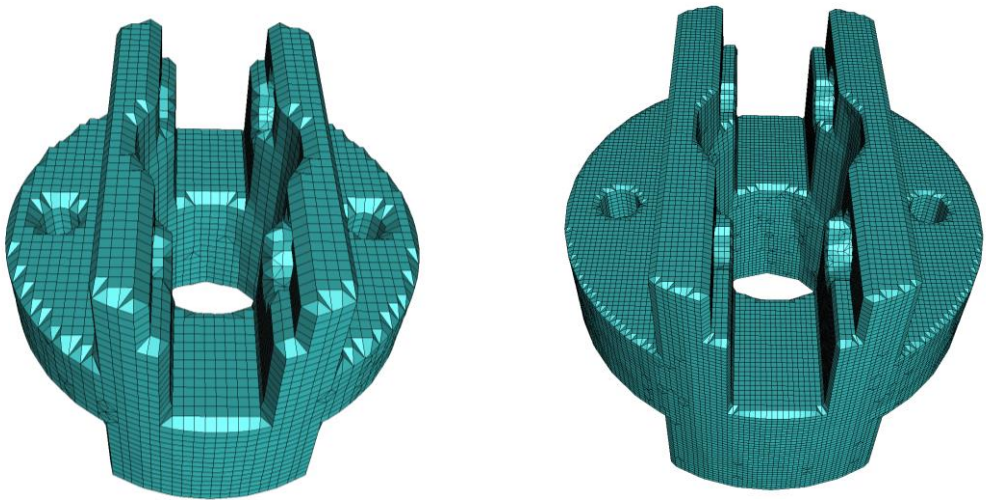
Ref. 5 Ref. 6

Figura 19: Mallas generadas usando la superficie *socket* (vista superior) como dominio de entrada en el algoritmo original (los refinamientos 3 y 4 se encuentra en el Anexo).



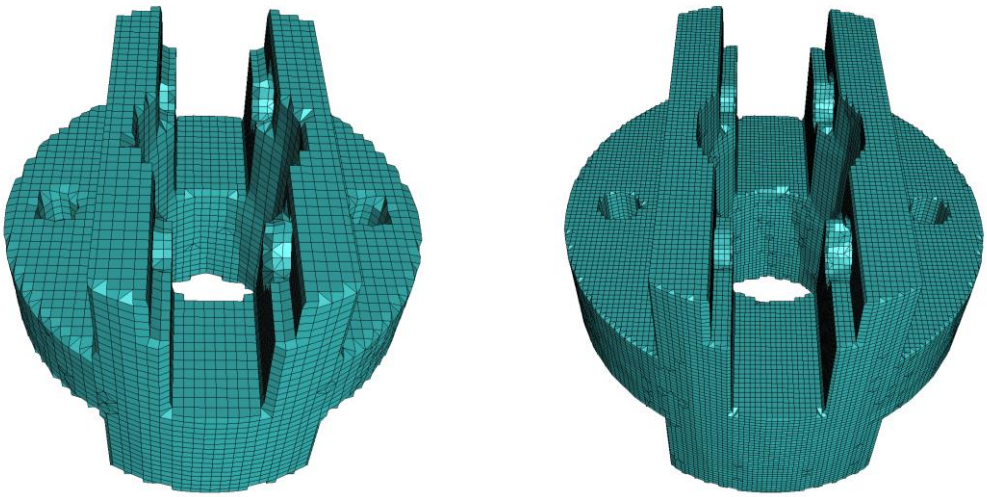
Ref. 5 Ref. 6

Figura 20: Mallas generadas usando la superficie *socket* (vista superior) como dominio de entrada en el algoritmo propuesto con un parámetro de calidad q = 0.5 (los refinamientos 3 y 4 se encuentra en el Anexo).



Ref. 5 Ref. 6

Figura 21: Mallas generadas usando la superficie *socket* (vista lateral) como dominio de entrada en el algoritmo original (los refinamientos 3 y 4 se encuentra en el Anexo).



Ref. 5 Ref. 6

Figura 22: Mallas generadas usando la superficie *socket* (vista lateral) como dominio de entrada en el algoritmo propuesto con un parámetro de calidad q = 0.5 (los refinamientos 3 y 4 se encuentra en el Anexo).

En el caso del dominio *socket*, se observa que las mallas generadas por el algoritmo original en la Figura 19 y la Figura 21 no representan de forma correcta ninguno de los ángulos rectos que se encuentran en la figura original, produciéndose en cambio geometrías más parecidas a “relieves”.

En cuanto al algoritmo propuesto, en la Figura 20 y la Figura 22 se observa en general una mejora significativa en la representación de ángulos rectos, en áreas como por ejemplo los agujeros laterales de la figura y los bordes rectos ubicados en el área media de la figura. Sin embargo, el algoritmo falla en representar correctamente los ángulos rectos que poseen una curvatura redonda similar a la de un cilindro (borde lateral de la figura), a pesar de que sí existe una mejora con respecto al algoritmo original.

CAPÍTULO 5: CONCLUSIONES

En el presente trabajo se abordó el problema de la representación de características finas en un generador de mallas Octree que, al estar diseñado para la representación de superficies suaves, no es capaz de representar superficies con ángulos rectos en forma correcta. Como el generador de mallas tiene un importante énfasis en el tiempo que demora la generación, cualquier modificación al algoritmo debe considerar esta restricción, además de evitar la existencia de elementos inválidos en las mallas resultantes. Teniendo en cuenta lo anterior, se diseñó e implementó una modificación al programa, y se realizaron distintas pruebas con el objetivo de comparar, tanto cuantitativamente como cualitativamente, la propuesta de solución realizada con el algoritmo original.

Antes de diseñar la modificación propuesta, se analizó paso por paso el algoritmo del generador de mallas, donde se descubrió que, durante las primeras etapas de generación de las mallas, el algoritmo es capaz de representar ciertas características finas a través de la generación de hexaedros, los cuales en un paso posterior son reemplazados debido a la aplicación de patrones de superficie.

Como la aplicación de patrones de superficie busca, entre otras cosas, reemplazar elementos de mala calidad, se optó por diseñar un filtro de calidad de elementos generados, con el objetivo de que los patrones de superficie no sean aplicados a todos los elementos como sucede en el algoritmo original. Para esto, se utilizó como filtro la métrica J_{ENS}, la cual representa la calidad individual de un elemento y tiene un valor mínimo de -1 y máximo de 1 para un elemento perfecto.

Para lograr esto, fue necesario implementar en el código del generador el cómputo del J_{ENS}, por lo que se extrajo y adaptó un fragmento de código externo para poder obtener la métrica J_{ENS} de un elemento individual. Esta métrica se debe calcular en todos los puntos del elemento, lo que implicó la creación de estructuras de datos eficientes como input de la función con el objetivo de evitar cualquier ralentización producto de los cálculos que se deben realizar.

De igual manera, se consideró que cualquier tipo de filtro va a implicar operaciones adicionales en el algoritmo, por lo que se añadió un booleano simple a la estructura de los puntos en los elementos a filtrar, teniendo como meta el diseño de un filtro lo más eficiente posible. La implementación del filtro se hizo antes de aplicar los patrones de superficie, y funciona a nivel de octante, por lo que basta con que un elemento no cumpla con el valor J_{ENS} requerido para que inmediatamente termine el proceso de verificación del booleano y se apliquen los patrones de superficie, como ocurre en el algoritmo original.

Con el fin de evitar el reemplazo innecesario de hexaedros, se debieron alterar los pasos del algoritmo original, dejando la aplicación de patrones de superficie junto con el filtro más cerca del final. El resultado de este diseño y posterior implementación es un generador de mallas Octree con un parámetro de calidad adicional q, el cual corresponde al valor mínimo de calidad requerida por todos los elementos de un octante para omitir la aplicación de patrones de superficie.

Para evaluar la propuesta de solución, se realizaron distintas pruebas usando como base de comparación el algoritmo original. Se evaluó el tiempo de ejecución del algoritmo, la cantidad de elementos generados en cada malla, la calidad del peor elemento en cada malla y la calidad promedio de todos los elementos de la malla. También, se evaluó cualitativamente la representación de características finas de las mallas resultantes.

Con respecto al tiempo, la propuesta de solución es, en el peor de los casos, tan solo un 1% más lenta que el algoritmo original. Este es el resultado no solo de un buen uso de estructuras de datos, sino también de un buen diseño algorítmico, ya que si bien se introdujeron cálculos adicionales, estos se compensan en parte gracias a que ya no se aplican los patrones de superficie a todos los elementos superficiales, ahorrando así tiempo. Entonces, se afirma que la modificación implementada posee un efecto prácticamente insignificante en el tiempo de ejecución.

En cuanto a la cantidad de elementos generados, en general las mallas generadas por la propuesta de solución poseen una menor cantidad de elementos que el algoritmo original, lo cual se condice con el diseño del algoritmo, pues se aprovechan los hexaedros generados en vez de reemplazarlos. Esto también explica que las mallas generadas por la propuesta tengan un mayor porcentaje de hexaedros, y mientras menor sea el parámetro de calidad, menos hexaedros son reemplazados, lo que implica una mayor proporción y una menor cantidad total de elementos. A medida que aumenta el nivel de refinamiento, las cantidades de elementos generados tienen un a valor similar, esto pues los elementos superficiales disminuyen en proporción al resto de los elementos de la malla de volumen.

El análisis del peor elemento generado en cada malla fue una prueba relevante ya que no era factible generar mallas con elementos inválidos. Los resultados muestran que en ningún caso se produjeron elementos inválidos, y que tan solo en 2 casos en el nivel de refinamiento 3 se generó al menos un elemento de peor calidad que el algoritmo original. En todos los otros casos, tanto el algoritmo original como la modificación poseen un peor elemento de la misma calidad. Esto se condice con la teoría ya que en general los elementos de peor calidad son aquellos generados como resultado de los patrones de superficie, sumado al hecho de que el filtro evita que se generen hexaedros de mala calidad.

Los resultados de la calidad promedio de los elementos de las mallas muestran que en todos los casos, el valor promedio es superior o igual al de los elementos generados por el algoritmo original. Teniendo en consideración el filtro de calidad y la utilización de hexaedros en mayor medida, estos resultados eran esperables. Los hexaedros tienden a poseer un valor de calidad J_{ENS} más cercano a 1 que el resto de los elementos, lo que se debe a que tienen formas más simples y fáciles de definir que el resto de los elementos de las mallas Octree. Entonces, si se utilizan más hexaedros que otras figuras, es natural que el promedio de los elementos generados aumente. Aún así, a medida que aumenta el nivel de refinamiento, la calidad promedio entre los algoritmos tiende a un valor similar debido a que, al igual que con la cantidad de elementos generados, los elementos en la superficie disminuyen en proporción al resto de los elementos.

El análisis cualitativo se realizó comparando la capacidad de representación de características finas entre ambos algoritmos, y en el caso de la propuesta con un parámetro de calidad q = 0.5. Se observó una mejora importante en la representación de ángulos rectos con respecto al algoritmo original, la cual se vuelve más sutil mientras aumenta el nivel de refinamiento y la cantidad de elementos que representan características finas disminuye proporcionalmente. Para el caso de un dominio de entrada simple como un cubo, se logró una representación prácticamente perfecta, mientras que en dominios más complejos los resultados variaron.

Si bien se logró representar correctamente los ángulos rectos sin inclinación (similar a la intersección de dos ejes normales), no se consiguió representar con la misma precisión los ángulos rectos que poseen cierta inclinación con respecto a los ejes (similar a un rombo), ni tampoco los ángulos rectos que forman una curvatura como la que se encuentra en un cilindro. El motivo de esto se debe a que el algoritmo utiliza hexaedros regulares, por lo que solo es posible alcanzar una representación perfecta de características finas en dominios que posean ángulos rectos capaces de ser modelados utilizando solo hexaedros.

Con respecto a la representación de características curvas, el algoritmo propuesto tiene un impacto menor en éstas, el cual depende principalmente del parámetro de calidad que se utilice. Un parámetro de calidad bajo va a tener un impacto mayor en superficies suaves, mientras que con un parámetro de calidad más cercano a 1 el impacto es prácticamente nulo, ya que mientras más exigente sea este parámetro menos octantes son filtrados, lo que implica que el algoritmo propuesto se comporta más parecido al algoritmo original en cuanto a la aplicación de patrones de superficie. En general, se observó que la modificación implementada no solo ayuda a representar de mejor forma las características finas, sino que también aprovecha de mejor manera los hexaedros generados.

Es necesario señalar que el uso de la métrica J_{ENS} como método de filtro es un diseño heurístico que se adapta bien a los pasos de este generador de mallas en particular. El valor J_{ENS} se calcula sobre un elemento individual, lo que significa que no es representativo de un conjunto de elementos como sí son las mallas geométricas. Más aún, al ser solo una métrica de la calidad del elemento, ésta no tiene ninguna relación con la topología del dominio, por lo que puede existir el caso en que un elemento de peor calidad represente de mejor forma cierta característica del dominio que un elemento perfecto.

Conseguir una representación perfecta de los dominios de entrada no es una tarea imposible, sin embargo se debe tener en cuenta el *trade-off* que se produce como consecuencia, el cual se refleja en el tiempo de ejecución. Considerando las características y el fin de este generador de mallas, una modificación costosa que mejore la representación de los dominios a cambio de un mayor tiempo de ejecución sería contraproducente.

Aún así, hay espacio para seguir mejorando el algoritmo y alcanzar una representación aún mejor de características finas. Una posible solución podría ser introducir otro elemento adicional a los 4 que existen, que represente de mejor manera los ángulos rectos con características cilíndricas, y añadiendo sus respectivos patrones de superficie. Sin embargo, una modificación como ésta podría tener un impacto significativo en el tiempo de ejecución, por lo que resulta necesario hacer un análisis de impacto en primer lugar.

Otro posible camino para desarrollar podría ser la creación de un módulo de preprocesamiento de características finas, el cual se encargaría de generar las características finas antes de aplicar los patrones de superficie. Para evitar el problema de analizar todos los elementos de la superficie, y teniendo en consideración que en muchos casos las características finas solo se presentan en algunas áreas, se podrían usar ciertas regiones de interés de ángulos rectos definidas por el usuario, con el objetivo de evitar que el generador realice cálculos innecesarios.

De todas maneras, es evidente que cualquier modificación que implique cómputo adicional va a afectar el tiempo de ejecución del algoritmo, por lo que quizás una forma distinta de abordar el problema podría ser desde el punto de vista de la optimización actual del algoritmo.

Durante la ejecución de las pruebas, se observó que el generador de mallas utiliza solamente un núcleo del procesador, a pesar de que este último posee múltiples núcleos. Esto significa que cada vez que se ejecuta el generador de mallas, hay a lo más un núcleo que trabaja al 100% de su capacidad mientras el resto se encuentra inactivo (para efectos del generador de mallas). Entonces, una optimización del algoritmo que utilice los múltiples núcleos de un procesador permitiría introducir pasos más costosos que ayuden a la

representación de características complejas del dominio sin producir tiempos de ejecución muy altos.

Siguiendo la misma línea anterior, hoy en día algunas tarjetas gráficas proveen interfaces de programación en C++ que permiten utilizar su capacidad de procesamiento para tareas que no están relacionadas con la generación de imágenes. Las tarjetas gráficas, a diferencia de los procesadores, utilizan miles de núcleos en paralelo para realizar cálculos en forma simultánea, por lo que vale la pena investigar qué tan factible es adaptar o transformar este generador de mallas para que utilice una tarjeta gráfica en vez del procesador, teniendo en cuenta que una optimización del tiempo de ejecución posibilita introducir operaciones más costosas.

REFERENCIAS BIBLIOGRÁFICAS

[1] C. Lobos y E. González, «Mixed-element Octree: a meshing technique toward fast and real-time simulations in biomedical applications,» *International Journal for Numerical Methods in Biomedical Engineering,* vol. 31, nº 12, 2015.

[2] T.-T. Cao, A. Nanjappa, M. Gao y T.-S. Tan, «A GPU accelerated algorithm for 3D Delaunay triangulation,» *I3D '14: Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games,* p. 47–54, 2014.

[3] D. Meagher, «Geometric modeling using octree encoding,» *Computer Graphics and Image Processing,* vol. 19, nº 2, pp. 129-147, 1982.

[4] M. Yerry y M. Shepard, «Automatic mesh generation for three-dimensional solids,» *Computers & Structures,* vol. Advances and Trends in Structures and Dynamics, nº 1-3, pp. 31-39, 1985.

[5] L. Maréchal, «Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features,» de *Proceedings of the 18th International Meshing Roundtable*, 2009.

[6] C. Arenas y C. Lobos, «Detection and representation of sharp features in octree–based meshes using different types of elements,» de *2018 37th International Conference of the Chilean Computer Science Society (SCCC)*, Santiago, 2018.

[7] Weierstrass Institute, [En línea]. Available: https://wias-berlin.de/software/tetgen/fformats.others.html. [Último acceso: 27 11 2023].

ANEXOS

Tabla 7: Calidad promedio y cantidad de hexaedros en las mallas generadas

	Hexaedro							
	Calidad promedio del elemento / Cantidad del elemento							
Algoritmo	Ref. 3		Ref. 4		Ref. 5		Ref. 6	
hex	Alg. 1	1.0000	368	1.0000	2192	1.0000	10448	1.0000	45392
	Alg. 2 (q = 0.2)	1.0000	448	1.0000	2368	1.0000	10816	1.0000	46144
	Alg. 2 (q = 0.5)	1.0000	448	1.0000	2368	1.0000	10816	1.0000	46144
	Alg. 2 (q = 0.8)	1.0000	448	1.0000	2368	1.0000	10816	1.0000	46144
fiducials1	Alg. 1	0.9957	8138	0.9964	39981	0.9967	184306	0.9967	781234
	Alg. 2 (q = 0.2)	0.9811	9806	0.9905	45329	0.9925	201769	0.9917	855030
	Alg. 2 (q = 0.5)	0.9829	9776	0.9906	45321	0.9925	201681	0.9918	854744
	Alg. 2 (q = 0.8)	0.9913	8878	0.9943	41731	0.9947	190873	0.9939	805518
cortex	Alg. 1	0.9524	87	0.9560	456	0.9617	2439	0.9668	11779
	Alg. 2 (q = 0.2)	0.7877	191	0.8874	778	0.9263	3630	0.9417	16566
	Alg. 2 (q = 0.5)	0.9387	115	0.9436	588	0.9529	2984	0.9597	13915
	Alg. 2 (q = 0.8)	0.9514	89	0.9548	468	0.9613	2457	0.9665	11837
socket	Alg. 1	0.9719	76	0.9551	1042	0.9819	7251	0.9848	34650
	Alg. 2 (q = 0.2)	0.8994	266	0.9305	1732	0.9639	9308	0.9758	40467
	Alg. 2 (q = 0.5)	0.9185	258	0.9399	1566	0.9675	9163	0.9767	40080
	Alg. 2 (q = 0.8)	0.9589	152	0.9511	1344	0.9804	7911	0.9832	36324

Tabla 8: Calidad promedio y cantidad de prismas en las mallas generadas

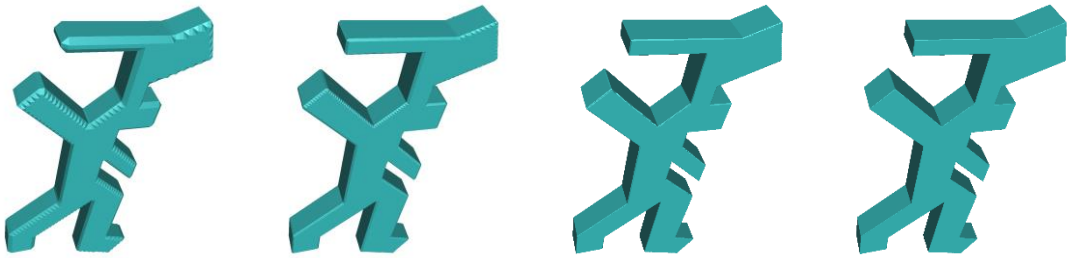
		Prisma (cuña)							
		Calidad promedio del elemento / Cantidad del elemento							
	Algoritmo	Ref. 3		Ref. 4		Ref. 5		Ref. 6	
hex	Alg. 1	0.8165	72	0.8165	264	0.8165	744	0.8165	1800
	Alg. 2 (q = 0.2)	0.0000	0	0.8165	96	0.8165	384	0.8165	1056
	Alg. 2 (q = 0.5)	0.0000	0	0.8165	96	0.8165	384	0.8165	1056
	Alg. 2 (q = 0.8)	0.0000	0	0.8165	96	0.8165	384	0.8165	1056
fiducials1	Alg. 1	0.7367	3422	0.7626	14857	0.7767	59164	0.7579	258014
	Alg. 2 (q = 0.2)	0.8238	1360	0.8284	6645	0.8317	29262	0.8290	120358
	Alg. 2 (q = 0.5)	0.8236	1384	0.8281	6653	0.8313	29402	0.8286	120694
	Alg. 2 (q = 0.8)	0.7692	2450	0.7802	12445	0.8041	48228	0.7884	213768
cortex	Alg. 1	0.6653	117	0.6818	622	0.6940	2806	0.7014	12101
	Alg. 2 (q = 0.2)	0.6819	38	0.7127	329	0.7294	1579	0.7402	6743
	Alg. 2 (q = 0.5)	0.6763	79	0.6936	445	0.7041	2025	0.7105	8766
	Alg. 2 (q = 0.8)	0.6650	115	0.6827	614	0.6945	2778	0.7021	11997
socket	Alg. 1	0.7127	144	0.6746	1007	0.7063	5192	0.7267	19098
	Alg. 2 (q = 0.2)	0.7933	30	0.6914	328	0.7571	2364	0.7725	9881
	Alg. 2 (q = 0.5)	0.7938	32	0.6791	417	0.7470	2499	0.7665	10412
	Alg. 2 (q = 0.8)	0.7131	78	0.6745	623	0.7104	4304	0.7364	16663

Tabla 9: Calidad promedio y cantidad de pirámides en las mallas generadas

		Pirámide							
		Calidad promedio del elemento / Cantidad del elemento							
	Algoritmo	Ref. 3		Ref. 4		Ref. 5		Ref. 6	
hex	Alg. 1	0.5303	96	0.4271	960	0.4251	6240	0.4308	31872
	Alg. 2 (q = 0.2)	0.5303	96	0.4271	960	0.4251	6240	0.4308	31872
	Alg. 2 (q = 0.5)	0.5303	96	0.4271	960	0.4251	6240	0.4308	31872
	Alg. 2 (q = 0.8)	0.5303	96	0.4271	960	0.4251	6240	0.4308	31872
fiducials1	Alg. 1	0.4422	5731	0.4365	36795	0.4363	184901	0.4349	833228
	Alg. 2 (q = 0.2)	0.4451	5221	0.4366	35725	0.4364	182835	0.4352	828844
	Alg. 2 (q = 0.5)	0.4451	5225	0.4366	35729	0.4364	182859	0.4352	828870
	Alg. 2 (q = 0.8)	0.4428	5739	0.4365	36807	0.4363	185027	0.4349	833284
cortex	Alg. 1	0.5161	302	0.4986	1938	0.4921	10287	0.4860	47741
	Alg. 2 (q = 0.2)	0.5467	202	0.5148	1592	0.5053	8988	0.4991	42836
	Alg. 2 (q = 0.5)	0.5269	278	0.5047	1834	0.4968	9912	0.4892	46655
	Alg. 2 (q = 0.8)	0.5173	302	0.4992	1922	0.4921	10280	0.4860	47726
socket	Alg. 1	0.4907	116	0.4965	1157	0.4442	7590	0.4424	47651
	Alg. 2 (q = 0.2)	0.4110	8	0.5148	899	0.4478	7052	0.4444	46817
	Alg. 2 (q = 0.5)	0.4110	8	0.5060	958	0.4471	7163	0.4441	46992
	Alg. 2 (q = 0.8)	0.4939	98	0.5013	1097	0.4443	7547	0.4425	47616

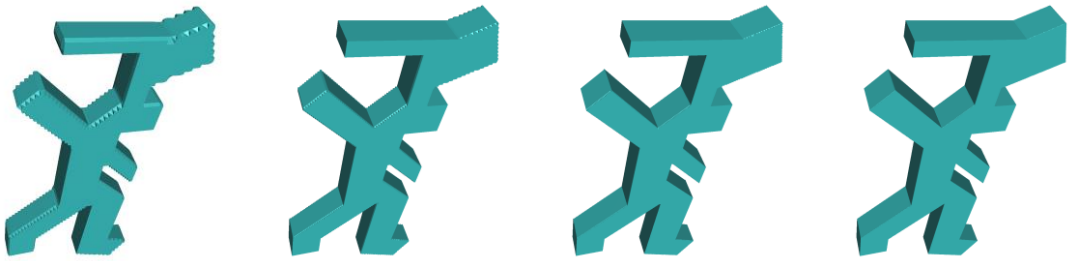
Tabla 10: Calidad promedio y cantidad de tetraedros en las mallas generadas

		Tetraedro							
		Calidad promedio del elemento / Cantidad del elemento							
	Algoritmo	Ref. 3		Ref. 4		Ref. 5		Ref. 6	
hex	Alg. 1	0.4302	104	0.5022	728	0.5178	4712	0.5188	24632
	Alg. 2 (q = 0.2)	0.4071	96	0.5000	720	0.5175	4704	0.5188	24624
	Alg. 2 (q = 0.5)	0.4071	96	0.5000	720	0.5175	4704	0.5188	24624
	Alg. 2 (q = 0.8)	0.4071	96	0.5000	720	0.5175	4704	0.5188	24624
fiducials1	Alg. 1	0.4776	4564	0.4971	26663	0.5097	129041	0.5152	569523
	Alg. 2 (q = 0.2)	0.4800	4384	0.4979	26447	0.5102	128471	0.5153	569031
	Alg. 2 (q = 0.5)	0.4802	4390	0.4979	26447	0.5102	128483	0.5153	569041
	Alg. 2 (q = 0.8)	0.4795	4462	0.4977	26533	0.5101	128673	0.5152	569353
cortex	Alg. 1	0.4984	292	0.4836	2095	0.4824	10677	0.4801	48289
	Alg. 2 (q = 0.2)	0.5104	202	0.4874	1786	0.4862	9628	0.4833	45103
	Alg. 2 (q = 0.5)	0.4988	280	0.4833	2028	0.4827	10493	0.4805	47608
	Alg. 2 (q = 0.8)	0.4986	286	0.4831	2073	0.4823	10667	0.4798	48283
socket	Alg. 1	0.4829	164	0.4380	1141	0.4850	5477	0.4976	33097
	Alg. 2 (q = 0.2)	0.5574	28	0.4363	969	0.4882	5231	0.4985	32693
	Alg. 2 (q = 0.5)	0.5443	34	0.4346	1051	0.4880	5293	0.4983	32738
	Alg. 2 (q = 0.8)	0.4781	151	0.4374	1130	0.4856	5453	0.4978	33037



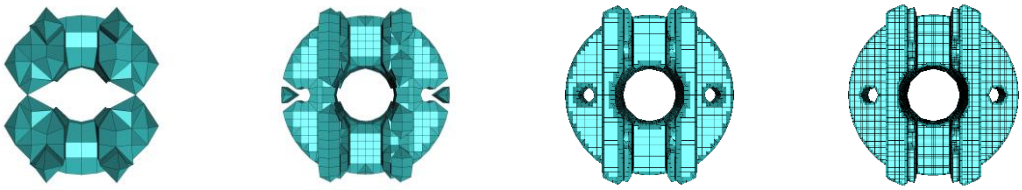
Ref. 3 Ref. 4 Ref. 5 Ref. 6

Figura 23: Malla generada usando la superficie *fiducials* como dominio de entrada en el algoritmo original.



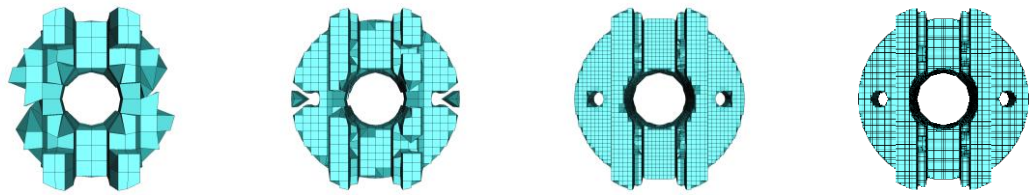
Ref. 3 Ref. 4 Ref. 5 Ref. 6

Figura 24: Malla generada usando la superficie *fiducials* como dominio de entrada en el algoritmo propuesto con parámetro q = 0.5.



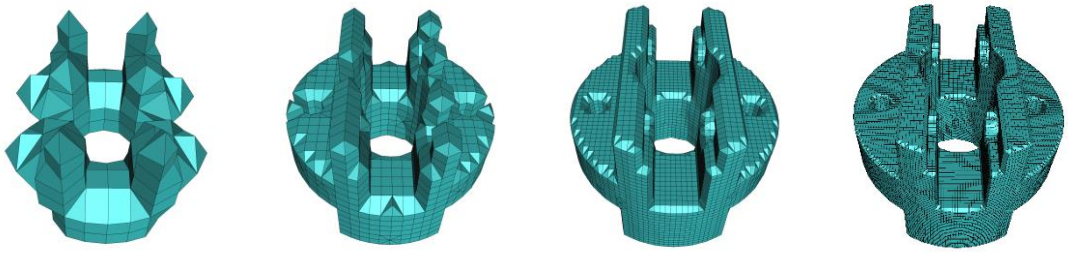
Ref. 3 Ref. 4 Ref. 5 Ref. 6

Figura 25: Malla generada usando la superficie *socket* como dominio de entrada en el algoritmo original (vista superior).



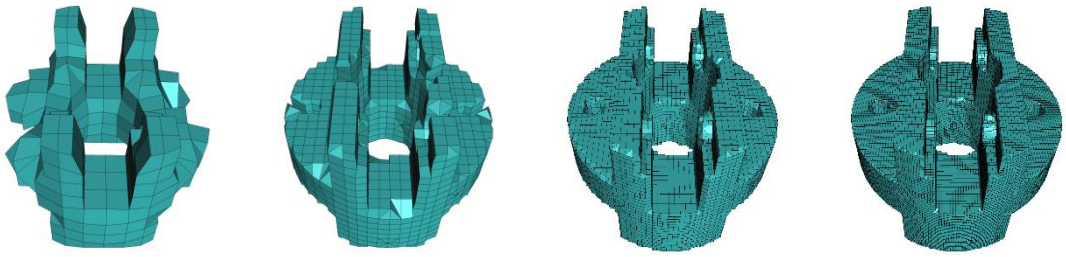
Ref. 3 Ref. 4 Ref. 5 Ref. 6

Figura 26: Malla generada usando la superficie *socket* como dominio de entrada en el algoritmo propuesto con parámetro q = 0.5 (vista superior).



Ref. 3 Ref. 4 Ref. 5 Ref. 6

Figura 27: Malla generada usando la superficie *socket* como dominio de entrada en el algoritmo original (vista lateral).



Ref. 3 Ref. 4 Ref. 5 Ref. 6

Figura 28: Malla generada usando la superficie *socket* como dominio de entrada en el algoritmo propuesto con parámetro q = 0.5 (vista lateral).