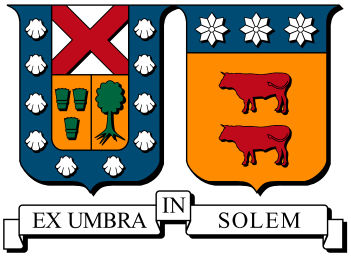
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



"PATRONES DE TRANSICIÓN DE POLIEDROS PARA MALLAS TIPO OCTREE"

RODRIGO ANDRÉS FUENTES HERRERA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: CLAUDIO LOBOS
PROFESOR CORREFERENTE: NANCY HITSCHFELD

JUNIO 2017

Agradecimientos

Quiero dar las gracias a todas las personas que han formado parte de este proceso, en especial a mi madre, Mónica, y padre, Juan, por darme todas las herramientas para poder desarrollar mis estudios sin tener que preocuparme de otras cosas, y con todo el apoyo que necesitaba.

A mi pareja Macarena por apoyarme en estos últimos años de estudio, siendo un pilar fundamental en los momentos más complicados y en los más alegres, además de enseñarme a vivir la vida y apoyarme siempre a realizar y cumplir los sueños que nos proponemos.

A mis amigos y compañeros de estudio, con los cuales aprendimos juntos a pasar los ramos y a también a reprobar algunos, pero siempre disfrutando cada momento dentro y fuera de la universidad.

A mis profesores Claudio y Nancy por su increíble disposición y ayuda en este trabajo, pese a todos los inconvenientes que tuve para poder realizarlo, siempre se prestaron amables y con consejos precisos para finalizar esta tarea.

Por último, agradecer a toda mi familia y amigos de infancia que me ayudaron a mantener la mente distraída y pasar excelentes momentos, mención especial a mi abuelita que siempre me ha apoyado económica y alimentariamente.

Resumen

Se analizan dos patrones para la generación de mallas volumétricas, los patrones de elementos mixtos basados en la técnica de refinamiento Octree, y los patrones de poliedros que satisfacen la condición de Delaunay. El problema consiste en la implementación de la librería de poliedros dentro del programa de elementos mixtos, además de determinar qué patrón genera mejores resultados tomando en consideración tiempos de ejecución, cantidad de poliedros, cantidad de polígonos, cantidad de vértices, y los tipos de poliedros utilizados en las mallas, que se estudiarán, y posteriormente realizar un análisis comparativo entre los resultados obtenidos.

Palabras Clave: Generación de Mallas, Octree, Tessellation, Delaunay, Poliedros

Abstract

Two patterns for volumetric mesh generation, mixed element patterns based on the Octree refinement technique, and the polyhedron patterns satisfying the Delaunay condition are discussed. The problem consists in the implementation of the library of polyhedral within the program of mixed elements, in addition to determining which pattern generates better results taking into consideration execution times, number of polyhedral, number of polygons, number of vertices, and types of polyhedral used in the final meshes, to be studied, and then perform a comparative analysis between the results obtained.

Keywords: Mesh Generation, Octree, Tessellation Delaunay, Polyhedral.

Índice de Contenidos

Introducción **1**

1. Definición del Problema **3**

1.1. Descripción 3

1.1.1. Malla de Superficie 3

1.1.2. Malla de Volumen 4

1.1.3. Octree 4

1.1.4. Punto Steiner 6

1.2. Objetivos 7

1.2.1. Objetivo Principal 7

1.2.2. Objetivos Secundarios 7

2. Estado del Arte **8**

2.1. Mixed-Element Octree 8

2.2. Polyhedral Delaunay Meshes 11

2.3. Medición de calidad de elementos mixtos 13

3. Metodología **14**

3.1. Mesher 14

3.2. Polyhedral Tessellation 17

4. Implementación **19**

4.1. Ambiente 19

4.2. Mesher 20

4.3. Polyhedral Tessellation 23

5. Resultados **30**

5.1. Detalle del experimento 30

5.2. Tipos de entrada 31

5.3. Resultados 32

5.3.1. Mesher - Córtex 32

5.3.2. Mesher - Fémur 33

5.3.3. Polyhedral Tessellation - Córtex 33

5.3.4. Polyhedral Tessellation - Fémur 34

5.4. Análisis Comparativo 35

5.4.1. Córtex 35

5.4.2. Fémur 37

A. Tablas **43**

Bibliografía **46**

Introducción

En el día de hoy, para poder estudiar objetos o fenómenos físicos, es necesario traducirlos a un lenguaje matemático que los represente. Utilizando el poder computacional para simular toda la información posible, se pueden realizar análisis más detallados y precisos, con el fin de simular el mundo en que vivimos.

Una forma de representar objetos es a través de una malla volumétrica. Esta malla se construye a partir de una malla de superficie, la cual está constituida por múltiples polígonos que representan la capa exterior del objeto en estudio.

A diferencia de las mallas de superficie, el elemento base de las mallas volumétricas son los poliedros, los cuales poseen un volumen definido por múltiples polígonos.

El trabajo presentado aquí se basa en la implementación y análisis comparativo de dos métodos para generar mallas volumétricas.

El primer método, Mixed-element Octree propuesto por el profesor Claudio Lobos, consiste en la utilización de patrones de elementos mixtos aplicados a una malla Octree. Mientras que el segundo, Polyhedral Delaunay Meshes propuesto por la profesora Nancy Hitschfeld-Kahler, consiste en la generación de poliedros que satisfacen la condición de Delaunay. Ambos métodos se basan en la técnica de refinamiento Octree, para la construcción de las mallas.

La técnica de refinamiento Octree utiliza un elemento base llamado octante. Este elemento posee la característica de poder dividirse en 8 elementos de menor volumen y que en su conjunto forman el octante original. Al utilizar esta técnica de forma recursiva se puede dividir un objeto en elementos más pequeños, lo que es utilizado con el fin de mejorar la calidad

de las mallas generadas. Puesto que, a mayor nivel de división, mayor será la precisión que tendrá la malla volumétrica final.

El siguiente trabajo contiene:

1. Definición del problema, explicación de conceptos base, planteamiento del problema y presentación de los objetivos a cumplir.
2. Estado del arte, estudio actual de los métodos a comparar.
3. Metodología, detalle del funcionamiento de los algoritmos.
4. Implementación, explicación de la integración del código.
5. Resultados, casos de prueba, análisis de resultados y comparación de las mallas obtenidas.
6. Conclusiones sobre el estudio y trabajo realizado.

Capítulo 1

Definición del Problema

1.1. Descripción

El proceso de generación de mallas volumétricas, en el área de la computación, se enfoca en resolver problemas de modelamiento y simulación de distintos fenómenos físicos.

Existen diversas técnicas que representan objetos mediante mallas, la diferencia radica en la cantidad de elementos que generan para construir la malla final, como también, en los procesos y algoritmos que se utilizan para generarlas.

Para crear mallas volumétricas se debe poseer un modelo de superficie, el cual determina los límites del dominio a trabajar.

1.1.1. Malla de Superficie

Para comprender la composición de una malla de superficie, es necesario explicar su elemento base, el polígono.

Un polígono se define como una figura geométrica plana compuesta por:

- Vértices: Puntos dentro del plano, que unen distintos ejes.

- Ejes: Segmentos de recta que unen dos vértices.
- Caras: Conjunto cerrado de ejes que forman un área.

Existen diversas clasificaciones de los polígonos, en base a su composición. Para este caso solo se utilizan polígonos simples convexos. Los cuales deben satisfacer lo siguiente:

- Ningún par de ejes puede cortarse entre sí.
- Los ángulos interiores formados por dos ejes de una cara, no pueden exceder los 180 grados.

La representación de un objeto mediante la unión de múltiples polígonos se denomina malla de superficie.

1.1.2. Malla de Volumen

Análogamente, a la malla de superficie, la malla volumétrica posee un elemento base llamado poliedro. Un poliedro es un cuerpo geométrico compuesto por la unión de polígonos que encierran un volumen finito.

Una de las clasificaciones más comunes es en base al número de caras que posee el poliedro. Dando origen a los tetraedros, poliedros de 4 caras, pentaedros, poliedros de 5 caras, hexaedros, poliedros de 6 caras, etc.

Una malla volumétrica está compuesta por la unión de múltiples poliedros, existen distintos tipos de algoritmos de generación de mallas que se clasifican según los poliedros que utilizan.

1.1.3. Octree

Una de las técnicas para generar mallas volumétricas es la de Octree [1] . Esta técnica define como elemento base un Octante, el cual debe poseer la función de dividirse. El número de

división puede variar, lo que da origen a distintos tipos de Octree. Para este caso se utiliza el método que divide el octante en 8 sub-octantes.

Para que el algoritmo funcione, se debe definir una malla de superficie y el nivel de refinamiento. Este último valor, representa la cantidad de veces que debe dividirse el octante original.

El proceso de refinamiento Octree consiste en encerrar completamente la malla de superficie en un octante. Luego se comienzan las divisiones recursivas de los sub-octantes, hasta alcanzar el nivel de refinamiento especificado.

Importante señalar que, por cada división producida se deben revisar si los octantes generados están o no dentro del dominio, es decir, si corresponden efectivamente al volumen generado por la malla de superficie. Pueden darse 3 casos:

1. Que el octante este completamente dentro del dominio.
2. Que el octante este parcialmente dentro del dominio.
3. Que el octante este fuera del dominio.

Si el nivel de refinamiento no se ha alcanzado, para el primer caso no es necesario seguir dividiendo ya que el octante se encuentra dentro del dominio, para el segundo caso el octante se puede volver a dividir, mientras que, para el tercer caso, ya no es necesario seguir dividiendo, dado que se encuentra fuera del dominio.

Una vez alcanzado el nivel de refinamiento, se tiene un conjunto de octantes que están completa o parcialmente dentro del dominio, y octantes fuera de él. Además, se pueden presentar diferencias en los tamaños de los octantes resultantes. Dado que al interior del volumen pueden generarse octantes de mayor tamaño, a diferencia de los octantes que están cerca de la superficie del objeto.

1.1.4. Punto Steiner

Para terminar de crear una malla volumétrica de tipo 1-irregular, se debe verificar que no existan más de dos puntos Steiner en cada eje.

Los puntos Steiner corresponden a los puntos en los cuales se intersecta un eje. Esto se provoca al tener octantes de distinto tamaño, dado que para generar un elemento 1-irregualr cada eje de un elemento no puede tener más de dos octantes vecinos. Esto se ve reflejado en la Figura 1.1.

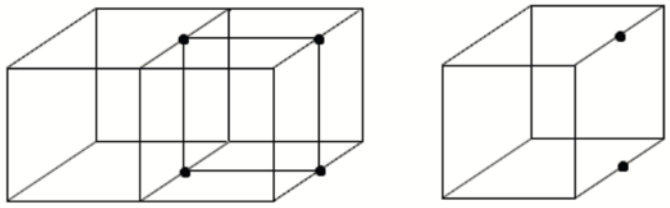


Figura 1.1: Izquierda: Unión de 3 octantes, generando un punto de Steiner en dos ejes. Derecha: Octante 1-irregular. Fuente: Hitschfeld-Kahler, Contreras[2]

Donde en la imagen de la izquierda, se ven 3 octantes, de los cuales el de la izquierda posee dos ejes que poseen intersecciones con dos octantes vecinos. Al apartar el octante de la izquierda se aprecia en la imagen de la derecha el octante 1-irregular.

En el caso que existan más de un punto Steiner en los ejes, se debe realizar una división en el octante, hasta que solo exista uno, con el fin de que todos los octantes de la malla sean de tipo 1-irregular.

Cuando todos los elementos de la malla son de tipo 1-irregular, se considera la malla como tal. Luego, se pueden generar las mallas de elementos mixtos, o las mallas de poliedros de Delaunay.

El problema está en que la librería para generar las mallas de poliedros de Delaunay, no se encuentra implementada en el software de generación de mallas de elementos mixtos. Por lo que la solución consiste en su implementación.

1.2. Objetivos

1.2.1. Objetivo Principal

El objetivo principal de este trabajo es realizar la implementación del método basado en poliedros de Delaunay, dentro del método basado en elementos mixtos, para la generación de mallas volumétricas.

1.2.2. Objetivos Secundarios

- Analizar en qué etapa del sistema del software actual, se debe implementar la librería de poliedros.
- Preparar el ambiente necesario para la instalación de ambos códigos.
- Instalar y visualizar resultados de las mallas de elementos mixtos.
- Instalar y visualizar los resultados de las mallas de poliedros.
- Registrar la cantidad de nodos y elementos generados por ambos métodos.
- Registrar las configuraciones de los elementos generados por ambos métodos.

Capítulo 2

Estado del Arte

2.1. Mixed-Element Octree

Las técnicas de generación de mallas nacen de la necesidad de generar métodos de simulación de procesos o fenómenos naturales para poder estudiarlos de manera computacional, para esto se debe modelar el dominio, obteniendo como resultado una malla.

Existen dos objetivos claves para la obtención de mallas, estos son la precisión y la velocidad. Para este caso el algoritmo de elementos mixtos se basa en la velocidad, ya que lo que busca es lograr una simulación, en el mejor de los casos, en tiempo real, de los fenómenos que se estén estudiando. Para lograr esto se utiliza una variación del algoritmo de generación de mallas Octree.

Octree es una estructura recursiva de datos, en la cual se posee inicialmente un octante el cual es dividido en 8, los cuales no pueden intersectarse entre sí. De esta manera se genera un árbol de octantes, esto genera el nombre de la técnica. Estos octantes generalmente son cubos, pero pueden tomar la forma de cualquier figura geométrica, lo importante es que posean la funcionalidad de dividirse. El resultado de estas divisiones genera una malla de tipo *"non-conformal"*, es decir, se generan octantes de tamaños muy distintos. Para pasar de una malla *"non-conformal"* a una *"conformal"* se debe cumplir que todos los octantes se

hayan dividido a lo menos una vez con respecto a sus octantes vecinos. La *Figura 2.1 (a)* representa una malla *"non-conformal* donde los elementos que estan en el interior poseen un tamaño más grande, generando multiples puntos steiner en sus ejes, por otro lado en la *Figura 2.1 (b)* se tiene una malla 1-irregular, donde todos los octantes poseen a lo más un punto Steiner en cada uno de sus ejes.

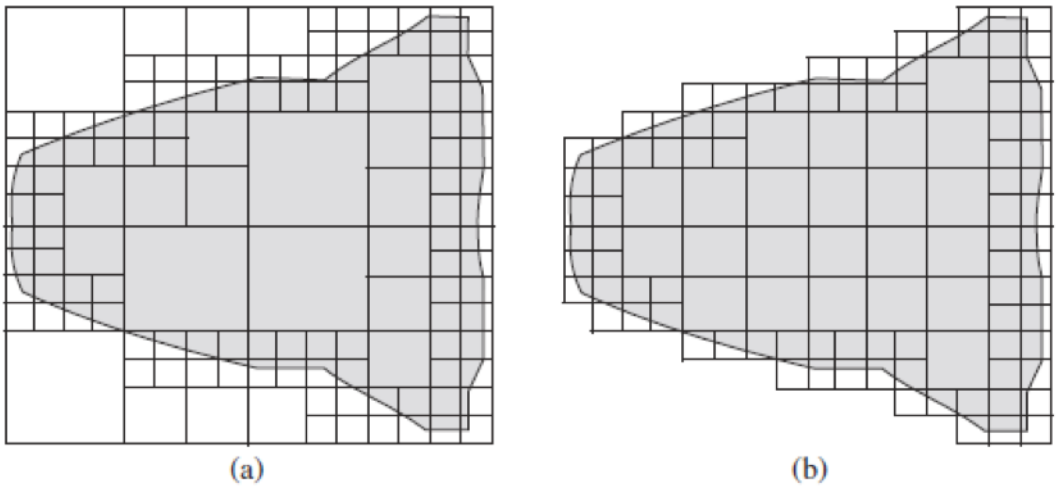


Figura 2.1: Ejemplo de (a) una malla "non-conformal" (Octree) y (b) una malla "conformal" (1-irregular). Fuente: Lobos, González[3]

Paralelo a esto, se tiene que algunas de las características más importantes para mejorar la velocidad de la ejecución del programa son la definición de zonas de interés y el nivel de refinamiento.

La zona de interés implica que, si se posee una malla de superficie de entrada, se puede definir una zona de interés, la cual puede quedar representada como una porción de la entrada original, a la cual se le desea mejorar la calidad de la malla resultante. Esta mejora en la calidad se logra definiendo un nivel de refinamiento mayor, lo que define esta última constante es la cantidad de veces que debe ser dividido el octante en nuevos octantes. El programa define tres tipos de nivel de refinamiento, uno para la zona de interés, otro para la zona de la superficie del dominio de entrada, y por último uno para definir la cantidad de divisiones para el resto del dominio. En la *Figura 2.2* se presentan distintos enfoques para una malla volumétrica, en la cual se aprecian los distintos niveles de refinamiento para las cada zona.

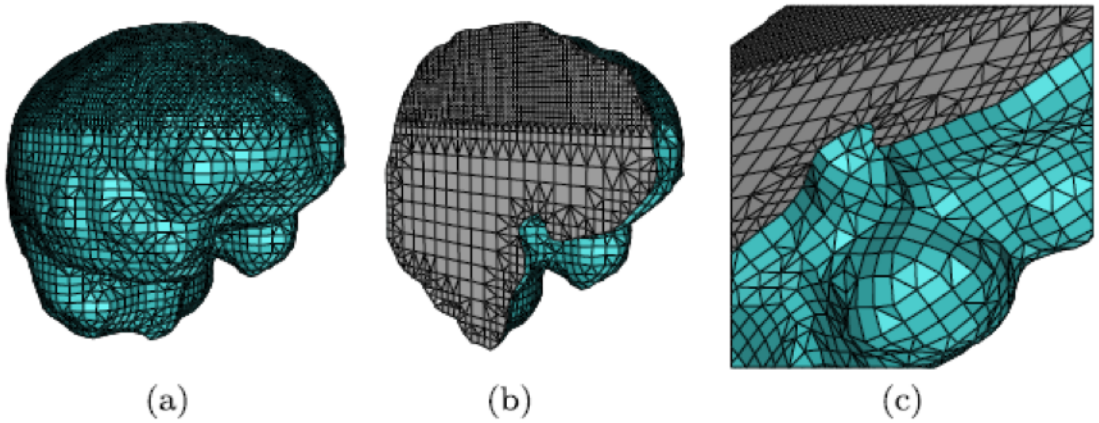


Figura 2.2: Ejemplo de los niveles de refinamiento, 6 para la zona de interés, 5 para la superficie, y 4 para el resto del dominio. (a) Vista de la superficie, (b) corte sagital y (c) zoom al corte sagital. Fuente: Lobos, González[3]

De la *Figura 2.2*, se aprecia que existe una zona de interés en la parte superior del córtex, en la cual se ha especificado que el nivel de refinamiento en dicha sección sea mayor al resto de la malla. Para los elementos que están al borde, en la superficie, se ha especificado otro valor distinto de refinamiento, mientras que los elementos restantes o del interior del córtex, presentan el nivel de refinamiento menor.

Para obtener una malla que posea una topología correcta entre regiones con distinto nivel de refinamiento, se utilizan los patrones de transición. Los patrones que se utilizan buscan en lo posible no agregar nuevos nodos, sino que dividir los octantes en elementos básicos, los cuales son el hexaedro, el prisma, la pirámide y el tetraedro. Para determinar que octantes deben pasar por los patrones de transición, se analizan los puntos de Steiner que posea cada eje. Si existe un punto Steiner se puede aplicar el patrón de transición respectivo.

Considerando todas las combinaciones posibles que pueden existir entre un octante y sus vecinos que posean distinto nivel de refinamiento, se han definido un total de 325 patrones de transiciones. De los cuales, solo 65 necesitan la inserción de un nodo extra.

Además de verificar la correcta topología de la malla, es importante verificar que los elementos y nodos generados estén, efectivamente, dentro del dominio de entrada.

2.2. Polyhedral Delaunay Meshes

Otro método para generar una malla volumétrica es el propuesto por la profesora Nancy Hitschfeld-Kahler, el cual es una modificación del algoritmo de generación de mallas Octree.

Este método busca reducir la cantidad de elementos de la malla, utilizando el computo de las teselaciones de Delaunay al conjunto de puntos de un casco convexo.

Una triangulación de Delaunay consiste en que, dado un conjunto de triángulos simples y convexos, cada circunferencia circunscrita en el triángulo, no puede contener ningún punto (vértice) de otro triangulo [4]. Para visualizar lo anterior se presenta la *Figura 2.3*, donde se tienen 4 puntos en el espacio 2D, y se tienen dos formas de construir los triángulos. En la *Figura 2.3 (b)* se forman dos triángulos que no satisfacen la condición, dado que las circunferencias circunscritas por ambos, poseen el vértice restante dentro de su superficie. Por otro lado, la *Figura 2.3 (c)* muestra la configuración correcta para el mismo conjunto de puntos.

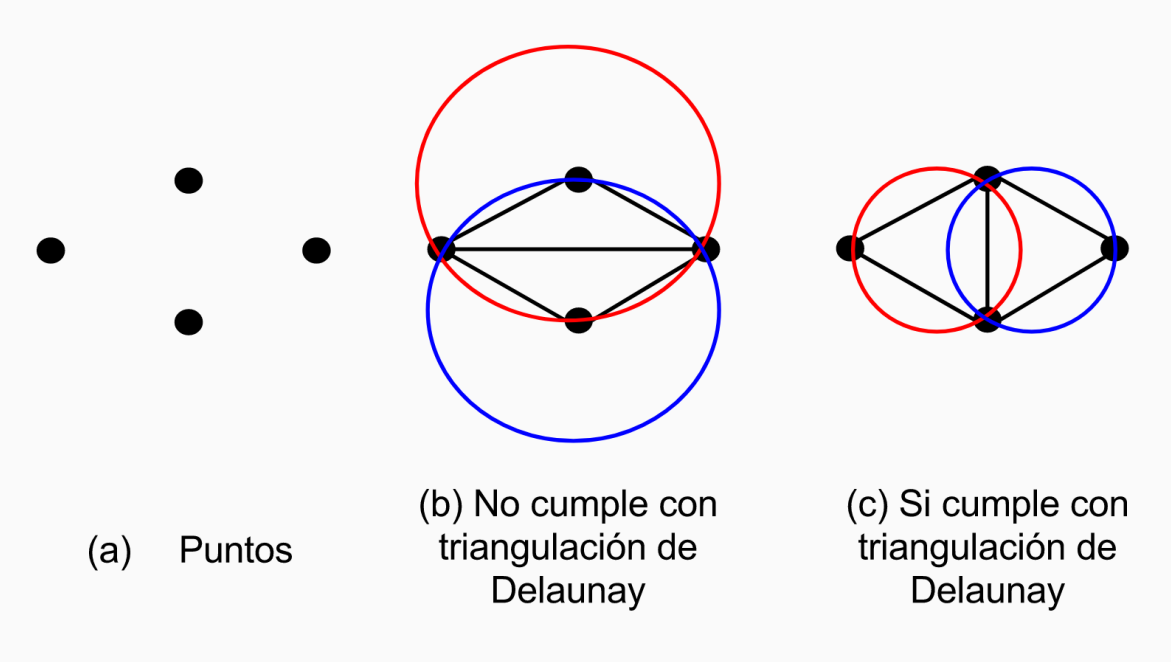


Figura 2.3: Ejemplo de triangulación de Delaunay. Fuente: Elaboración propia.

A diferencia de lo anterior, una Teselación de Delaunay incorpora otros elementos aparte de

los triángulos en 2D o poliedros que contengan polígonos triangulares en 3D.

De esta manera, solo se necesitan los puntos de una malla 1-irregular para construir la malla de tetraedros de Delaunay. Luego se deben unir los tetraedros para formar los elementos más grandes posibles, para finalmente generar poliedros co-esféricos.

El generador de mallas utilizado para este caso[5], entrega 7 poliedros iniciales. Sin embargo, luego de aplicar las teselaciones de Delaunay aparecieron 17 nuevos elementos. Generando un total de 24 elementos para construir una malla volumétrica. Cabe señalar que todos los elementos son poliedros co-esféricos, construidos a partir de un cubo 1-irregular y considerando las 4096 configuraciones posibles del cubo 1-irregular.

Dentro de estos 24 elementos, existen poliedros que pueden ser divididos en dos elementos diferentes, pero que están en la lista de las 24 iniciales. En particular analizando la *Figura 2.4*, se puede apreciar que el elemento puede ser dividido en dos sub-poliedros una pirámide cuadrilátera y un cuboide.

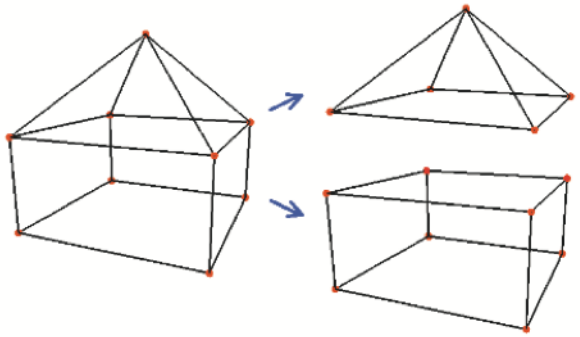


Figura 2.4: División de elementos. Fuente: Hitschfeld-Kahler, Contreras[2]

Realizando este estudio con el resto de los poliedros, se dio como resultado que la cantidad de poliedros finales se reduce a 16. Por último, se analizó la opción de añadir ejes y caras internas dentro de algunos poliedros, lo cual disminuyo a 13 los elementos finales utilizados por el algoritmo.

2.3. Medición de calidad de elementos mixtos

Dada la variedad de elementos que se pueden utilizar al momento de generar las mallas, se tienen métricas para cada tipo de elemento[6], a continuación, se explican las métricas de calidad para los Hexaedros y Tetraedros.

Para elementos de tipo hexaedros, se utiliza la métrica de Scaled Jacobian. Esta métrica se basa en tomar un nodo base, y calcular el jacobiano del tetraedro conformado por el nodo mismo y los tres nodos conectados a él. Cabe señalar que el valor resultante de esta métrica, puede otorgar tres conclusiones distintas, la primera es que el elemento sea inválido, esto ocurre cuando el valor del Scaled Jacobian es menor a cero, la segunda es que el elemento tenga una calidad relativamente buena, cuando es mayor a 0,2, y por último cuando el elemento es perfecto, cuando se obtiene un valor igual a 1.

Para el caso de los Tetraedros la métrica más utilizada es Aspect Ratio Gamma, para este caso se consideran las aristas y el volumen de cada elemento. Sin embargo, esta métrica no es compatible con los elementos de tipo Hexaedros, de igual manera que el Scaled Jacobian no es aplicable a los Tetraedros, dado que los resultados mixtos no son los mismos, en algunos casos, sin relación alguna.

Para lograr unificar ambas métricas, de modo que se pueda cuantificar la calidad de una malla generada a partir de elementos mixtos, se define Element Normalized Scaled Jacobian, la cual está basada en la métrica de Scaled Jacobian. La diferencia con respecto a ésta última, es que considera un factor constante de ajuste. El valor final Element Normalized Scaled Jacobian, se calcula en base a una función dependiente del valor del Scaled Jacobian del elemento. De esta forma, se calcula de forma particular en base a cada elemento que se esté analizando, incluyendo elementos de distintas formas, tales como los ya mencionados Tetraedros y Hexaedros, como también los Prismas y Pirámides.

Capítulo 3

Metodología

Para cumplir con los objetivos propuestos, se realiza un análisis de los programas para generar las mallas volumétricas.

3.1. Mesher

El profesor Claudio Lobos desarrolló un programa generador de mallas volumétricas[3], basado en Octree, el cual tiene como principal característica, la utilización de patrones de transición y superficies, con elementos mixtos.

El algoritmo se puede resumir según la *Figura 3.1*, la cual indica los pasos para transformar una malla de superficie en una malla volumétrica.

El primer punto a destacar es el ingreso de la malla de superficie, la cual representa el dominio en el cual se desea generar la malla volumétrica. Por otro lado, se ingresan las regiones de refinamiento, las cuales están representadas por la zona de interés, la zona de la superficie y el dominio restante, cada una de estas zonas posee un valor de refinamiento, el cual puede ser el mismo o bien diferente.

Con los parámetros de entrada definidos, se explica el paso siguiente, el cuál es la implementación de la técnica Octree. La primera línea indica que la malla estará compuesta por

Algorithm 1 Mixed elements meshing main algorithm

Require: input surface domain: Ω^s, set of regions of refinement: RoI_set.

```
1: mesh ← base_octree
2: while refinement is not reached do
3:     list_octants ← detect_octants_for_refinement(RoI_set)
4:     mesh ← split_octants(Ω^s, mesh, list_octants)
5: end while
6: while mesh is not 1-irregular do
7:     list_octants ← detect_not_1_irregular()
8:     mesh ← split_octants(Ω^s, mesh, list_octants)
9: end while
10: mesh ← apply_transition_patterns(mesh)
11: mesh ← remove_close_to_boundary_elements(mesh)
12: mesh ← apply_surface_patterns(mesh)
13: mesh ← shrink_outside_nodes(Ω^s, mesh)
14: mesh ← decrease_staircase_effect(Ω^s, mesh)
15: write_eoutput(mesh)
```

Figura 3.1: Algoritmo de generación de mallas volumétricas utilizando elementos mixtos[3].

el octante base de Octree, es decir se encierra todo el dominio dentro de un octante. Luego comienza el primer ciclo que divide recursivamente el octante original hasta alcanzar el nivel de refinamiento en cada zona. Una vez que se alcanza el nivel, se analizan los octantes que posean más de un punto Steiner, es decir, los octantes que no sean 1-irregulares, los cuales son nuevamente divididos. Cuando los ciclos terminan, se tiene una malla resultante del tipo 1-irregular.

Posterior a esto, se aplican los patrones de transición a los octantes que posean vecinos con distinto nivel de refinamiento, con el fin de obtener una malla topológicamente correcta.

Finalmente, se analizan los elementos que están en la superficie de la malla, removiendo los elementos fuera de ella, aplicando los patrones de superficie, ajustando los nodos que están fuera del dominio, y disminuyendo el efecto escalera entre los elementos de la superficie.

El código del programa posee el patrón de diseño visitante implementado por Sebastián Tobar[7], con el fin de mejorar la calidad del software, en particular mejorar el desempeño en tiempo de ejecución y uso de memoria.

El diagrama de clases resultante se puede apreciar en la *Figura 3.2*, en la cual se presentan

las clases del programa, como también los atributos y métodos más importantes.

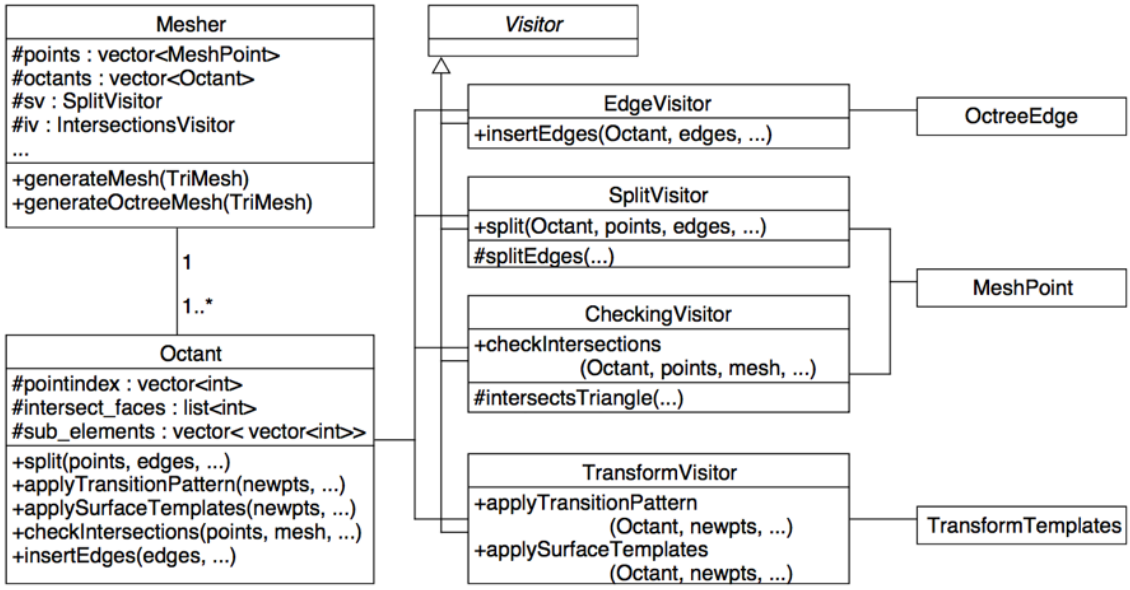


Figura 3.2: Diagrama de clases software mesher[7].

3.2. Polyhedral Tessellation

Por otro lado, la profesora Nancy desarrolló una librería en C++[2], para la generación de mallas basada en la técnica de refinamiento Octree utilizando teselaciones de Delaunay.

A diferencia del software mesher, esta librería trabaja a partir de una malla 1-irregular previamente generada[8]. En particular, se comienza trabajando con los vértices de la malla, los cuales son utilizados para calcular la Teselación de Delaunay. El resultado es un conjunto de tetraedros, el cual no define los elementos que posee la Teselación.

Para determinar que tetraedros conforman un elemento, se analizan los vértices. Si los vértices de los tetraedros son co-esféricos, estos se pueden unir para formar un elemento final.

Una vez que todos los tetraedros han sido analizados, se posee una configuración base de los elementos, es decir, se conocen los vértices que posee cada uno, pero no se tiene la información ni de sus ejes ni de sus caras.

Dado que los vértices de los elementos encontrados anteriormente son co-esféricos, esto implica que también son convexos. Por lo que, al encontrar la cerradura convexa, se obtiene la información de las caras y ejes. Una vez completados estos ciclos, se tiene como resultado una malla volumétrica.

La *Figura 3.3*, representa el diagrama de clases de la librería. En el método generate, de la clase *Polyhedraltessellation*, se aprecia que la librería solo necesita de un conjunto de puntos para calcular la teselación correspondiente. Para este caso se ingresará como input el conjunto de puntos de una malla 1-irregular.

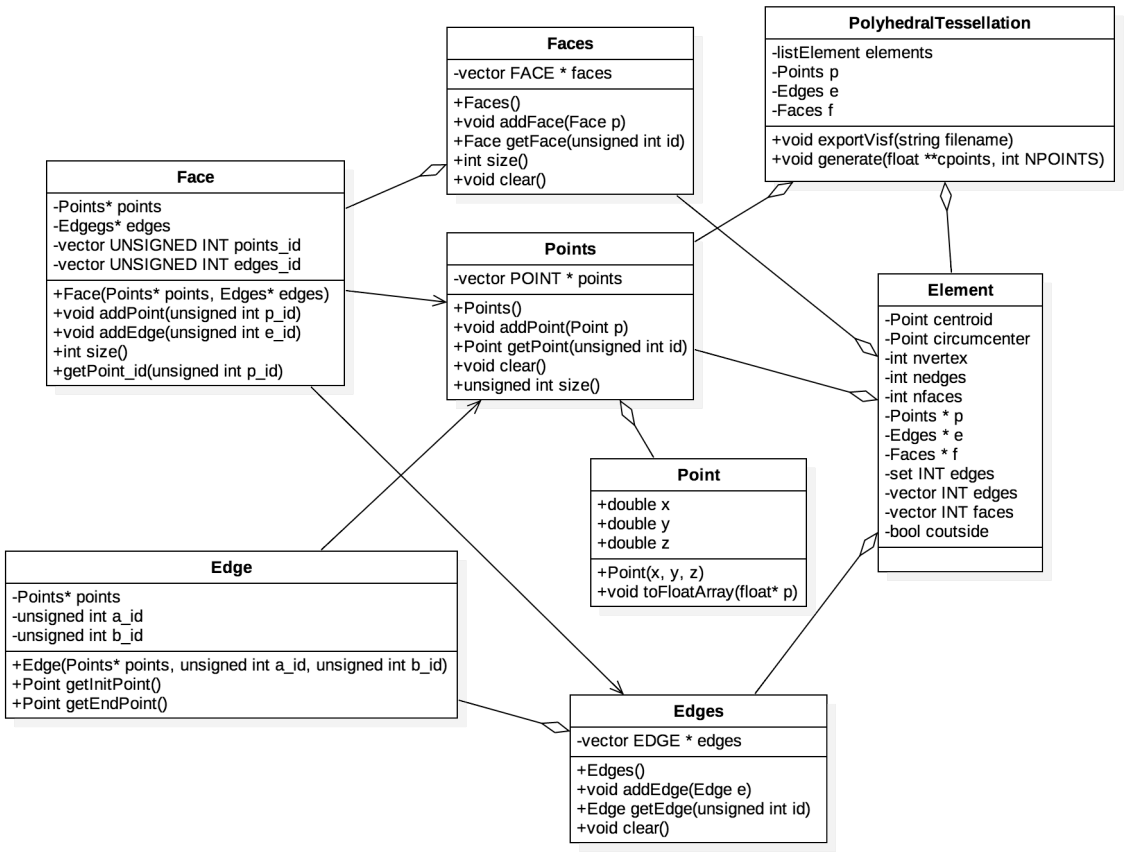


Figura 3.3: Diagrama de Clases librería PolyhedralTessellation.

Capítulo 4

Implementación

4.1. Ambiente

La implementación de la librería, *Polyhedral Tessellation*, en el código *mesher* se llevó a cabo en un notebook, con las siguientes características:

- Modelo: MacBook Pro (Retina, 13-inch, Late 2012)
- Procesador: 2,5 GHz Intel Core i5
- Memoria: 8 GB 1.600 MHz DDR3
- Gráficos: Intel HD Graphics 4.000 1.536 MB
- Almacenamiento: APPLE SSD SD128E
- Sistema Operativo: macOS Sierra 10.12.5
- Editor de código: Sublime Text 2, versión 2.0.2
- Apple LLVM versión 8.1.0 (clang-802.0.42)
- Cmake versión 3.6.0

4.2. Mesher

El software *mesher* está en código C++, y es compatible con las especificaciones mencionadas anteriormente. Sin embargo, para poder visualizar los resultados obtenidos mediante el método de elementos mixtos, se requiere del programa *viewer*.

Viewer consiste en un programa de C++ que permite la visualización de las mallas volumétricas resultantes del software *mesher*. Para instalar correctamente el software *viewer*, se debe contar con la aplicación *geomview*.

Geomview es una aplicación interactiva para visualizar elementos 3D. Está desarrollado para ambientes *Unix*, por lo que el equipo cumple con las características para su correcta ejecución. Para instalar la aplicación, bajo el sistema operativo *macOS Sierra*, se utilizó *MacPorts*, el cual es un proyecto open-source, diseñado para facilitar la compilación, instalación y actualización de softwares para este sistema operativo. Para instalar geomview mediante *MacPorts*, se utiliza la siguiente línea dentro de un terminal:

```
1    sudo port install geomview
```

Código 4.1: Comando instalación geomview utilizando MacPorts

La versión instalada de *geomview* es la 1.9.5.

Con el software de *geomview* instalado, se procede a instalar el programa *viewer*. En primer lugar, se debe ubicar dentro de la carpeta contenedora de los archivos de *viewer* mediante un terminal, luego ejecutar las siguientes líneas:

```
1 cmake src/
2 make
```

Código 4.2: Comandos de instalación de software viewer

Una vez instalado el programa *viewer*, se debe ingresar como parámetro una malla volumétrica o de superficie. Los formatos que recibe son *mdl*, *off*, *m3d* y *mvm*, además se puede habilitar la visualización de los ejes interiores para las mallas volumétricas. *Viewer* viene con un archivo de prueba *test2.mvm*, el cual puede ser visualizado mediante el siguiente comando:

```
./viewer -v test2.mvm
```

Código 4.3: Comando ejecución del software viewer

Al ejecutar la línea anterior, se cargará el programa *geomview*, mostrando una visualización 3D de dos elementos.

Análogamente a *viewer*, el programa *mesher* se instala en la carpeta contenedora del código, y ejecutando:

```
cmake src/
make
```

Código 4.4: Comandos de instalación del software mesher

El programa *mesher* recibe los siguientes parámetros:

- Input: Malla de superficie en formato *mdl*.
- Output: Nombre de la malla volumétrica de salida, la cual se construye en formato *m3d*.
- Nivel de refinamiento en la superficie: Número que indica la cantidad de veces que se dividirá el octante inicial en la superficie de la malla de entrada.
- Nivel de refinamiento de todos los elementos: Número que indica la cantidad de veces que se dividirá el octante inicial en todo el dominio restante de la malla de entrada.
- Región de interés: Archivo de tipo *reg*, que especifica una región dentro del dominio, donde se posee un nivel de refinamiento particular.

Para validar la instalación, el programa viene con una malla de superficie de ejemplo *cortex.mdl*. Para generar la malla volumétrica asociada al archivo córtex, se ejecuta la siguiente línea:

```
1       ./mesher_roi -i cortex.mdl -o volumen -s 5 -a 2
```

Código 4.5: Comando ejecución del software mesher

Donde *-i cortex.mdl*, representa la malla de superficie de entrada, *-o volumen*, representa la malla volumétrica en formato *m3d* generada por el software, *-s 5*, corresponde al nivel de refinamiento de la superficie de la malla, y *-a 2*, corresponde al nivel de refinamiento del resto de la malla.

Para visualizar el resultado, se utiliza el software *viewer*:

```
1       ./viewer -m volumen.m3d
```

Código 4.6: Comando ejecución del software viewer

Además, se ha realizado un corte en el eje z, tomando el espacio entre $0 < z < 95$, para poder visualizar los elementos internos de la malla final.

En la *Figura 4.1* se puede apreciar que el funcionamiento del programa mesher es correcto, dado que, la malla resultante posee en la superficie elementos que han sido refinados (divididos) una mayor cantidad de veces (5) que el resto de los elementos (2).

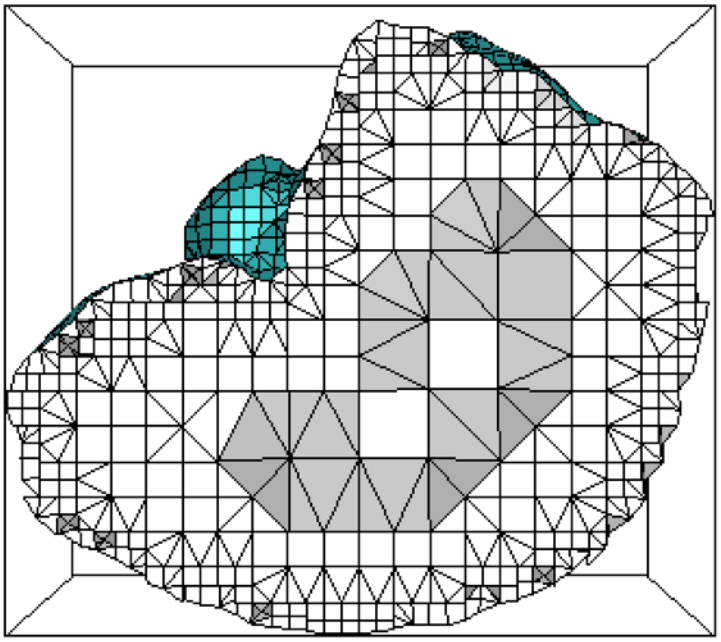


Figura 4.1: Malla volumétrica generada con el software mesher. Elaboración propia

4.3. Polyhedral Tessellation

Previo a implementar la librería *Polyhedral Tessellation*, se debe instalar el programa *Qhull*[9] y la librería *PStreams*[10].

El programa *Qhull* permite la construcción de estructuras tales como la cerradura convexa de puntos, las triangulaciones de Delaunay, diagramas de Voronoi, entre otras. Para este trabajo se utilizará la función relacionada con la construcción de las triangulaciones de Delaunay y el cálculo de la cerradura convexa.

Qhull está desarrollado en C++ y se puede instalar mediante el uso de *Cmake*, para completar la instalación, se debe nuevamente ir al directorio donde se encuentran los archivos de compilación, para este caso en la carpeta build, y ejecutar:

```
cmake ..
make
make install
```

Código 4.7: Comando instalación Qhull

Con el programa ya instalado, se pueden obtener las triangulaciones de Delaunay para generar las mallas volumétricas. Para obtener lo anterior, *Qhull* necesita definir parámetros y un archivo con la información de los vértices de la malla 1 irregular.

Los parámetros que se ingresan en el comando son para indicar que se deben obtener las triangulaciones de Delaunay y que la salida de los vértices debe ser a través de tipo estándar.

Con respecto al formato del archivo, que contiene la información de los vértices, se menciona que debe contener en la primera línea el número de la dimensión en la que se está trabajando. Para nuestro caso el valor será 3, dado que estamos trabajando sobre la representación de elementos en el espacio tridimensional. La segunda línea debe indicar la cantidad de puntos (vértices) que posee la malla 1 irregular. Finalmente, las restantes líneas del archivo poseen la información de las coordenadas de los puntos.

Por otro lado, falta realizar la comunicación entre la librería *Polyhedral Tessellation* y el programa *Qhull*, para esto se utiliza la librería *PStreams*.

PStreams es una librería en C++ que permite la transferencia de información mediante dos programas, utilizando el componente iostream del lenguaje C++. Los beneficios de utilizar esta librería son el uso de interfaces estándares y flexibles de C++, entradas y salidas bidireccionales, entre otros. Otro punto importante a mencionar es que la librería es de software libre.

La librería *Polyhedral Tessellation*, utiliza ambas herramientas dentro del método generate de la clase *PolyhedralTessellation*. Para demostrar lo anterior, se tiene la siguiente sección de código en C++:

```
1    redi::pstream fstcomm("qhull d Qz i");
2    fstcomm << vertexfile.str() << redi::peof;
```

Código 4.8: Ejemplo de uso Qhull y PStreams

Donde *fstcomm* inicia el comando de sistema *qhull* con los parámetros necesarios para generar las triangulaciones de Delaunay, luego se ingresa la información de los puntos mediante la variable *vertexfile.str()*.

Con todos los programas instalados y correctamente implementados en los códigos, se realiza la implementación final de la librería *Polyhedral Tessellation* dentro del código *mesher*.

Como se mencionó en la sección de metodología, el código *mesher* posee un flujo principal en el cual se ingresa una malla de superficie, junto con los niveles de refinamiento especificados en el input del programa. Luego, a través de la técnica Octree, se realiza la división de octantes hasta llegar a una malla 1-irregular.

En este punto se lleva a cabo la implementación de la librería, dado que para generar la malla volumétrica solo se requiere tener como entrada los puntos de la malla 1-irregular.

Antes de introducir la librería al código *mesher*, se prepara el ingreso del parámetro que indica si se debe o no generar la malla volumétrica utilizando el método de los poliedros de Delaunay. Para esto se agrega un nuevo parámetro de inicio, el cual es opcional, dejando el mensaje de ayuda de la siguiente manera:

```
1  void endMsg(){
2      cout << "use: ./mesher [-i] input.mdl [-o] output [-s] ref_level
3          [-a] ref_level [-r] file.reg [-u] input_surface rl
4          [-t] tesselation\n";
5      cout << "where:\n";
6      cout << " input.mdl is the surface input mesh in mdl format\n";
7      cout << " output is the output mesh to be saved in output.m3d\n";
8      cout << " the use one of the following options:\n";
9      cout << "-s (refine surface elements) ref_level is the refinement
10         level\n";
11     cout << " -a (refine all elements) ref_level is the refinement
12         level\n";
13     cout << " -r (refine regions) and file.reg is a text file
14         specifying ";
15     cout << " regions of refinement with a particular level\n";
16     cout << " -u (refine surface region) will refine all the elements
17         in the input_surface at level rl\n";
18     cout << " -t (tessellation patterns) will use polyhedral
19         tessellation\n";
20     }
```

Código 4.9: Ingreso de parámetro tessellation

Se debe indicar un valor -t 1 para que se ejecute el método.

Por otro lado, en la clase mesher se debe incluir la librería mediante:

```
1  #include "Tessellation/polyhedraltessellation.h"
```

Código 4.10: Include de la librería

Luego, se modifica el flujo principal del software *mesher*, en particular en el método *generateOctreeMesh* de la clase *mesher*, donde ahora se recibe el parámetro de *tessellation*.

```
1    void Mesher::generateOctreeMesh(
2        const unsigned short &rl,
3        TriMesh &input,
4        list<RefinementRegion *> &all_reg,
5        const string &name,
6        bool apply_tessellaton){
```

Código 4.11: Método generateOctreeMesh de la clase Mesher

Luego de generar la malla 1-irregular, se verifica el valor del parámetro de *tessellation*, para ver si se aplica o no el método. En caso que se apliquen los patrones de *tessellation*, se ejecuta el siguiente código:

```
1  PolyhedralTessellation tessellation(path, "");
2  int NPOINTS = points.size();
3  float **cpoints = new float*[NPOINTS] ;
4  for (int i=0; i<NPOINTS; i++){
5      cpoints[i] = new float[3];
6      cpoints[i][0] = points[i].getPoint()[0];
7      cpoints[i][1] = points[i].getPoint()[1]
8      cpoints[i][2] = points[i].getPoint()[2];
9  }
10 tessellation.generate(cpoints, NPOINTS);
11 string tess_name = name + "_Tessellation.visf";
12 tessellation.exportVisfBinaryRemovingDuplicatedFaces(tess_name);
```

Código 4.12: Instacia de una teselación, a través de los puntos de la malla 1-irregular

Donde se crea un objeto de la clase *PolyhedralTessellation*, con parámetros *path* y *string* vacío, la variable *path* indica la ruta donde está el ejecutable. Una vez instanciado el objeto, se define una variable con la cantidad de puntos que posee la malla. Luego, se preparan los puntos para ser ingresados como parámetros a través de la función *generate*. En la cual, se genera la malla volumétrica. Finalmente, se exporta la malla mediante el método *exportVisf-BinaryRemovingDuplicatedFaces(tess_name)*.

Para validar la implementación realizada, se utiliza el mismo ejemplo que para mesher, obteniendo la malla de la *Figura 4.2*.

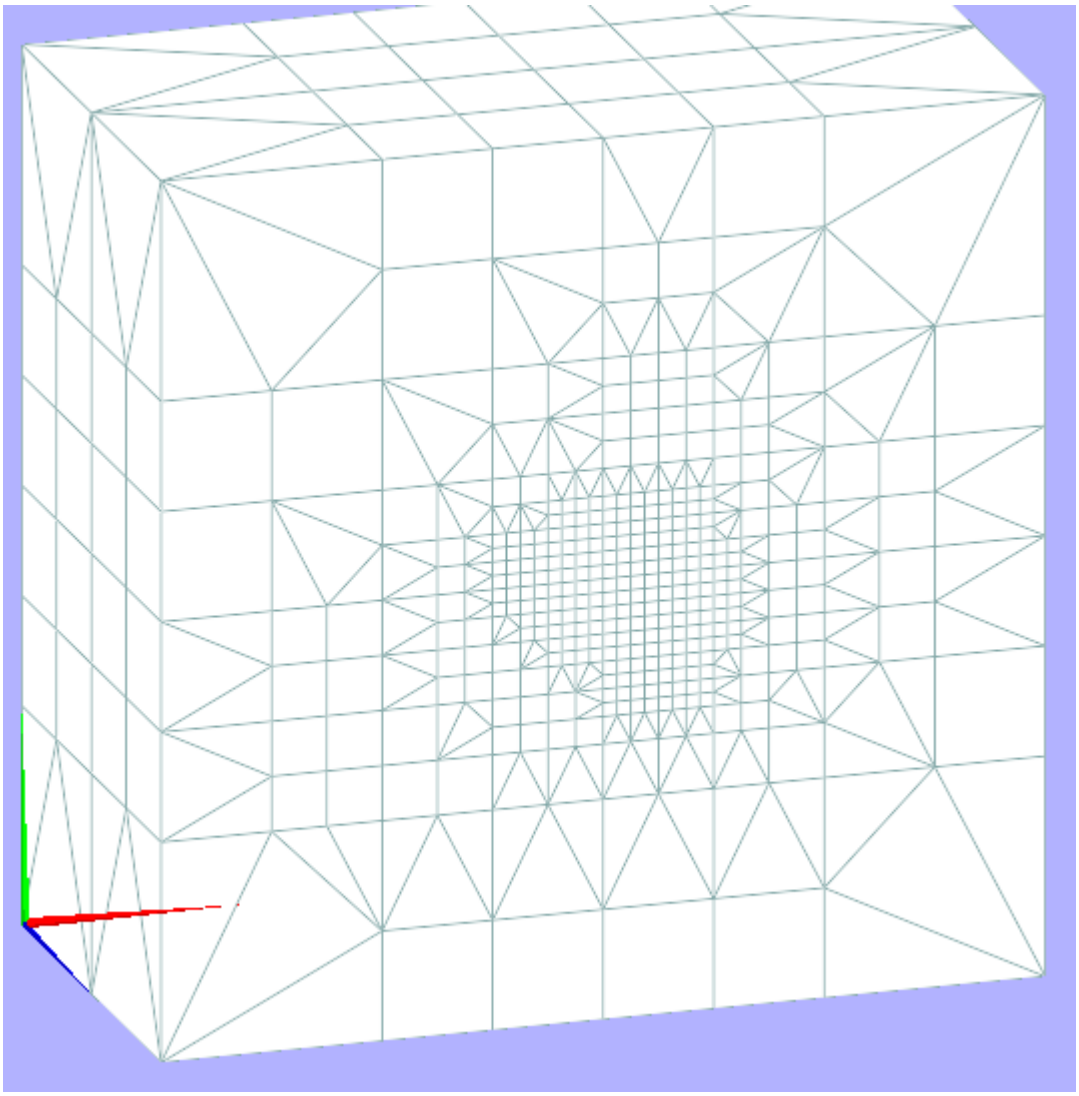


Figura 4.2: Malla volumétrica generada con el software Polyhedral Tessellation. Elaboración propia

De esta manera se logra implementar la librería *Polyhedral Tessellation*, dentro del código mesher.

Para analizar la configuración de las mallas resultantes de esta implementación, se implementó además el software Camarón. La versión utilizada es la de Gonzalo Infante [11]. Para su instalación, se requiere de dos librerías:

- GLEW, librería que provee mecanismos eficientes para determinar las extensiones de OpenGL soportadas por cada máquina.
- GLM, librería matemática desarrollada en C++ diseñada para ser utilizada en aplicaciones gráficas. El principal uso es el trabajo con vectores y matrices.

El programa está desarrollado bajo el ambiente de desarrollo *Qt Creator*[12], el cual es una herramienta para trabajar con aplicaciones con interfaces gráficas de usuarios. Tanto las librerías como la herramienta *Qt Creator* son compatibles con el ambiente de trabajo. Sin embargo, la visualización de las mallas mediante el software Camarón no es posible, dado que, dentro de los códigos utilizados para presentar los elementos, se utilizaron versiones no compatibles con la tarjeta gráfica presente en el dispositivo. Por lo que el software Camarón se utilizará para analizar las estadísticas de las mallas generadas. Dentro de las cuales se enumeran:

1. Cantidad total de poliedros generados.
2. Cantidad total de polígonos generados.
3. Cantidad total de vértices generados.
4. Tipos de Poliedros, cantidades de poliedros en base a la cantidad de caras que poseen.
5. Tipos de polígonos, cantidad de polígonos en base a la cantidad de vértices que poseen.

Capítulo 5

Resultados

Con la librería implementada, se procede a obtener resultados utilizando ambos métodos, analizando la cantidad de poliedros, polígonos, vértices y el tiempo de ejecución de cada algoritmo.

5.1. Detalle del experimento

Para obtener resultados, se trabajó con dos mallas de superficies, córtex y fémur. Además, se utilizó la funcionalidad para definir zonas de interés, generando de esta forma, distintas mallas volumétricas utilizando las mismas mallas de superficies.

El experimento consiste en generar mallas volumétricas con distintos niveles de refinamiento y distintas zonas de interés para dos mallas de superficies. Se registra la cantidad de poliedros, polígonos, vértices y el tiempo de ejecución de cada algoritmo.

5.2. Tipos de entrada

Se generaron 40 tipos de entrada, los cuales se desglosan de la siguiente manera:

- 20 experimentos utilizando el software mesher
  - 10 sobre la malla de superficie córtex
  - 10 sobre la malla de superficie fémur
- 20 experimentos utilizando la librería Polyhedral Tessellation
  - 10 sobre la malla de superficie córtex
  - 10 sobre la malla de superficie fémur

Los 10 experimentos base fueron los mismos tanto para el tipo de programa utilizado, como el tipo de malla de superficie ingresada, se detallan de la siguiente manera:

Exp	RL Superficie	RL Restante	ZI	RL	X %	Y %	Z %
1	4	2	No	-	-	-	-
2	5	2	No	-	-	-	-
3	4	3	No	-	-	-	-
4	4	4	No	-	-	-	-
5	3	2	No	-	-	-	-
6	4	2	Si	5	0-100	0-100	0-30
7	5	2	Si	5	0-10	0-100	0-100
8	4	3	Si	5	0-100	0-100	0-30
9	4	4	Si	5	0-10	0-100	0-100
10	3	2	Si	4	45-55	45-55	45-55

Tabla 5.1: Experimentos realizados sobre los software mesher y Polyhedral Tessellation

Donde:

- NR Superficie, corresponde al nivel de refinamiento indicado para la superficie.

- NR Restante, corresponde al nivel de refinamiento indicado para el resto del dominio.
- ZI, corresponde a si se ingresa o no, una zona de interés.
- NR, corresponde al nivel de refinamiento indicado para la zona de interés.
- X, Y, Z, corresponden a la porción de la zona de interés en cada plano.

5.3. Resultados

Los resultados de los experimentos, se agrupan en base al tipo de programa y malla de superficie utilizadas.

5.3.1. Mesher - Córtex

Los resultados obtenidos mediante el software *mesher*, y utilizando la malla de superficie córtex, se ven en la *Tabla 5.2*, donde se puede ver inmediatamente que el experimento más complejo fue el número 8, dado que presenta una mayor cantidad de poliedros, polígonos, vértices y un mayor tiempo de ejecución. En su contraparte, el experimento 5 fue el que necesito menos tiempo de ejecución como también menor cantidad de elementos generados. Esto está directamente relacionado con los niveles de refinamiento definidos para cada experimento.

El experimento 8, posee una zona de interés del 30 % del volumen total con nivel 5, mientras que para la superficie un nivel 4 y para el resto del dominio un nivel 3. El experimento 5 a su vez, no posee zona de interés, el nivel de refinamiento de la superficie es de 3 y el del resto del dominio solo 2.

Otro valor a mencionar son los tipos de poliedros utilizados por este método, se registraron poliedros de 4 caras (tetraedros), 5 caras (pentaedros) y 6 caras (hexaedros). Para todos los experimentos a excepción del número 8, los poliedros más utilizados fueron los pentaedros, para la excepción el poliedro más utilizado fue el hexaedro. Estos datos se ven reflejados en la *Tabla A.1* del anexo.

Exp	Poliedros	Polígonos	Vértices	Tiempo (ms)
1	5.111	12.542	2.230	1.871
2	26.209	63.805	10.941	5.538
3	5.111	12.542	2.230	1.885
4	4.506	12.094	2.967	2.075
5	798	2.026	411	1.241
6	13.767	35.622	7.869	3.127
7	26.147	63.766	11.018	5.769
8	68.360	186.897	49.447	9.171
9	12.837	33.499	7.636	3.687
10	4.930	12.194	2.288	1.369

Tabla 5.2: Resultados obtenidos para la el software mesher y malla de superficie córtex.

5.3.2. Mesher - Fémur

Los resultados obtenidos para la malla de superficie del fémur, se ven en la *Tabla 5.3*. El análisis que se puede realizar sobre este caso es similar. Dado que nuevamente el experimento 8 fue el que más elementos y tiempo utilizo. Mientras que el experimento 5 fue nuevamente el que menos tiempo necesito y menos elementos generó.

Con respecto a los poliedros utilizados, hubo mayor diferencia, dado que para algunos experimentos se utilizaron más comúnmente los pentaedros y para otros los hexaedros. Los resultados de esta medición se encuentran en la *Tabla A.2* del anexo.

5.3.3. Polyhedral Tessellation - Córtex

Los resultados obtenidos mediante el uso de la librería de poliedros de Delaunay, y utilizando la malla de superficie córtex, se ven en la *Tabla 5.4*. Para este método, el experimento que generó mayor cantidad de elementos generados fue el número 7. Sin embargo, el que más tiempo utilizó fue el número 2. Mientras que el experimento que genero menos elementos y utilizo menos tiempo fue el mismo que para mesher, es decir, el experimento número 5.

Exp	Poliedros	Polígonos	Vértices	Tiempo (ms)
1	9.101	23.177	4.875	2551
2	47.380	119.560	23.950	8440
3	9.101	23.177	4.875	2421
4	8.451	23.610	6.611	2519
5	1.271	3.377	837	911
6	29.671	82.167	22.482	5367
7	47.327	119.791	24.293	8735
8	178.186	511.303	153.477	19429
9	23.746	62.908	15.186	5068
10	9.121	23.134	4.768	1839

Tabla 5.3: Resultados obtenidos para la el software mesher y malla de superficie fémur.

Con respecto a los poliedros utilizados para generar las mallas, el método de las teselaciones genero elementos con una mayor cantidad de caras, utilizando en casi todos octaedros (poliedros de 8 caras). El resumen con la cantidad de poliedros agrupados por la cantidad de caras, está en la *Tabla A.3* del anexo.

5.3.4. Polyhedral Tessellation - Fémur

Por último, los resultados obtenidos para la superficie del fémur, se ven en la *Tabla 5.5*. Para este experimento, se presentaron problemas serios de rendimiento, donde el uso de memoria colapso el ambiente de pruebas. Logrando registrar resultados en tan solo 5 de los 10 experimentos. El experimento que menos elementos generó y menos tiempo utilizó fue el número 5. Mientras que no se pudo determinar cuál fue el que más costó generar, dado que el sistema no fue capaz de finalizar los experimentos 2, 6, 7, 8 y 9.

Con respecto a los poliedros utilizados, para este caso se amplió aún más la variedad, llegando a generar poliedros de hasta 10 caras en 3 de los 5 experimentos, los cuales se pueden observar en la *Tabla A.4* del anexo.

Exp	Poliedros	Polígonos	Vértices	Tiempo (ms)
1	8.163	21.534	4.961	188.242
2	35.397	91.769	20.281	4.083.702
3	8.163	21.534	4.961	197.430
4	8.087	22.078	5.732	266.024
5	1.751	4.861	1.299	24.973
6	20.067	53.376	12.978	1.419.343
7	35.419	91.886	20.364	3.908.036
8	20.067	53.376	12.978	1.362.857
9	10.438	27.895	6.830	412.493
10	2.058	5.598	1.409	26.851

Tabla 5.4: Resultados obtenidos para la librería Polyhedral Tessellation y malla de superficie córtex.

5.4. Análisis Comparativo

Dada la variedad en los resultados, es importante realizar un análisis comparativo entre ambas técnicas bajo los mismos experimentos. Este análisis se agrupará en dos partes, uno para comparar los resultados ante el uso de la malla de superficie del córtex, y otro para la malla de superficie del fémur.

5.4.1. Córtex

De los resultados obtenidos en la sección 5.3.1 y 5.3.3, se generaron 4 gráficos *Figura 5.1* que reflejan el comportamiento que tuvieron ambos algoritmos ante un mismo set de experimentos. En ellos se aprecia que en cuanto a la cantidad de poliedros, polígonos y vértices actuaron de manera muy similar.

El único experimento que generó una diferencia considerable fue el número 8. Esta diferencia puede estar asociada a los poliedros que se utilizaron para generar la malla. Dado que

Exp	Poliedros	Polígonos	Vértices	Tiempo (ms)
1	18340	47686	10786	1095281
2	-	-	-	-
3	18340	47686	10786	1197965
4	18086	48825	12548	1249716
5	3779	10280	2653	64223
6	-	-	-	-
7	-	-	-	-
8	-	-	-	-
9	-	-	-	-
10	11689	30194	6610	647834

Tabla 5.5: Resultados obtenidos para la librería Polyhedral Tessellation y malla de superficie fémur.

el software *mesher* genera tetraedros, pentaedros y hexaedros, mientras que, para el método de las teselaciones, los poliedros utilizados van desde los tetraedros hasta los octaedros (poliedros de 8 caras).

Con respecto al tiempo de ejecución, la diferencia es considerable, mientras que para el método mesher todos los experimentos no superaron los 10 segundos de ejecución, para el software de las teselaciones el experimento que registró menos tiempo fue el 5 con 24 segundos, mientras que el más alto fue el número 2 con aproximadamente 68 minutos.

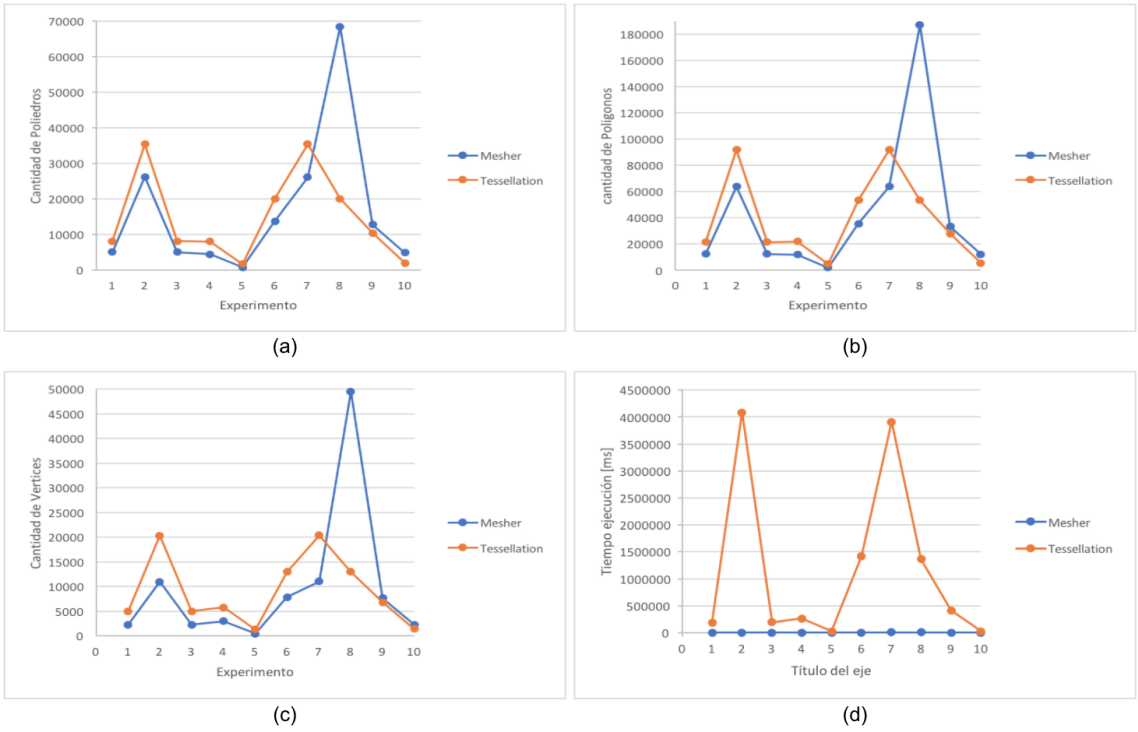


Figura 5.1: (a) Gráfico de cantidad de poliedros vs experimento, (b) gráfico de cantidad de polígonos vs experimento, (c) gráfico de cantidad de vértices vs experimento y (d) gráfico de tiempo vs experimento.

5.4.2. Fémur

Análogamente, a la comparación anterior se generaron 4 gráficos representando la cantidad de elementos generados y el tiempo de ejecución de los algoritmos. Para este análisis solo se consideraron los experimentos que pudieron completarse para el método de las teselaciones, es decir, solo se consideraron los experimentos 1, 3, 4, 5 y 10.

Con respecto a la cantidad de elementos generados, el método de las teselaciones registro el doble de elementos en casi todos los experimentos en comparación con el método mesher. Mientras que, el tiempo de ejecución de nuevo generó una brecha importante entre ambos. Dejando al software *mesher* con un máximo de 19 segundos para el experimento 8, mientras que el mínimo de las teselaciones fue de 1 minuto para el experimento 5.

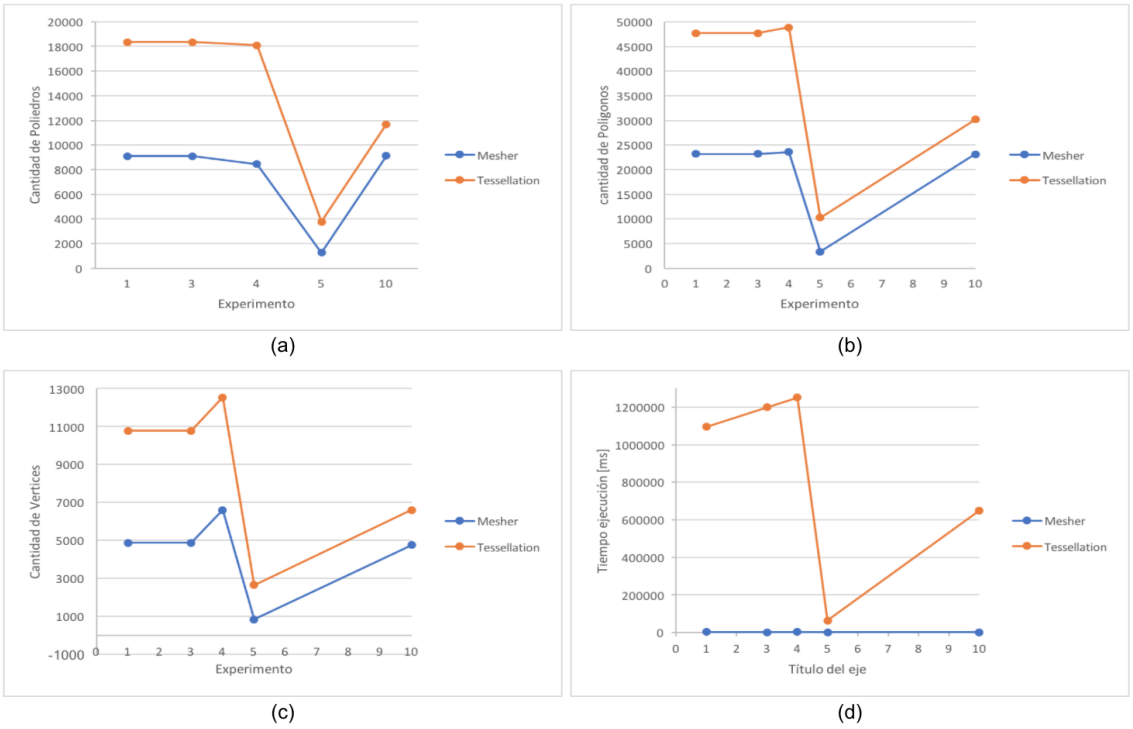


Figura 5.2: (a) Gráfico de cantidad de poliedros vs experimento, (b) gráfico de cantidad de polígonos vs experimento, (c) gráfico de cantidad de vértices vs experimento y (d) gráfico de tiempo vs experimento.

Conclusiones

Dada la necesidad de generar mejoras a los algoritmos de generación de mallas volumétricas a partir de mallas de superficie, se han estudiado y comparado dos métodos.

Ambos se basan en la técnica de refinamiento Octree, el cual consiste en encerrar el objeto en estudio dentro de un octante, el cual es dividido recursivamente hasta alcanzar el punto deseado. Una vez finalizada la división, se analizan los puntos de Steiner, los cuales son los que se encuentran dentro de un eje. Para que una malla se considere 1-irregular debe satisfacer la condición de que no puede existir más de un punto Steiner en cada uno de sus ejes. Para solucionar esta problemática, si un octante posee dentro de uno de sus ejes más de un punto Steiner, este debe dividirse. Una vez que todos los ejes de todos los octantes, es decir, todos los ejes de la malla, posean a lo más un punto Steiner, la malla se considera 1-irregular, por lo que se pueden aplicar los dos algoritmos de generación de mallas volumétricas.

Por un lado, se tienen los patrones de transición y superficies, utilizando elementos mixtos. Los cuales mediante la utilización de un algoritmo modificado de Octree, genera mallas tipo 1-irregulares. Para los octantes que presentan puntos de Steiner, se utilizan 325 patrones de transición para generar elementos mixtos que ayudan a generar una malla topológicamente correcta. Luego de aplicar estos patrones, se aplican los patrones de superficie, los cuales ayudan a mejorar la calidad de las mallas en los bordes del objeto.

Por el otro lado, se tiene el proceso de generación de mallas utilizando poliedros de Delaunay. Estos poliedros tienen la característica de satisfacer las teselaciones de Delaunay, la cual verifica que cada poliedro circunscrito dentro de una esfera, no posea nodos de otros poliedros dentro de ella. Uno de los beneficios de esto, es que todos los elementos generados son

co-esféricos, por lo que, al calcular la cerradura convexa, se pueden definir los ejes y caras de cada uno de ellos, generando finalmente una malla volumétrica.

Luego del análisis de los programas, se concluyó que la implementación de la librería Polyhedral Tessellation, debía hacerse dentro del método generateOctreeMesh de la clase Mesher, dado que ahí es donde se genera la malla 1-irregular, y desde donde se dividen los caminos para generar una malla volumétrica con uno u otro método.

Dado el trabajo realizado se puede concluir que el objetivo principal fue cumplido, pues fue posible integrar de manera sencilla la librería Polyhedral Tessellation dentro del software Mesher. Esto debido a que, en el análisis de los códigos, se pudo apreciar rápidamente donde realizar la integración, puesto que, ambos algoritmos trabajan sobre la base del método Octree, no fue necesario realizar ningún ajuste a la malla 1-irregular generada, ni tampoco a las clases utilizadas por ambos métodos, ya que funcionan de manera independiente.

Previo a lo anterior, se tuvo que preparar el ambiente de trabajo, el cual fue montado exitosamente para todas las herramientas, salvo el caso del software Camarón para la visualización de las mallas volumétricas. Sin embargo, se pudo utilizar el software Viewer para observar las mallas 1-irregulares y las generadas por la aplicación mesher.

Para obtener los resultados estadísticos, se utilizó el software Camarón, el cual posee una opción para desplegar la información relacionada con las cantidades de elementos generados. Además, presenta una clasificación de los tipos de elementos generados. Se registraron las cantidades de poliedros, polígonos, y vértices de las mallas generadas.

Con respecto a los poliedros, se registró la cantidad total como también los tipos de poliedros utilizados, registrando para mesher poliedros de 4, 5 y 6 caras, en los 20 experimentos realizados. Mientras que, para la librería de las teselaciones, se registraron poliedros de 4 a 8 caras para el caso de la superficie córtex, y de 4 a 10 caras para el caso del fémur. De este último, se observó que mientras más poliedros de múltiples caras se utilizan, menor será la cantidad total de elementos necesarios en la malla final.

El factor que genero gran diferencia entre ambos métodos fue el tiempo de ejecución, el cual en todos los experimentos fue menor en el software mesher, con tiempos entre los 0,911

segundos para el experimento 5 del fémur, y 19,429 segundos en el experimento 8 de la misma superficie, mientras que el uso de la librería Polyhedral Tessellation utilizó tiempos entre los 24,973 segundos en el experimento 5 del córtex, y registro el máximo tiempo en 68 minutos para el experimento 2 de la misma superficie.

Punto aparte son los experimentos que no se pudieron completar para el caso de la malla del fémur, utilizando las teselaciones. Puesto que, dos de los ciclos para preparar las caras y ejes de la malla utilizaban acceso al disco, mediante un archivo temporal. Al tener un nivel de refinamiento mayor, implicaba generar demasiados elementos, los cuales no alcanzaban a ser almacenados, generando el colapso del sistema y dejando inconclusa la ejecución del programa. Pese a esto, se pudo observar que uno de los experimentos generó menor cantidad de elementos en comparación a mesher, debido a que se utilizan elementos más complejos para generar las mallas, tomando por ejemplo poliedros de 10 caras

A nivel general el software mesher presenta un mejor rendimiento ante esto, dado que es capaz de recibir mallas y niveles de refinamiento mucho más complejas, sin aumentar tanto los costos de ejecución.

Con respecto al trabajo a futuro a realizar se nombran:

- Emplear el método de generación de poliedros de Delaunay a cada elemento de la malla 1-irregular.

  En esta memoria se trabajó sobre la generación de los poliedros tomando todos los puntos de la malla 1-irregular. Esto afecta directamente el rendimiento del algoritmo, ya que luego de obtener la configuración de las teselaciones, el algoritmo calcula la información sobre las caras y ejes que no están definidos. Al trabajar sobre todo el espacio, los ciclos que recuperan esta información incrementan su tiempo de ejecución. Por lo que si se trabaja por cada elemento 1-irregular encontrando su Teselación asociada, y luego juntar todos los elementos. Se podría disminuir los costos de ejecución y obtener quizás mejores resultados.

- Mejorar el rendimiento de la librería Polyhedral Tessellation

Como se comentó anteriormente, uno de los puntos que más genero tiempo de ejecución y calculo, fue el de reconstruir las caras y ejes de los elementos. Esto se realizó mediante el uso de ciclos extensos y el acceso recurrente al disco del sistema. Se podría trabajar en mejorar el diseño de los ciclos y buscar alguna alternativa para no acceder al disco de manera recurrente, guardando información más precisa.

- Agregar otro tipo de poliedros al algoritmo mesher.

  La librería de poliedros de Delaunay, entrego resultados interesantes sobre la superficie del fémur. En la cual, al poder incorporar poliedros de mayor número de caras, se obtuvieron menos elementos generados. Por lo que, se podría analizar la posibilidad de incorporar más tipos de poliedros al software mesher, incorporando los respectivos patrones de transición asociados, en post de generar mallas volumétricas con menor cantidad de poliedros.

Apéndice A

Tablas

Exp	Nº Poliedros	Nº 4 caras	Nº 5 caras	Nº 6 caras
1	5.111	2.095	2.560	456
2	26.209	10.677	13.093	2.439
3	5.111	2.095	2.560	456
4	4.506	1.133	1.750	1.623
5	798	292	419	87
6	13.767	4.309	5.262	4.196
7	26.147	10.557	13.039	2.551
8	68.360	13.482	17.199	37.679
9	12.837	3.973	4.742	4.122
10	4.930	1.740	2.082	1.108

Tabla A.1: Detalle de los poliedros generados mediante el software mesher, tomando la malla de superficie córtex.

Exp	Nº Poliedros	Nº 4 caras	Nº 5 caras	Nº 6 caras
1	9.101	2.911	4.842	1.348
2	47.380	4.386	26.005	6.989
3	9.101	2.911	4.842	1.348
4	8.451	1.422	3.054	3.975
5	1.271	341	723	207
6	29.671	5.023	8.903	15.745
7	47.327	14.116	25.765	7.446
8	178.186	16.914	29.626	131.646
9	23.746	6.829	8.731	8.186
10	9.121	2.707	3.890	2.524

Tabla A.2: Detalle de los poliedros generados mediante el software mesher, tomando la malla de superficie fémur.

Exp	Nº Poliedros	Nº 4 caras	Nº 5 caras	Nº 6 caras	Nº 7 caras	Nº 8 caras
1	8163	1839	3168	3022	129	5
2	35397	8347	14036	12496	499	19
3	8163	1839	3168	3022	129	5
4	8087	1463	2306	4253	61	4
5	1751	358	552	808	33	0
6	20067	4307	6309	9299	147	5
7	35419	8330	13960	12619	491	19
8	20067	4307	6309	9299	147	5
9	10438	2177	3353	6840	64	4
10	2058	450	730	852	25	1

Tabla A.3: Detalle de los poliedros generados mediante la librería Polyhedral Tessellation, tomando la malla de superficie córtex.

Nº Caras / Exp	4	5	6	7	8	9	10
1	4370	7141	6697	126	4	1	1
2	-	-	-	-	-	-	-
3	4370	7141	6697	126	4	1	1
4	3516	4973	9538	54	3	1	1
5	794	1316	1641	28	0	0	0
6	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-
10	2970	4618	4051	50	0	0	0

Tabla A.4: Detalle de los poliedros generados mediante la librería Polyhedral Tessellation, tomando la malla de superficie fémur.

Bibliografía

[1] M. Yerry, M. ;Shephard. Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal of Numerical Methods in Engineering 20*, 1984.

[2] N. Contreras, D. ; Hitschfeld-Kahler. Generation of polyhedral delaunay meshes. *23rd International Meshing Roundtable IMR23*, 2014.

[3] Lobos C Gonzalez E. Mixed-element octree: a meshing technique toward fast and real-time simulations in biomedical applications. *International journal for numerical methods in biomedical engineering*, 2015.

[4] B. Delaunay. Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 1934.

[5] P. ;Fichtner W. Hitschfeld, N ; Conti. Mixed elements trees: A generalization of modified octrees for the generation of meshes for the simulation of complex 3-d semiconductor devices. *IEEE Trans. on CAD/ICAS 12*, 1993.

[6] Lobos C. Towards a unified measurement of quality for mixed–elements. *technical report n 2015/01, di–utfsm*, 2014.

[7] Tobar S. Mejoramiento de la mantenibilidad y extensibilidad de una herramienta de generación de mallas volumétricas. *Memoria de titulación para optar al título de Ingeniero Civil Informático, UTFSM, Chile*, 2016.

[8] N. Hitschfeld-Kahler. Generation of 3d mixed element meshes using a flexible refinement approach. *Engineering with Computers 21*, 2005.

[9] D. P.; Huhdanpaa H. Barber, C. B. ; Dobkin. The quickhull algorithm for convex hulls. *ACM Transactions on mathematical software 22*, 1996.

[10] Pstreams. `http://pstreams.sourceforge.net/`. Consultado el 01 de Junio 2017.

[11] Software camarón. `https://sourceforge.net/u/gloix/camaron/ci/master/tree/`. Consultado el 01 de Junio 2017.

[12] Qt creator. https://www.qt.io/ide/. Consultado el 01 de Junio 2017.