



UNIVERSIDAD NACIONAL AUTÓNOMA  
DE MÉXICO

---

---

FACULTAD DE CIENCIAS

Generación de Mallas para la Visualización de Estructuras  
Tubulares

T E S I S

QUE PARA OBTENER EL TÍTULO DE:  
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

MARCO ANTONIO CABALLERO GUERRERO

TUTOR: DRA. MARÍA ELENA MARTÍNEZ PÉREZ

COTUTOR: M. EN C. ALEJANDRO AGUILAR SIERRA

2011





Universidad Nacional  
Autónoma de México



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

## Hoja de Datos del Jurado

<b>1. Datos del Alumno</b> Caballero Guerrero Marco Antonio 55 40 75 00 11 Universidad Nacional Autónoma de México Facultad de Ciencias Ciencias de la Computación 406061342
<b>2. Datos del Tutor</b> Dra. María Elena Martínez Pérez
<b>4. Datos del Sinodal 1</b> Mat. Ana Luisa Solís González-Cosío
<b>5. Datos del Sinodal 2</b> M. en C. Guilmer Ferdinand González Flores
<b>3. Datos del Sinodal 3</b> M. en C. Alejandro Aguilar Sierra
<b>6. Datos del Sinodal 4</b> M. en C. Joel Espinosa Longi
<b>7. Datos del trabajo escrito</b> Generación de Mallas para la Visualización de Mallas Tubulares 102 páginas 2011

# Resumen

El trabajo realizado en la presente tesis comprende el desarrollo de un algoritmo de generación de mallas tubulares, el módulo correspondiente que permite la visualización por computadora de dichas estructuras, así como el diseño y codificación de una interfaz gráfica que integra las dos funcionalidades anteriores, con la cuál se pueden observar fácilmente los modelos construidos.

La aplicación, con fines de asistencia médica en la detección y tratamiento de enfermedades de los vasos retinales, constituye la siguiente fase sobre el proceso realizado por el sistema anterior: RISA (*Retinal multiScale System Analysis*) [Martinez-Perez *et al.*, 2007], con el fin de representar con una vista en tres dimensiones, los vasos sanguíneos señalados, interpretando los datos de entrada proporcionados por el mencionado sistema.

El procedimiento para la generación de las mallas en este trabajo utiliza un refinamiento iterativo basado en el algoritmo original de subdivisión de Catmull-Clark, el cual es aplicado a una malla de puntos de estructura sencilla que es construida mediante el método de *cilindros generalizados* y el uso de *índices de referencia* de las posiciones de ciertos vértices de aquellos segmentos que forman cada bifurcación.

Por último, las bifurcaciones son tratadas como bloques de estructuras “independientes“, por lo que puede aplicarse subdivisión a cada uno de esos bloques que la conforman de manera individual. Es ésta característica la que puede ayudar a mejorar el desempeño de la aplicación, optimizando los procedimientos como parte del trabajo a futuro.

# Agradecimientos

Quisiera dedicar este trabajo a mis padres, a mi hermano y a cada uno de los miembros de mi familia y amigos que estuvieron apoyándome en todo momento.

Agradezco a la UNAM, a la Facultad de Ciencias y a mis tutores, Dra. Elena Martínez y M. en C. Alejandro Aguilar, por su tiempo y paciencia en esta etapa de mi formación; a CONACyT por otorgarme la beca como parte del proyecto CB-2007-83088.

Gracias.

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	2
1.1.1. Importancia de los vasos sanguíneos . . . . .	2
1.1.2. Los vasos sanguíneos en el ojo humano . . . . .	3
1.2. Definición del problema . . . . .	4
1.3. Objetivos . . . . .	6
1.4. Estructura . . . . .	6
<b>2. Generación de Mallas</b>	<b>7</b>
2.1. Métodos numéricos y mallas . . . . .	7
2.2. Elementos de una malla . . . . .	8
2.3. Clasificación de métodos de mallado . . . . .	9
2.4. Remallado ( <i>Remeshing</i> ) . . . . .	10
2.4.1. Clasificación . . . . .	11
2.5. Subdivisión . . . . .	15
2.5.1. Características . . . . .	16
2.5.2. <i>Puntos polo</i> . . . . .	19
2.6. Mallas tubulares . . . . .	21
<b>3. Método propuesto</b>	<b>27</b>
3.1. Primera aproximación . . . . .	27
3.1.1. Obtención de puntos de la malla . . . . .	28
3.1.2. Trazado de líneas . . . . .	28
3.1.3. Resultados preliminares . . . . .	28
3.2. Construcción del modelo de malla . . . . .	31
3.2.1. Cilindros generalizados . . . . .	31
3.2.2. Malla canónica . . . . .	33
3.3. Refinamiento . . . . .	40
<b>4. Diseño y Desarrollo del Sistema</b>	<b>47</b>
4.1. Especificación de requerimientos . . . . .	47

4.2. Diseño de la aplicación . . . . .	49
4.2.1. Arquitectura . . . . .	49
4.2.2. Patrones del diseño . . . . .	50
4.2.3. Estructura de clases . . . . .	51
4.2.4. Diagramas de secuencia . . . . .	53
<b>5. Experimentación y Resultados</b>	<b>59</b>
5.1. Interfaz de Usuario . . . . .	59
5.2. Visualización de las mallas . . . . .	65
5.3. Evaluación y Análisis . . . . .	75
5.3.1. Dimensiones de los modelos . . . . .	76
5.3.2. Volumen . . . . .	81
5.3.3. Rendimiento . . . . .	84
<b>6. Conclusiones y trabajo a futuro</b>	<b>86</b>
<b>Apéndices</b>	<b>89</b>
<b>A. Manual del Sistema</b>	<b>89</b>
A.1. Primeros Pasos . . . . .	89
A.1.1. Ajuste de la cámara . . . . .	90
A.2. Cargar archivo . . . . .	91
A.3. Generar malla . . . . .	91
A.4. Subdividir malla . . . . .	92
A.5. Modificar atributos . . . . .	92
A.6. Capturar escena . . . . .	93
A.7. Guardar proyecto . . . . .	94
<b>Bibliografía</b>	<b>95</b>

# Capítulo 1

## Introducción

La visualización por computadora se puede definir como el proceso de explorar, transformar y mostrar datos en forma de imágenes y simulaciones con el fin de apreciar y comprender adecuadamente las características de los mismos. En tal sentido, la disciplina de la visualización médica surge como una necesidad de aprovechar, así como de asimilar, la enorme cantidad de datos producidos por los modernos sistemas de captura y procesamiento de imágenes en dicho ámbito.

Hoy en día, las aplicaciones de tratamiento y análisis de imágenes aparecen frecuentemente integradas en software de visualización especializado, que además de detectar y localizar zonas de interés, permite a los usuarios (radiólogos, cirujanos, etc.) tener diagnósticos más acertados. De esta forma, la disciplina de la visualización encuentra en tales tareas un extenso campo de trabajo, en el que médicos y especialistas demandan de forma creciente sistemas avanzados que interpreten fiel y claramente los datos procesados y proporcionen herramientas que faciliten el análisis médico. Por tanto, resulta de importancia el uso de métodos y esquemas apropiados para su representación, entre otras, la generación de modelos 3D mediante mallas.

Así, desde reducidos mapas de alturas representando los estratos de la piel, pasando por variadas geometrías partícipes en la generación de estructuras y mallas, hasta llegar a completos modelos del cuerpo humano, o bien, de órganos específicos, se presentan técnicas diversas de visualización científica capaces de recrear los complejos sistemas y funciones de los organismos vivos, que además de evidenciar el uso del poder de cómputo actual muestran un perfecto escenario de encuentro entre la teoría y la práctica de las Ciencias de la Computación.

## 1.1. Motivación

Como habrá notado todo estudiante, docente y demás personas conocedoras del tema, en la literatura se reportan numerosos procedimientos para adquirir imágenes, detectar regiones a evaluar, reconstruir determinado órgano físico, etc; todo ello con el objetivo de brindar un análisis más preciso acerca de lo que en realidad ocurre con tal estructura.

En particular, dentro de las posibles estructuras tubulares que podemos encontrar en el cuerpo humano, se encuentran los vasos sanguíneos, que son de suma importancia. Por ello, no es sorpresa que, desde hace varias décadas, tan sólo en las técnicas de modelado y captura fotomédica hallemos una amplia gama de éstos ejemplos, entre los que se encuentran modalidades como angiografías de rayos X, resonancia magnética, angiografías por ultrasonido, o algún otra. Conducir a una adecuada interpretación de los datos que éstas arrojan se convierte en un reto considerable, más aún cuando las exigencias médicas así lo determinan, pues cuanto mejor sea la representación mejor es el diagnóstico y el tratamiento.

Con pequeñas observaciones como ésta última, es imprescindible distinguir por tanto qué método de modelado se adapta mejor a los datos de entrada conseguidos y a los requerimientos de los usuarios.

### 1.1.1. Importancia de los vasos sanguíneos

Comúnmente, la geometría vascular observada aparece constituida por un cuerpo central rodeado de ramificaciones periféricas, o todo un completo esquema arborescente de ramas y bifurcaciones, donde el estudio médico se enfoca en detectar aquellas zonas a lo largo de los tubos que presentan anomalías, distinguiendo, en ocasiones, ciertos cambios peculiares en el grosor y/o bifurcación de los mismos, dictando posiblemente la presencia de enfermedades como *ateroesclerosis*<sup>1</sup>, *aneurismas*<sup>2</sup>, *trombosis*<sup>3</sup>, entre otras.

Ahora bien, es necesario recordar que el sistema circulatorio forma parte de la gran mayoría de los procesos de nuestro organismo, transportando los nutrientes de los alimentos y oxígeno a todas las células del cuerpo a través de la sangre, se encarga

---

<sup>1</sup>La aterosclerosis es un síndrome caracterizado por el depósito e infiltración de sustancias lipídicas en las paredes de las arterias de mediano y grueso calibre. Es la forma más común de arteriosclerosis. Provoca una reacción inflamatoria y la multiplicación y migración de las células musculares lisas de la pared, que van produciendo estrechamientos de la luz arterial.

<sup>2</sup>Un aneurisma es una dilatación parecida a un globo que se produce en una arteria. Se produce cuando la presión de la sangre que pasa por una parte de una arteria debilitada empuja la pared hacia afuera, formando lo que podría describirse como una ampolla.

<sup>3</sup>Formación de coágulos dentro de los vasos sanguíneos.

de recoger los desechos metabólicos que se han de eliminar, interviene en las defensas del organismo, regula la temperatura corporal, etc. Por ende, parece lógico pensar que si en determinado momento los conductores encargados, esto es, las venas y arterias, padecen alguna afección, ello repercutirá en el resto de las funciones vitales. Además, no todo termina ahí, la comunicación es tal que la situación inversa también aparece, la falla de algún órgano podría llevar a un estado de malestar sobre algún otro órgano o a los vasos sanguíneos directamente, debido al continuo e inevitable envío y recepción de sustancias.

### 1.1.2. Los vasos sanguíneos en el ojo humano

Hablar de los vasos sanguíneos de un ser vivo, incluso sólo en el cuerpo humano, presenta una enorme cantidad de procesos y enfermedades relacionados con el funcionamiento del mismo. Bajo esta premisa, resulta importante señalar la relevancia de un grupo reducido: la vascularidad de la retina, debido a que las estructuras del ojo son transparentes, son las únicas en todo el cuerpo humano que puede ser observada a detalle directamente desde afuera, sin la necesidad de realizar daño a tejido.

De esta manera, un médico puede apreciar internamente el ojo y, lo más importante, síntomas de numerosas afecciones del cuerpo. Además, el ojo está compuesto de varias capas de tejidos, característica que lo hace susceptible a una gran variedad de enfermedades así como también dar entrada a muchos sistemas del cuerpo. Casi cualquier parte del ojo puede ser una clave importante hacia el primer diagnóstico de una *enfermedad sistémica*<sup>4</sup>. Señales de una *enfermedad sistémica* pueden ser evidentes en la superficie exterior del ojo (párpado, conjuntiva y córnea), centro del ojo o la parte trasera del ojo (retina).

Numerosos estudios han mostrado que muchos de los padecimientos oculares se manifiestan en sus etapas tempranas a nivel de la retina. Algunos ejemplos de indicadores morfológicos (detectables a partir de imágenes conocidas como de *fondo de ojo*) son la aparición de *neovasos*<sup>5</sup>, que es un síntoma indicador de la presencia de una enfermedad en el ojo como la retinopatía diabética proliferante y la trombosis isquémica; por otro lado, la tortuosidad de los vasos y su grosor son parámetros que ayudan a detectar la presencia de la leucemia y, por otra parte, permiten clasificar el grado de avance de la retinopatía hipertensiva [Aldana Iuit, 2010].

---

<sup>4</sup>Las enfermedades sistémicas son aquellas que involucran varios órganos o todo el cuerpo

<sup>5</sup>Los *neovasos* son pequeños vasos sanguíneos que se generan en las zonas del ojo afectadas por la falta de riego de sangre; cuya aparición rápida, masiva y desordenada provoca graves perturbaciones en el ojo tales como hemorragias en el interior del ojo, obstrucción de las vías de drenaje de la presión ocular y, en casos más severos, pérdida progresiva de las fibras nerviosas de la retina, cambios en el aspecto del nervio óptico y desprendimientos de retina.

Asimismo, el nervio óptico y los movimientos del ojo frecuentemente reflejan cambios en el sistema nervioso central, pues gran parte del cerebro ayuda a procesar la información visual y controla los movimientos del ojo, lo que incentiva aún más el estudio del ojo; es decir, el hecho de que la microvascularidad retiniana y cerebral tienen propiedades morfológicas y fisiológicas en común, hace que el estudio y análisis ocular puede considerarse como un monitor del estado del cerebro [Aldana Iuit, 2010].

## 1.2. Definición del problema

Con base en lo expuesto anteriormente, la tarea parece ya más clara, construir un modelo para visualizar sistemas vasculares, que reproduzca lo más exacto y real posible la superficie tubular, para facilitar un diagnóstico oportuno y preciso de alguna u otra de las enfermedades citadas anteriormente. En concreto, el fin de la presente tesis consiste en buscar, estudiar y mostrar un esquema de representación tridimensional de superficies de estructuras tubulares del sistema vascular retinal: los vasos sanguíneos (venas y arterias) del interior del globo ocular conocidos como vasos retinales, distinguiendo como punto clave construir una malla apropiada que resuelva el problema de las bifurcaciones que suelen presentarse.

Para ello, el punto de partida son los datos numéricos extraídos con el sistema RISA [Martinez-Perez *et al.*, 2007] (Figura 1.1) y posteriormente reconstruidos en 3D [Martinez-Perez y Espinosa-Romero, 2004], los cuales están dispuestos en una colección de archivos de texto, cada uno de ellos conteniendo un listado de cinco columnas  $n_{k_i}, x_i, y_i, z_i, \lambda_{k_i}$ , donde la terna  $(x_i, y_i, z_i)$ ,  $i = 1, \dots, n$  corresponde a un punto  $P_i$  en  $\mathbb{R}^3$ , cuya etiqueta  $n_{k_i}$  es el identificador de los puntos  $P_i$  del segmento  $k$  en el árbol vascular, donde  $k = 1, \dots, M$ , con  $M$  el número de segmentos del árbol.

Finalmente,  $\lambda_{k_i}$  es la relación entre el largo  $l_k$  y ancho  $\omega_k$  del  $k$ -ésimo segmento, tal que  $\lambda_{k_i} = l_k/\omega_k$ , de modo que se puede obtener el valor  $r_{k_i} = \omega_k/2$ , que represente el valor del radio para cada punto del segmento correspondiente (Figura 1.2). Además, el conjunto de puntos  $P_i$ , que serán llamados indistintamente *puntos de control*, forman únicamente el esqueleto central de los tubos.

Como se observa en la Figura 1.2 la reconstrucción de la superficie con base en los datos de entrada sería simple si se consideran a los segmentos como tubos separados; sin embargo, la región de la bifurcación que los une no es trivial de definir.

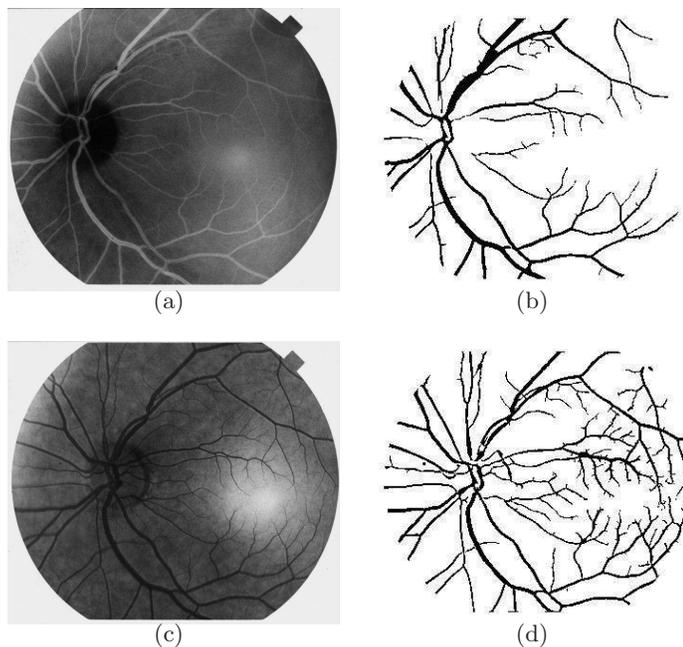


Figura 1.1: Segmentación obtenida con el sistema RISA usando imágenes de *fondo de ojo*. (a) La imagen en filtro rojo y (b) la segmentación correspondiente de los vasos retinales. (c) Imagen del mismo ojo aplicando una preparación fluorescente. (d) La segmentación de la toma anterior conseguida con RISA. Fuente: [http://turing.iimas.unam.mx/~elena/Projects/segmenta/StMary\\_2.html](http://turing.iimas.unam.mx/~elena/Projects/segmenta/StMary_2.html); Martinez-Perez *et al.* [2007].

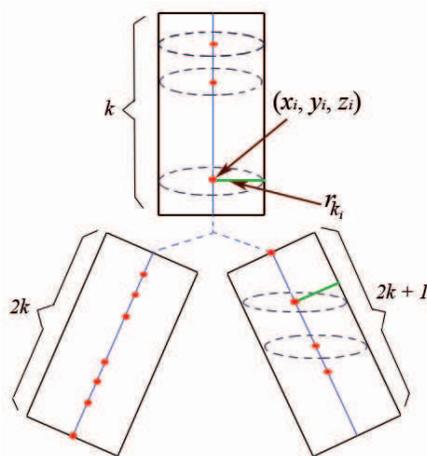


Figura 1.2: Representación interna de los datos procesados. Los puntos en rojo indican algunos de los puntos  $P_i$  de la lista; en verde  $r_{k_i}$ , el radio del vaso en el punto  $P_i$  correspondiente. Las etiquetas (identificadores) de los segmentos se representan bajo la regla  $[k, 2k, 2k + 1]$ , donde  $k$  indica el identificador para el primer segmento antes de la bifurcación, y  $2k, 2k + 1$  son las etiquetas de las ramas hijas de la bifurcación.

### 1.3. Objetivos

Esencialmente, la principal meta de este trabajo consiste en realizar un breve análisis de las estructuras tubulares, adaptando alguna de las técnicas de generación de mallas existentes, para programar una aplicación que satisfaga:

1. Que sea fácilmente manipulable por los usuarios finales.
2. El número de puntos que forman la malla sea *suficiente*, procurando que la ejecución de la visualización se realice lo más veloz posible.
3. Que resuelva efectivamente el problema de las bifurcaciones en las ramas de los vasos sanguíneos.
4. Conserve, en lo posible, las dimensiones (largo, ancho y ángulos de bifurcación) de las estructuras tubulares empleando los datos de los que se dispone.

### 1.4. Estructura

En resumen, la tesis está organizada de la siguiente manera: en el Capítulo 1 se proporciona una breve introducción del problema a resolver, así como la importancia y finalidad del presente trabajo listando los principales objetivos. En el Capítulo 2 se describen a grandes rasgos los distintos tipos de mallado; en general, las técnicas más usuales y en particular para estructuras tubulares, con lo cual se determina los modelos mejor aproximados al problema planteado, eligiendo aquél más óptimo según se considere.

Posteriormente, en el Capítulo 3 se explica a detalle propiamente la técnica seleccionada, así como la construcción del procedimiento usando algunos resultados preliminares. En el Capítulo 4 se hace mención del diseño de software a seguir y una breve explicación del proceso de desarrollo. Hecho lo anterior, en el Capítulo 5 se muestra la salida obtenida con la experimentación correspondiente y se analizan a detalle los resultados en las pruebas de la visualización, sobre todo en las regiones donde aparecen bifurcaciones.

Finalmente, en el Capítulo 6 se discuten las conclusiones acerca del sistema y los resultados, además de señalar los siguientes pasos en la mejora del modelo de malla y el posible trabajo a futuro.

## Capítulo 2

# Generación de Mallas

Cuando se genera un modelo tridimensional, la representación visual lograda consiste en una colección de objetos, elementos y propiedades entre ellos, capaces de reproducir determinadas características de algún fenómeno físico. Sin embargo, concebir la mejor descripción matemática que permita conseguir esto no es sencillo.

En la vida real, los modelos matemáticos de muchos problemas utilizan ecuaciones diferenciales parciales (EDPs) para describir su comportamiento; sin embargo, no existe un método general para resolver analíticamente todas las EDPs o incluso la solución de muchas de éstas es inexistente, por lo que se buscan soluciones numéricas, las cuales asumen que el dominio de interés puede ser dividido en pequeños elementos cuya unión y continuidad forman una malla [García *et al.*, 2009].

Así, para resolver numéricamente un problema de modelado continuo, es necesario convertirlo de continuo a un conjunto finito de puntos, de tal modo que la selección de puntos es determinada por la generación de mallas [Rivera Loaiza, 2000]. Por tanto, resulta evidente que la elección de puntos más apropiada dependerá, en gran parte, de la cantidad y forma de las regiones complejas a representar, en el caso de esta tesis las zonas de bifurcación entre segmentos tubulares.

### 2.1. Métodos numéricos y mallas

La idea general de cualquier método numérico es discretizar, con el fin de conseguir una aproximación del fenómeno a analizar; o bien, modelar una predicción del comportamiento cuantitativo y cualitativo del sistema, resolviendo el modelo matemático asociado.

Ahora bien, existen dos métodos principales para llevar a cabo las simulaciones numéricas de modelos basados en ecuaciones diferenciales parciales, los cuales son explicados como sigue:

El *Método de Diferencias Finitas* (MDF), es un procedimiento en el cual las derivadas son reemplazadas por aproximaciones en diferencias finitas<sup>1</sup>, convirtiendo entonces un problema de ecuaciones diferenciales en un problema algebraico “fácilmente” resoluble por medios especialmente matriciales.

El *Método de Elemento Finito* (MEF) consiste en la división de un espacio continuo, que es regido por una ecuación diferencial o un sistema de ecuaciones diferenciales, a un sistema cuyo comportamiento se modela por un sistema de ecuaciones, lineales o no. El dominio del problema se aproxima dividiéndolo en subdominios denominados *elementos finitos*, usando puntos (en el caso lineal), líneas (en el caso bidimensional) o superficies (en el tridimensional).

En ambos métodos, se requiere que el problema se encuentre definido en un espacio geométrico que pueda ser discretizado en pequeñas regiones formando una especie de red o malla (*mesh*), cuyas intersecciones reciben el nombre de *nodos*. Esencialmente, la desventaja del primer método es que requiere de un orden en los nodos; además, los esquemas de diferencias finitas pueden aplicarse solamente cuando el arreglo de nodos de la malla es rectangular, lo cual es una seria limitación. Por otra parte, el segundo método tiene como principal desventaja el que su implementación es difícil y hasta cierto grado artesanal [Rivera Loaiza, 2000].

## 2.2. Elementos de una malla

En general, sea cuál sea el método elegido con el que se desee generar una malla de ese estilo o, caso contrario, una malla simplemente *geométrica* con formas arbitrarias incluso independiente de algún modelo matemático particular, deben considerarse los componentes esenciales de la misma, conocidos como *elementos*. Un *elemento* está definido por su naturaleza geométrica (triángulo, cuadrilátero, etc.) y una lista de vértices, incluyendo además la definición de las aristas y caras que lo conforman [Frey y George, 2008]:

**Definición 2.2.1** *La conectividad de un elemento de una malla es la definición de las conexiones entre los vértices a nivel elemento.*

**Definición 2.2.2** *La topología de un elemento de una malla es la definición de las relaciones entre sus vértices, aristas y caras.*

**Definición 2.2.3** *Una malla  $M$  es conforme (conformal mesh) si la intersección de dos elementos en  $M$  es el conjunto vacío, un vértice, una arista, o bien, una cara.*

---

<sup>1</sup>Una diferencia finita es una expresión matemática de la forma  $f(x + b) - f(x + a)$ . Si una diferencia finita se divide por  $b - a$  se obtiene una expresión similar al cociente diferencial, que difiere en que se emplean cantidades finitas en lugar de infinitesimales.

## 2.3. Clasificación de métodos de mallado

A pesar de algunas diferencias conceptuales, contextos y/o aplicaciones para los cuales se han propuesto, los métodos de generación de mallas basados en *diferencias finitas*, *elementos finitos* y otras técnicas con diversas geometrías, se pueden clasificar en las siguientes categorías [Frey y George, 2008]:

i) **Métodos manuales y semi-automáticos.**

En los métodos manuales o *enumerativos* toda la información es proporcionada por el usuario, tanto la forma de los elementos como las coordenadas de los nodos, los atributos físicos, etc. En cambio, los métodos semi-automáticos o *explícitos* aprovechan que el dominio a discretizar tiene formas geométricas sencillas.

ii) **Métodos por parametrización.**

Aquí la malla final es el resultado de una transformación inversa, de una *lattice* regular de puntos en un espacio paramétrico a un espacio físico. Dependiendo de la función de mapeo que se utilice, implícita o explícita, se definen:

*Métodos de Interpolación algebraica.*

Se obtiene la malla a partir de interpolación de curvas (superficies) o alguna otra técnica relacionada que sea explícitamente definida.

*Métodos basados en soluciones.*

La malla es generada como solución a un sistema de ecuaciones diferenciales parciales (elíptico, hiperbólico o parabólico), por lo que depende de una función definida analíticamente.

iii) **Métodos de descomposición de dominios.**

La malla es el resultado de un análisis (*top-down*) que consiste en separar el dominio en pequeños dominios que son geoméricamente cerrados a un dominio de referencia.

iv) **Métodos de inserción de puntos.**

Éstos métodos comienzan con una discretización del dominio y principalmente consiste en agregar e insertar nodos internos a éste. *Advancing-Front* (creación de elementos) y aproximación por *Delaunay* (inserción de puntos) son dos métodos de ésta clase.

v) **Métodos constructivos.**

Las mallas finales son el resultado de mezclar varias mallas mediante transformaciones topológicas o geométricas, tal que cada malla ha sido creada por cualquiera de los métodos anteriores.

## 2.4. Remallado (*Remeshing*)

Como es bien sabido, las superficies generadas por mallas son usadas en muchas aplicaciones de cómputo gráfico (modelado, edición, visualización, animación, etc.) para representar diversas formas. Muchas de esas mallas son generadas por dispositivos de escaneo o mediante representaciones implícitas de *isosuperficies*<sup>2</sup>. Desafortunadamente, tales procesos están propensos a errores y el resultado son mallas pocas veces satisfactorias. Por tanto, es frecuente que la calidad de dichas mallas, en términos de cantidad de vértices, posición y diseño de la geometría de sus elementos (Sección 2.2), sea mejorada. A este proceso de mejoramiento se le llama remallado (*remeshing*).

**Definición 2.4.1** *Dada una malla  $M$  en  $3D$ , el proceso de remallado consiste en computar otra malla  $M'$ , cuyos elementos satisfagan ciertos requerimientos de calidad, mientras aproximan los valores de entrada aceptablemente [De Floriani y Spagnuolo, 2008].*

A grandes rasgos, De Floriani y Spagnuolo [2008] señalan algunas características deseadas que se buscan optimizar o evitar durante el proceso de remallado, las cuales se mencionan a continuación:

- **Validez.** La malla resultante tiene que ser una malla *válida*; es decir, debe ser un espacio matemático simple y cerrado.
- **Calidad.** Se requiere que posea robustez y estabilidad numérica, con el fin de minimizar errores numéricos y otras singularidades que puedan ocurrir.
- **Fidelidad.** La nueva malla generada debe aproximar la forma geométrica original deseada tanto como sea posible, mientras mantiene su complejidad. Ésto involucra elegir una medida de error para valorar la diferencia analíticamente.
- **Valores de entrada.** Normalmente, los valores de entrada de la malla inicial son dados como una malla discreta, lo cual es usualmente sólo una aproximación de alguna (desconocida) forma continua.
- **Dimensión del conjunto de datos.** En muchas ocasiones los escáneres modernos generan enormes bases de datos con el fin de asegurar que no haya pérdidas de detalle de las formas de la superficie a representar. Como resultado, el muestreo y las formas geométricas son excesivas, e incluso, la información por sí misma está repleta de redundancias.

---

<sup>2</sup>Una *isosuperficie* es un objeto cuya superficie se define matemáticamente por medio de una función, por lo que puede ser deformado y sometido a desplazamientos de superficie.

- **Incertidumbre.** Asimismo, con frecuencia la obtención de datos es contaminada por ruido electrónico, óptico o mecánico presente en el proceso de escaneo y la interpretación suele ser difícil.
- **Correspondencia.** Un propósito esencial es encontrar la posición correspondiente de un nuevo vértice, lo cual se logra típicamente parametrizando la malla de entrada. Sin embargo, este es un problema complejo y costoso, ya que el mapeo de una estructura 3D no trivial (posiblemente una malla 3D con un número arbitrario de picos y agujeros) a un dominio paramétrico bidimensional introduce inevitablemente distorsiones métricas y puede conducir a pérdida de información importante. La ventaja consiste en que las operaciones de remallado en un plano 2D se realizan considerablemente más rápido que las optimizaciones en 3D; por ello, la distorsión causada por el mapeo puede ser reducida usando traslape de parches y reducción en la acumulación de error.

### 2.4.1. Clasificación

De manera similar, los métodos de remallado suelen agruparse de distintas formas; en particular, una de estas clasificaciones consiste en que la obtención de la malla cumpla una meta y características específicas, en vez de enfocarse en la técnica y algoritmos que emplea. En resumen, las 5 categorías de remallado identificadas en tal clasificación son las siguientes [De Floriani y Spagnuolo, 2008]:

- A) Estructurado.** Reemplaza una malla de entrada no estructurada por una estructurada, mejor conocida como malla *regular*; es decir, una malla en la que todos los vértices internos están rodeados por un número constante de elementos. Una malla *semi-regular* (Figura 2.1) es obtenida por una subdivisión regular de una malla irregular, donde todos los vértices son regulares excepto para un pequeño número de vértices llamados *puntos extraordinarios* (*extraordinary points*). Una malla *altamente regular* es aquella en la cual la gran mayoría de los vértices son regulares, aún cuando la malla no haya sido generada por subdivisión.

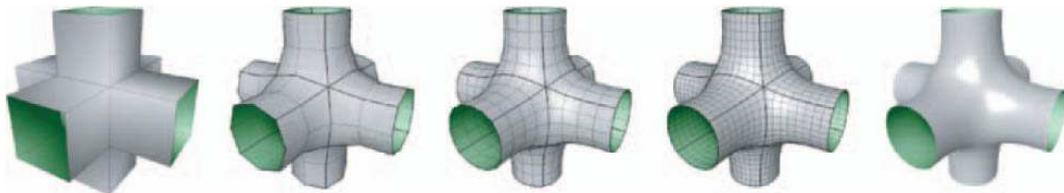


Figura 2.1: Malla *semi-regular* generada por subdivisión recursiva de una malla inicial base. Fuente: De Floriani y Spagnuolo [2008].

En principio, una malla *regular* mantiene una gráfica de conectividad mucho

más simple, por lo tanto permite un análisis más eficiente en los algoritmos de localización de vértices. Por su parte las mallas *semi-regulares* son esencialmente regulares por secciones y ofrecen una apropiada transición entre la simplicidad de las mallas regulares y la flexibilidad de las mallas no estructuradas.

**B) Compatible.** En muchas aplicaciones, tales como *morphing*, *shape blending*<sup>3</sup>, transferencia de texturas y materiales, entre otras, se necesita un mapeo biyectivo conocido como *parametrización cruzada* (*cross-parameterization*), con el fin de preservar la forma y características de los modelos parametrizados, mapeando así las estructuras comunes entre ellos (Figura 2.2). La mayoría de estas aplicaciones requieren que los modelos sean representados por mallas *compatibles*; es decir, mallas con idéntica conectividad, por lo que, usando dicho mapeo, los algoritmos de *remallado compatible* (*Compatible Remeshing*) preservan la correspondencia entre los vértices de los rasgos definidos, permitiendo al mismo tiempo la generación de nuevos modelos mezclados (Figura 2.3).

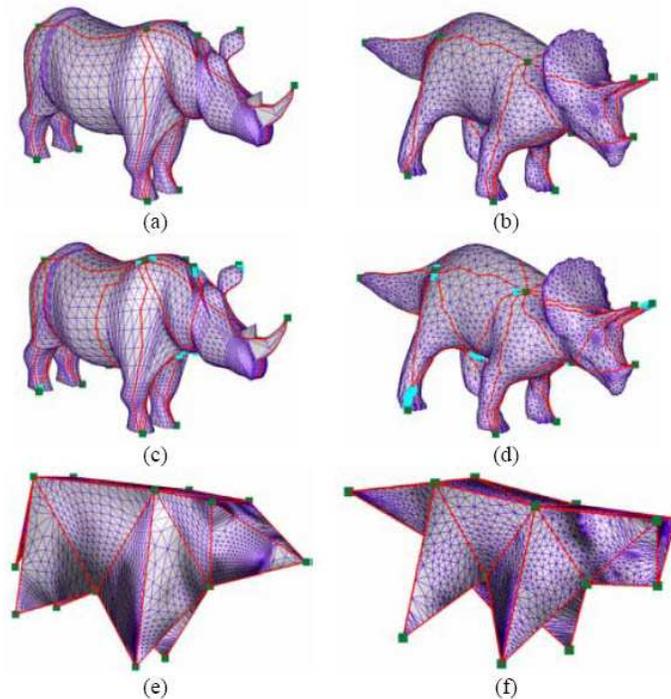


Figura 2.2: Dominios base de construcción obtenidos mediante *parametrización cruzada*. (a),(b) Rastreo de aristas; (c),(d) Rastreo de caras poligonales; (e),(f) Mallas base. Fuente: Kraevoy y Sheffer [2004].

<sup>3</sup>El *morphing* y el *shape blending* son efectos especiales utilizados para modificar las características de los objetos hasta transformarlos en otros.

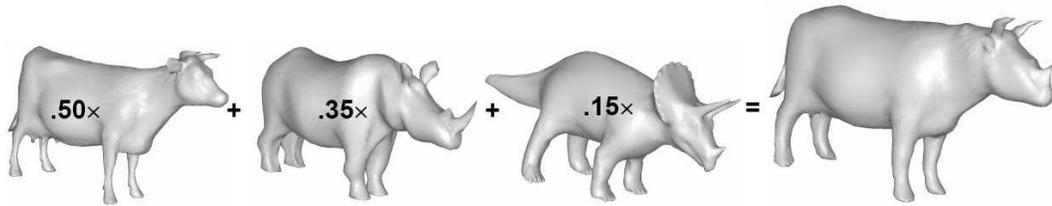


Figura 2.3: *Blending* y *morphing* de distintas superficies. Los números en los modelos son la proporción del peso de cada modelo en la malla resultante. Fuente: Kraevoy y Sheffer [2004].

- C) De alta calidad.** Como se mencionó anteriormente, el muestreo y adquisición de los datos y vértices por escáner suele tener muchos problemas que, aunado a la forma (comúnmente compleja) de la estructura original sobre la que se realiza el escaneo, conlleva a la generación de mallas que poco aproximan a la curvatura y torsión de las funciones que definen la superficie. En ese sentido, usando la difusa distribución de vértices conseguidos, el *remallado de alta calidad* (*High Quality Remeshing*) busca adaptar el muestreo a la geometría para obtener una nueva discretización con una malla que exhiba propiedades como: elementos bien formados, muestreo uniforme o isotrópico<sup>4</sup> y muestreo gradual suave<sup>5</sup>. Un ejemplo conocido de este tipo de *remallado* es la generación de mallas adaptativas, mediante la utilización de algoritmos basados en refinamientos de Delaunay, diagramas de Voronoi, entre otros (Figura 2.4).
- D) Por características.** La mayoría de las técnicas anteriores están restringidas a asumir que cada vértice del muestreo yace en una curva o línea específica cuya posición está completamente definida por un patrón (función) preestablecido. No obstante, en la mayoría de los casos, tal patrón no puede ser ajustado para coincidir con esquinas y crestas de algunos modelos debido a la discontinuidad de las curvas en tales puntos y, por tanto, casi ningún vértice del muestreo definirá las crestas, removiendo así los picos y esquinas de la estructura original. La meta principal de esta categoría es entonces preservar o incluso restaurar características de este tipo (Figura 2.5).
- E) Guiado por error.** El *remallado conducido por error* (*Error-driven remeshing*) se enfoca en generar una malla que maximice la relación entre la complejidad y la precisión de los algoritmos, minimizando las diferencias entre la superficie de entrada y la resultante. La complejidad es expresada en términos del número de elementos de la malla, mientras que la precisión geométrica es una medida relativa a la malla de entrada y relacionada con la medida de distorsión del

<sup>4</sup>Un muestreo *isotrópico* (*isotropic sampling*) es aquél que es localmente uniforme en todas direcciones.

<sup>5</sup>El muestreo *gradual suave* (*smooth gradation sampling*) significa que aún si la densidad del muestreo no es uniforme, éste puede variar de una forma suave.

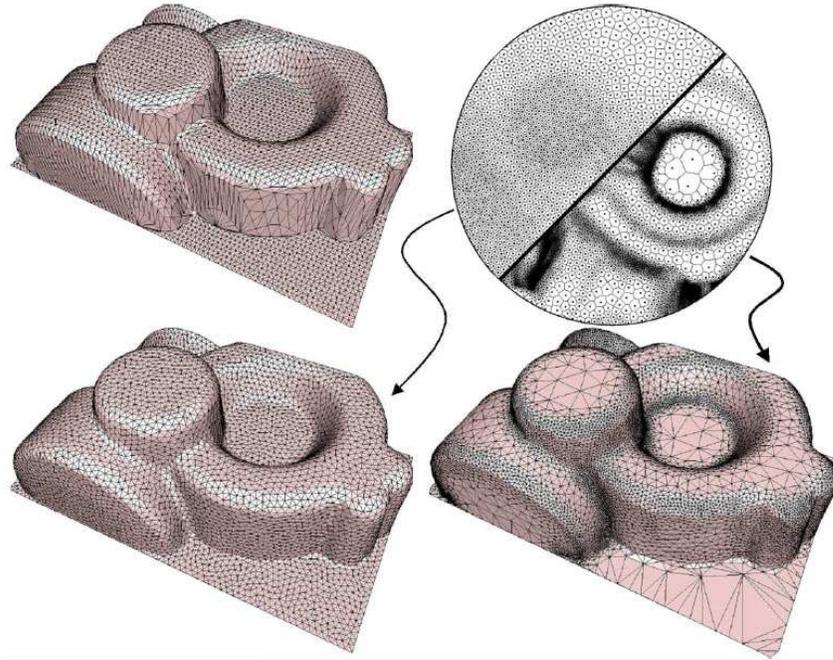


Figura 2.4: Generación de malla por refinamiento basado en diagramas de Voronoi. La imagen superior izquierda representa la malla de entrada. Las imágenes inferiores corresponden al *remallado* adaptado a la curvatura de la superficie con distinta cantidad de vértices. La imagen superior derecha muestra un acercamiento a cada malla con los centros de los diagramas computados. Fuente: Alliez *et al.* [2003].



Figura 2.5: Malla triangular (imagen izquierda) reconstruida por mejoramiento de crestas usando *Feature-Remeshing*. Las regiones suaves y los triángulos de frontera son identificados (imagen central). Como resultado, se obtiene una transición con curvas suaves definiendo la superficie (imagen derecha). Fuente: De Floriani y Spagnuolo [2008].

error predefinida. De este modo, la simplificación de mallas o los métodos de refinamiento son formas comunes de generar mallas más eficientes.

## 2.5. Subdivisión

Para lograr entonces resultados aceptables en la generación de una malla es indispensable tomar en cuenta que la calidad de los datos de entrada puede no ser la esperada; asimismo, la complejidad de las formas a representar, el nivel de detalle deseado, o bien, la meta y características buscadas determinarán los procedimientos necesarios para la obtención de la malla final.

Evidentemente, una malla de estructura “simple“ no contará con la cantidad de información suficiente y carecerá de fidelidad visual, en cambio una malla “saturada“ de vértices, aunque permita muestreo, dificultará ese mismo análisis sobre la distribución de los puntos, además del impacto en el consumo de recursos de procesamiento. Una forma de lidiar con ello y encontrar un punto intermedio consiste en los algoritmos de *subdivisión* basados en refinamientos.

Hace más de 30 años, los trabajos simultáneos de Catmull y Clark [1978], y de Doo y Sabin [1978], marcaron el inicio de las técnicas de modelado de superficies utilizando *subdivisión*; la idea de conseguir una malla poligonal mediante la especificación de una malla menos detallada permitía mejorar la suavidad de las curvas y superficies a través de un proceso recursivo. De esta forma, la subdivisión ha encontrado variadas aplicaciones en computación gráfica y diseño geométrico asistido por computadora, pues su desarrollo aparece fuertemente ligado a técnicas que requieren enormes y complejas geometrías, tales como la multiresolución y herramientas matemáticas tradicionales como la *transformada de ondeleta*<sup>6</sup>.

**Definición 2.5.1** *La subdivisión es un procedimiento que define una curva o superficie suave como el límite de una secuencia de refinamientos sucesivos [Schröder, 2000].*

El proceso inicia con una malla poligonal dada, posteriormente se aplica un esquema de refinamiento que toma la malla y la subdivide, creando nuevos vértices y caras. Las posiciones de los nuevos vértices son computadas con base a la posición de los vértices antiguos y sus vecinos cercanos. En algunos esquemas de refinamiento, la posición de los vértices anteriores puede o no ser alterada. Naturalmente, se produce una malla más densa que la original, conteniendo más caras poligonales; no obstante, el nivel de calidad se incrementa, reduciendo al mismo tiempo la cantidad

---

<sup>6</sup>La *transformada de ondeleta* (o *transformada wavelet*) es un tipo especial de transformada de Fourier que representa una señal en términos de versiones trasladadas y dilatadas de una onda finita denominada ondeleta madre.

de redundancias y muestreo. La malla resultante puede volver a ser procesada con el mismo esquema de refinamiento una y otra vez (Figura 2.6). El límite de subdivisiones de una superficie es iterable hasta el infinito. En la práctica, este algoritmo es aplicado sólo un pequeño número de veces, ya que en tanto aumenta la cantidad de iteraciones también se incrementa rápidamente la cantidad de vértices definidos, los cuales se encontrarán cada vez más y más cerca uno del otro sin obtenerse una mejora mayor, eso sin mencionar las consecuencias que trae consigo el total de operaciones asociadas a cada vértice.

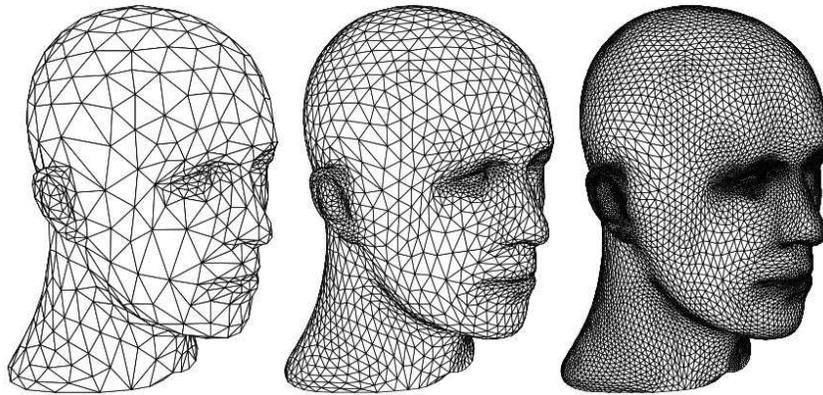


Figura 2.6: Subdivisión de una superficie mostrando 3 niveles sucesivos de refinamiento. Del lado izquierdo la malla triangular inicial aproximando a la superficie. Cada triángulo es dividido en 4 de acuerdo a una regla de subdivisión particular. Del lado derecho, la malla es subdividida una vez más. Fuente: Schröder [2000].

Los esquemas de refinamiento de superficies pueden ser clasificados en dos principales categorías: por *interpolación* y por *aproximación*. Los esquemas por *interpolación* requieren ajustarse a la posición de los vértices de la malla original. Los esquemas por *aproximación* no, pues pueden ajustarse a estas posiciones según sea necesario. En general, los esquemas de aproximación tienen mejor suavidad, pero el usuario tiene menor control del resultado. Precisamente, un esquema de refinamiento por *aproximación* parece útil en cualquier aplicación que pretenda visualizar una superficie representativa, más que “exacta”; mientras que un esquema basado en *interpolación* sería más apropiado, por ejemplo, en el *remallado por características* (Sección 2.4.1) con el fin de conservar las esquinas, crestas y determinadas aristas importantes.

### 2.5.1. Características

A pesar de que el proceso de *subdivisión* es en cierto modo una operación constructiva puntual, en comparación con algunos otros modelos, cumple ciertas características que se resumen brevemente a continuación [Schröder, 2000]:

- 1.- **Eficiencia.** La *subdivisión* es fácil de implementar y es computacionalmente eficiente. Para el cómputo de los nuevos puntos de la malla resultante, por cada vértice de la malla de entrada se usa sólo un pequeño número de vértices adyacentes. Ésto es similar a los métodos de inserción de nodos presentes en el *modelado por splines*<sup>7</sup> y de hecho muchos métodos de *subdivisión* son simplemente una generalización de la inserción de nodos. En caso contrario, generar mallas usando superficies implícitas es mucho más costoso, algoritmos como *marching cubes* son requeridos para generar las aproximaciones poligonales necesarias para el renderizado.
  
- 2.- **Topología arbitraria.** Esta propiedad constituye uno de los puntos claves de los algoritmos de *subdivisión*. Primeramente, los tipos de mallas y las superficies asociadas a ellas pueden ser arbitrarias; en segundo lugar, la estructura de las gráficas formadas por las aristas y vértices de tales mallas pueden también ser arbitrarias; específicamente, cada vértice puede tener un grado arbitrario. Mucho más todavía, sin importar la geometría (cuadriláteros o triángulos) de los elementos formados por los los puntos de control de la malla inicial, la valencia de los nuevos vértices en iteraciones posteriores se conserva (Figura 2.7), incluso para los *puntos extraordinarios* (Sección 2.4). Además, aún cuando haya otros procedimientos capaces de reproducir topologías diversas, el impacto sobre el costo computacional puede ser más elevado, por lo que la *subdivisión* se convierte en un método flexible para manejar cualquier topología sin perder tanta eficiencia.

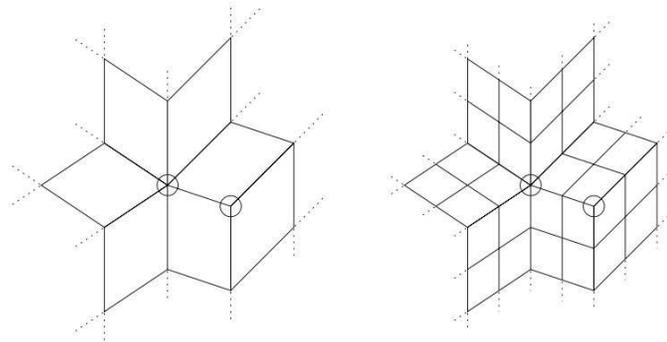


Figura 2.7: Una malla con dos *puntos extraordinarios*, uno de grado 3 y otro de grado 6; el resto de los vértices en la estructura con grado 4. Nótese que la valencia de los vértices se mantiene. Además la cantidad de *puntos extraordinarios* puede conocerse, ya que es constante en cada iteración. Fuente: Schröder [2000].

<sup>7</sup>El *modelado por splines* (*Spline modeling*, también conocido como *patch modeling*) es una alternativa al uso de polígonos en modelado orgánico. Un *spline* en 3D es una línea que describe una curva definida por un número de puntos, que puede ser deformada para crear geometría en tres dimensiones. Una malla creada de la intersección de *splines* consiste de áreas llamadas *patches* (patches).

- 3.- Calidad de los detalles.** Como se vió en la sección anterior, controlar la forma y tamaño de los rasgos y detalles tales como crestas, surcos y picos es otro punto importante. En el caso de los procedimientos que hacen uso de isosuperficies resulta difícil de controlar tales características, ya que el modelado se desarrolla indirectamente y existen muchas interacciones indeseables entre diferentes partes de la superficie. Por su parte, la generación por *splines* suele ser lenta debido a la resolución por métodos generalmente matriciales, aparte que la incorporación de nuevos rasgos en posiciones arbitrarias no se realiza adecuadamente. En cambio, el proceso de *subdivisión* permite manipular los coeficientes de los algoritmos de refinamiento para lograr efectos tales como crestas, o controlar el comportamiento en las fronteras de las curvas.
- 4.- Geometría compleja.** Debido a que la *subdivisión* está basada en refinamientos repetitivos es muy sencillo incorporar ideas tales como renderizado a nivel de detalle y compresión para aplicaciones en línea. Asimismo, durante la edición local interactiva en la *subdivisión* adaptativa pueden generarse múltiples refinamientos. Para aplicaciones que sólo requieren visualización de geometrías fijas, representaciones con mallas progresivas suele ser más cómodo.
- 5.- Variación del volumen.** Cabe aclararse que el proceso de *subdivisión* suele reducir el volumen de la superficie original, sobretodo en regiones donde el modelo presente geometrías más simples o con pocos puntos (Figura 2.8). No obstante, la implementación del algoritmo puede complicarse tanto como se desee o necesite, siendo posible entonces agregar nuevos vértices conservando algunos de los anteriores, con el fin de evitar o compensar esa pérdida de volumen.

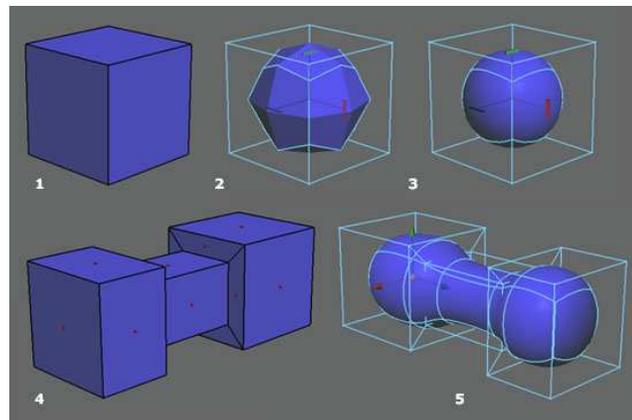


Figura 2.8: Subdivisión aplicada a dos estructuras diferentes. Nótese cómo en ambos casos el volumen se modifica. En las 3 primeras figuras se distingue la obtención de una superficie esférica a partir de un cubo. Las figuras 4 y 5 muestran el cambio de volumen usando 3 cuerpos cuboides. Fuente: <http://www.subdivisionmodeling.com/page2.htm>

### 2.5.2. Puntos polo

Cuando se trabaja con algoritmos de subdivisión es muy usual encontrar dificultades en su implementación, principalmente como consecuencia de la aparición de los *puntos extraordinarios*, conocidos en el modelado por subdivisión como *puntos polo* (*poles*). Por esta razón es importante destacar los dos tipos de *puntos polo* más importantes:

- 1) *Polo de Extrusión*<sup>8</sup> (*Extrude-Pole*). Abreviado E(5)-pole, es un *punto extraordinario* en el que inciden 5 aristas (Figura 2.9(a)). Idealmente, en la modelación “correcta” de cráneos y caras, la extrusión de regiones como la boca, ojos y orejas conseguiría 4 polos de este tipo en cada una de esas zonas (Figura 2.9(b)).

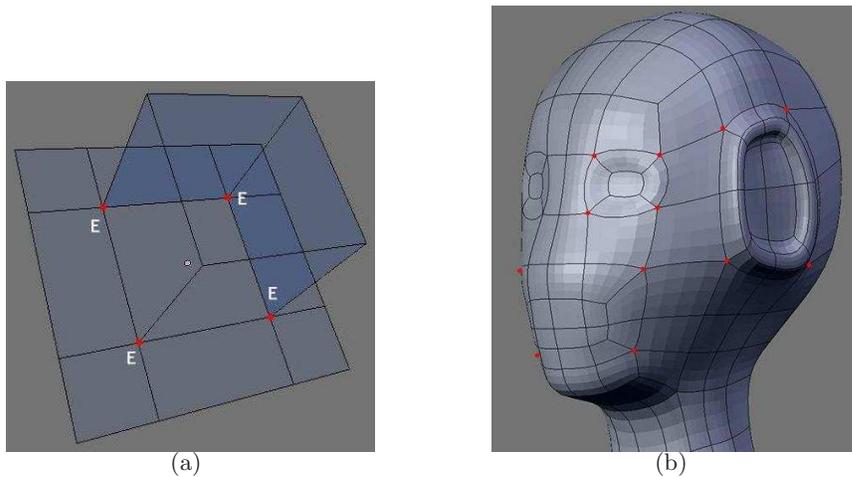


Figura 2.9: Formación de *puntos polo de extrusión*. (a) La deformación de un cuadrilátero genera 4 polos de este tipo. (b) En rojo se marcan los *polos de extrusión* correspondientes a las regiones de la boca, orejas y ojos del modelo. Fuente: <http://www.subdivisionmodeling.com/forums/showthread.php?t=907>

- 2) *Polo Nasal* (*Nose-Pole*)<sup>9</sup>. Este tipo de punto, abreviado como N(3)-pole, es un *punto extraordinario* con una valencia igual a 3, usado frecuentemente, también en la modelación de rostros, para simular el efecto de una nariz (Figura 2.10).

---

<sup>8</sup>El proceso de *extrusión* se define como: dar forma a una masa metálica, plástica, etc., haciéndola salir por una abertura especialmente dispuesta. Por lo que en el contexto de modelado geométrico podría considerarse como una deformación de los elementos de una superficie bajo algún criterio, tales como la eliminación o inserción de puntos.

<sup>9</sup>Una mejor traducción podría ser *polo interno* o *de intromisión*, ya que a diferencia del *polo de extrusión* se formaría en el interior de una cara definida. En el otro sentido, si un *Nose-Pole* es un punto que define una nariz, un *Extrude-Pole* podría equiparar el nombre de *Eye-Pole* o *Ear-Pole*.

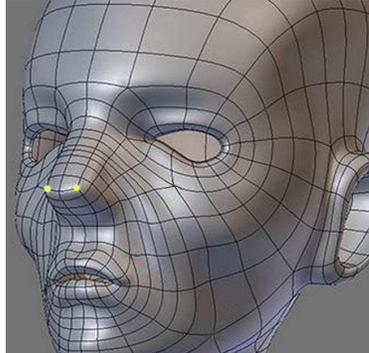


Figura 2.10: Definición de *puntos polos nasal*. Se muestran en color amarillo estos puntos, empleados para formar la superficie de la nariz. Fuente: <http://www.subdivisionmodeling.com/forums/showthread.php?t=907>

Como puede notarse en la Figura 2.11, el algoritmo solución que se utilice no debe dejar de lado la aplicación de un tratamiento especial en las zonas donde haya *puntos polo*, pues su posición inicial y su referencia posterior en el refinamiento iterativo afinará con mejor o peor calidad los detalles de la forma deseada.

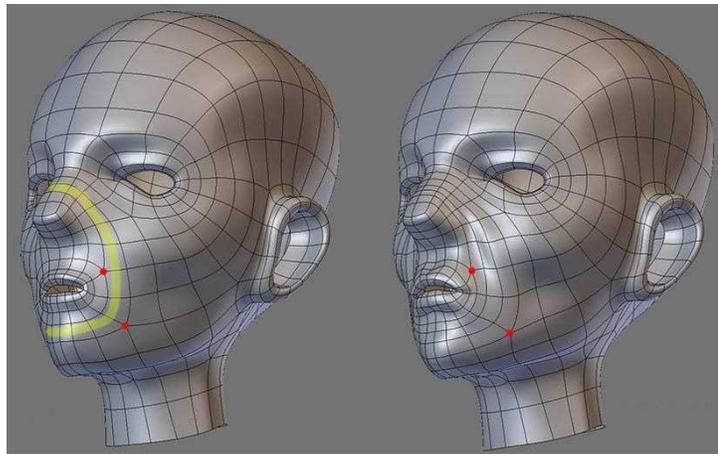


Figura 2.11: Modelado de un rostro con *puntos polo* en distintas posiciones. La definición de puntos en la imagen del lado izquierdo ofrece una cara con mayor redondez en las mejillas y el contorno de los labios, lo que reduce el tamaño de la nariz. Del lado derecho se observa una apariencia claramente distinta, con surcos más marcados en las regiones mencionadas. En rojo aparecen algunos *polos de extrusión*. Fuente: <http://www.subdivisionmodeling.com/forums/showthread.php?t=907>

## 2.6. Mallas tubulares

Al igual que en el Capítulo 1, donde hablar de estructuras tubulares conducía a un amplio grupo como los vasos sanguíneos; en este capítulo, seguir describiendo la infinidad de métodos y técnicas de mallado excedería por mucho el fin de la presente tesis. Por ende, nuestra atención deberá centrarse desde ahora en analizar un poco la bibliografía relacionada específicamente con la generación de mallas tubulares.

En particular, se han implementado varios métodos para visualizar este tipo de estructuras, principalmente segmentos del citado sistema vascular sanguíneo. De esta manera, se tienen modelos para obtener estimaciones precisas sobre las dimensiones de los vasos, realizando para ello una parametrización por medio de un proceso de ajuste no lineal [La Cruz *et al.*, 2004]. En trabajos de este tipo, se remarca la importancia de conseguir parámetros adecuados que permitan usar técnicas de visualización tales como Reformación de Planos Curvados (*Curved Planar Reformation*).

Se verá que resulta común modelar los tubos mediante estructuras elípticas y cilíndricas; así, La Cruz *et al.* [2004] propone describir la superficie tubular empleando dimensión, orientación y densidad específicos del interior y exterior de la misma. Las secciones elípticas de cruce son modeladas mediante un punto elegido como centro  $(x_0, y_0)$ , dimensiones del radio  $(r_x, r_y)$ , coeficientes de densidad interna del tubo  $V$  y externa  $b$  (Figura 2.12), obteniendo que para una elipse con parámetro de rotación  $\alpha$ , su función implícita está dada por:

$$f(x, y) = \frac{[(x-x_0)\cos(\alpha)-(y-y_0)\sin(\alpha)]^2}{r_x^2} + \frac{[(x-x_0)\sin(\alpha)+(y-y_0)\cos(\alpha)]^2}{r_y^2} - 1 \quad (2.1)$$

o bien, su equivalente en la reconstrucción 3D

$$f(x, y) = \frac{[(x-x_0)\cos(\beta)+(y-y_0)\sin(\alpha)\sin\beta+(z-z_0)\cos(\alpha)\sin(\beta)]^2}{r_x^2} + \frac{[(y-y_0)\cos(\alpha)-(z-z_0)\sin(\alpha)]^2}{r_y^2} - 1 \quad (2.2)$$

Distinguiendo los contornos de las paredes internas y externas, y utilizando una serie de imágenes bidimensionales capturadas por sistemas de ultrasonido, Barratt *et al.* [2004] muestra la reconstrucción de bifurcaciones de la arteria carótida, donde las superficies son representadas por esquemas geométricos definidos a través de curvas *splines* paramétricas. Además, éste método requiere de tomas de cortes transversales a la dirección vertical del tubo (ver Figura 2.13).

Similarmente, Behrens *et al.* [2003] optó también por parametrizaciones tanto en la detección de objetos, como en el uso de cilindros generalizados. Aquí los autores utilizan cinco puntos coplanares para determinar la única elipse (si es que existe) que pase ellos:

$$x^2 + y^2 + U(y_2 - x_2) - 2Vxy - Rx - Sy - T = 0 \quad (2.3)$$

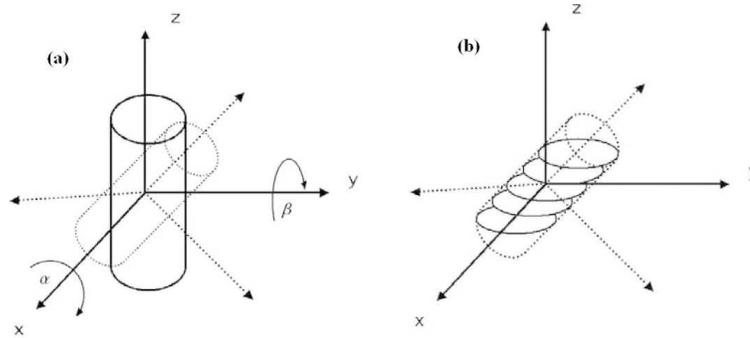


Figura 2.12: (a) Cilindro a lo largo del eje  $Z$  con ángulos de rotación  $\alpha$  y  $\beta$  sobre los ejes  $X$  y  $Y$ , respectivamente. (b) Secciones elípticas a lo largo del eje  $Z$  del cilindro girado. Fuente: La Cruz *et al.* [2004]

tal que

$$\begin{aligned}
 U &= \cos(2\theta) \frac{1 - e^2}{1 + e^2}, & V &= \sin(2\theta) \frac{1 - e^2}{1 + e^2} & (2.4) \\
 R &= 2x_c(1 - U) - 2y_cV \\
 S &= 2y_c(1 + U) - 2x_cV \\
 T &= \frac{2a^2b^2}{a^2 + b^2} - \frac{x_cR}{2} - \frac{y_cS}{2} & e &= \frac{b}{a}
 \end{aligned}$$

siendo el par  $(x_c, y_c)$  el centro correspondiente de la elipse,  $a$  y  $b$  los valores de los semiejes y  $\theta$  el ángulo de la elipse en el plano respectivo.

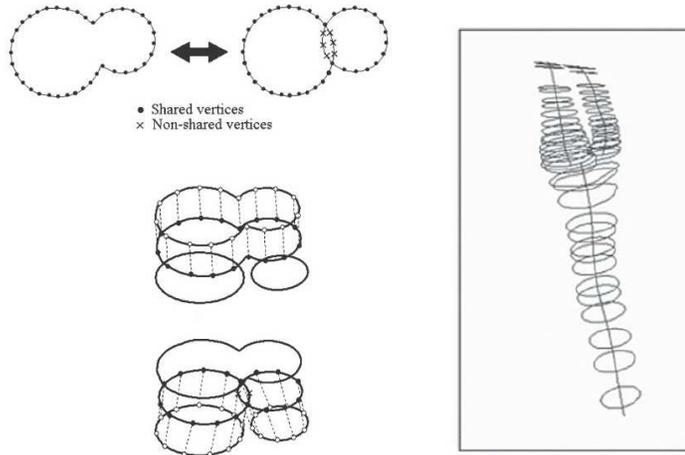


Figura 2.13: Modelo de cilindros generalizados utilizando secciones elípticas transversales. Fuente: Barratt *et al.* [2004]

En cambio, algoritmos de reconstrucción de la arteria coronaria [Chen *et al.*, 2002] han considerado analizar la deformación del vaso, ya que dispositivos intracoronarios además del movimiento del corazón modifican su comportamiento curvilíneo. En este caso, los procedimientos ahora basados en geometría diferencial como la teoría *Frenet-Serret*<sup>10</sup>, fueron ajustados para seguir tal cambio, calculando la longitud, la curvatura (Figura 2.14(a)) y la torsión (Figura 2.14(b)) con respecto a la base ortonormal en cada cuadro de tiempo del ciclo cardiaco.

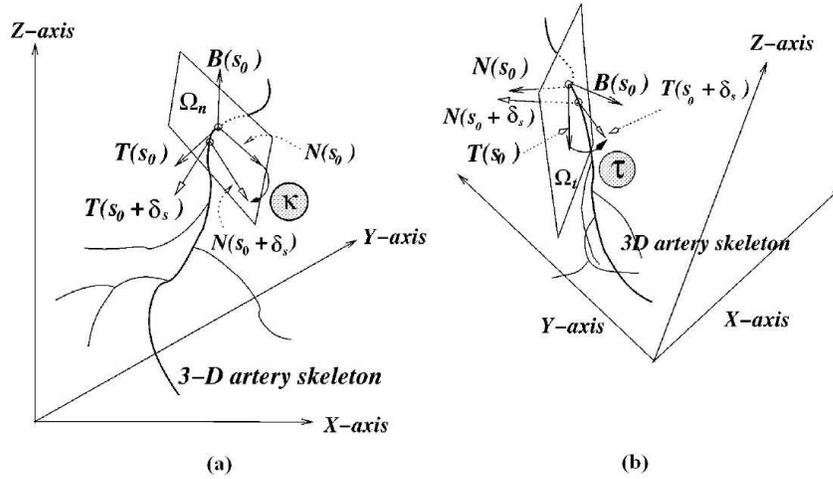


Figura 2.14: (a) Cambio en la curvatura  $\kappa$  del esqueleto arterial. (b) Esquema de representación en la torsión  $\tau$  de la curva. Fuente: Chen *et al.* [2002]

Por otro lado, los métodos deformables construyen modelos a partir de voxels o mallas adaptativas. Tal es el caso de la aproximación del esqueleto vascular a través de curvas NURBS<sup>11</sup>, para posteriormente cubrir el volumen correspondiente extendiendo círculos deformables, cuyos radios varían de acuerdo a la posición de los puntos de control a lo largo de la curva [Li *et al.*, 2007]. Para ello, se requiere ajustar los NURBS conservando la continuidad de las curvas en las zonas de bifurcación. La definición de la curva NURBS puede describirse como sigue:

$$B(t) = \frac{\sum_{i=0}^n N_{i,k}(t)w_i P_i}{\sum_{i=0}^n N_{i,d}(t)w_i} \quad (2.5)$$

tal que  $P_i$ ,  $i = 0, \dots, n$  son los puntos de control,  $N_{i,k}$  es la función base del *B-Spline*,  $d$  es el grado de la curva y  $w_i$  el peso de  $P_i$ .

<sup>10</sup>La teoría *Frenet-Serret* relaciona la base ortonormal de vectores en el espacio con la velocidad y cambio en la dirección de una partícula en movimiento.

<sup>11</sup>NURBS (acrónimo inglés de la expresión *Non Uniform Rational B-Splines*) es un modelo matemático muy utilizado en la computación gráfica para generar y representar curvas y superficies.

Asimismo, la superficie NURBS está dada por la ecuación:

$$S(u, v) = \frac{\sum_{i=0}^m \sum_{j=0}^n N_{i,d}(u) N_{j,e}(v) w_{i,j} P_{i,j}}{\sum_{i=0}^m \sum_{j=0}^n N_{i,d}(u) N_{j,e}(v) w_{i,j}} \quad (2.6)$$

con  $N_{i,d}$  y  $N_{j,e}$  funciones bases del *B-Spline*,  $P_{i,j}$  puntos de control y  $w_{i,j}$  el peso de  $P_{i,j}$  (Figura 2.15).

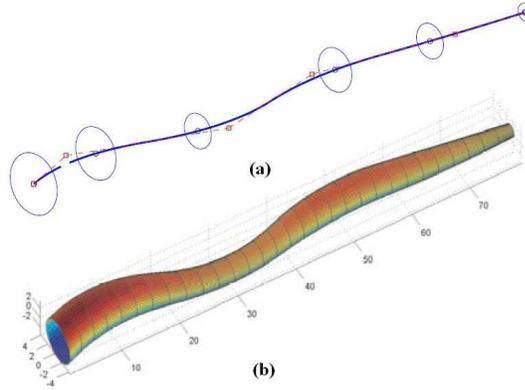


Figura 2.15: (a) Una serie de círculos con tamaño deformable y una curva NURBS pasando a través de sus centros. (b) La correspondiente superficie NURBS generada. Fuente: Li *et al.* [2007]

Yim *et al.* [2001] además de definir un sistema paramétrico de coordenadas tubulares (Figura 2.16), lleva a cabo un procedimiento de mezcla de vértices, como se observa en la Figura 2.17.

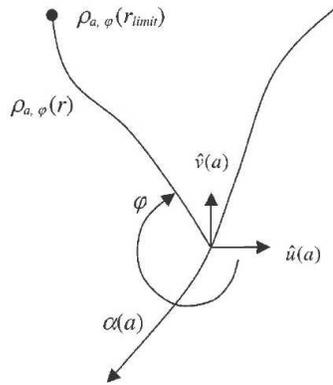


Figura 2.16: Definición de los parámetros del sistema de coordenadas tubulares. A diferencia del sistema de coordenadas cilíndricas, las líneas axial y radial del sistema de coordenadas tubulares pueden ser curvilíneas. Las líneas axial y radial del modelo deformable tubular  $\alpha$  y  $\rho$  son curvas parametrizadas por las variables  $a$  y  $r$ , respectivamente.  $\hat{u}(a)$  es la referencia radial de dirección en cada posición axial.  $\hat{v}(a)$  es perpendicular a  $\hat{u}(a)$ .  $\phi$  es la posición angular de cada línea radial. Fuente: Yim *et al.* [2001]

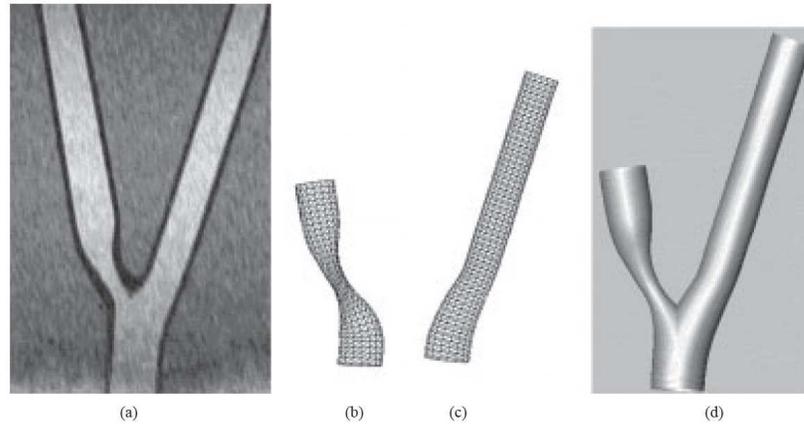


Figura 2.17: Reconstrucción de superficie de la bifurcación carótida. (a) Detección de las regiones tubulares. (b) y (c) La superficie se reconstruye mediante tubos deformables traslapando segmentos y mezclando vértices. (d) La superposición de dos superficies representa el modelo de la bifurcación. Fuente: Yim *et al.* [2001]

No obstante, otros autores han optado por el uso de diferentes tipos de superficies en vez de curvas elípticas y cilindros. Como referencia, se encuentra la concatenación de conos truncados propuesta por Hahn *et al.* [2001], donde emplean voxels para generar una estructura de tipo árbol que represente el esqueleto del vaso, suprimen aquellas ramas cuya longitud esté por debajo de cierto umbral definido y generan una bifurcación con el mismo ángulo de apertura para cada rama hija si tal bifurcación tiene un comportamiento simétrico. Además se requiere de un proceso de *suavizado* (Figura 2.18) considerando el tamaño de los diámetros calculados en el preproceso de las imágenes y la detección de los tubos.

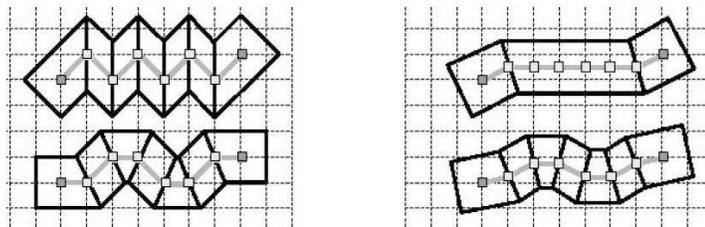


Figura 2.18: Suavizado con un filtro binomial. El diagrama con picos (izquierda) es transformado en un suave esqueleto (derecha). Los pequeños cuadrados centrales representan los centros de los voxels. Nótese que los puntos inicial y final del esqueleto no se modifican. Fuente: Hahn *et al.* [2001]

Bajo un criterio parecido, existen técnicas de teselado<sup>12</sup> usando voxels para rellenar los planos horizontales [Cebral y Löhner, 1999]. Asimismo, Cebral y Löhner [1999] consideran necesario un procedimiento para armar el modelo como grupos de elementos separados. Sin embargo, es claro que la geometría de los voxels genera, por tanto, un esqueleto irregular con demasiados picos y saltos bruscos sobre la superficie, por lo que es necesario aplicar - como en [Hahn *et al.*, 2001] - un primer suavizado, cuya idea básica es calcular nuevas coordenadas de cada punto de la malla, recubriéndola al mismo tiempo con triángulos sobre determinadas caras de los voxels. Opcionalmente, se aplica un mejoramiento del mallado, colapsando aristas y diagonales innecesarias, para finalmente colocar parches discretos en huecos de la malla que así lo requieran. Las fases del proceso pueden observarse en la Figura 2.19.

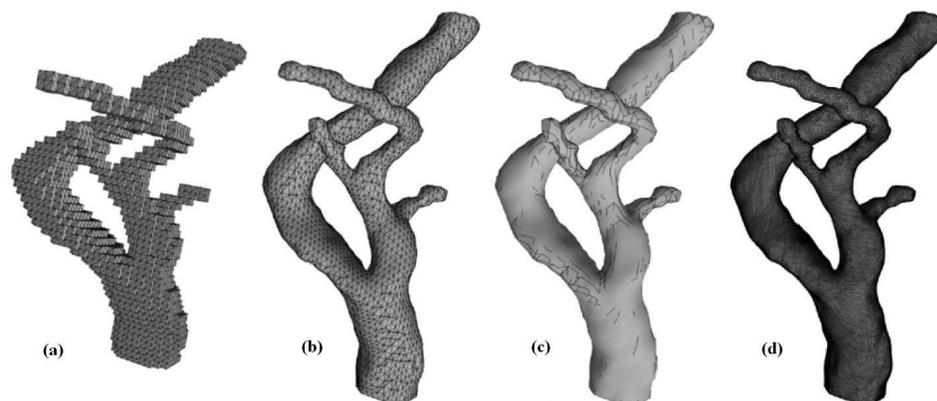


Figura 2.19: Reconstrucción de la carótica derecha. (a) Teselado (Tessellation), (b) Suavizado (Smoothing), (c) Reacomodo (Discrete patches) y (d) Malla final. Fuente: Cebral y Löhner [1999]

---

<sup>12</sup>Del latín *Tessellatus*, se define como una partición de un plano mediante polígonos idénticos, o un grupo de polígonos idénticos, que convenientemente agrupados recubren enteramente dicho plano. Un teselado también es una regularidad o patrón de figuras que cubre o pavimenta completamente una superficie plana que cumple con dos requisitos: (a) no quedan huecos y (b) no se superponen las figuras.

## Capítulo 3

# Método propuesto

Hechas las observaciones pertinentes en los capítulos anteriores, se describe a continuación el desarrollo del método elegido que permita la generación de una malla tubular *correcta* que satisfaga los objetivos propuestos para esta tesis. Cabe aclarar que la explicación del método final es consecuencia de la pre-experimentación con algunos de los datos y, por tanto, de la mejora de los algoritmos originales.

Como se vió en el capítulo introductorio, la naturaleza de la información de los datos de entrada podríamos definirla como *básica*, ya que contando únicamente con el esqueleto central del árbol vascular se define el conjunto de vértices necesarios para el modelo. Por tal razón, a través de algunas pruebas, se buscaron los ajustes que mejoraran la calidad de las mallas resultantes y la visualización de las mismas.

Debido a que los datos están dispuestos en forma de un árbol binario (ver Figura 1.2), se utiliza una estructura de datos de este tipo donde cada *nodo* contiene toda la información correspondiente al  $k$ -ésimo segmento. De esta forma, se pueden definir recursivamente los procedimientos que el algoritmo requiera para descender en el árbol y aplicarlos a todos los *nodos*, al mismo tiempo que se puede operar sobre aquellos que forman implícitamente las bifurcaciones.

### 3.1. Primera aproximación

Inicialmente, el método consistió en “seguir“ la idea general presentada por Barratt *et al.* [2004], trazando circunferencias horizontales al *eje Z*, aprovechando que la naturaleza de los *puntos de control* con los que se dispone son puntos en  $\mathbb{R}^3$ . Así, cada entrada  $(x_i, y_i, z_i)$  de la lista se convierte en la posición del centro de un círculo y el valor  $r_{k_i}$  es entonces el radio del mismo. Además, se utilizaron líneas y mallas de triángulos para visualizar el modelo tridimensional.

### 3.1.1. Obtención de puntos de la malla

Con base en la idea de curvas paramétricas y cilindros, se propuso construir la malla proporcionando una colección de vértices y, utilizando las primitivas adecuadas de la biblioteca gráfica *OpenGL* sobre el lenguaje de programación *C++*, pintar el trazado de las líneas entre dichos puntos.

Se realizó la lectura del listado de datos descrito en la Sección 1.2 para generar puntos sobre la circunferencia paramétrica  $C_i$  determinada por el punto  $P_i$  y el valor  $r_{k_i}$  (centro y radio de la circunferencia, respectivamente), tal que:

$$C_i(t) = \{(x, y) \in \mathbb{R}^2 | X^2 + Y^2 = r_{k_i}^2, t \in [0, 2\pi]\}, \text{ donde } \begin{cases} X = x_i + r_{k_i} * \cos(t) \\ Y = y_i + r_{k_i} * \sin(t) \end{cases} \quad (3.1)$$

Como vemos, la coordenada  $z_i$  del punto solo fue considerada para determinar la posición del plano  $XY$  sobre la que se colocaba la circunferencia, de modo que tales círculos consisten en secciones transversales al eje  $Z$ . Una vez conseguidos los vértices (los cuales son obtenidos en tiempo de ejecución), se realiza una inspección sobre los datos de entrada seleccionados previamente para determinar el primero y el último punto del segmento  $k$  en turno.

### 3.1.2. Trazado de líneas

En esta parte del proceso se unen los vértices mediante líneas que definan las caras del modelo; en particular, utilizando `GL_LINES_STRIP`<sup>1</sup> se trazaron las circunferencias transversales al eje  $Z$  (Figura 3.1(a)). Asimismo, para generar el cuerpo cilíndrico se trazan líneas o mallas de triángulos (`GL_LINES` y `GL_TRIANGLE_STRIP`, respectivamente, según determine el usuario) entre dos círculos siempre y cuando los identificadores  $n_{k_i}$  de las entradas correspondientes pertenezcan al mismo segmento  $k$  (Figura 3.1(b)).

Hasta aquí fueron generados los cilindros de cada segmento sin considerar la zona de bifurcación que hay entre ellos. Para resolver la malla en esta región, de manera preliminar, se unieron los vértices finales de la rama padre con los vértices iniciales de cada una de las ramas hijas, sin considerar por ahora las posibles intersecciones y superposiciones generadas entre líneas de la malla (Figura 3.1(c)).

### 3.1.3. Resultados preliminares

Utilizando el procedimiento anterior, se realizó la ejecución del programa con algunos archivos de puntos generados por RISA [Martinez-Perez *et al.*, 2007; Martinez-

---

<sup>1</sup>`GL_LINES_STRIP` es una de las primitivas de *OpenGL*

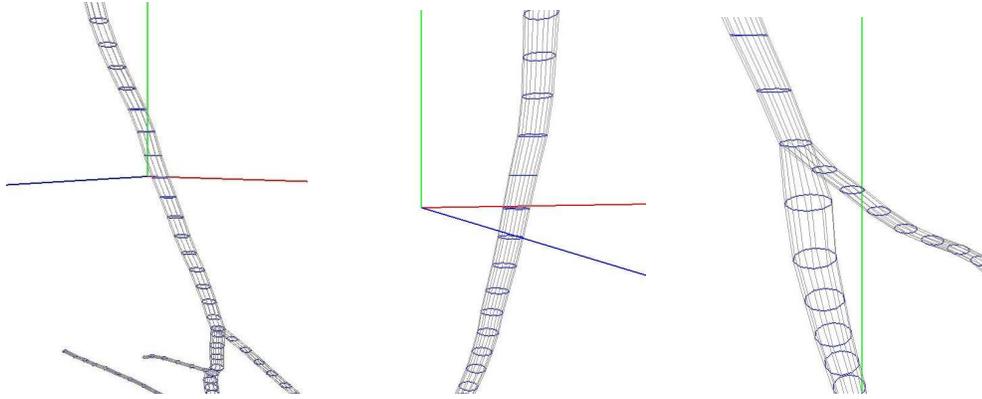


Figura 3.1: Visualización tridimensional de una estructura tubular. (a) Representación 3-D del vaso (el eje Z en color verde), (b) región del cuerpo tubular (las circunferencias en color azul, la malla en color gris) y (c) bifurcación, obsérvese la malla que une los segmentos.

Perez y Espinosa-Romero, 2004]. Moviendo la posición de la cámara para observar los detalles más relevantes, vemos en la Figura 3.2 una representación tubular aceptable, donde la malla parece suave, debido a la relativa simetría de la estructura. La Figura 3.2(a) muestra la malla usando `GL_LINES` entre circunferencias, mientras que en Figura 3.2(b) se emplea `GL_TRIANGLE_STRIP` rellenando el volumen de la superficie.

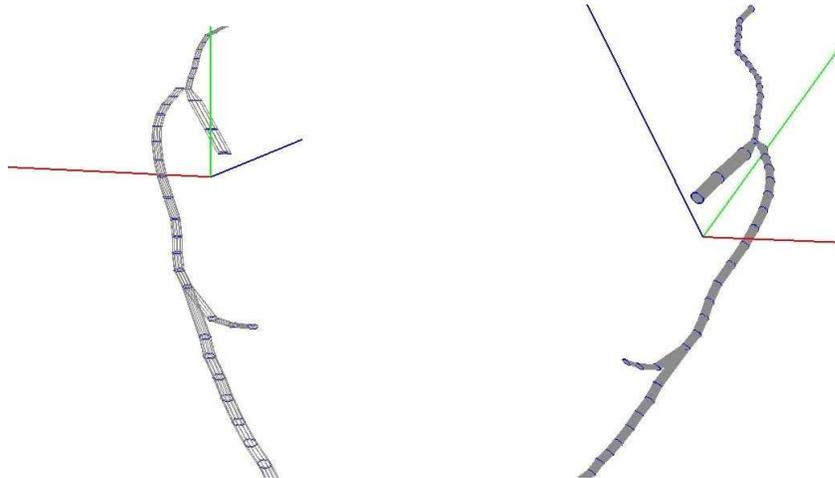


Figura 3.2: Reconstrucción tubular apropiada, favorecida por la simetría de los puntos centrales del esqueleto. (a) Malla trazada con `GL_LINES` y (b) malla trazada con `GL_TRIANGLE_STRIP` vista desde otro ángulo.

Nótese que en las regiones donde el vaso mantiene una dirección uniforme la malla está mejor definida. En cambio, en la Figura 3.3 en algunas ramas próximas a una bifurcación las circunferencias se encuentran más separadas tanto horizontal como verticalmente, consecuencia de su mala orientación (Figura 3.3(a)). Esto claramente dificulta el correcto trazado del tubo, pues si se realiza la triangulación entre dos circunferencias contiguas donde se presente este problema se obtiene una superficie con volumen reducido y casi completamente plana (Figura 3.3(b), (c)).

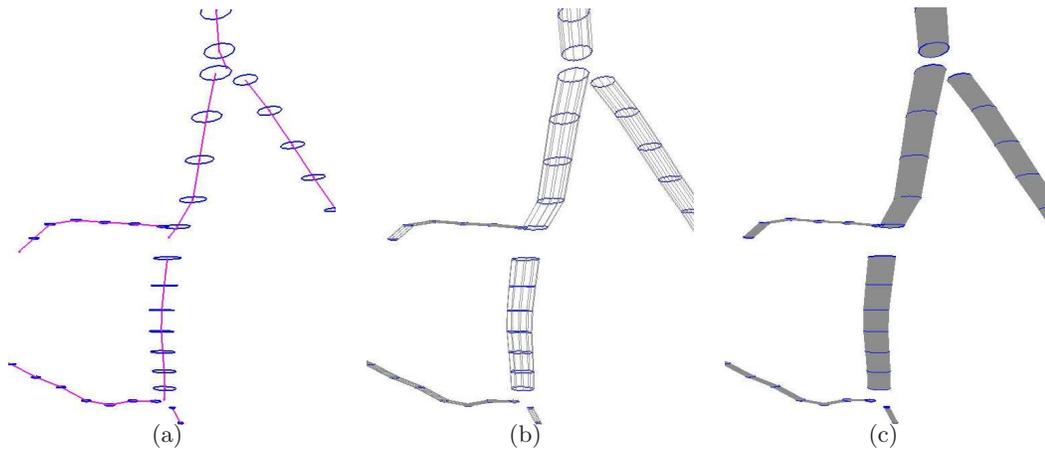


Figura 3.3: Visualización del trazado de la malla. (a) Circunferencias transversales al eje Z (obsérvese la separación entre circunferencias contiguas), (b) trazado de líneas de la estructura y (c) malla generada con `GL_TRIANGLE_STRIP` (se distingue una superficie con poco volumen).

En principio, éste es un algoritmo que, en términos generales, adapta de manera sencilla los datos que se tienen al modelo tridimensional logrado. Asimismo, la generación de los puntos sobre la malla se consigue fácilmente mediante una parametrización muy simple y poco costosa en cuanto a operaciones aritméticas (Figura 3.1(a), (b) y Figura 3.2).

Sin embargo, aunque todavía no se resuelve correctamente las zonas de bifurcación (de modo que evite la superposición de líneas entre segmentos), uno de los segmentos de forma sencilla y rápida si se desea tan sólo apreciar la estructura cerrada del tubo (Figura 3.1(c)). Por otra parte, una desventaja evidente es la disminución del volumen en las regiones donde las circunferencias se encuentran mal orientadas, lo anterior debido a que se toman los círculos en secciones transversales al eje Z siempre (Figura 3.3).

## 3.2. Construcción del modelo de malla

Hasta ahora, la malla conseguida se aproxima poco a la representación deseada; asimismo, la colección de puntos y la definición de las caras en las zonas de bifurcación es inapropiada y no corresponde a una malla. Por otro lado, es importante también mencionar que si bien podría utilizarse alguna técnica como la expuesta por Cebral y Löhner [1999] y únicamente visualizar las estructuras como agrupamientos de voxels partiendo desde el esqueleto central de los tubos, no se hará de esa manera; ya que se pretende reducir el uso de recursos y trabajar directamente con mallas y los vértices necesarios para éstas, además de aprovechar tales datos en otras aplicaciones que serán mencionadas más adelante.

Ahora bien, a pesar de los resultados anteriores, utilizar curvas y superficies como círculos y cilindros, respectivamente, aún parece ser el primer paso para obtener el modelo. Para ello, es necesario analizar las condiciones de los datos iniciales y hacer las correcciones necesarias. A diferencia de Barratt *et al.* [2004], la información no proviene de cortes transversales de los vasos sanguíneos, tampoco se deriva de un escaneo que proporcione ya un conjunto de vértices; concretamente, RISA opera sobre imágenes de *fondo de ojo*, por ende, imágenes planas que capturan horizontalmente el contenido del interior del globo ocular. Una vez que se realiza la segmentación y los datos son reconstruidos en 3D [Martinez-Perez y Espinosa-Romero, 2004], se tiene la colección de puntos  $P_i$  definiendo una curva central por cada vaso.

### 3.2.1. Cilindros generalizados

Como puede verse, la generación de los vértices sobre circunferencias transversales al eje Z no contempla la información existente entre los *puntos de control* de los tubos; la primera mejora es entonces aplicar el método de *cilindros generalizados* usando circunferencias ahora perpendiculares a la dirección del esqueleto de cada segmento tubular, para cada posición  $P_i$ .

Para cada segmento del árbol se procede como sigue:

```
1 cilindrosGeneralizados_1() {
2   L <- this.segmento.datos.size();
3
4   for i <- 0 to L - 1: //for(i = 0; i < L - 1; i++)...
5     // Se calcula el vector de orientacion
6     pI <- this.segmento.datos.at(i).p;
7     pF <- this.segmento.datos.at(i + 1).p;
8     v <- Vector(pI, pF);
9
10    // Se definen los valores para generar una circunferencia
11    // con los parametros indicados
12    c <- Circle(pI, this.segmento.radio, v);
```

```

13
14         // Se obtienen los N vertices que definen cada
15         // circunferencia
16         vtxs <- c.getVertexs_1(N);
17         this.segmento.vertices.add(vtxs);
18     end-for
19
20     // Un nodo es 'final' si el segmento no genera
21     // una bifurcacion
22     if final() then:
23         // Se utiliza el ultimo vector calculado
24         // en el ciclo 'for' anterior
25         c <- Circle(pI, this.segmento.radio, v);
26         vtxs <- c.getVertexs_1(N);
27         this.segmento.vertices.add(vtxs);
28     else
29         // Se obtiene el ultimo punto del esqueleto
30         // del segmento en turno
31         pI <- this.segmento.datos.last().p;
32
33         // Los primeros puntos de los nodos izquierdo
34         // y derecho, respectivamente
35         pL1 <- left.segmento.first().p;
36         pR1 <- right.segmento.first().p;
37         // El punto final se consigue como promedio de los
38         // 3 puntos anteriores
39         pF <- puntoIntermedio(pI, pL1, pR1);
40
41         v <- Vector(pI, pF);
42         c <- Circle(pI, this.segmento.radio, v);
43         vtxs <- c.getVertexs_1(N);
44         this.segmento.vertices.add(vtxs);
45
46         // Se hace la llamada recursiva
47         left.cilindrosGeneralizados();
48         right.cilindrosGeneralizados();
49     end-if
50 }
51
52 getVertexs_1(n) {
53     // Se computa la matriz de rotacion correspondiente al
54     // vector 'v' de la circunferencia e internamente
55     // respecto al vector canonico (0, 0, 1)
56     MRot <- Matrix(v);
57
58     // Se aplica la matriz de rotacion 'MRot' y el radio 'r'
59     // para obtener los 'n' puntos sobre la circunferencia
60     for i <- 0 to n:
61         t <- i/n * (2.0 * PI);
62         x <- r*cos(t);
63         y <- r*sin(t);

```

```

64     tmp <- Point(x, y, 0);
65     rho = MRot.getPosition(tmp);
66
67     // Se calcula el vertice final trasladando el punto
68     // re-orientado 'rho' con respecto al centro 'pI'
69     // de la circunferencia (vease el codigo de la funcion
70     // anterior)
71     vertex <- rho + pI;
72     list.add(vertex);
73
74   end-for
75
76   return list;
77 }

```

En la Figura 3.4 se observa que, aplicando el algoritmo anterior, se resuelve adecuadamente la geometría de la malla para los segmentos tubulares, formando las caras de los cilindros por medio de cuadriláteros para visualizar la malla. Además, la función *getVertices\_1* arriba indicada modifica la ecuación 3.1, de modo que:

$$C_i(t) = \{(x, y, z) \in \mathbb{R}^3 | X^2 + Y^2 + Z^2 = r_{k_i}^2, t \in [0, 2\pi]\}, \quad \text{donde} \quad \begin{cases} X = x_i + x_\rho \\ Y = y_i + y_\rho \\ Z = z_i + z_\rho \end{cases} \quad (3.2)$$

$$\begin{pmatrix} x_\rho \\ y_\rho \\ z_\rho \end{pmatrix} = R * \begin{pmatrix} u \\ v \\ 0 \end{pmatrix}, \quad \begin{cases} u = r_{k_i} * \sin(t) \\ v = r_{k_i} * \cos(t) \end{cases} \quad (3.3)$$

### 3.2.2. Malla canónica

En general, el algoritmo permite construir los tubos usando una cantidad arbitraria de  $N$  puntos por cada circunferencia definida; sin embargo, elegir un número muy grande y con ello intentar armar los elementos en las zonas de bifurcación no sería sencillo; además, realizar dicha tarea únicamente con los puntos de las ramas más cercanos entre sí se vería muy forzado, ya que se requiere que la transición sea suave, aunque bien definida.

Por tanto, se propone comenzar con una malla más simple, específicamente una malla *semi-regular* que será definida como malla *canónica*, la cual consistirá en un modelo de cilindros en el que cada circunferencia se genera con sólo 4 vértices (ver Figura 3.5).

Asimismo, la elección de ese pequeño número de puntos no es arbitraria, pues es un punto clave para solucionar las bifurcaciones, construyéndolas, de la siguiente manera:

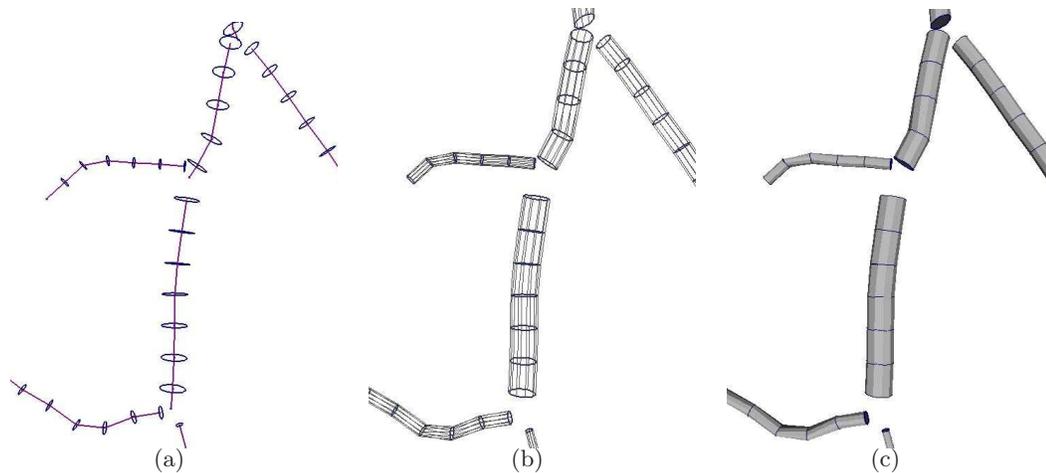


Figura 3.4: Malla obtenida usando el método de *cilindros generalizados*. (a) El procedimiento orienta correctamente las circunferencias, ahora perpendiculares a la dirección del esqueleto de cada segmento. (b) Se representan las superficies ya sin pérdida de volumen. (c) La estructura cerrada de los tubos está mejor definida. (Comparar con la Figura 3.3)

```

1 puntosBifurcacion_1() {
2     if (!final()) then:
3         L <- this.segmento.datos.size();
4         // Se toman los ultimos 4 vertices del segmento padre
5         // y los 4 primeros vertices de cada segmento hijo
6         // NOTA: 'take' excluye el segundo indice
7         PP <- this.segmento.datos.take(L - 4, L);
8         PR <- right.segmento.datos.take(0, 4);
9         PL <- left.segmento.datos.take(0, 4);
10
11         // Se obtienen uno a uno - siguiendo el orden en que
12         // fueron generados - los puntos medios respectivos
13         for i <- 0 to 4:
14             MPL.add(puntoMedio(PP[i], PL[i]));
15             MPR.add(puntoMedio(PP[i], PR[i]));
16         end-for
17
18         // Se calculan, 4 nuevos vertices usando 4 puntos,
19         // 2 de MPL y 2 MPR, ademas de dos valores de dos
20         // parametros elegidos 'arbitrariamente' cercanos
21         t1 <- 0.45;
22         t2 <- 0.55;
23         NVtxs.add(nuevoPunto(MPL[0], MPR[3], t1));
24         NVtxs.add(nuevoPunto(MPL[1], MPR[2], t2));
25         NVtxs.add(nuevoPunto(MPL[1], MPR[2], t2));
26         NVtxs.add(nuevoPunto(MPL[0], MPR[3], t1));
27

```

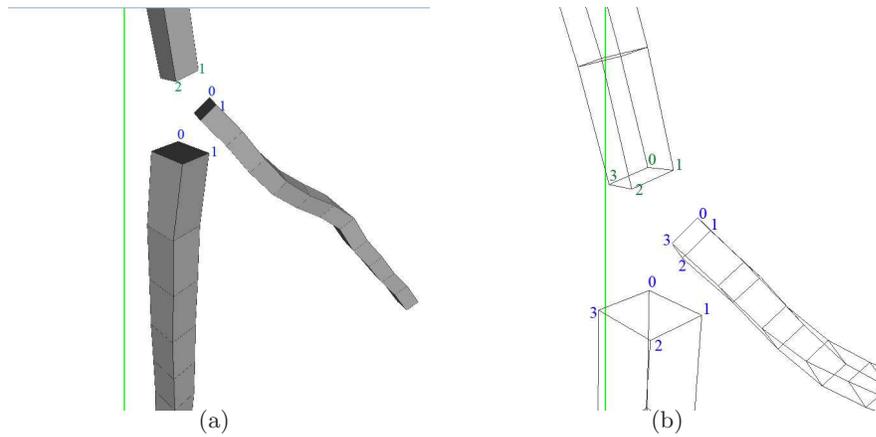


Figura 3.5: Estructura inicial de los cilindros que conforman la malla *canónica*. (a) Cada circunferencia se define únicamente con 4 vértices. (b) Los vértices sobre las circunferencias son generados siempre en sentido de las manecillas del reloj.

```

28 // Se insertan tales puntos al final del segmento actual
29 this . segmento . datos . add ( NVtxs );
30
31 // Se realizan las asignaciones de los nuevos vertices
32 // generados para armar las 5 caras de la bifurcacion
33 reasignar ( ) ;
34
35 // Por ultimo , la llamada recursiva
36 left -> puntosBifurcacion_1 ( ) ;
37 right -> puntosBifurcacion_1 ( ) ;
38 end-if
39 }

```

Nótese que para la construcción de los nuevos vértices se eligen los puntos que se supondrían más cercanos, ya que se sigue el mismo orden en que se generan los vértices de cada segmento por separado (véase Figura 3.5(b)). No obstante, utilizar ese mismo orden en todos los segmentos no es el problema, sino la referencia al primer punto - y en consecuencia, los posteriores - que define la circunferencia inmediata de cada cilindro hijo en la bifurcación; pues, en algunos casos, se obtendrán vértices nada útiles, generando traslapes entre los cilindros (Figura 3.6(a) y (b)). De la misma manera, asignar dichos vértices a los cilindros hijos no siempre se hace correctamente debido a que la elección de puntos medios entre los segmentos no garantiza un orden que evite la superposición de las superficies (Figura 3.6(c)).

Por ello, es necesario *referir* a los vértices del cilindro padre de modo que se pueda resolver cuáles son los que tienen que emplearse, permitiendo al mismo tiempo ge-

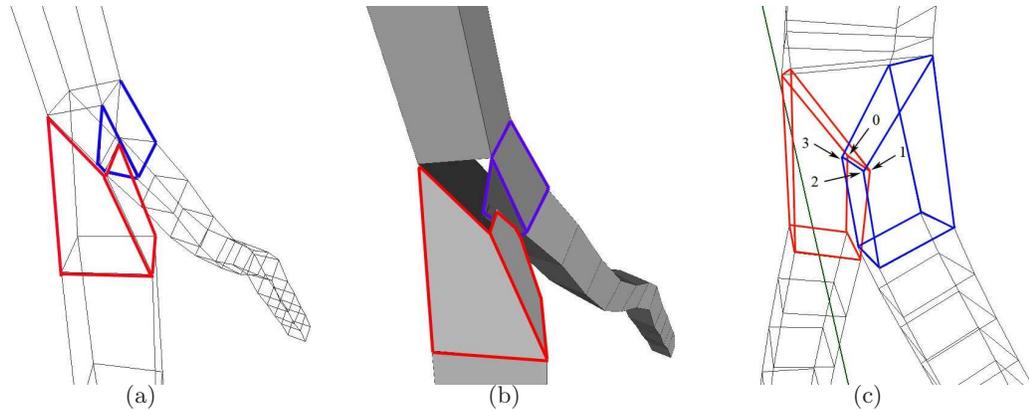


Figura 3.6: Traslapo de cilindros. (a) Se observan las aristas que producen la superposición; (b) se distingue claramente el ensamble entre las superficies. (c) Orden invertido en la asignación de los vértices intermedios generados; las flechas señalan los puntos cruzados.

nerar los vértices de los cilindros hijos calculando estratégicamente la posición del vértice inicial, al menos, para la primer circunferencia paramétrica de cada uno.

La propuesta para el ajuste consiste en usar un índice, que se llamará *índice de referencia* ( $idx_{ref}$ ), el cual señalará la posición del primer vértice *útil* en la última circunferencia del cilindro padre y usando ese mismo dato cambiar el valor donde arranque el parámetro  $t$  en las ecuaciones 3.2 y 3.3. Asimismo, para reordenar y asignar los vértices calculados, se buscará la distancia que sea menor desde el centro de la circunferencia inicial de uno de los cilindros hijos hacia esos vértices, para determinar si existe o no una inversión. La solución se comenta a detalle a continuación:

```

1  cilindrosGeneralizados_2() {
2      L <- this.segmento.datos.size();
3      for i <- 0 to L - 1:
4          // Se recupera el índice de referencia; el valor por
5          // defecto es idx = 0, lo cual se mantiene para el
6          // primer segmento, pero puede cambiar en la recursion
7          idx <- this.segmento.idx_ref;
8
9          pI <- this.segmento.datos.at(i).p;
10         pF <- this.segmento.datos.at(i + 1).p;
11         v <- Vector(pI, pF);
12         c <- Circle(pI, this.segmento.radio, v);
13
14         // Ahora, la funcion 'getVertex' recibe ahora dos
15         // parametros, uno de ellos el índice de referencia 'idx'
16         // NOTA: Se usa el mismo valor para cada circunferencia
17         // del segmento en turno

```

```

18         vtxs <- c.getVertexs_2(N, idx);
19         this.segmento.vertices.add(vtxs);
20     end-for
21
22     if final() then:
23         c <- Circle(pI, this.segmento.radio, v);
24         vtxs <- c.getVertexs_2(N, idx);
25         this.segmento.vertices.add(vtxs);
26     else
27         pI <- this.segmento.datos.last().p;
28         pL1 <- left.segmento.first().p;
29         pR1 <- right.segmento.first().p;
30         pF <- puntoIntermedio(pI, pL1, pR1);
31         v <- Vector(pI, pF);
32
33         c <- Circle(pI, this.segmento.radio, v);
34         vtxs <- c.getVertexs_2(N, idx);
35         this.segmento.vertices.add(vtxs);
36
37         // Por ultimo, se obtiene el indice de ajuste y se
38         // actualiza el valor correspondiente en los datos
39         // de cada segmento hijo
40         idx = getIndex();
41         left.data.idx_ref = (idx + 2)%4;
42         right.data.idx_ref = idx;
43
44         left.cilindrosGeneralizados();
45         right.cilindrosGeneralizados();
46     end-if
47 }
48
49 getIndex() {
50     // Se utilizan los valores necesarios de cada
51     // segmento hijo
52     pL1 <- left.segmento.first().p;
53     vL1 <- left.segmento.first().v;
54     rL <- left.segment.radio;
55
56     pR1 <- right.segmento.first().p;
57     vR1 <- right.segmento.first().v;
58     rR <- right.segment.radio;
59
60     // Se calculan vertices temporales sobre las
61     // circunferencias correspondientes
62     cL <- Circle(pL1, rL, vL1);
63     vtxsL <- cL.getVertexs_2(N, 0);
64
65     cR <- Circle(pR1, rR, vR1);
66     vtxsR <- cR.getVertexs_2(N, 0);
67
68     distMax <- MIN_VALUE;

```

```

69     dist <- -1;
70     for i <- 0 to 4:
71         // Notese que el calculo de los puntos medios se
72         // realiza siempre con puntos 'opuestos' respecto
73         // al orden en que se generan arriba
74         mR <- puntoMedio(cR[i], cR[(i + 1)%4]);
75         mL <- puntoMedio(cL[(i + 2)%N], cL[(i + 3)%4]);
76
77         dist <- distance(mL, mR);
78         if distMax < dist then:
79             // Si la distancia se maximiza, entonces se ha
80             // encontrado un mejor 'indice de referencia'
81             distMax <- dist;
82             idx <- i;
83         end-if
84     end-for
85
86     return idx;
87 }

```

En la Figura 3.7 se puede apreciar el proceso y el resultado del algoritmo basado en las dos funciones anteriores. Las ecuaciones resultantes usando el *índice de referencia* serían las siguientes:

$$C_i(t, idx_{ref}) = \{(x, y, z) \in \mathbb{R}^3 | X^2 + Y^2 + Z^2 = r_{k_i}^2, t \in [0, 2\pi]\}, \quad (3.4)$$

$$X = x_i + x_\rho, \quad Y = y_i + y_\rho, \quad Z = z_i + z_\rho,$$

$$\begin{pmatrix} x_\rho \\ y_\rho \\ z_\rho \end{pmatrix} = R * \begin{pmatrix} u \\ v \\ 0 \end{pmatrix}, \quad \begin{aligned} u &= r_{k_i} * \sin(t) \\ v &= r_{k_i} * \cos(t) \end{aligned} \quad (3.5)$$

$$t = \frac{(j+idx_{ref})\%N}{N} * (2\pi), \quad j = 0, \dots, N - 1; \quad N = 4$$

donde N es el número de vértices por circunferencia.

De esta manera, el método para completar la malla *canónica* en las bifurcaciones quedaría como sigue:

```

1     puntosBifurcacion_2() {
2         if (!final()) then:
3             L <- this.segmento.datos.size();
4             PP <- this.segmento.datos.take(L - 4, L);
5             PR <- right.segmento.datos.take(0, 4);
6             PL <- left.segmento.datos.take(0, 4);
7
8             // Se calculan los valores finales de los indices

```

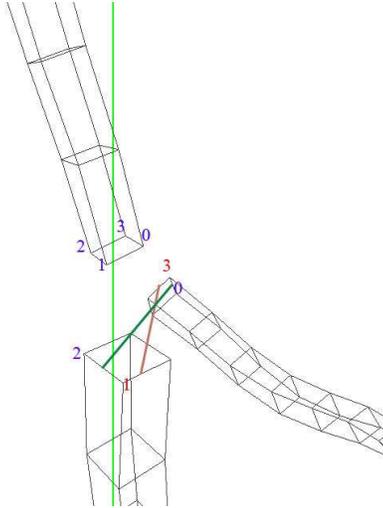


Figura 3.7: Cálculo del *índice de referencia*. La función *getIndex* obtiene el valor del índice que se relaciona con la distancia máxima entre los puntos opuestos de las primeras circunferencias de cada cilindro.

```

9      // utiles , usando el 'índice de referencia' del
10     // nodo padre y el de cada uno de los nodos hijos
11     idx = this.segmento.idx_ref;
12     idxL <- |4 - (idx - left.segmento.idx_ref)|%4;
13     idxR <- |4 - (idx - right.segmento.idx_ref)|%4;
14
15     for i <- 0 to 4:
16         MPL.add(puntoMedio(PP[(i + idxL)%4], PL[i]));
17         MPR.add(puntoMedio(PP[(i + idxR)%4], PR[i]));
18     end-for
19
20     // En seguida , los 4 nuevos vertices generados se
21     // obtienen , con los 4 puntos medios - bien
22     // posicionados - mas cercanos
23     t1 <- 0.45;
24     t2 <- 0.55;
25     aux.add(nuevoPunto(MPL[2], MPR[3], t1));
26     aux.add(nuevoPunto(MPL[3], MPR[2], t2));
27     aux.add(nuevoPunto(MPL[3], MPR[2], t2));
28     aux.add(nuevoPunto(MPL[2], MPR[3], t1));
29
30     // Dos puntos medios mas son calculados usando los
31     // puntos opuestos , guardados en la lista 'aux'
32     pAux01 <- puntoMedio(aux[0], aux[1]);
33     pAux23 <- puntoMedio(aux[2], aux[3]);
34
35     // Ahora se obtiene el centro de la primer
36     // circunferencia del cilindro derecho

```

```

37     m <- Point();
38     for i <- 0 to 4:
39         m <- m + PR[i];
40     end-for
41     m <- m * (0.25);
42
43     // Se determina si existe o no inversion,
44     // comparando las distancias del punto centro
45     // 'm' a los puntos 'pAux01' y 'pAux23'
46     distA <- m.distance(pAux01);
47     distB <- m.distance(pAux23);
48
49     // Si la comparacion es cierta, existe inversion
50     // y habran de reordenarse los vertices
51     if distA >= distB then:
52         for i <- aux.size() - 1 to 0:
53             NVtxs.add(aux[i]);
54         end-for
55     else
56         for i <- 0 to 4:
57             NVtxs.add(aux[i]);
58         end-for
59     end-if
60
61     this.segmento.datos.add(NVtxs);
62
63     // En la reasignacion de vertices, se emplea
64     // tambien el 'indice de referencia'
65     reasignar();
66
67     left -> puntosBifurcacion_2();
68     right -> puntosBifurcacion_2();
69 end-if
70 }

```

Por último, la finalidad del procedimiento de reasignación de vértices mencionado es simplificar el renderizado y el algoritmo de subdivisión que será descrito en la siguiente sección, ya que sirve para completar los cilindros y formar las caras faltantes en las bifurcaciones del modelo. Un ejemplo de esto puede verse en la Figura 3.8.

### 3.3. Refinamiento

Evidentemente, tratar de representar las superficies tubulares únicamente con la malla *canónica* sería poco útil para los fines médicos que se buscan satisfacer con la visualización de los tubos. En vez de eso, se mejorará el nivel de detalle y la fidelidad del modelo incrementando la cantidad de vértices de la malla, por medio de refinamiento iterativo aplicando el algoritmo de subdivisión desarrollado por Catmull y Clark [1978].

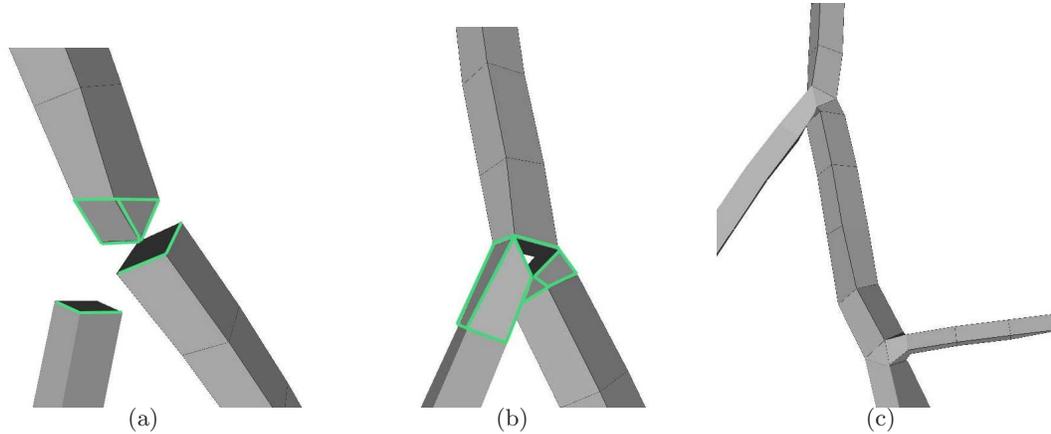


Figura 3.8: Reasignación de vértices y formación de las caras de la bifurcación. (a) Una vez que se generan los últimos vértices del cilindro padre con la función *puntosBifurcacion*, (b) se reacomodan los vértices en los cilindros de los nodos hijos para formar las *caras laterales* de la bifurcación. (c) Finalmente, se arman las *caras restantes* completando el modelo.

Además, para lograr un mejor efecto de suavizado de las superficies de los modelos y distinguir los detalles de la malla, se utiliza un modelo de iluminación sencillo, activando una sola fuente de luz ambiental, `glEnable(GL_LIGHT0)`, colocada en una posición fija. Asimismo, se calculan las normales por cada cuadrilátero que define una cara de la malla y se asocia el color aplicado a la superficie de cada malla como un material, `glEnable(GL_COLOR_MATERIAL)`.

Por su parte, el algoritmo de subdivisión original propuesto por Catmull y Clark [1978], además de ser matemáticamente simple, parece suficiente para conseguir un refinamiento aceptable de la malla *canónica*. En resumen, dicho procedimiento se basa en calcular 3 conjuntos de puntos utilizando como entrada una malla poligonal, mediante las siguientes reglas:

1. *Puntos Cara (Face Points)*: Corresponden al promedio de los viejos vértices que definen cada cara (polígono) de la malla previa.
2. *Puntos Arista (Edge Points)*: Se consiguen promediando los puntos medios de cada vieja arista de la malla con el promedio de los dos nuevos *puntos cara* de las caras que comparten la arista en turno.
3. *Puntos Vértice (Vertex Points)*: Serán los nuevos puntos que reemplacen las viejas posiciones de los vértices de la malla de entrada. Se obtienen con la siguiente expresión:

$$S' = \frac{Q}{n} + \frac{2R}{n} + \frac{S(n-3)}{n} \quad (3.6)$$

donde  $Q$  es el promedio de los nuevos *puntos cara* de todas las caras adyacentes a  $S$ ,  $R$  es el promedio de los puntos medios de todas las viejas aristas incidentes en  $S$ ,  $S$  es el viejo vértice,  $n$  es el número de vértices adyacentes a  $S$  y  $S'$  es el nuevo vértice calculado.

Para entender mejor como funciona el algoritmo, observése la Figura 3.9, donde se ilustra la generación de los nuevos puntos, aplicando las reglas anteriores sobre la malla poligonal definida por sus vértices y aristas (Figura 3.9(a)). Asimismo, debe ser claro que tanto los *puntos cara*, *puntos arista* y *puntos vértice* serán todos nuevos vértices de la malla refinada. Así, por ejemplo, el cálculo del *punto arista*  $q_{12}$  (Figura 3.9(b) y (c)) está dado por:

$$q_{12} = \frac{\frac{(C+D)}{2} + \frac{(p_{12}+p_{22})}{2}}{\frac{p_{22}}{2}}, \quad (3.7)$$

$$C = q_{11} = \frac{p_{11}+p_{12}+p_{21}+p_{22}}{4}, \quad D = q_{13} = \frac{p_{12}+p_{13}+p_{22}+p_{23}}{4}$$

Con la tercer regla, por ejemplo, el nuevo *punto vértice*  $q_{22}$  (Figura 3.9 (d)) se obtiene sustituyendo los valores en la ecuación 3.6 y haciendo los cálculos requeridos:

$$q_{22} = \frac{Q}{4} + \frac{2R}{4} + \frac{p_{22}*(4-1)}{4} = \frac{Q}{4} + \frac{R}{2} + \frac{p_{22}}{4} \quad (3.8)$$

tal que,

$$Q = \frac{q_{11}+q_{13}+q_{31}+q_{33}}{4}, \quad R = \frac{1}{4} * \left[ \frac{(p_{22}+p_{21})}{2} + \frac{(p_{22}+p_{23})}{2} + \frac{(p_{22}+p_{12})}{2} + \frac{(p_{22}+p_{32})}{2} \right] \quad (3.9)$$

donde

$$\begin{aligned} q_{11} &= \frac{p_{11}+p_{12}+p_{21}+p_{22}}{4}, & q_{13} &= \frac{p_{12}+p_{13}+p_{22}+p_{23}}{4}, \\ q_{31} &= \frac{p_{21}+p_{22}+p_{31}+p_{32}}{4}, & q_{33} &= \frac{p_{22}+p_{23}+p_{32}+p_{33}}{4} \end{aligned} \quad (3.10)$$

Como puede notarse las reglas de este algoritmo de subdivisión son aplicables, en principio, para mallas con cualquier geometría, aunque se recomienda utilizar mallas en las que la mayoría de sus elementos tengan la misma forma y vértices internos con la misma valencia, de tal modo que el número de *puntos extraordinarios* (Sección 2.5.2) sea reducido. En la Figura 3.9(a) se distinguen como *puntos extraordinarios* los vértices en el perímetro de la malla, por lo que se considerarán en los respectivos cálculos sólo los puntos necesarios correspondientes.

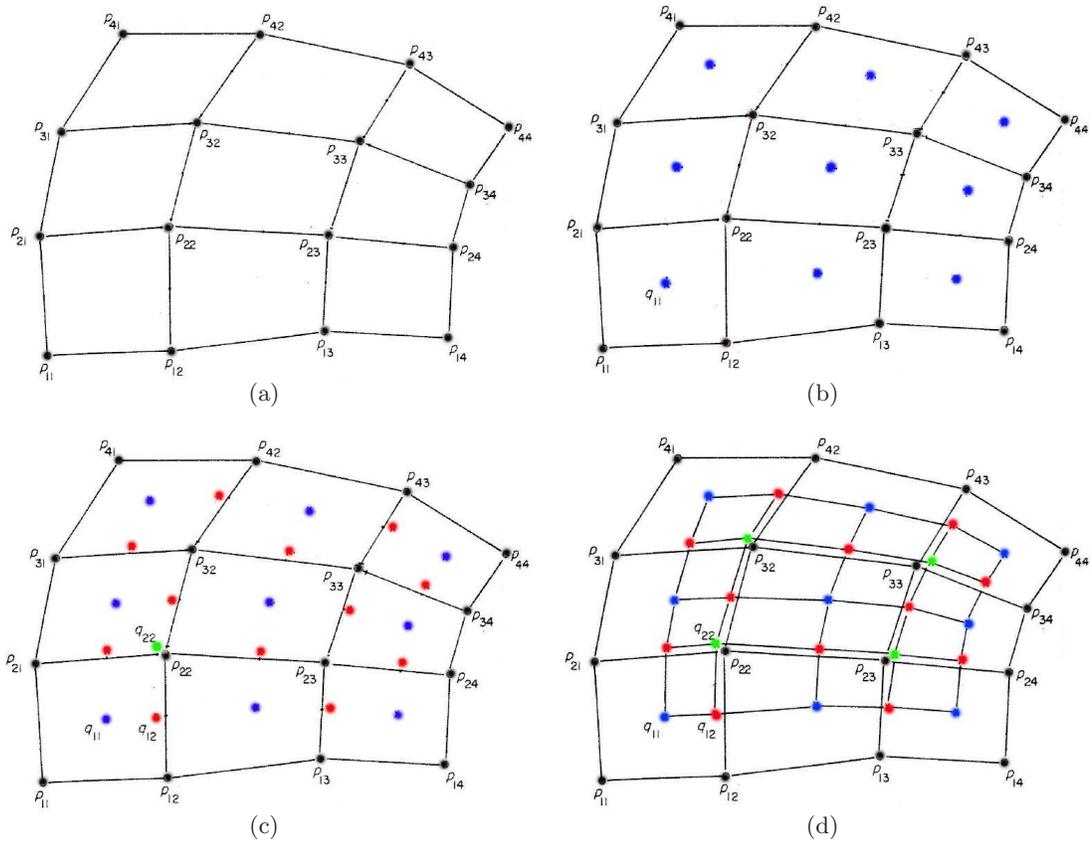


Figura 3.9: Fases del algoritmo de subdivisión de Catmull-Clark. (a) Una malla de entrada de estructura simple; los vértices se remarcan en color negro. (b) Se calculan los *puntos cara*, en azul. (c) Se generan los *puntos arista*, en color rojo. (d) Así, se calculan finalmente los nuevos *puntos vértice*, en verde.

Ahora bien, en el caso de la malla canónica construida se puede aplicar fácilmente el algoritmo para cada uno de los cilindros de manera independiente ya que la valencia de los vértices internos siempre es 4. Con base en esa idea, debido a que los cilindros no son superficies completamente cerradas, cada uno tendría inicialmente 8 *puntos extraordinarios*, 4 al inicio del cilindro y 4 al final, que también se pueden tratar de manera sencilla. Por último, ya como estructura completa, el modelo de cada malla canónica tendrá  $W$  *puntos extraordinarios* por nivel de detalle:

$$W = M + 4 * \beta + \omega * M \quad (3.11)$$

tal que  $M = 4 * 2^{L-1}$ ,  $L$  es el número de iteraciones (nivel de refinamiento),  $\beta$  el número de bifurcaciones y  $\omega$  el número de nodos finales.

En la Figura 3.10 se muestran sobre la malla, aún sin subdividir, los puntos extraordinarios correspondientes a cada sumando de la ecuación 3.11.

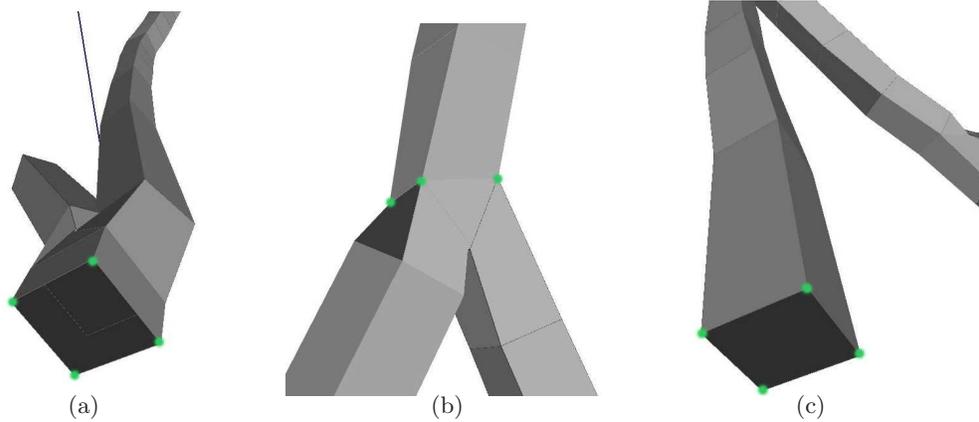


Figura 3.10: Puntos extraordinarios de la malla *canónica*: (a) los vértices en la primer circunferencia del segmento raíz, (b) los 4 vértices que unen cada una de las bifurcaciones y (c) los vértices en la última circunferencia de cada segmento *final*.

Finalmente, calcular los nuevos vértices en las zonas de bifurcación será lo más complicado de todo, porque cada una de las secciones (anterior, posterior e inferior) de las mismas, utilizarán determinados vértices tanto del nodo padre como de los nodos hijos. Para completar la solución se seguirán los siguientes pasos:

1. Subdividir cada cilindro por separado (Figura 3.11(a)).

La malla de cada segmento se obtiene de manera sencilla aplicando las 3 reglas del método de refinamiento, para un valor constante de 4 vértices adyacentes a cada vértice interno procesado. Los *puntos extraordinarios* al inicio y final del tubo se interpolan para únicamente completar la cantidad de puntos necesarios en los extremos del cilindro y después hacer el reemplazo de un número constante de vértices (Paso 3).

2. Aplicar refinamiento a cada una de las secciones restantes en la bifurcación, también por separado (Figura 3.11(b)), incluyendo, por ende, el cálculo de los *nuevos puntos extraordinarios* en tales regiones.

El cálculo de los nuevos vértices en cada sección de la bifurcación se hace formando una rejilla de puntos como un arreglo bidimensional; se utilizan vértices del final del cilindro padre y del inicio de los cilindros hijos para calcular algunos de los *puntos cara, arista y vértice* faltantes.

Los 4 *puntos extraordinarios* de la cada bifurcación tienen valencia igual a 5 y el cómputo de las nuevas posiciones de éstos es la implementación más extensa del método, debido al manejo de las posiciones de un mayor número de vértices en los cálculos, mediante el uso de los *índices de referencia* y valores relacionados con el incremento en la cantidad de vértices que se van generando en cada nivel del refinamiento.

3. Reemplazar los vértices correspondientes necesarios (Figura 3.11(c)) obtenidos en el Paso 1, por aquellos que se generan en el Paso 2, ya que las posiciones de los segundos vértices son las correctas.

Una vez que se consiguen los vértices que forman las bifurcaciones, aquellos en el perímetro de la rejilla de cada sección deben reemplazar a los vértices en las posiciones correspondientes de los cilindros padre e hijos, ya que el cálculo de los vértices en las bifurcaciones si usa la información de los 3 segmentos.

El resultado final de la malla subdividida se muestra en la Figura 3.11(d), en la cual se indica el reacomodo de los nuevos vértices en la región de la bifurcación.

En resumen, el algoritmo del método de generación de mallas tubulares aquí presentado facilita la aproximación de las superficies mediante cálculos geométricos sencillos y referencias a las posiciones de los vértices, en general, de una manera simple; principalmente para resolver las bifurcaciones. Utilizar alguna otra técnica que construya la malla y no facilite una indexación de tales posiciones de los datos, que requiera de almacenar la definición de los elementos de la malla o necesite uno o varios métodos de suavizado adicional, además de incrementar probablemente el tiempo de procesamiento de cómputo, sería más laborioso y tardado de implementar.

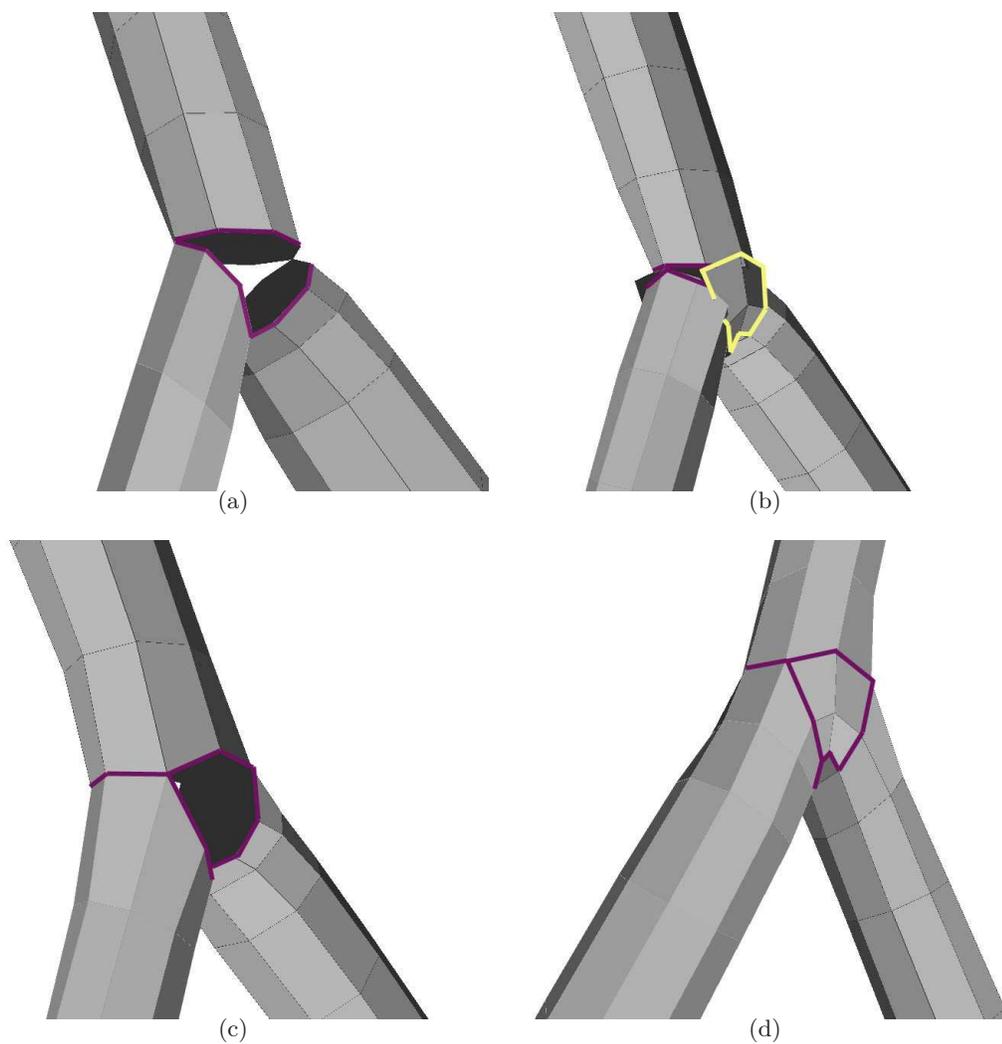


Figura 3.11: Resultado del algoritmo de subdivisión sobre la malla *canónica* (a) El refinamiento en cada cilindro por separado. (b) Se aplica subdivisión en las secciones de la bifurcación. (c) Reemplazo de los vértices en las secciones correspondientes de los 3 segmentos que forman la bifurcación. (d) La malla *canónica* final con un nivel de refinamiento.

## Capítulo 4

# Diseño y Desarrollo del Sistema

Como se vió en la Sección 1.1.2, los numerosos padecimientos que pueden presentarse en el ojo humano, promueven la construcción de un sistema que permita distinguir esas y otras anomalías en los vasos retinales con mayor anticipación y certeza. Aún hoy en día, ese diagnóstico muchas veces es realizado tomando en cuenta sólo las vistas *planas* de las imágenes de *fondo de ojo* u otros recursos limitados, pues se carece de muchas más herramientas con otras representaciones.

Por tanto, la codificación ligada a esta tesis consiste, en principio, en una aplicación que propocione una vista en tercera dimensión de vasos sanguíneos retinales, para apoyar a médicos especialistas (oftalmólogos y cirujanos) en la detección temprana y seguimiento de las enfermedades que suelen presentarse en ellos.

### 4.1. Especificación de requerimientos

Si bien algunas de las necesidades deseadas fueron señaladas en el primer capítulo, es importante mencionar claramente cada una de las características que deben satisfacer el programa, las mallas y el modelo tridimensional completo. Para ésto, se cubrirán los siguientes requerimientos de los usuarios:

- 1.- Que el sistema permita obtener las mallas de superficie de los segmentos tubulares y las bifurcaciones que se presenten en tales vasos.
- 2.- Almacene en disco las mallas generadas para su uso posterior en el mismo programa.
- 3.- Que pueda integrar cualquiera de los árboles de los distintos conjuntos de vasos, ya sea usando los datos proporcionados por RISA o mallas guardadas con anterioridad por el sistema.

- 4.- Facilite además a los usuarios la manipulación de la vista de la escena con el fin de distinguir los detalles importantes de los vasos.
- 5.- Que permita capturar vistas de la escena actual.
- 6.- Que cada modelo sea lo más independiente del resto de los objetos de la escena y modificable para ciertos atributos esenciales.

En resumen, en el diagrama de la Figura 4.1 se agrupan las necesidades citadas señalando los casos de uso a satisfacer.

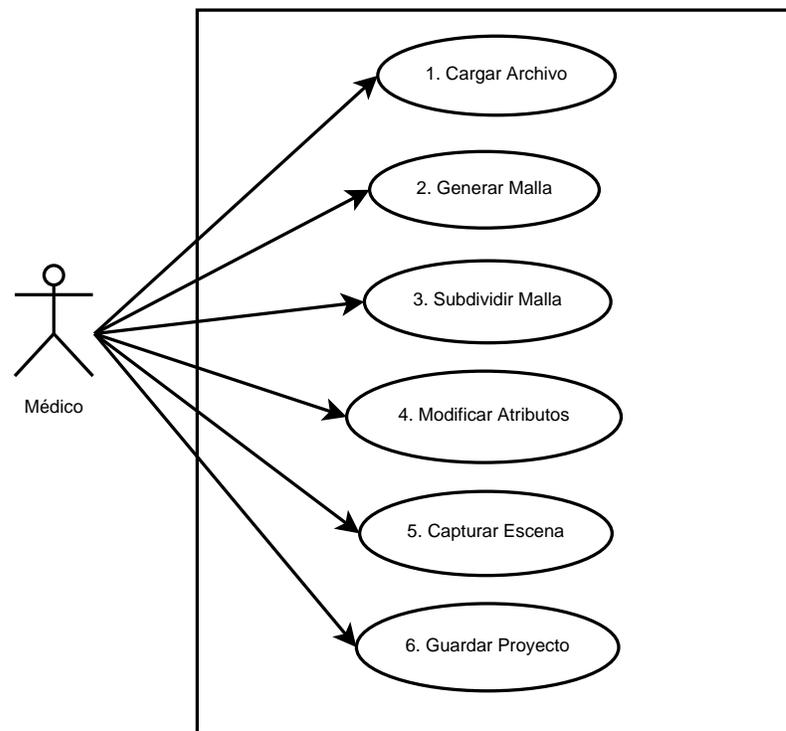


Figura 4.1: Diagrama General de Casos de Uso.

Por su parte, las mallas generadas y el modelo tridimensional completo de cada una deberán cumplir con:

- ✓ Que las mallas no sean mixtas; es decir, que todos sus elementos tengan la misma geometría.
- ✓ Sean mallas lo más regulares posibles (Sección 2.4.1).
- ✓ Que sean mallas conformes (ver Sección 2.2).

- ✓ Que no exista redundancia de vértices ni caras que las forman.
- ✓ El modelo de toda estructura debe ser *claro*, esto es, que la visualización muestre nítidamente los detalles de las superficies.
- ✓ Que tal representación de los vasos altere lo menos posible los ángulos de bifurcación, así como el grosor y la longitud de los tubos.

Nótese que algunas de éstas propiedades ya han sido cubiertas por medio de la malla *canónica* descrita en la Sección 3.2.2, ya que la simplicidad de su estructura se adapta adecuadamente a ellas.

## 4.2. Diseño de la aplicación

Para permitir un mejor manejo del programa en cuestión se planeó la construcción de una interfaz gráfica desarrollada en *Qt*, versión 4.6.2.0 (2010.02.01), usando el paradigma orientado a objetos en el lenguaje *C++*, además de las bibliotecas gráficas de *OpenGL* correspondientes, incluidas en la versión *Qt* 4.6.2.0.

La razón de emplear esta versión es que la aplicación sea un programa de código libre que pueda ser instalado en cualquier máquina que lo soporte, con fines médicos o de desarrollo posterior; asimismo, no se codificará haciendo uso de tecnología *GLSL* por medio de *shaders*, debido a que se desconoce aún las limitaciones de hardware de los equipos finales que lo utilicen, principalmente aquellos destinados al análisis médico.

### 4.2.1. Arquitectura

La arquitectura más apropiada será el patrón *Modelo Vista Controlador* debido a la fácil integración de los módulos en el ambiente gráfico de *Qt* y la finalidad de interacción buscada entre el usuario y el sistema.

Los componentes de dicho patrón se considerarán de la siguiente manera:

- *Modelo*. Incluirá dos conjuntos: el primero corresponde a los datos de entrada que definen los *puntos de control* de cada segmento tubular; el segundo comprende los vértices generados con el primer modelo, que conforman la malla de la estructura tubular en turno.

Asimismo, podría considerarse la definición de las caras poligonales de las superficies; sin embargo, se asumirá que los vértices pueden ser generados de una forma ordenada y luego referenciarlos, por lo que no es necesario el uso de memoria para almacenar la información de dichos polígonos.

En resumen, según el modelo utilizado, se tienen 3 formas de visualizar los vasos:

- El esqueleto central de los tubos representado por los vectores directores de cada segmento y los vectores formados por el segmento padre y los segmentos hijos en cada bifurcación.
  - La malla *canónica* obtenida mediante cilindros generalizados y los ajustes en las zonas de bifurcación.
  - La malla conseguida por subdivisión aplicando el algoritmo de Catmull-Clark sobre la malla *canónica*.
- *Vista*. Este componente corresponde propiamente a los elementos gráficos (*wid-gets*) de la interfaz gráfica, el *panel de control*, y al módulo encargado de la visualización tridimensional de las superficies y la escena completa, que es también parte de la interfaz.
  - *Controlador*. Comprenderá la implementación de las funciones que resuelvan las solicitudes que haga el usuario al sistema desde los elementos de la interfaz, el teclado o el mouse; sobre todo tomando en cuenta el tipo de modelo representado en la vista. Así, el controlador tendrá que programarse para cambiar en tiempo de ejecución; por ejemplo, no tendría mucho sentido permitir la subdivisión de las mallas si se está visualizando el modelo usando sólo los *puntos de control*.

#### 4.2.2. Patrones del diseño

Con el fin de cubrir la implementación de cada componente anterior, los patrones parcialmente utilizados para el diseño de clases serán el *Método de Plantilla* (*Template Method*), *Fábrica Abstracta* (*Abstract Factory*) y *Estrategia* (*Strategy Method*) [Gamma *et al.*, 1998].

Ahora bien, específicamente el método de *Fábrica Abstracta* y el *Método de Plantilla* serán empleados para definir las clases del controlador con el objetivo de que sea extensible e integre otras ventanas relacionadas con la captura y segmentación de las imágenes de fondo de ojo, o en su caso, una posterior simulación de flujo sanguíneo con las mallas obtenidas hasta ahora. Inicialmente, la *Fábrica Abstracta* permitirá definir y generar componentes gráficos básicos, además de establecer la jerarquía de tales clases; de este modo, el *Método de Plantilla* delegará la implementación de las funciones comunes a las subclases.

Por otra parte, el *Método de Estrategia* es más bien una variación del patrón, ya que consistirá en separar en módulos independientes los 4 estados esenciales en el

flujo del programa: la lectura de los archivos de datos o mallas y el uso de una estructura de árbol binario para almacenar la información de los segmentos, la generación de la malla de cada árbol, la aplicación del refinamiento por subdivisión y por último, las clases para el renderizado.

### 4.2.3. Estructura de clases

Usando los patrones anteriores, en la Figura 4.2 se muestra el diagrama de clases resultante. Debe ser claro que, en su mayoría, las clases del *Controlador* no son *interfaces* ni clases *abstractas*, ya que son definiciones precisas para esta sección de la aplicación encargada de visualizar los modelos.

Como notas importantes, la clase del *widget* principal, *MainWindow*, contempla cada llamada a las funciones que modifican la escena donde se despliega el render de los modelos, además contiene los disparadores a funciones comunes de ventanas internas que la aplicación puede tener.

En lo que respecta a los módulos de la clase *SBT* (**B**inary **T**ree of **S**egments), todas las funciones internas son recursivas, de modo que es difícil simplificar el código, sobre todo la subdivisión; en cambio, utilizar otra estructura de datos no proporcionaría el beneficio de aplicar automáticamente el mismo algoritmo a cada nodo.

Finalmente, la jerarquía de clases *DataProcessor* corresponde a los objetos encargados de realizar la lectura y escritura de los archivos de datos, de acuerdo a alguno de los formatos que serán permitidos, con el fin de proporcionar flexibilidad en el uso e intercambio de información.

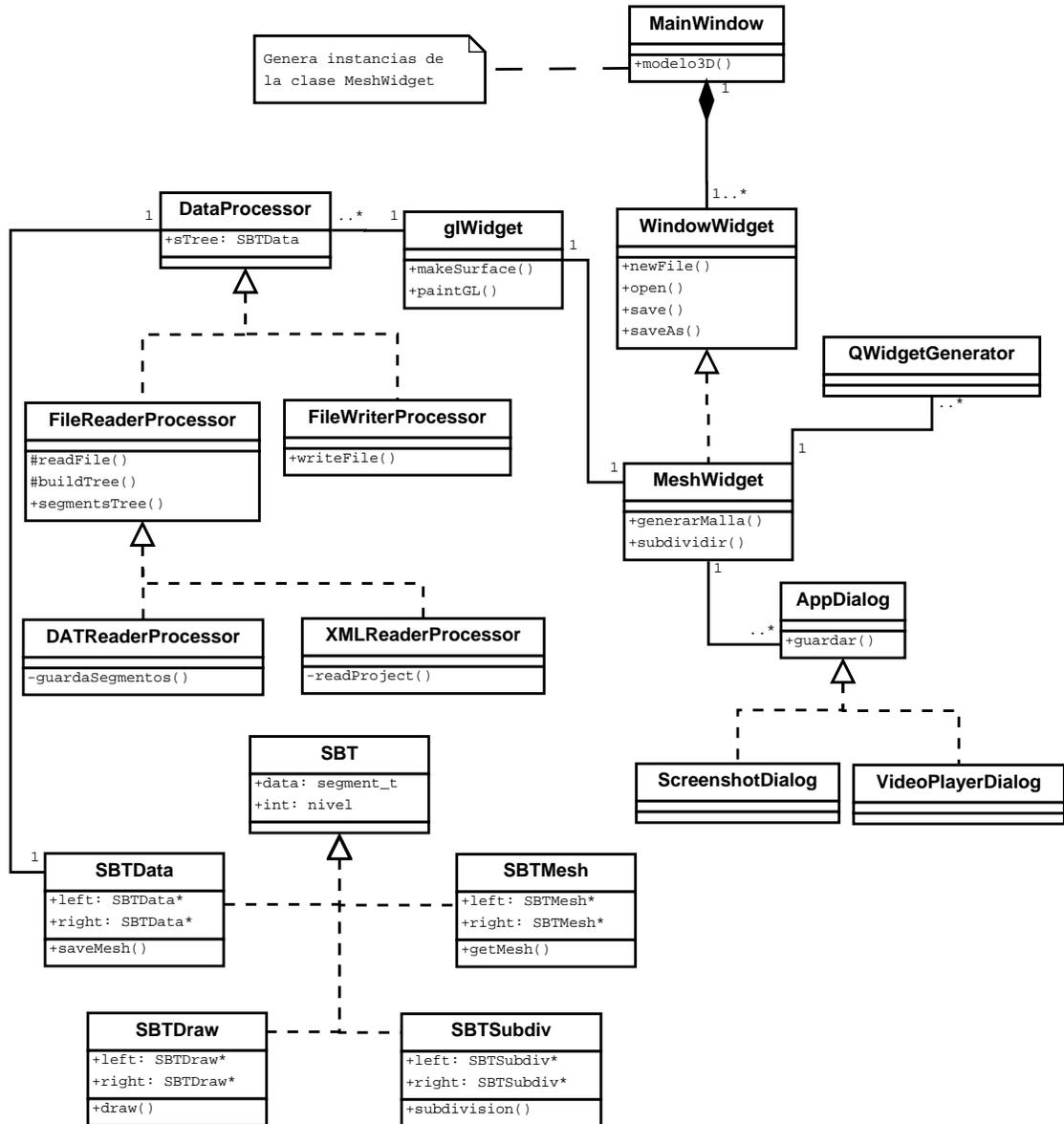


Figura 4.2: Diagrama general del diseño de clases.

#### 4.2.4. Diagramas de secuencia

A continuación, en las Figuras 4.3 a 4.11 se incluyen las secuencias de interacción del usuario con el sistema:

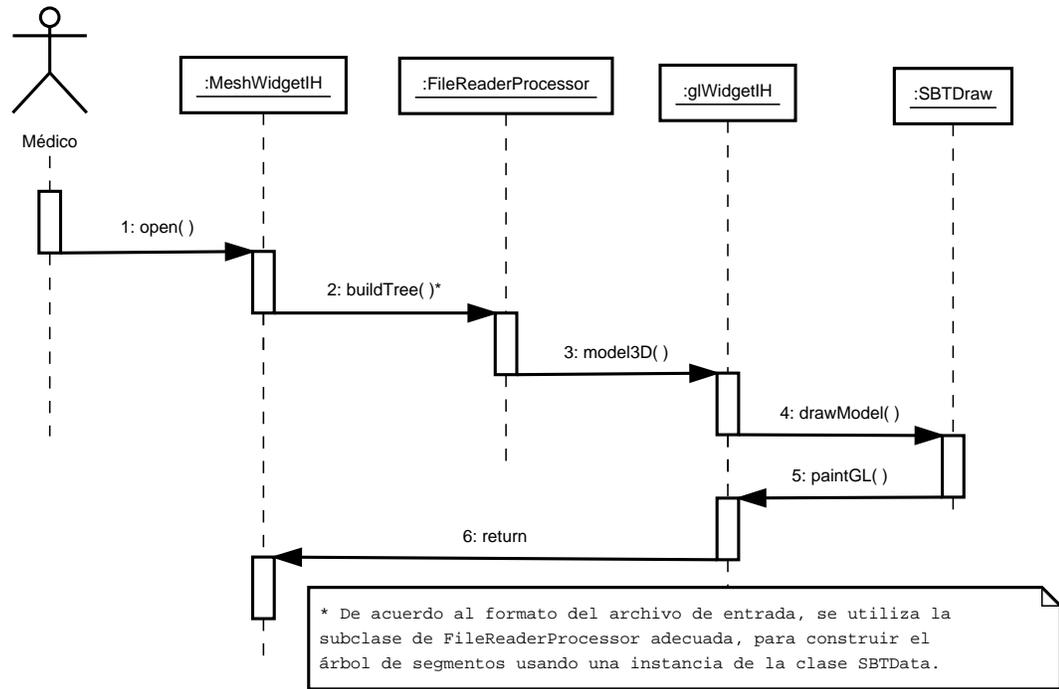


Figura 4.3: Caso de Uso.- *Cargar Archivo*.

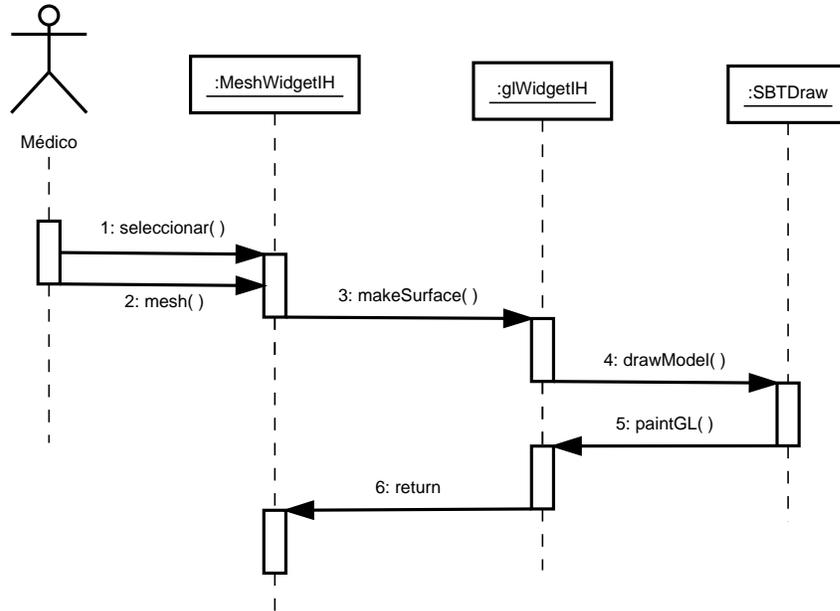


Figura 4.4: Caso de Uso.- *Generar Malla*.

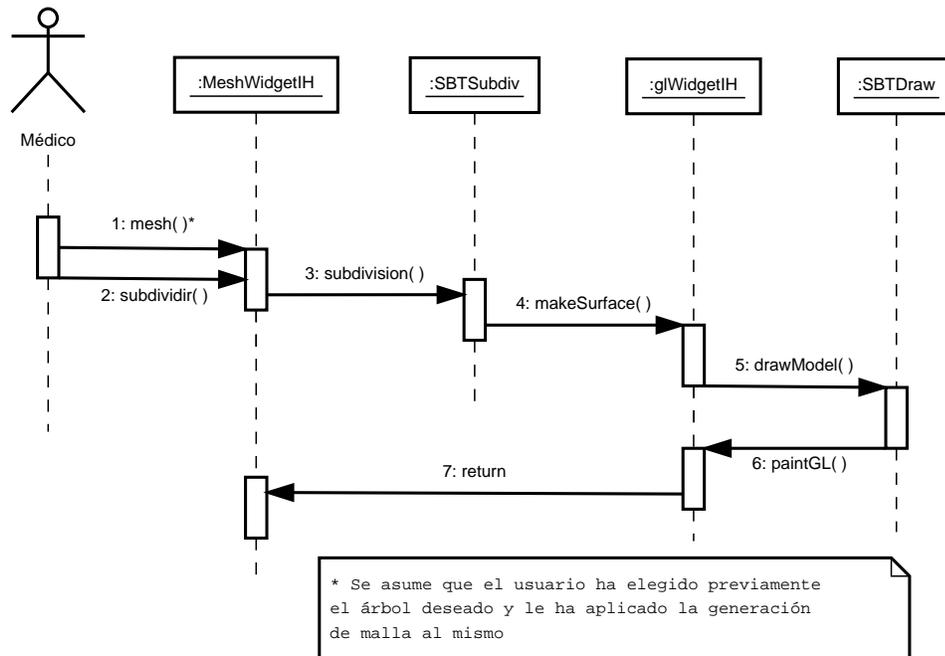


Figura 4.5: Caso de Uso.- *Subdividir Malla*.

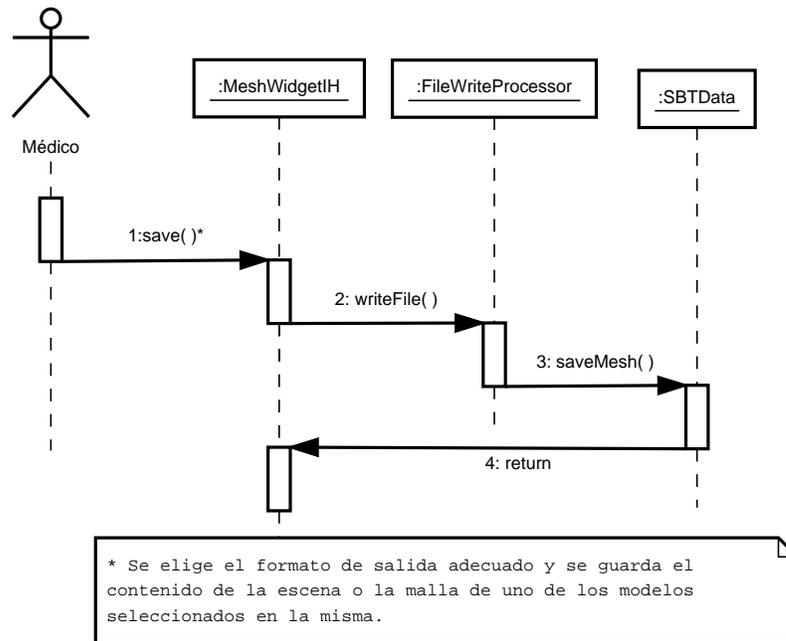


Figura 4.6: Caso de Uso.- *Guardar Proyecto*.

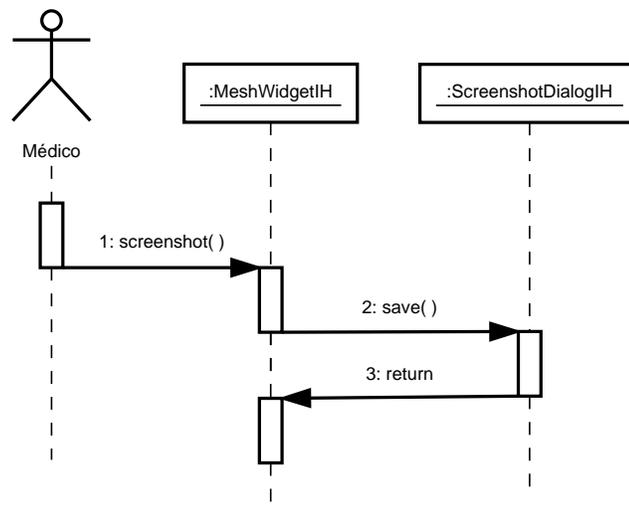


Figura 4.7: Caso de Uso.- *Capturar Escena (Pantalla)*.

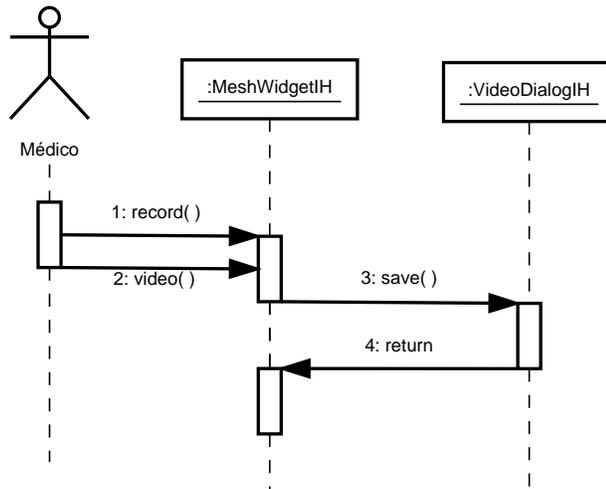


Figura 4.8: Caso de Uso.- *Capturar Escena (Video)*.

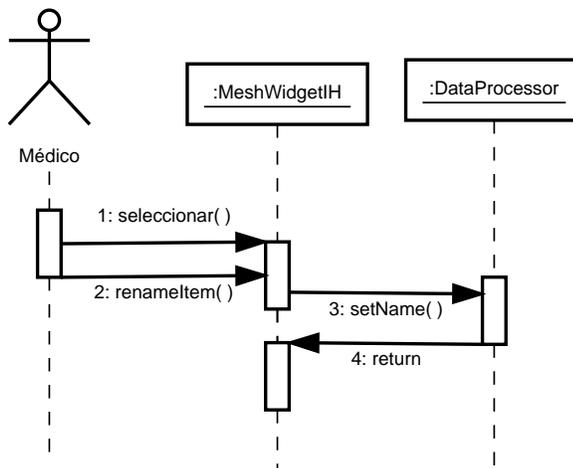


Figura 4.9: Caso de Uso.- *Modificar Atributos del Modelo (Nombre)*.

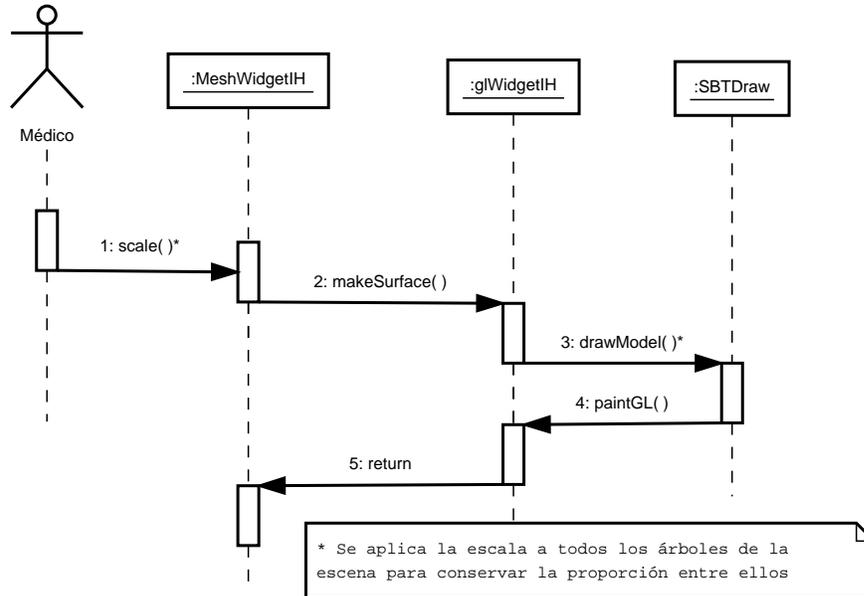


Figura 4.10: Caso de Uso.- *Modificar Atributos del Modelo (Escala).*

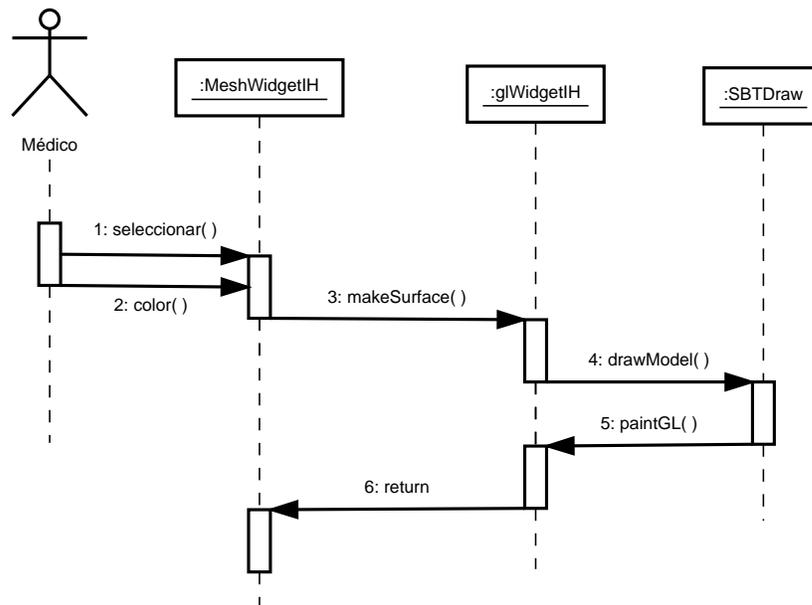


Figura 4.11: Caso de Uso.- *Modificar Atributos del Modelo (Color de material).*

Finalmente, el flujo de control del programa completo aparece en la Figura 4.12.

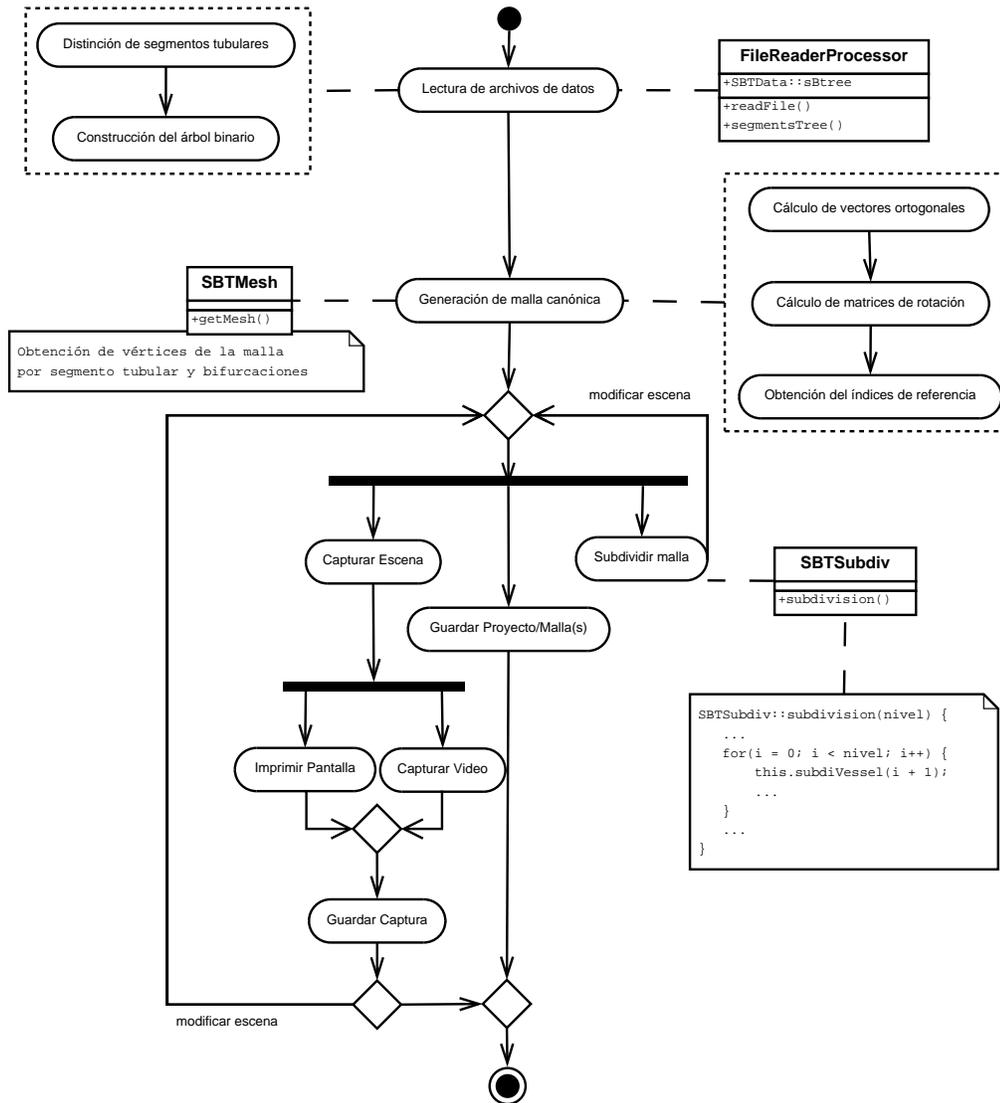


Figura 4.12: Flujo de control de actividades realizadas mediante la aplicación.

## Capítulo 5

# Experimentación y Resultados

Con base en los algoritmos propuestos en la metodología del Capítulo 3 y las especificaciones señaladas en el Capítulo 4, se llevaron a cabo las pruebas experimentales utilizando la aplicación desarrollada, cuyos resultados serán mostrados en las próximas secciones.

Las características del equipo de cómputo en el cual se realizaron las pruebas se describen a continuación:

- Dell Inspiron 1300
- Procesador Intel® Celeron® M (1.50 GHz/1MB Cache/400MHz FSB)
- 512 MB de memoria RAM
- Microsoft Windows XP Profesional, Versión 2002; Service Pack 3
- Tarjeta gráfica integrada Intel® Media Accelerator 900

### 5.1. Interfaz de Usuario

Dado que la interfaz gráfica constituye una parte importante del programa, además de ayudar a llevar a cabo la experimentación usando los datos de entrada, se requiere indicar cómo se interactúa con la aplicación y cómo se generan las mallas correspondientes, pues todo esto conforma el primer grupo de resultados.

Una vez que se haya iniciado la ejecución del programa, los pasos necesarios se resumen en la siguiente secuencia de acciones:

- a) Se oprime el botón o la opción del menú que corresponda a la herramienta para visualizar los modelos de las superficies tubulares (Figura 5.1(a) y 5.1(b)). Como resultado aparece una nueva ventana interna con dos apartados: el área de la escena y el panel de control (Figura 5.1(c)).

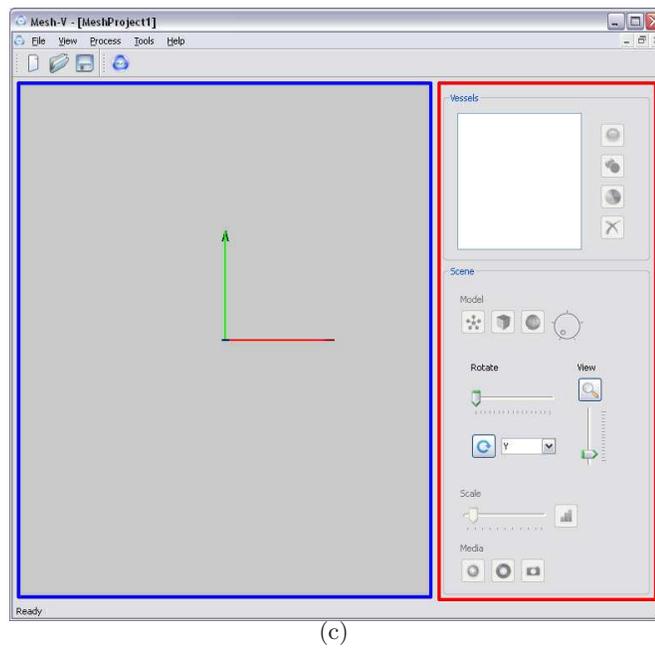
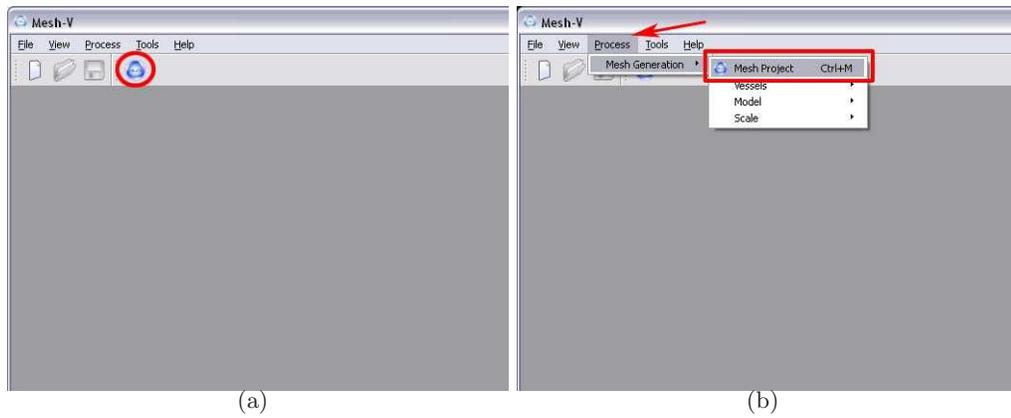


Figura 5.1: Iniciando el programa. (a) Arranque del sistema desde el botón en la barra de herramientas y (b) selección desde el menú principal. (c) Ventana donde se visualizan los modelos de las mallas tubulares.

- b) Además, es posible cambiar la vista de la escena en todo momento que se requiera, modificando la posición de la cámara, haciendo uso del ratón, el teclado o los controles del panel, según el ajuste buscado (Figura 5.2).

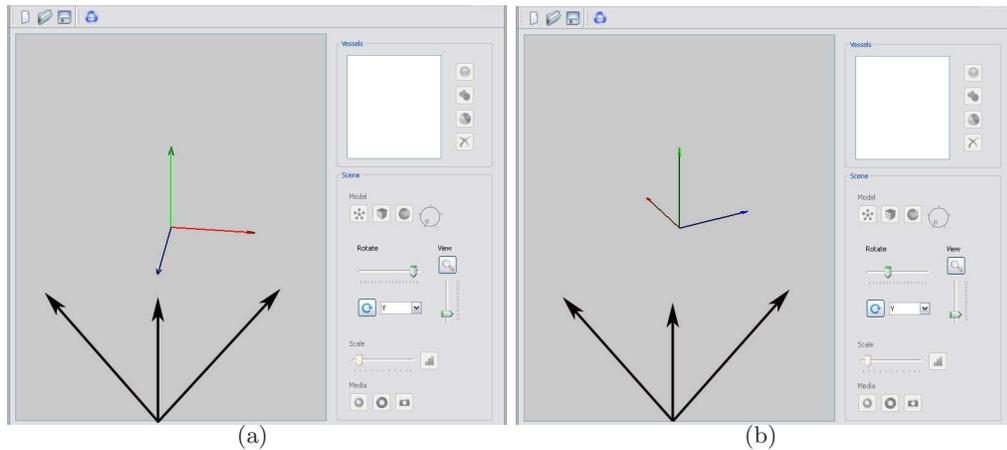


Figura 5.2: Cambio de la posición de la cámara para mejorar la vista de la escena. (a) La escena vacía con un ajuste sencillo de cámara y (b) la misma escena observada desde otro ángulo. Las flechas en negro indican una nueva orientación.

- c) Se procede a agregar uno o varios modelos a la escena, presionando la opción de la barra de herramientas, para cargar los datos seleccionando los archivos deseados en el formato correcto (Figura 5.3(a)). Se mostrará en el área de la escena el árbol de segmentos de cada modelo usando el esqueleto central que forman los puntos de los datos (Figura 5.3(b)).

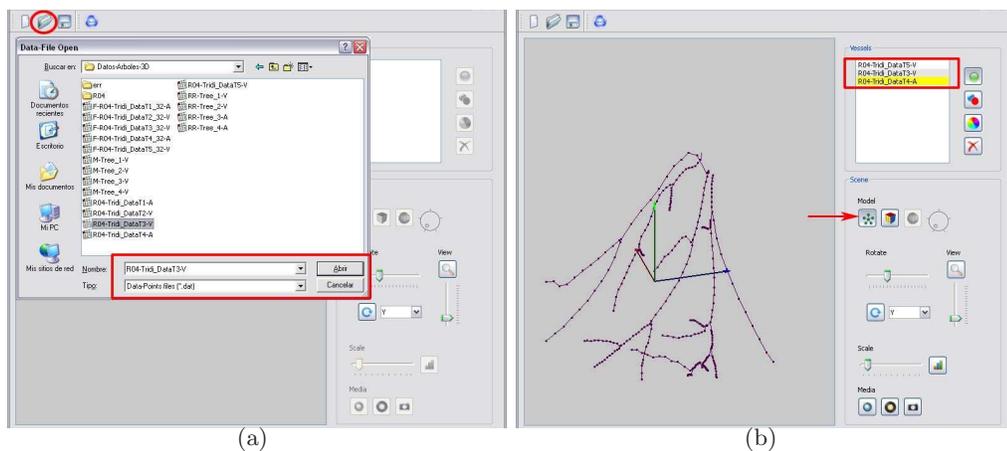


Figura 5.3: Despliegue en pantalla de los modelos agregados a la escena. (a) Selección de los archivos de datos a utilizar y (b) visualización de los árboles de segmentos correspondientes.

- d) En seguida, se aplica la generación de la(s) malla(s); inicialmente, la malla *canónica* de cada modelo agregado; en otro caso, la malla pudo haber sido refinada y será ésta la que aparezca en pantalla (Figura 5.4(a)).
- e) Si, en efecto, son varios modelos y se desactiva la opción que modifica a todos ellos a la vez, deberá seleccionarse de la lista que aparece en la parte superior del panel de control el árbol deseado, para aplicar los cambios sólo a éste (Figura 5.4(b)).

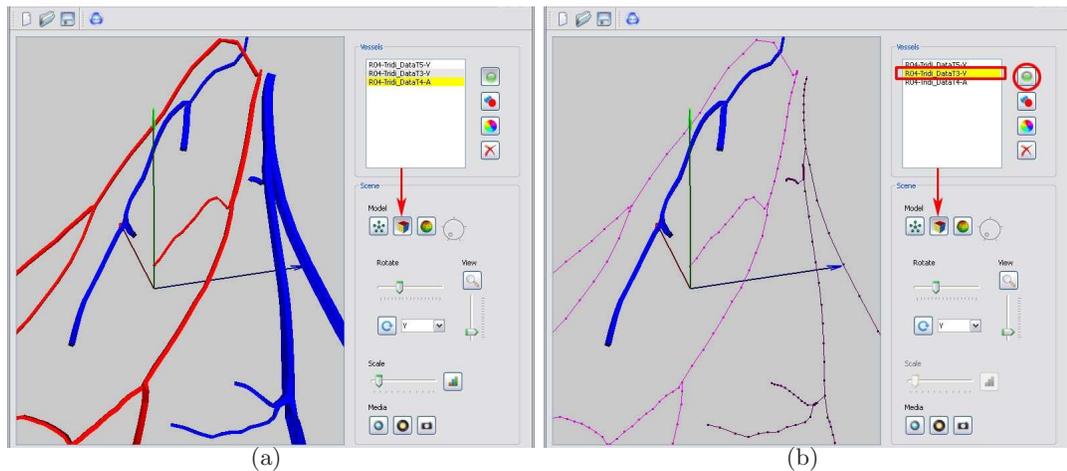
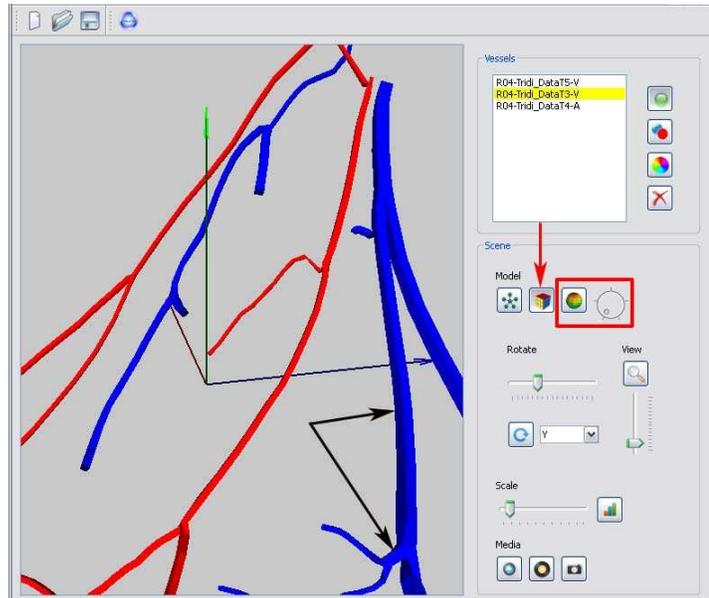
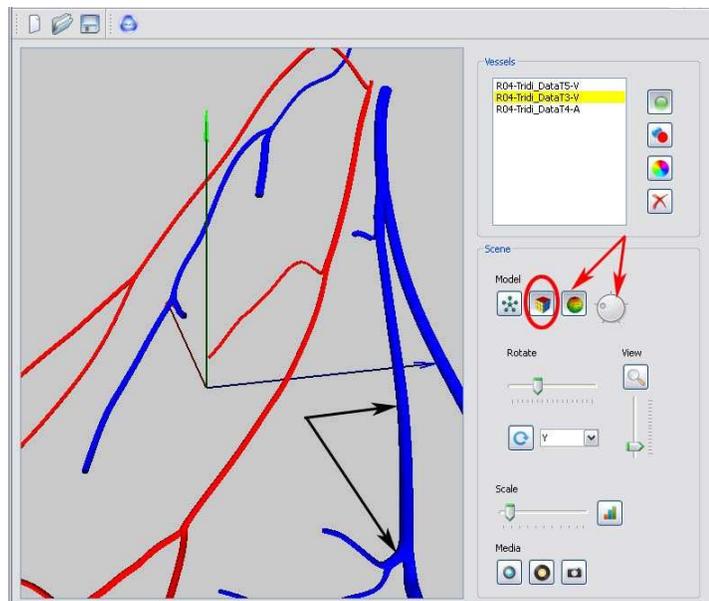


Figura 5.4: Generación de la malla *canónica* según el(los) vaso(s) seleccionado(s). (a) La malla se aplica a todos los modelos de la escena. (b) Un sólo modelo de malla; se desactivan y activan las opciones necesarias.

- f) Sea cuál sea la decisión tomada en el paso anterior y siempre que se haya generado el modelo de malla, puede aplicarse uno o varios refinamientos por subdivisión habilitando la opción correspondiente. Si tal elección se deshabilita, se observará de nuevo la malla *canónica* del árbol en turno, o de todos, según sea el caso (Figura 5.5).
- g) Asimismo, estrictamente seleccionando uno de los árboles de la lista del panel de control, puede modificarse el nombre que mantiene en esa lista o bien el color del material de la superficie de la malla respectiva (Figura 5.6).
- h) Alternativamente, se puede capturar la vista actual de la escena, guardándola en una imagen (Figura 5.7(a)) o capturando algunos *cuadros* de video.
- i) Por último, puede(n) guardarse la(s) malla(s) generada(s) para su uso posterior en el sistema, de acuerdo a los criterios elegidos en los pasos e) y f) (Figura 5.7(b)).



(a)



(b)

Figura 5.5: Aplicando refinamiento por subdivisión. (a) Escena con varios modelos únicamente con malla *canónica* y (b) los mismos modelos con dos niveles de refinamiento. Las flechas en negro enfatizan el suavizado logrado.

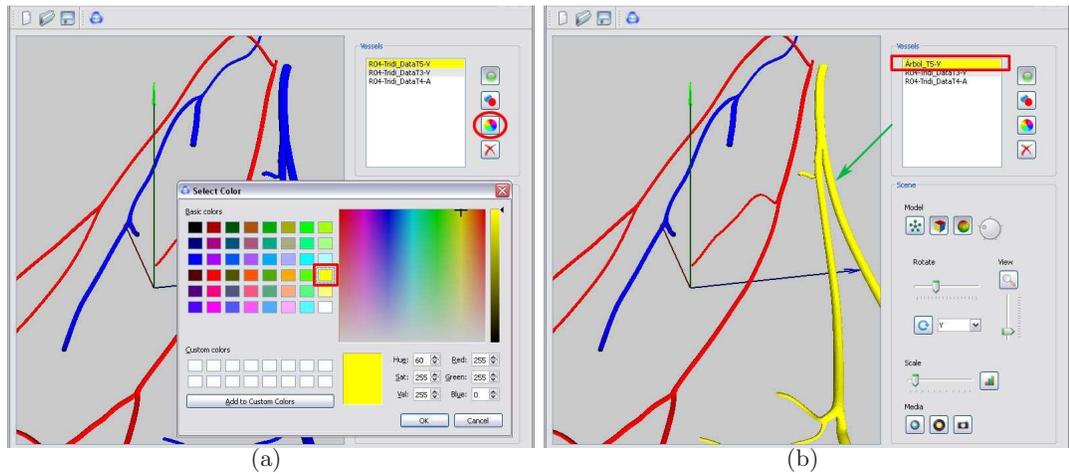


Figura 5.6: Modificación de los atributos de los modelos. (a) Selección de un árbol de la escena actual para modificar su color de material. (b) Resultado de cambiar el color y el nombre de la etiqueta asociada al árbol.

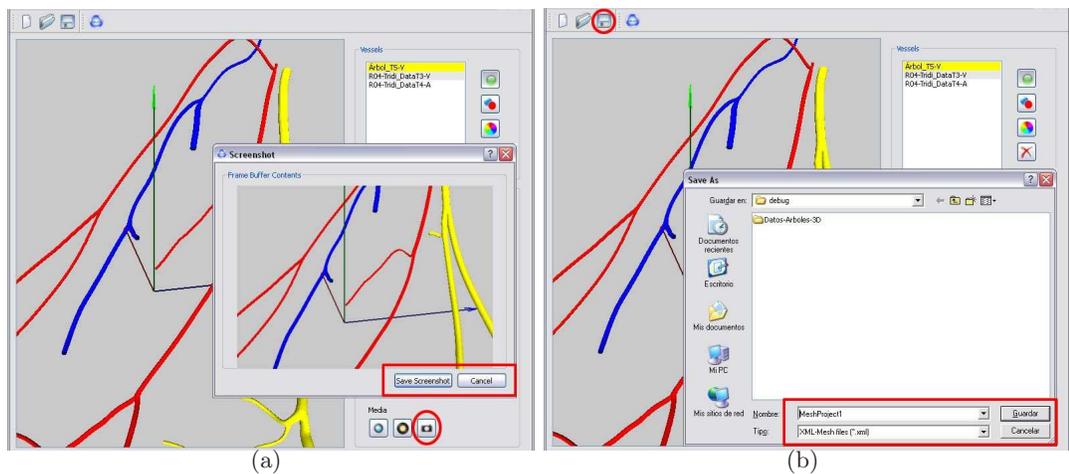


Figura 5.7: (a) Captura del contenido de la escena como imagen y (b) guardado de la(s) malla(s) correspondiente(s) a un archivo en disco.

## 5.2. Visualización de las mallas

En esta sección se muestran algunas imágenes de las mallas generadas, sobretodo de las regiones de bifurcación. Para realizar esta experimentación se cuentan con 3 conjuntos de datos, cada uno con 4 vasos sanguíneos del mismo sujeto.

A continuación, se anotan las especificaciones de cada conjunto y se incluyen las imágenes asociadas que fueron segmentadas con RISA [Martinez-Perez *et al.*, 2007], con las cuáles se obtuvieron los datos.

### ※ Experimento 1.

Los datos de los vasos sanguíneos de este conjunto fueron extraídos con RISA y reconstruidos en 3D [Martinez-Perez y Espinosa-Romero, 2004] utilizando las siguientes imágenes muestra:

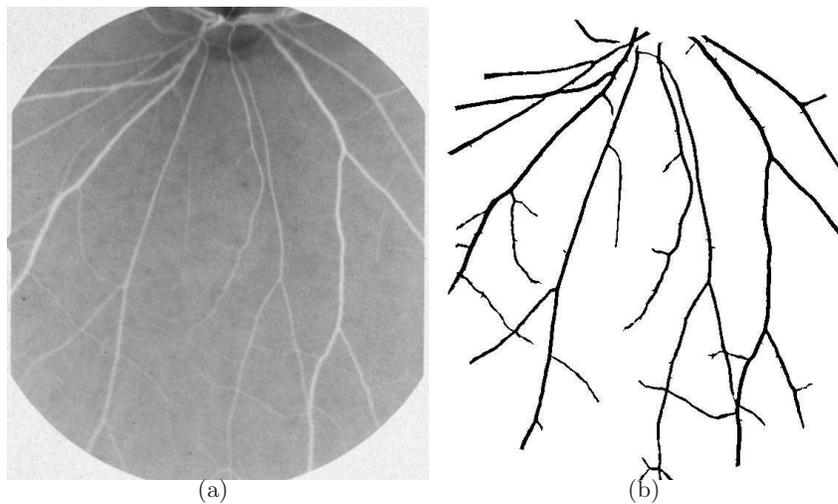


Figura 5.8: Imágenes asociadas al primer conjunto de datos. (a) La imagen de *fondo de ojo* procesada y (b) el resultado de la segmentación.

En la Figura 5.9 aparece la malla correspondiente a cada uno de los vasos sanguíneos extraídos. Asimismo, la Figura 5.10 muestra algunas de las bifurcaciones pertenecientes a cada vaso.

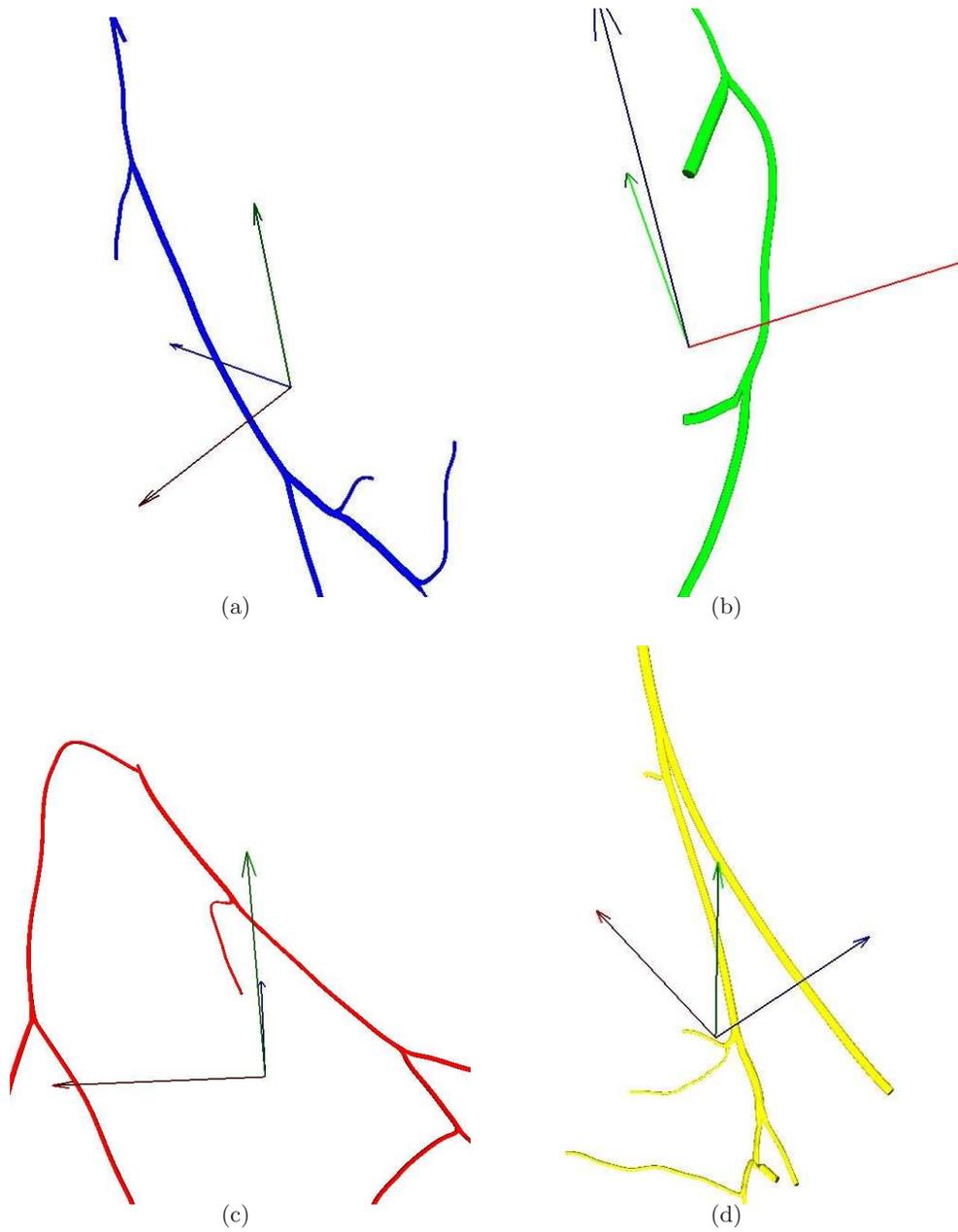


Figura 5.9: Modelos en tercera dimensión de las estructuras tubulares, obtenidas usando el conjunto de datos procesado. (a) - (d) Árboles del T1 al T4, primer experimento.

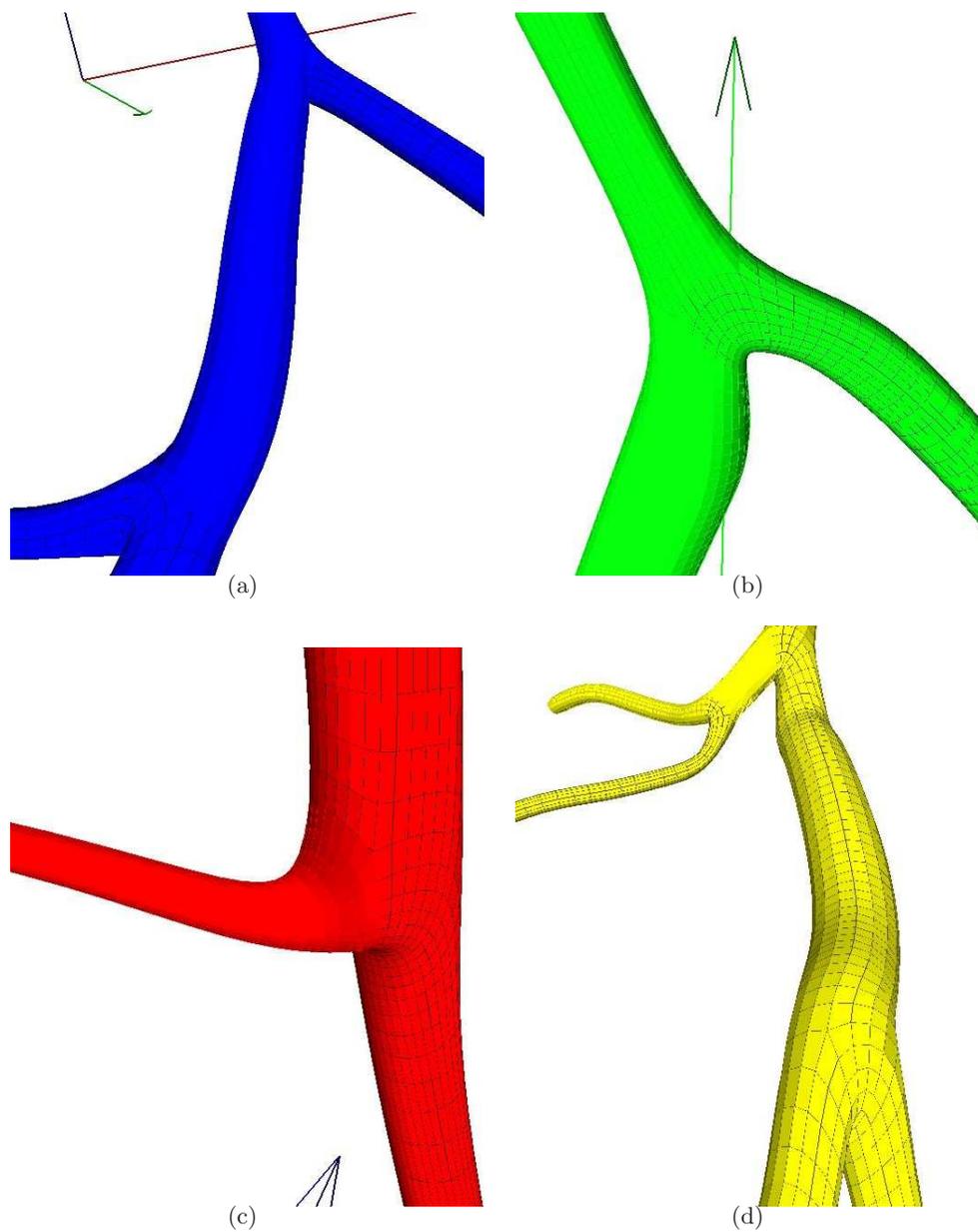


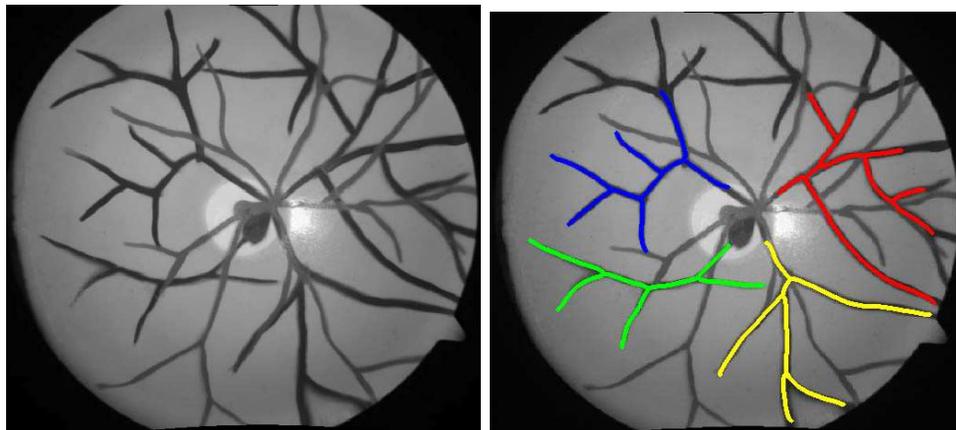
Figura 5.10: (a) - (d) Acercamientos a algunas bifurcaciones de los vasos sanguíneos T1 al T4 del primer conjunto.

※ **Experimento 2.**

Este conjunto se obtuvo de la segmentación y reconstrucción correspondientes tomando como sujeto un maniquí de ojo (Figura 5.11(a)), con el objetivo de trabajar en la parte de calibración del sistema de reconstrucción [Aldana Iuit, 2010].



(a)



(b)

(c)

Figura 5.11: Imágenes muestra del segundo sujeto. (a) El modelo de maniquí de ojo. (b) Imagen de *fondo de ojo* obtenida una vez calibrado el sistema. (c) La segmentación de los vasos resuelta por RISA.

De manera análoga al experimento anterior, las Figuras 5.12 y 5.13 ilustran, respectivamente, los modelos resultantes y bifurcaciones de los mismos.

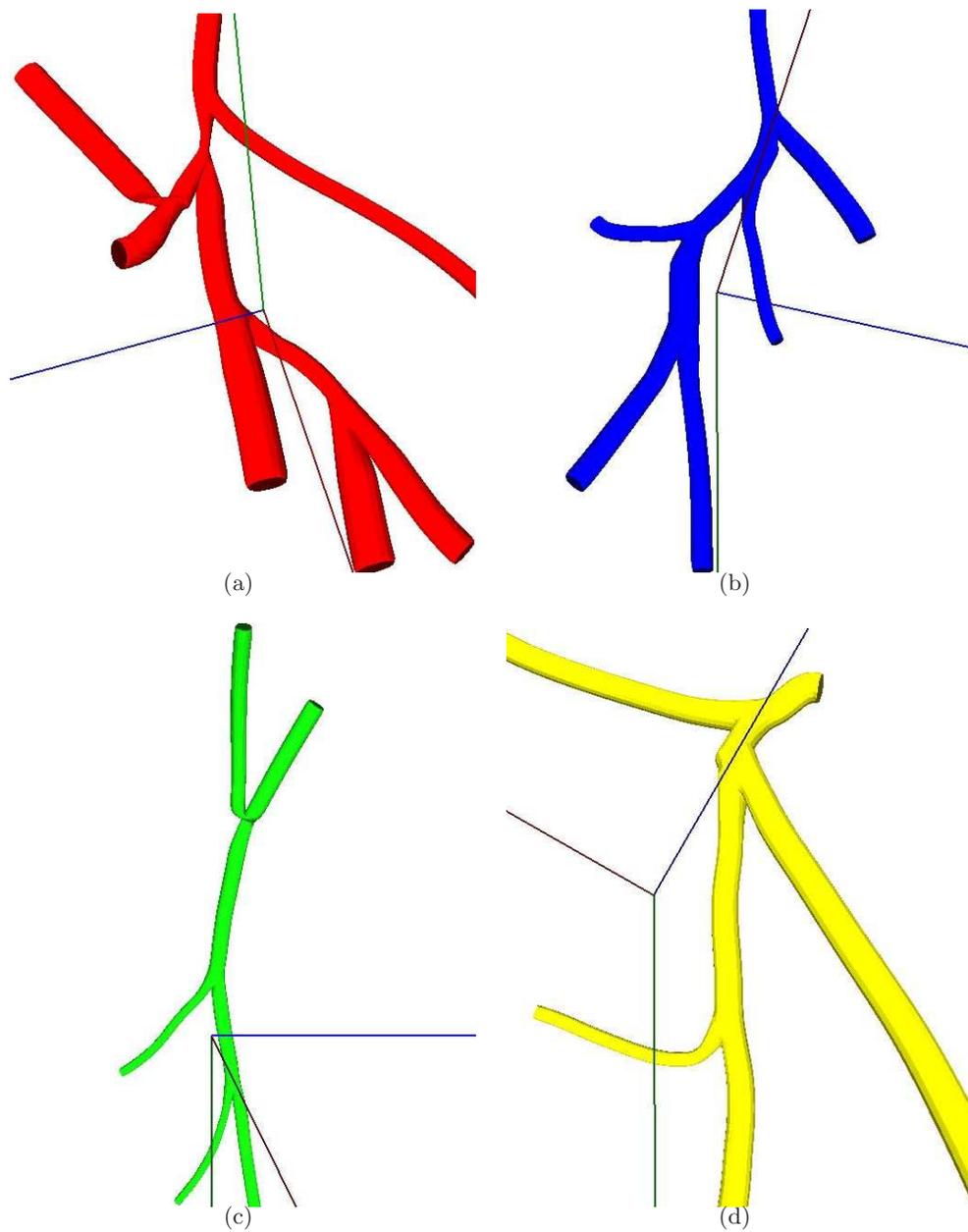


Figura 5.12: Representación de las estructuras tubulares de conjunto de datos del maniquí de ojo. (a) - (d) Árboles del T1 al T4, segundo experimento.

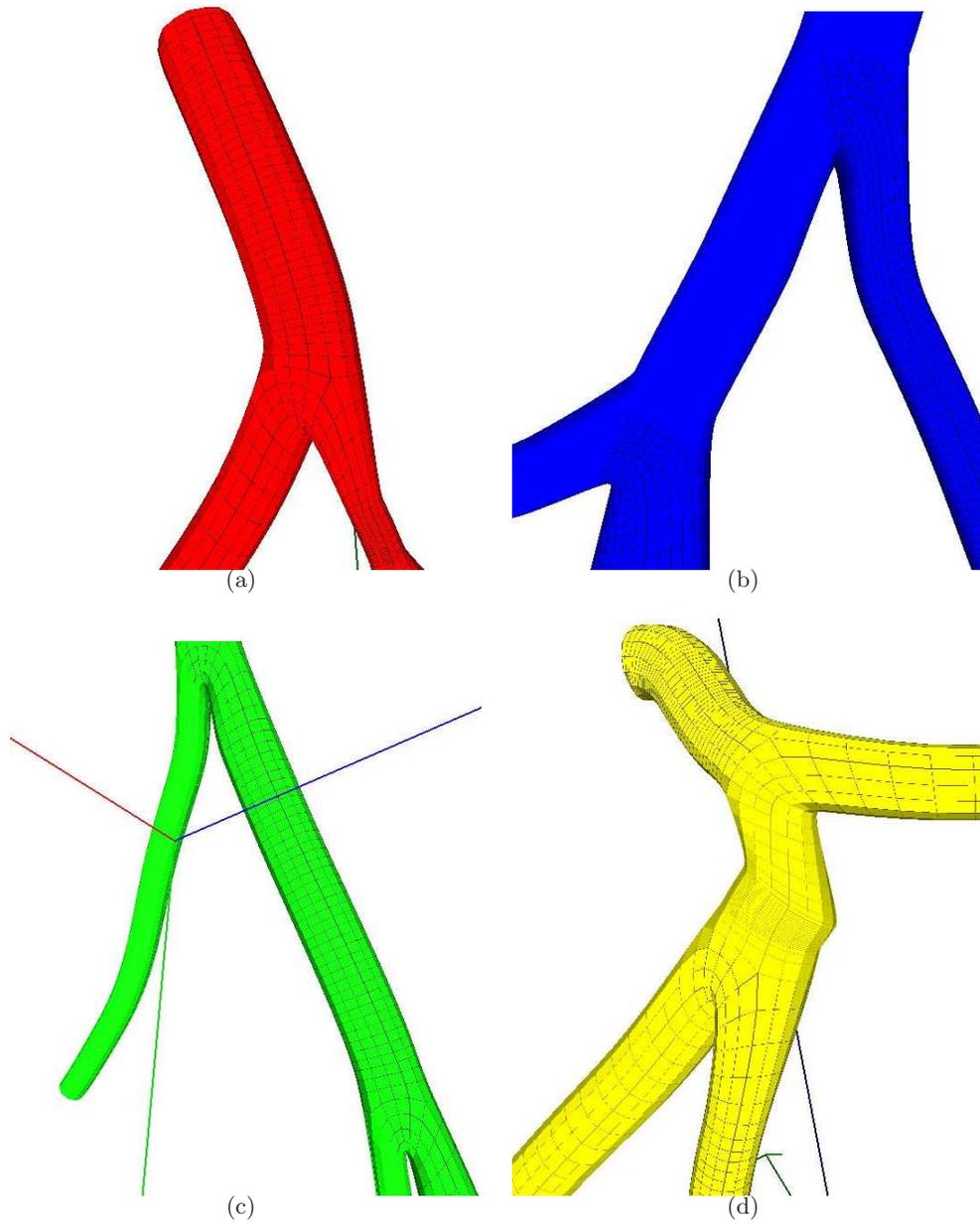


Figura 5.13: (a) - (d) Acercamiento a bifurcaciones resueltas en los modelos de los árboles T1 al T4 del segundo conjunto de datos.

※ **Experimento 3.**

De igual manera, el tercer conjunto de datos fue procesado como uno de los resultados en la calibración del sistema de reconstrucción [Aldana Iuit, 2010], esta vez sobre una imagen de retina real (Figura 5.14). Los resultados de este experimento se presentan en las Figuras 5.15 y 5.16.

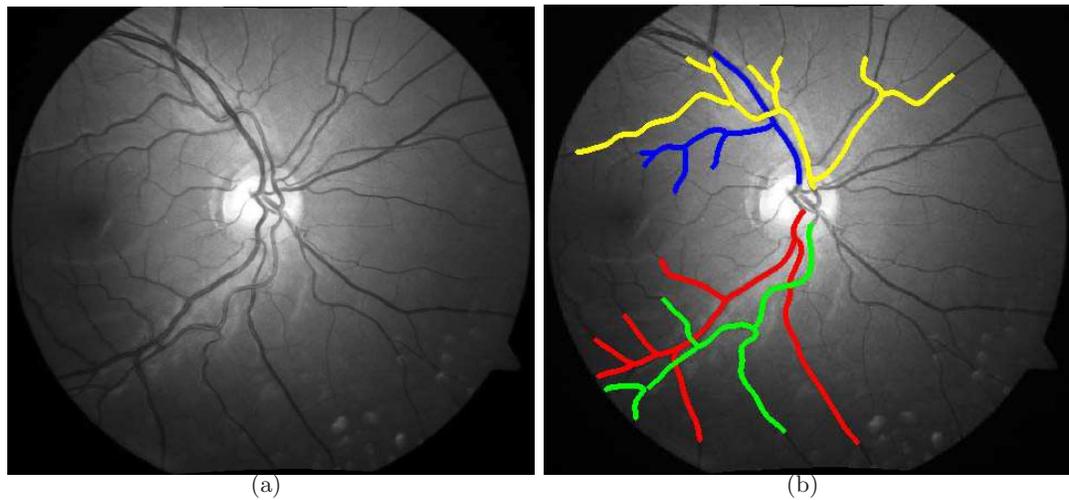


Figura 5.14: Imágenes muestra del tercer sujeto. (a) Imagen de una retina real. (b) El resultado de la segmentación correspondiente.

Por último, en la Figura 5.17 se muestran varias bifurcaciones y segmentos, de distintos árboles de los 3 conjuntos, que no fueron resueltos adecuadamente; el análisis de estos resultados se discute en la Sección 5.3.

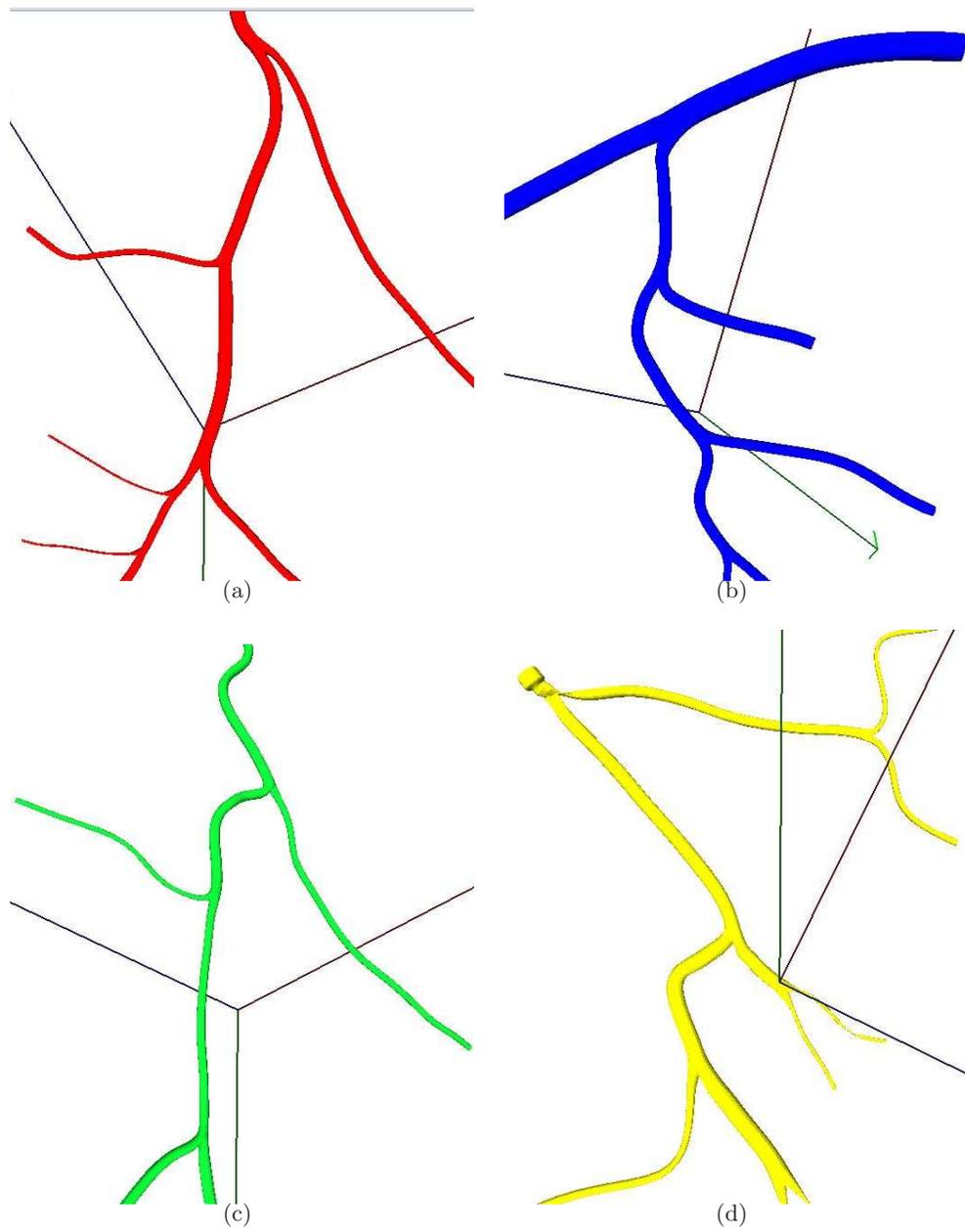


Figura 5.15: Visualización de las mallas de los vasos segmentados de la imagen de retina real. (a) - (d) Árboles T1 a T4 del experimento correspondiente.

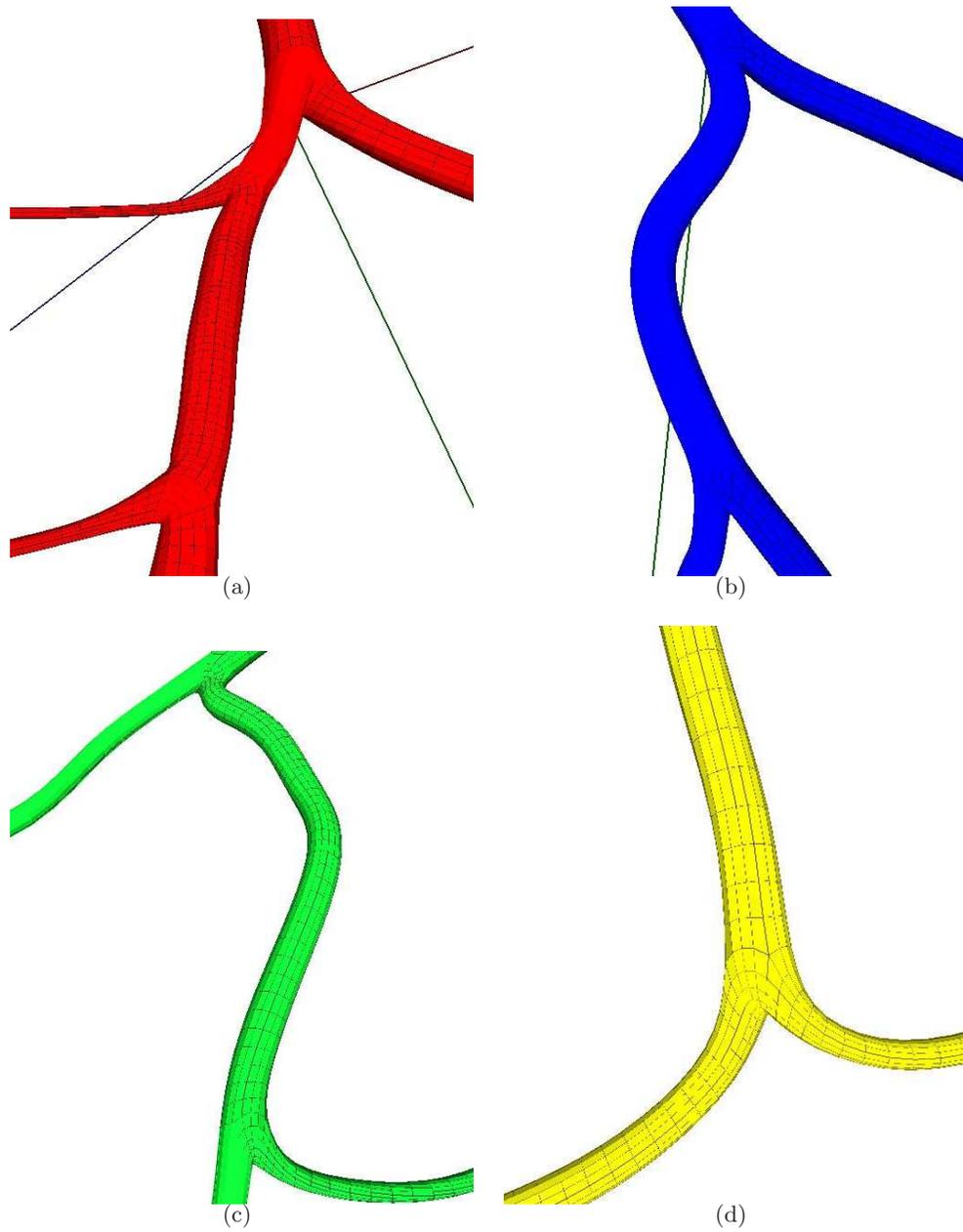


Figura 5.16: (a) - (d) Acercamiento a algunas de las bifurcaciones resueltas apropiadamente con el tercer conjunto de datos.

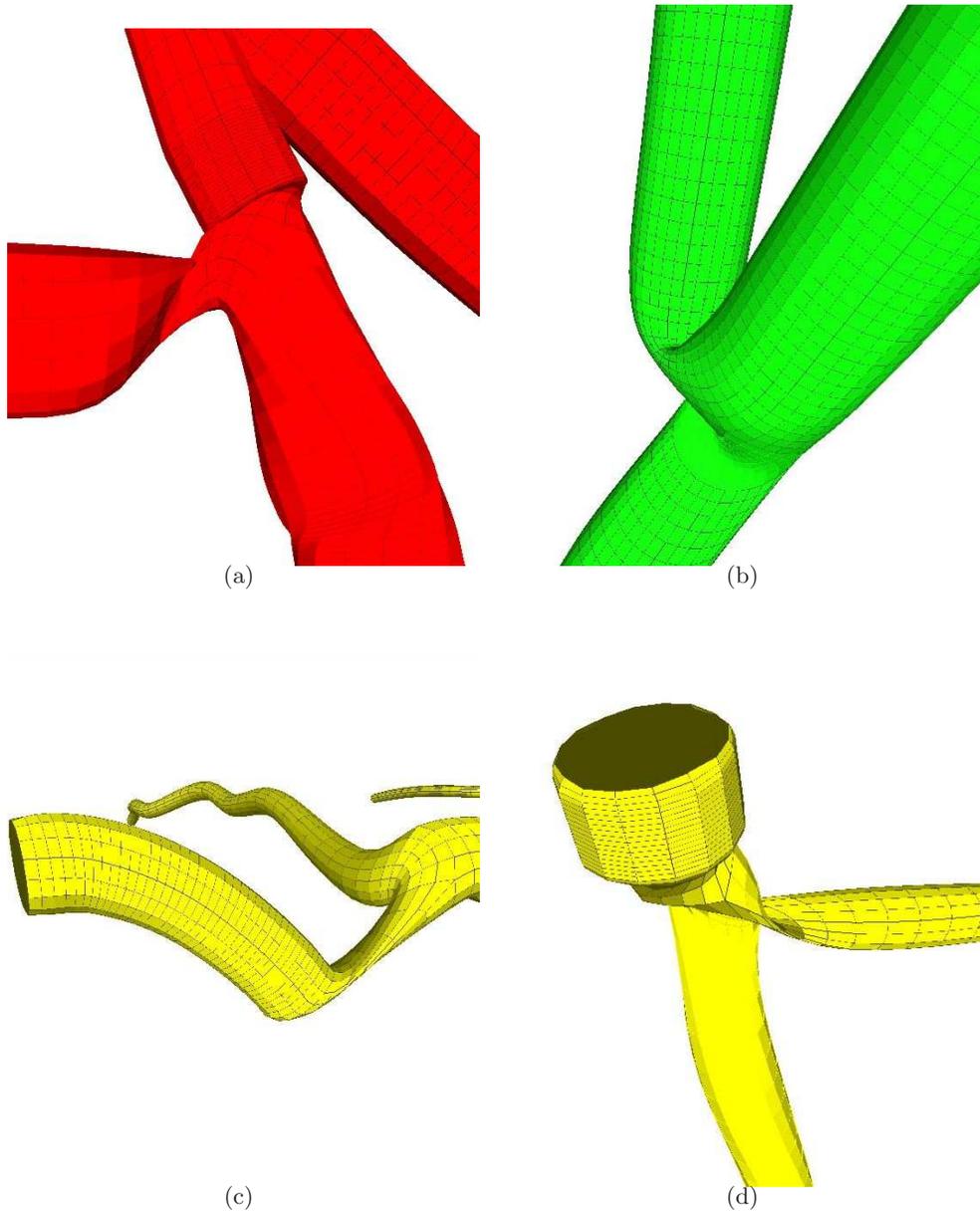


Figura 5.17: Segmentos y bifurcaciones que presentan errores en la generación de la malla. (a) y (b) construcción forzada de la malla en las bifurcaciones respectivas. (b) Reducción del volumen en una sección de un segmento. (d) Generación incorrecta de la malla y reducción de volumen en una bifurcación.

### 5.3. Evaluación y Análisis

Como se vió en la Sección 5.2, la visualización de las mallas tubulares lograda es buena, en relación con el número de bifurcaciones resueltas en las que la transición entre cilindros padre e hijos sí es suave, a excepción de algunas bifurcaciones y tubos como los mostrados en la Figura 5.17 donde se presentan errores.

Específicamente, la malla en la bifurcación de la Figura 5.17(a) tiene elementos muy cercanos entre sí e incluso algunos de ellos, próximos al segmento padre, no son visibles. La malla de la Figura 5.17(b) aparentemente no presenta el problema anterior; sin embargo, uno de los segmentos hijos de la bifurcación se encuentra orientado casi en la misma dirección que el segmento padre, formándose entre ellos un ángulo superior a los  $90^\circ$  y forzando la definición de la malla. En particular, la Figura 5.17(c) muestra reducción de volumen en la malla de uno de los cilindros, consecuencia del cambio en la curvatura de la superficie. Por último, la Figura 5.17(d) presenta ambos problemas, la disminución del volumen y la generación forzada de la malla.

Ahora bien, en el caso de las Figuras 5.17(a) y (d) la falla en la construcción del modelo puede estar en la orientación final que deben tener las circunferencias originales de la malla *canónica*, pues hasta ahora se hace de manera muy simple usando sólo los *índices de referencia* correspondientes (ver Sección 3.2.2); es decir, se encuentra únicamente un mejor valor donde inicie la parametrización para obtener los vértices sobre dichas circunferencias, pero con eso no se consigue el valor óptimo que defina los vértices.

Algo similar está ocurriendo en la Figura 5.17(b), considerando además que el ángulo de bifurcación tampoco ayuda. En general, el algoritmo *puntosBifurcacion\_2*, de la Sección 3.2.2 favorece las bifurcaciones que no presentan este último problema, usando siempre puntos medios entre los segmentos padre e hijos, asumiendo exactamente que éstos siguen un comportamiento de ramas descendentes en el árbol. Otra posible causa del error puede estar en que los vasos del modelo de maniquí de ojo, al cual pertenece este árbol, hayan sido trazados sin seguir una referencia clínica o ningún ejemplo de retina real ya que, en general, vasos con ésta característica excesivamente marcada, no suelen presentarse en los vasos sanguíneos reales. Tómese en cuenta también que el maniquí fue originalmente utilizado para otro fin, la calibración del *Sistema Cámara-Ojo* [Aldana Iuit, 2010] y posteriormente se hizo la segmentación.

Por otro lado, el volumen reducido del cilindro de la Figura 5.17(c) se debe más a la lectura del archivo de entrada y la selección de los puntos del esqueleto del árbol, ya que se hace un filtrado de los *puntos de control* de manera automática quitando aquellos puntos al final de los segmentos padres y al inicio de los segmentos

hijos, que estén a distancia menor que el valor del máximo radio de los 3 segmentos que forman cada bifurcación, con el fin de evitar el traslape de los cilindros que se generarían; por ello, sólo se resuelve la malla *canónica* en las zonas de bifurcación aproximando las superficies.

### 5.3.1. Dimensiones de los modelos

Entre los valores que interesaban que no fueran modificados con la representación se tenían el ancho y largo de los tubos, así como los ángulos de bifurcación. Con respecto a los últimos dos, no son modificados por el método propuesto, ya que la malla inicial generada por cilindros generalizados contempla siempre la obtención de circunferencias usando los puntos centrales del esqueleto; en todo caso, el algoritmo que podría afectar tales valores sería el filtrado previo de esos puntos que se realiza al construir el árbol de segmentos. Por su parte, completar la malla *canónica* sólo es resolverla en las regiones de bifurcación, cuyos ángulos en esa zona están considerados implícitamente según la posición original de los *puntos de control* restantes de cada rama hija de manera independiente.

En relación con el grosor de los tubos si deben efectuarse algunas mediciones ya que, como se explicó en la Sección 2.5.1, la subdivisión suele reducir el volumen de las superficies, lo que ocurre en este caso. Por tal razón, uno de los ajustes previos al procesamiento de la malla durante la construcción de la estructura de árbol de segmentos, consiste en incrementar el valor del radio de las circunferencias paramétricas de modo que  $r_{Fk_i} = \sqrt{2 * r_{k_i}^2} = \sqrt{2} * r_{k_i}$ , lo cual se traduce en incrementar el ancho o grosor de cada tubo según lo descrito en la Sección 1.2. Ésto hace que exista una gran diferencia entre el volumen de la malla *canónica* con tales radios  $r_{Fk_i}$  y la superficie de la malla que tendría los radios originales  $r_{k_i}$ ; asimismo, aunque no sean las mallas *canónicas* las que se visualicen como modelos “finales”, serán los valores  $r_{Fk_i}$  de cada uno de sus segmentos los que determinen las dimensiones en refinamientos subsecuentes.

Así, dado que un médico especialista espera ver en pantalla los modelos usando tubos; es decir, que los cortes transversales sean circunferencias y no cuadriláteros como sucede con la malla *canónica*; se necesita calcular el valor efectivo del ancho de los tubos para dicha malla inicial y para las mallas en cada iteración del refinamiento que se le aplique.

Para hacer ésta medición se tomó de cada conjunto de datos un árbol de segmentos al azar, y de ellos uno de los puntos internos  $P_{k_j}$  para algún  $j \in 1, \dots, n_k$ , por cada segmento  $k$  del esqueleto correspondiente; calculando entonces la distancia  $d_{k_l}$ , desde  $P_{k_j}$  hacia uno de los vértices que se encuentra en la circunferencia  $C_{k_j}$  de la malla con  $l$  refinamientos (Figura 5.18). Las Tablas 5.1 a 5.6 registran el ancho  $d_{k_l}$  de cada

segmento de los vasos seleccionados y su diferencia respecto a  $r_{k_i}$  en la iteración correspondiente, así como los radios y diferencias promedio  $\bar{x}$ , su desviación estándar  $\sigma$  y el porcentaje de error relativo promedio comparado con los radios reales.

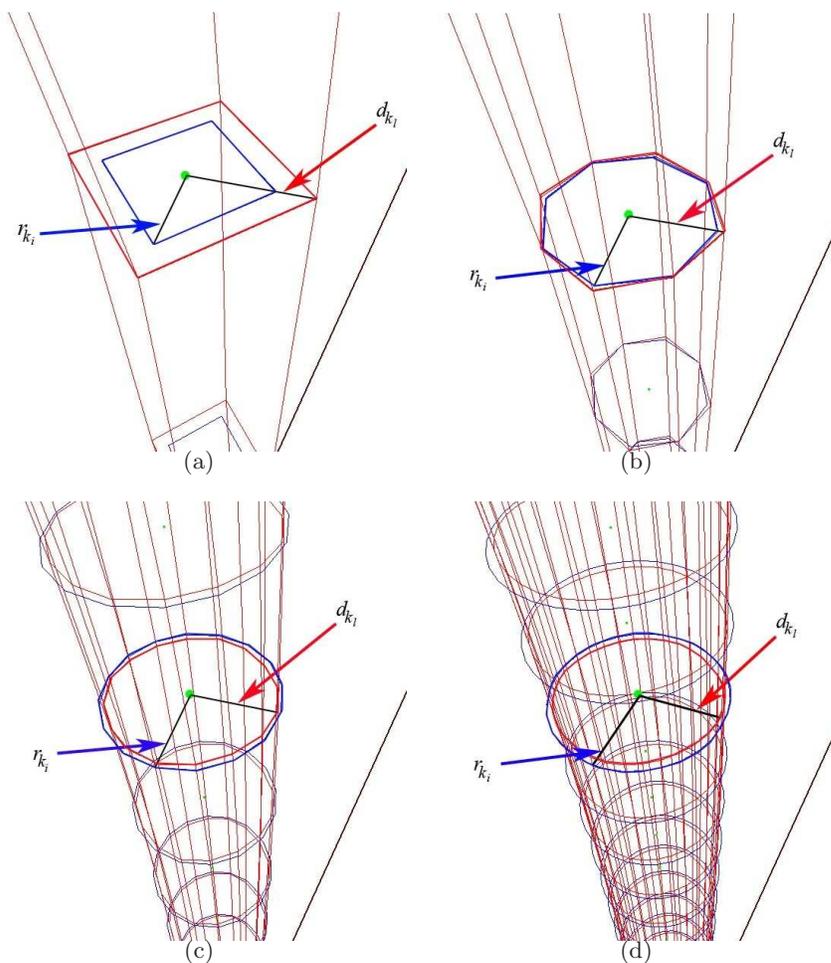


Figura 5.18: Medición de radio de un segmento a distintos niveles de refinamiento  $l$ , señalando las circunferencias con los valores de radio real  $r_{k_i}$  (curvas en color azul) y distancia  $d_{k_l}$  hacia el centro (puntos en color verde) de la circunferencia  $C_{k_j}$  correspondiente (curvas en color rojo). (a) Medición de radio en la malla *canónica*, sin refinamiento. (b) Circunferencias con radio real y medición de distancia con un nivel de refinamiento. (c) Valores de radio correspondientes aplicando 2 niveles de refinamiento. (d)  $r_{k_i}$  y  $d_{k_l}$  para 3 niveles de refinamiento.

T4-Conjunto1		$d_{k_l}$			
$k$	$r_{k_i}$	$l = 0$	$l = 1$	$l = 2$	$l = 3$
1	0.0042767	0.00604817	0.00453612	0.00415811	0.0040636
2	0.00733064	0.0103671	0.00777458	0.00712636	0.00696426
3	0.00547972	0.00774949	0.00577788	0.00529907	0.00518226
4	0.00571174	0.00807762	0.00605324	0.00554919	0.00542359
5	0.00365018	0.00516214	0.00386714	0.00354539	0.00346535
6	0.0052645	0.00744513	0.00557582	0.00511203	0.00499656
7	0.00679826	0.00961419	0.00720871	0.00660807	0.00645799
8	0.0058034	0.00820725	0.00614616	0.00563418	0.00550684
9	0.00549769	0.00777491	0.0058178	0.00533503	0.00521544
18	0.0042138	0.00595921	0.00446836	0.00409254	0.00399751
19	0.00438279	0.0061982	0.00464584	0.00425909	0.0041626
$\bar{x}$	$5.31 \times 10^{-3}$	$7.5094 \times 10^{-3}$	$5.6247 \times 10^{-3}$	$5.1563 \times 10^{-3}$	$5.0396 \times 10^{-3}$
$\sigma$	$1.1248 \times 10^{-3}$	$1.5907 \times 10^{-3}$	$1.1924 \times 10^{-3}$	$1.0933 \times 10^{-3}$	$1.0686 \times 10^{-3}$
% error		41.4213837	5.92751991	2.8939852	5.0906515

Tabla 5.1: Ancho de los segmentos de uno de los árboles del primer conjunto de datos. El valor de radio registrado  $d_{k_l}$  fue calculado para  $l$  niveles de subdivisión,  $l = 0, \dots, 3$ . El radio real del  $k$ -ésimo segmento del árbol corresponde a la columna etiquetada como  $r_{k_i}$ .

T4-Conjunto1		$ (d_{k_l} - r_{k_i})/r_{k_i} $			
$k$	$l = 0$	$l = 1$	$l = 2$	$l = 3$	
1	0.414214231	0.060658919	0.027729324	0.049828139	
2	0.414214857	0.060559515	.027866598	0.049979265	
3	0.41421277	0.054411539	0.032967013	0.054283796	
4	0.414213532	0.059789136	0.028458928	0.050448725	
5	0.414215189	0.059438165	0.028708173	0.050635859	
6	0.414214075	0.05913572	0.028961915	0.050895622	
7	0.414213343	0.060375743	0.027976276	0.050052513	
8	0.414214081	0.059061929	0.029158769	0.051101079	
9	0.41421397	0.058226273	0.029586972	0.051339745	
18	0.414212825	0.06041103	0.028776876	0.051328967	
19	0.414213321	0.060018846	0.028224031	0.050239688	
$\bar{x}$	0.414213836	$5.9281 \times 10^{-2}$	$2.8947 \times 10^{-2}$	$5.0921 \times 10^{-2}$	
$\sigma$	$7.698 \times 10^{-7}$	$1.7813 \times 10^{-3}$	$1.4510 \times 10^{-3}$	$1.2364 \times 10^{-3}$	

Tabla 5.2: Error relativo existente entre el radio original  $r_{k_i}$  y los valores obtenidos aplicando  $l$  veces el algoritmo de subdivisión al árbol seleccionado del primer conjunto de datos. El promedio  $\bar{x}$  del error relativo corresponde correctamente con el porcentaje de error promedio de la tabla anterior.

T2-Conjunto2		$d_{k_l}$			
$k$	$r_{k_i}$	$l = 0$	$l = 1$	$l = 2$	$l = 3$
1	0.00607969	0.00859799	0.00644797	0.0059106	0.00577628
2	0.00826593	0.0116898	0.00876675	0.00803617	0.00785357
3	0.00720092	0.0101836	0.00763589	0.0069997	0.00684081
4	0.00442755	0.00626149	0.00469563	0.0043043	0.00420649
5	0.00687629	0.00972454	0.00729323	0.00668523	0.00653313
10	0.00927527	0.0131172	0.00983778	0.00901797	0.00881303
11	0.00509005	0.00719842	0.00539739	0.00494758	0.00483522
20	0.00525966	0.00743829	0.00557799	0.00511298	0.00499673
21	0.00629666	0.00890482	0.00667746	0.00612107	0.00598207
$\bar{x}$	$6.5302 \times 10^{-3}$	$9.2351 \times 10^{-3}$	$6.9257 \times 10^{-3}$	$6.3484 \times 10^{-3}$	$6.2042 \times 10^{-3}$
$\sigma$	$1.5606 \times 10^{-3}$	$2.207 \times 10^{-3}$	$1.6554 \times 10^{-3}$	$1.5175 \times 10^{-3}$	$1.483 \times 10^{-3}$
% error		41.42129197	6.054020263	2.78435215	4.9933455

Tabla 5.3: Valores del ancho del  $k$ -ésimo segmento del árbol seleccionado del conjunto del maniquí de ojo;  $r_{k_i}$  es el valor real del ancho de cada tubo y  $d_{k_l}$  el radio en cada refinamiento  $l$ .

T2-Conjunto2		$ (d_{k_l} - r_{k_i})/r_{k_i} $			
$k$	$l = 0$	$l = 1$	$l = 2$	$l = 3$	
1	0.414215198	0.060575457	0.027812273	0.049905505	
2	0.414214734	0.060588464	0.027796025	0.049886704	
3	0.414208185	0.060404782	0.027943652	0.050008888	
4	0.41421102	0.060548159	0.027837066	0.04992829	
5	0.414213188	0.060634441	0.027785332	0.049904818	
10	0.414212201	0.060646213	0.027740432	0.049835746	
11	0.414214006	0.060380546	0.027989902	0.050064341	
20	0.414214987	0.060522924	0.027887734	0.049989923	
21	0.41421325	0.060476507	0.027886213	0.049961408	
$\bar{x}$	0.414212974	$6.0531 \times 10^{-2}$	$2.7853 \times 10^{-2}$	$4.9943 \times 10^{-2}$	
$\sigma$	$2.255 \times 10^{-6}$	$9.4304 \times 10^{-5}$	$8.0373 \times 10^{-5}$	$7.0141 \times 10^{-5}$	

Tabla 5.4: Error relativo de la distancia  $d_{k_l}$  respecto al radio real  $r_{k_i}$  en cada iteración  $l$  de refinamiento, de acuerdo a los valores de la tabla anterior.

T2-Conjunto3		$d_{k_l}$			
$k$	$r_{k_i}$	$l = 0$	$l = 1$	$l = 2$	$l = 3$
1	0.00477983	0.0067597	0.00498657	0.00455441	0.0044474
2	0.0022665	0.00320532	0.00240225	0.00220229	0.00215246
3	0.00424553	0.00600409	0.00449901	0.00412305	0.00402897
4	0.00220811	0.00312274	0.00231626	0.00212552	0.00208008
5	0.00218715	0.0030931	0.00231955	0.00212611	0.00207774
8	0.00181823	0.00257137	0.00191592	0.00175224	0.00171104
9	0.00176052	0.00248975	0.00186726	0.00171066	0.00167116
16	0.00124927	0.00176673	0.00132442	0.00121341	0.0011854
17	0.00156446	0.00221249	0.00165919	0.00152091	0.00148635
$\bar{x}$	$2.4533 \times 10^{-3}$	$3.4698 \times 10^{-3}$	$2.5878 \times 10^{-3}$	$2.3698 \times 10^{-3}$	$2.3156 \times 10^{-3}$
$\sigma$	$1.2203 \times 10^{-3}$	$1.7258 \times 10^{-3}$	$1.2755 \times 10^{-3}$	$1.1655 \times 10^{-3}$	$1.1382 \times 10^{-3}$
% error		41.42144785	5.483930868	3.40132973	5.6115147

Tabla 5.5: Ancho de cada segmento  $k$  de un árbol del tercer conjunto (retina real). En la tabla se registran el radio original  $r_{k_i}$  y el valor  $d_{k_l}$  con  $l$  iteraciones de refinamiento sobre la malla.

T2-Conjunto3		$ (d_{k_l} - r_{k_i})/r_{k_i} $			
$k$	$l = 0$	$l = 1$	$l = 2$	$l = 3$	
1	0.414213476	0.043252584	0.047160673	0.069548499	
2	0.414215751	0.05989411	0.028330024	0.050315464	
3	0.41421448	0.059705149	0.028849166	0.051008944	
4	0.414213966	0.048978538	0.037403028	0.057981713	
5	0.414214846	0.0605354	0.027908465	0.050024004	
8	0.414216023	0.053728076	0.036293538	0.058952938	
9	0.414212846	0.060629814	0.028321178	0.050757731	
16	0.414209899	0.060155131	0.028704764	0.051125858	
17	0.414219603	0.060551245	0.027837081	0.049927771	
$\bar{x}$	0.414214543	$5.6381 \times 10^{-2}$	$3.2312 \times 10^{-2}$	$5.4405 \times 10^{-2}$	
$\sigma$	$2.6285 \times 10^{-6}$	$6.3691 \times 10^{-3}$	$6.6938 \times 10^{-3}$	$6.6554 \times 10^{-3}$	

Tabla 5.6: Error relativo entre  $r_{k_i}$  y los valores de radio  $d_{k_l}$  para distintos niveles de subdivisión  $l$ , usando los datos de la tabla anterior.

En la Tabla 5.7 aparecen los promedios del porcentaje de error que presentan los radios de los cilindros que definen las mallas de los árboles seleccionados. Nótese que para 2 niveles de subdivisión ya existe una pérdida de volumen consecuencia de la disminución del tamaño de los radios correspondientes; con 3 refinamientos el valor es incluso más pequeño, pero no se aleja demasiado de los valores de los datos de entrada reales.

Árbol	% error			
	$l = 0$	$l = 1$	$l = 2$	$l = 3$
T3-Conjunto1	41.4213837	5.92751991	2.893985251	5.090651474
T2-Conjunto2	41.421292	6.054020263	2.784352146	4.993345473
T2-Conjunto3	41.4214479	5.483930868	3.401329734	5.611514701
$\bar{x}$	41.4213745	5.821823681	3.02655571	5.231837216
$\sigma$	$7.835 \times 10^{-5}$	0.299381445	0.329160343	0.332390372

Tabla 5.7: Porcentaje de error relativo promedio por nivel de refinamiento en cada uno de los árboles sometidos a medición. El porcentaje total promedio  $\bar{x}$  de los 3 árboles es aproximadamente del 3.026 % con 2 iteraciones y 5.232 % para 3 iteraciones de subdivisión.

### 5.3.2. Volumen

Otra medición calculada fue el volumen interno que ocupan los modelos como superficies cerradas, para ello se obtuvo el centro de cada circunferencia definida en los cilindros de los tubos y el punto central de las curvas “transversales“ formadas en las zonas de bifurcación; de esta manera, pueden formarse poliedros de 5 caras como se muestra en la Figura 5.19(a) y posteriormente dividir cada uno de esos poliedros en 3 tetraedros irregulares (Figura 5.19(b)).

Por cada tetraedro  $T$  se elige como vértice inicial uno de los 4 puntos que lo forman, para obtener los vectores  $\vec{u}$ ,  $\vec{v}$  y  $\vec{w}$  usando los puntos restantes, generando así un paralelepípedo  $H$  ((Figura 5.20(a)), cuyo volumen puede ser calculado como  $\text{volumen}(H) = |\vec{u} \cdot \vec{n}|$ , donde  $\vec{n} = \vec{v} \times \vec{w}$ .

Observando la (Figura 5.20(b) puede notarse que los tetraedros  $ABCD$  y  $BCDF$  son equivalentes, ya que sus respectivas bases  $ACD$  y  $FCD$  son equivalentes y tienen la misma altura, la distancia de  $B$  a la cara  $ACDF$ . Análogamente,  $BCDF$  es equivalente a  $EBDF$ , puesto que las bases  $CBF$  y  $EBF$  son equivalentes y tienen la misma altura, la distancia del punto  $D$  a la cara  $BCFE$ . Por lo tanto, el volumen buscado del tetraedro  $T$  se consigue al dividir el volumen del paralelepípedo entre 6, ya que ese número de tetraedros, del mismo volumen, están contenidos en el paralelepípedo en cuestión.

Finalmente, el volumen de cada modelo de malla corresponde a la suma de los volúmenes de todos los tetraedros que caben dentro de él.

Los resultados pueden observarse en las Tablas 5.8 y 5.9, junto con el valor de volumen promedio  $\bar{x}$ , la desviación estándar correspondiente y el valor del error relativo respecto al volumen  $V_l$  y a  $V_{l-1}$ , donde  $V_l$  es el volumen en el la iteración de subdivisión del nivel correspondiente y  $V_{l-1}$  el volumen en el nivel de iteración inmediato anterior.

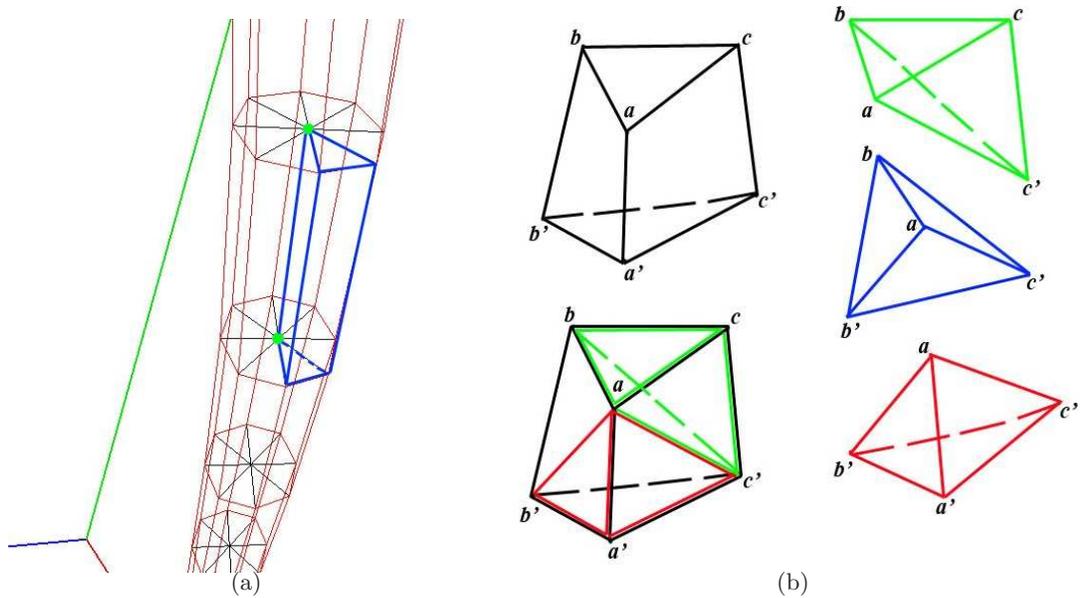


Figura 5.19: Descomposición de la malla en poliedros para el cálculo del volumen. (a) Los poliedros de 5 caras se forman tomando los puntos centrales (puntos en color verde) de dos curvas consecutivas y tomando, de 2 en 2, los puntos sobre cada curva. (b) Cada poliedro puede dividirse en 3 tetraedros, posiblemente irregulares, de la manera indicada en la figura.

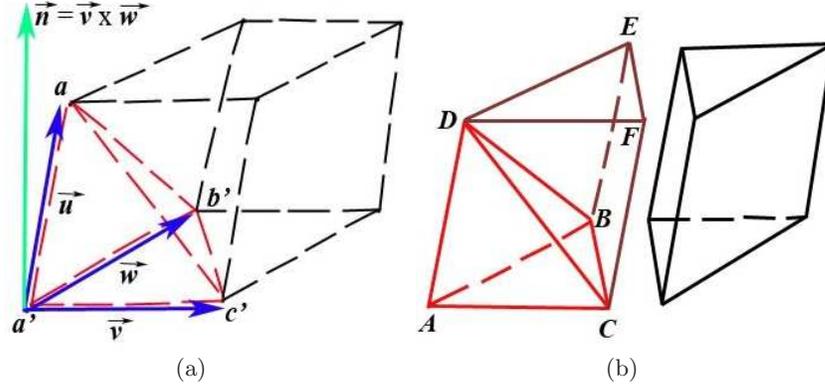


Figura 5.20: Cálculo del volumen del tetraedro (líneas en color rojo) formado por 4 puntos del poliedro padre al que pertenece. (a) Se obtiene el volumen del paralelepípedo como  $|\vec{u} \cdot (\vec{v} \times \vec{w})|$ . (b) El volumen buscado del tetraedro  $ABCD$  corresponde a la sexta parte del volumen del paralelepípedo.

Datos	volumen ( $u^3$ )			
	$l = 0$	$l = 1$	$l = 2$	$l = 3$
Conjunto1	$7.967 \times 10^{-4}$	$5.923 \times 10^{-4}$	$5.487 \times 10^{-4}$	$5.383 \times 10^{-4}$
Conjunto2	$1.083 \times 10^{-4}$	$8.146 \times 10^{-5}$	$7.553 \times 10^{-5}$	$7.409 \times 10^{-5}$
Conjunto3	$2.32 \times 10^{-5}$	$1.71 \times 10^{-5}$	$1.58 \times 10^{-5}$	$1.55 \times 10^{-5}$
$\bar{x}$	$3.094 \times 10^{-4}$	$2.303 \times 10^{-4}$	$2.134 \times 10^{-4}$	$2.093 \times 10^{-4}$
$\sigma$	$4.242 \times 10^{-4}$	$3.152 \times 10^{-4}$	$2.92 \times 10^{-4}$	$2.864 \times 10^{-4}$
% error		25.55898302	7.356507143	1.901819597

Tabla 5.8: Unidades de volumen promedio por nivel de refinamiento  $l$ , usando los 4 árboles de cada conjunto de datos. El volumen total promedio  $\bar{x}$  y la desviación estándar  $\sigma$  correspondientes a los 3 conjuntos aparecen en las últimas filas de la tabla, así como el porcentaje de error relativo promedio respecto a  $V_l$  y  $V_{l-1}$ ;  $V_0$  es el volumen de la malla *canónica*.

Datos	$ (V_l - V_{l-1})/V_{l-1} $		
	$l = 1$	$l = 2$	$l = 3$
Conjunto1	0.256487186	0.073596313	0.01899675
Conjunto2	0.247710694	0.072885413	0.019042723
Conjunto3	0.261558167	0.075718884	0.019645002
$\bar{x}$	0.255252016	0.07406687	0.019228159
$\sigma$	$7.0058 \times 10^{-3}$	$1.4742 \times 10^{-3}$	$3.6173 \times 10^{-3}$

Tabla 5.9: Error relativo promedio del volumen de cada conjunto de datos correspondiente, respecto al volumen en cada nivel de refinamiento  $l$  y el volumen en el nivel inmediato anterior;  $V_0$  corresponde al volumen de la malla *canónica*.

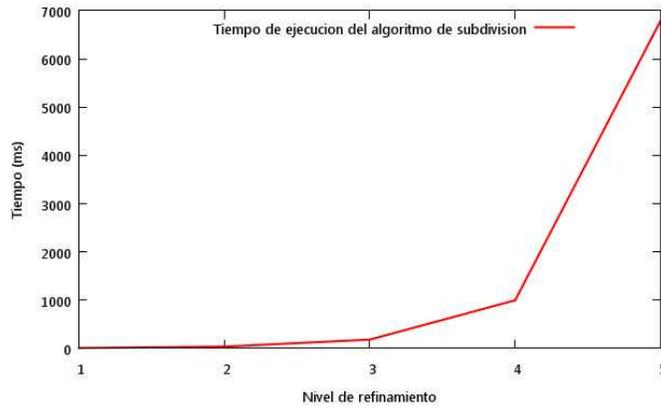
### 5.3.3. Rendimiento

En cuanto a la complejidad, la ejecución de cada procedimiento sobre la estructura de datos, teóricamente, tarda tiempos estimados entre  $O(Vl_{min} \times (M \log M))$  y  $O(Vl_{max} \times (M \log M))$ , donde  $M$  es el número de segmentos por árbol,  $l_{min}$  es el número de elementos de la lista de puntos del esqueleto más pequeño de los  $M$  segmentos,  $l_{max}$  el tamaño de la lista de longitud máxima y  $V$  el número de vértices por cada circunferencia que se define.

En efecto, la complejidad dependerá, de entre otras cosas, del número de vértices de cada malla; sin embargo, ese valor está determinado por el número de refinamientos que se apliquen a la malla *canónica* en cuestión. Por tanto, se registró el tiempo promedio que tarda el procesamiento de cada árbol de segmentos con el algoritmo de subdivisión, pues es la operación primordial para obtener, visualmente, una mejor malla. Se utilizaron los datos de los 3 experimentos para realizar la prueba, aplicando el algoritmo 20 iteraciones por árbol. Los resultados se describen en la Figura 5.21.

Nivel	Tiempo (milisegundos)
1	7.169549167
2	36.11296667
3	181.004025
4	995.828
5	6790.125

(a)



(b)

Figura 5.21: Tiempo de ejecución del algoritmo de subdivisión aplicado al método de generación de mallas tubulares. (a) Costo promedio por cada nivel de refinamiento. (b) Gráfica del tiempo calculado en función del nivel.

Analizando la figura anterior, puede notarse que el tiempo promedio de ejecución para niveles mayores a 3 crece rápidamente; cuando el nivel de refinamiento es igual a 5, el procesamiento tarda cerca de 7 segundos en completarse. En general, considérese un elemento de la malla en forma aislada, el cual está formado por 4 vértices; por ende, si se aplica subdivisión una vez se obtienen 9 nuevas posiciones de vértices en la malla: un *punto cara*, 4 *puntos arista* y 4 nuevos *puntos vértice* reemplazando los anteriores. Si el nivel es 2, se generan 25 vértices; con 3, 81 vértices. En resumen, hay  $(2^l + 1)^2$  nuevos vértices después de  $l$  refinamientos. Por lo tanto, si el número

de vértices crece siguiendo un comportamiento exponencial, sucederá lo mismo con la complejidad del algoritmo.

No obstante, revisando las figuras de la Sección 5.2, aplicar 2 o hasta 3 veces el algoritmo de subdivisión genera mallas con superficies suaves y en su mayoría bien formadas. Además, el porcentaje de error relativo del 5.232% a 3 iteraciones es bajo de acuerdo a los valores de las tablas 5.1 a 5.7. Finalmente, el tiempo de ejecución del algoritmo de subdivisión es aceptable hasta ese nivel de refinamiento; por lo tanto, 3 iteraciones son suficientes para obtener buenos resultados y que la aplicación sea aún eficiente.

## Capítulo 6

# Conclusiones y trabajo a futuro

Se propuso un algoritmo para generar modelos de mallas tubulares, cuya visualización satisface, en la gran mayoría de los casos, los requerimientos especificados. Asimismo, la interfaz gráfica que lo incluye es fácilmente manipulable, de modo que los usuarios finales puedan operarla aún con el mínimo de práctica.

Esencialmente, la aplicación representa conjuntos de venas y arterias retinales, obteniendo las estructuras de una manera consistente; es decir, en vez de procurar el ajuste de los datos de entrada para formar la superficie deseada, la construcción de ésta busca sólo interpretarlos sin hacer modificaciones a los mismos, para asegurar, en general, modelos con mayor fidelidad.

No obstante, un pequeño número de resultados erróneos se muestran en la Figura 5.17, donde algunas bifurcaciones presentan reducción de volumen o generación de malla forzada, consecuencia de varias razones como: el filtrado de los *puntos de control* en la lectura de los datos de entrada, la definición de los vértices sobre las circunferencias paramétricas de la malla *canónica* usando únicamente los *índices de referencia* y, en algunos casos, los ángulos de bifurcación de gran tamaño y poco probables en los vasos de retinas reales.

A pesar de ello, los valores originales como el largo de los segmentos tubulares o los ángulos de bifurcación no son alterados directamente por el algoritmo que construye el modelo de malla. En el caso del grosor de cada tubo y teniendo en cuenta que lo que se espera visualizar son superficies cilíndricas, los resultados conseguidos no difieren mucho de los reales, una vez que se aplican refinamientos a la malla *canónica*.

Así, cada modelo puede mejorarse usando el refinamiento por subdivisión, generando superficies con transiciones suaves en las zonas de bifurcación. Como se vió en el análisis de la Sección 5.3, con un porcentaje de error relativo promedio del 5.232 %

para los valores de los radios y un tiempo de ejecución aproximado de 181 milisegundos, en este caso, basta con 3 refinamientos por subdivisión para generar mallas tubulares aceptables para cada árbol de segmentos.

En particular, es el procedimiento de subdivisión el que permite resolver la malla en las regiones de bifurcación sin causar un “caos”, pues para formarlas se necesita seguir un orden al definir los nuevos vértices y caras. Es ese orden en los datos lo que reduce hasta cierto punto el uso de memoria, al no tener que almacenarse explícitamente la información correspondiente a cada polígono que define la malla.

Sin embargo, ahorrar memoria queda un poco de lado cuando se habla de la complejidad del algoritmo de subdivisión por sí mismo. Si bien es cierto que la estructura de datos de árbol binario, correctamente elegida, facilita la definición de los algoritmos mediante las llamadas recursivas y tiene tiempos estimados muy buenos, una vez aplicada la subdivisión, la complejidad de las llamadas a procedimientos subsecuentes no parece ser la mejor. Es importante recordar también que la subdivisión se acostumbra aplicarla sólo un pequeño número de veces, con justa razón; aparte que definir más y más vértices resulta poco útil si los modelos no mejoran. Por otro lado, el asunto de la complejidad no es evitable en esta área de las Ciencias de la Computación; no debe olvidarse que el nivel de detalle siempre tiene un fuerte impacto en el rendimiento, ya que lo mismo ocurre con muchas otras aplicaciones gráficas, visuales o en animación, incluso el análisis de algunos algoritmos en el procesamiento de imágenes suelen ser demasiado tardado con el fin de no perder información valiosa.

En resumen, el algoritmo completo obtiene en un buen tiempo promedio modelos aceptables realizando los cálculos necesarios; la cuestión sería entonces cómo optimizar su desempeño y los modelos de malla que así lo requieran. En primer lugar, considerando los errores mencionados en el análisis del Capítulo 5, podría modificarse el filtrado de *puntos de control* mediante un método especializado de supresión de puntos, posiblemente usando interpolación y, por otra parte, habrá que modificar el método de generación de *malla canónica* para obtener vértices mejor orientados y formar las bifurcaciones, o bien, suavizar en esas zonas la malla definida entre los segmentos mediante interpolación o reparametrización según se necesite.

Por su parte, para mejorar la eficiencia de las rutinas del sistema, sobre todo el algoritmo de subdivisión, podrían utilizarse procedimientos basados en nivel de detalle, *Level of Detail (LOD)*, ya que la idea de tales algoritmos es simplificar la geometría poligonal de objetos pequeños o distantes. Por ende, dado que las estructuras de los vasos sanguíneos a representar tienen regiones apenas perceptibles desde cierta distancia, la combinación de subdivisión con rutinas *LOD* mejoraría los tiempos de ejecución.

Otra de las características que quizás ha sido poco atendida es el hecho de que se ha trabajado con las zonas de bifurcación como estructuras “independientes“ de los cilindros de cada vaso involucrado. La ventaja de ésto podría radicar en que pueda paralelizarse, con o sin optimización por *LOD*, el algoritmo de subdivisión.

También, parece factible añadir a todo ello aplicar subdivisión de manera específica, restringiendo la generación de una mayor cantidad de vértices sólo para regiones próximas a una bifurcación; pues, como los cilindros tienen el mismo valor de radio para cada *punto de control* en el mismo segmento, no es necesario definir más vértices para esas superficies, es suficiente con reparametrizar las circunferencias originales de la malla *canónica* según el nivel de refinamiento.

En consecuencia, el siguiente paso a esta tesis es obtener un mejor modelo de malla *canónica* y, posiblemente, la optimización del proceso de subdivisión, probando las alternativas anteriores y estudiando al mismo tiempo alguna otra que se adapte al problema, como las técnicas de *remallado por rasgos* o las mallas adaptativas usando *remallado de alta calidad*.

Finalmente, una vez solucionados los inconvenientes anteriores, el fin inmediato de la construcción de las mallas tubulares hecha aquí es usarlas para simulación de flujo sanguíneo. En efecto, se ha buscado que la aplicación sea extendible, de forma tal que los módulos de la visualización y la interfaz gráfica permitan incluir, hasta donde sea posible, la aplicación de la simulación mencionada. En ese sentido, uno de los siguientes algoritmos a implementar será sin duda el que someta a las mallas a algún tipo de *remallado estructurado* o de *alta calidad*, ya que se necesita que los elementos de las mallas finales sean fuertemente uniformes entre sí, en relación con su área, forma y ángulos internos, para atribuir las propiedades necesarias a los vértices y conseguir así una simulación de mayor calidad.

Posteriormente, cuando los elementos de las mallas posean dichas cualidades se procederá a continuar con el desarrollo de la simulación, sometiendo a consideración los procedimientos y técnicas más adecuados que, como en este caso, sean suficientes para el fin buscado.

# Apéndice A

## Manual del Sistema

Las siguientes secciones describen la manera de interactuar con la aplicación en cualquiera de las plataformas, para lo cual se indica cómo ejecutar y operar el programa.

### A.1. Primeros Pasos

Inicie el sistema usando la herramienta *Qt Creator* o corra el ejecutable correspondiente que se encuentra en el directorio principal del sistema. Si es necesario compile la aplicación con dicha herramienta o hágalo mediante la línea de comandos, según la plataforma que utilice.

1. Cuando comience la aplicación aparecerá una ventana mostrando la *barra de menus* y la *barra de herramientas* en la parte superior, y la *barra de estado* en la parte inferior.
2. Para trabajar con el sistema de generación de mallas seleccione la opción **Process**→**Mesh Generation**→**Mesh Project**, presione  en la barra de herramientas o **Ctrl+M**.

Se abrirá una nueva ventana interna con dos secciones principales: el *área de la escena*, donde se visualizan los modelos de los vasos, y el *panel de control* formado por la lista de etiquetas y los tableros de controles.

*NOTA:* Si utiliza el ratón para manipular los modelos y la escena, presione la tecla **ESC**, si es necesario, para recuperar el foco sobre el área de la escena y utilizar el teclado si lo desea.

### A.1.1. Ajuste de la cámara

Modifique la posición de la cámara sobre la escena buscando la vista más apropiada si así lo requiere, para observar los detalles importantes de los vasos o la escena completa.

- a) Para hacer un acercamiento presione, tantas veces como se necesite, la combinación de teclas **Shift+Z**, gire la rueda del ratón hacia atrás o mueva hacia abajo el slider correspondiente al tablero de control *View*.

En caso contrario, para alejarse presione **Z** o realice las otras acciones anteriores en las direcciones opuestas.

- b) Para hacer una rotación sobre la escena también hay varias formas:

- Mantenga presionado el botón izquierdo o derecho del ratón para rotar la vista actual, girando el ratón en la dirección deseada.
- Elija un eje de rotación de la lista desplegable en el tablero *Rotate* y desplace la posición del slider para girar la escena alrededor del eje seleccionado.
- Use las flechas de dirección del teclado:  $\leftarrow$  y  $\rightarrow$  para rotar alrededor del eje *Y*,  $\uparrow$  y  $\downarrow$  para girar sobre el eje *X* y los comandos **Ctrl+ $\uparrow$**  y **Ctrl+ $\downarrow$**  para el eje *Z*.
- Se puede además rotar la escena automáticamente, para ello presione el botón , **View→Rotate→Automatic** o las teclas **Shift+R**. Seleccione el eje de rotación de la lista desplegable en el tablero en cuestión, use las opciones disponibles desde el menú **View→Rotate** o las combinaciones **Alt+X**, **Alt+Y** o **Alt+Z**.

*NOTA:* Tome en cuenta que activando la rotación automática se bloquea el slider y varias opciones asociadas con el ratón y las flechas de dirección. Para desactivarla oprima de manera similar alguna de las opciones indicadas arriba.

- c) Para modificar la vista haciendo una traslación oprima las teclas **W**, **S**, **A** o **D** para moverse hacia arriba, hacia abajo, a la izquierda o a la derecha, respectivamente; o bien, presione el botón derecho del ratón sobre el *área de la escena* y sin soltarlo desplácelo hacia otro punto dentro de la misma.
- d) Por último, al presionar , **View→Restore Position** o **Shift+V** puede restaurar la vista de la cámara a la posición original cuando inicia la aplicación.

## A.2. Cargar archivo

Agregue uno o varios modelos a la escena, cargando los datos desde archivos con el formato adecuado.

1. Presione el botón , la opción **File**→**Open** o **Ctrl+O**, para que aparezca en pantalla el diálogo de carga de archivos.
2. Dependiendo del formato del archivo, el resultado varía.
  - *Archivos .dat*. Se añadirán a la escena del proyecto en turno los datos del archivo seleccionado.
  - *Archivos .mshv*. Abrirá un proyecto existente en la misma área de trabajo si la escena no contiene modelo alguno; en caso contrario, una nueva ventana extrayendo los modelos contenidos en tal proyecto.
  - *Archivos .xml*. Funciona igual que el archivo en formato *.dat*, pero el *.xml* contiene la información organizada de distinta manera, incluyendo la lista de *puntos de control* y la malla *canónica* de un modelo.

En cualquier caso, si los archivos corresponden a esta aplicación, en la escena se mostrará el modelo de cada árbol de segmentos de acuerdo al modo previamente seleccionado por el usuario; además, aparecerá una nueva etiqueta, por modelo, en la lista del *panel de control*.

## A.3. Generar malla

1. Seleccione **Process**→**Mesh Generation**→**Model**→**Surface**, o bien ; se mostrará por defecto la malla *canónica* de cada modelo. En cambio, si posteriormente los modelos de la escena son modificados mediante subdivisión serán las mallas refinadas las que aparezcan en pantalla.

Para acceder a esta opción desde el teclado haga lo siguiente:

- Presione el comando **Shift+M** para obtener las mallas como superficies *lisas*; es decir, sin distinguir sus elementos.
- Si oprime la tecla **M** observará las mallas como estructuras de alambre, usando líneas para formar las caras de su geometría.
- Presionando la tecla del número **0** se visualizará una combinación de las dos representaciones anteriores.

Si se elige con el teclado una opción distinta a la predefinida será tal representación la que se siga mostrando con la acción de generación de malla, en el proyecto actual hasta que se cierre la ventana del proyecto actual o se eliminen todos los árboles de la escena.

2. Para regresar a la representación por esqueleto oprima el botón , elija **Process**→**Mesh Generation**→**Model**→**Skeleton** desde la barra de menus o presione la tecla **E** o **Shift+E**.
3. Desactive la opción **Process**→**Mesh Generation**→**Vessels**→**All**, utilice el botón  del tablero *Model* o presione **Shift+A**, para intercambiar entre la representación por malla o por esqueleto de un sólo modelo a la vez. Seleccione un elemento de la lista de etiquetas y modifique la representación de ese vaso según lo necesite.

Si requiere trabajar con todos los modelos por igual, active de nuevo la opción señalada; cada modelo será mostrado de acuerdo a la representación del último seleccionado.

#### A.4. Subdividir malla

Aplique refinamiento por subdivisión a las mallas de la escena; para habilitar esta opción se necesita activar la representación del modelo por malla (Sección A.3).

1. Presione el botón , el comando **Shift+D** o la opción correspondiente del menu **Process**→**Mesh Generation**→**Model**→**Subdivision** para refinar las mallas de los modelos seleccionados.

Además, puede incrementar y decrementar el nivel de refinamiento girando la rueda del tablero *Model* o pulsando **+** ó **-** desde el teclado.

2. Deshabilite la opción de subdivisión para mostrar nuevamente en pantalla los modelos elegidos sólo con la malla *canónica*.

#### A.5. Modificar atributos

Es posible cambiar algunas características de los modelos desplegados en la escena.

1. Modifique la escala del conjunto de modelos completo; oprima alguna de las opciones del menu **Process**→**Mesh Generation**→**Model**→**Scale**, o desplace el valor del slider del tablero *Scale* para aumentar o reducir las dimensiones de los modelos.

Presione el botón , la combinación **Shif+S** o la opción en el menú arriba mencionado para restaurar la escala al tamaño original.

2. Seleccione un elemento de la lista de etiquetas con el ratón o use la combinación **Ctrl+L** para acceder a dicha lista y desplácese sobre ella con las flechas de dirección.
3. Modifique la etiqueta seleccionada y asigne otro valor distinto al resto de la lista, presione **Enter** una vez que lo haya escrito o **Esc** para conservar el nombre anterior.

*NOTA:* El acceso a la lista de etiquetas sirve únicamente para cambiar el nombre de alguna etiqueta o aplicar cambios a un árbol a la vez desde el teclado, si está desactivada la opción *All*. Presione **Esc** para salir del modo de edición de la lista de etiquetas o aplicar los cambios al árbol seleccionado.

4. Cambie el color del modelo asociado a la etiqueta seleccionada:
  - Pulse el botón , **Process**→**Mesh Generation**→**Model**→**Color** o **Shift+C**.
  - Elija un nuevo color en el diálogo emergente y presione Aceptar.
5. Alternativamente, puede cambiar el color de todos los modelos usando la opción **Process**→**Mesh Generation**→**Vessels**→**Group**, el botón  o el comando **Shift+G**, reagrupando las venas en color azul y las arterias en color rojo.

## A.6. Capturar escena

Guarde algunas vistas de los modelos de la escena actual con un pantallazo o con un video de varios segundos. Para activar el tablero *Media* y las opciones del menú **Tools**→**Media** se debe agregar por lo menos un modelo al proyecto.

1. Obtenga una imagen con el contenido de la escena presionando el botón , **Tools**→**Media**→**Screenshot** o las teclas **Ctrl+T**. Guarde la imagen resultante oprimiendo el botón *Save* del diálogo que aparece en pantalla.
2. Utilice el botón  del tablero o presione **Ctrl+R** para grabar algunos cuadros de video capturando la escena. Oprima  o **Ctrl+Y** para abrir un nuevo diálogo donde podrá reproducir la secuencia grabada; guarde el video mencionado presionando el botón *Save* en el último diálogo.

## A.7. Guardar proyecto

Almacene en un archivo en disco las mallas de los modelos del proyecto en turno.

1. Seleccione **File**→**Save As**,  en la *barra de herramientas* o el comando **Ctrl+S** para abrir el diálogo correspondiente.
2. Elija el formato de salida que sea más apropiado, de acuerdo a los siguientes criterios:
  - *Archivo .mshv*. Guarda los cambios y características generales de los modelos en la escena, según las opciones seleccionadas.
  - *Archivo .xml*. Almacena en disco sólo la malla *canónica* y los atributos (Sección A.5) del modelo en turno seleccionado.
  - *Archivo .obj*. Guarda las mallas, incluso con refinamiento, de los modelos seleccionados, además del color del material de cada uno.
3. Finalmente, pulse el botón *Save* para guardar los cambios y volver a la escena.

*NOTA:* Los archivos *.obj* no pueden abrirse con esta aplicación (ver Sección A.2), en cambio pueden visualizarse en otros programas, como *Blender*, para aplicar modificaciones adicionales y especializadas a los modelos.

# Bibliografía

- Aldana Iuit, J. A. (2010). *Calibración del Sistema Cámara-ojo para la Reconstrucción 3D de Estructuras en Imágenes de Retina*. Master's thesis, Universidad Nacional Autónoma de México, Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, D.F., México.
- Alliez, P., Verdière, E. C. d., Devillers, O., y Isenburg, M. (2003). Isotropic surface remeshing. In *Proceedings of the Shape Modeling International 2003*, pages 49–, Washington, DC, USA. IEEE Computer Society.
- Barratt, D., Ariff, B., Humphries, K., Thom, S., y Hughes, A. (2004). Reconstruction and quantification of the carotid artery bifurcation from 3-D ultrasound images. *IEEE Transactions on Medical Imaging*, **23**(5), 567–583.
- Behrens, T., Rohr, K., y Stiehl, H. (2003). Robust segmentation of tubular structures in 3-D medical images by parametric object detection and tracking. *Systems, Man and Cybernetics, Part B: IEEE Transactions on Cybernetics*, **33**(4), 554–561.
- Catmull, E. y Clark, J. (1978). *Recursively generated B-spline surfaces on arbitrary topological meshes*, pages 183–188. ACM, New York, NY, USA.
- Cebal, J. y Löhner, R. (1999). From medical images to CFD meshes. In *Proceedings of the 8th International Meshing Roundtable*, pages 321–331, South Lake Tahoe, CA, U.S.A.
- Chen, S.-Y., Carroll, J., y Messenger, J. (2002). Quantitative analysis of reconstructed 3-D coronary arterial tree and intracoronary devices. *IEEE Transactions on Medical Imaging*, **21**(7), 724–740.
- De Floriani, L. y Spagnuolo, M. (2008). *Shape Analysis and Structuring*, chapter 2, Recent Advances in Remeshing of Surfaces, pages 53–78. Springer, Serie: Mathematics+Visualization.
- Doo, D. y Sabin, M. (1978). *Behaviour of recursive division surfaces near extraordinary points*, pages 177–181. ACM, New York, NY, USA.

- Frey, P. J. y George, P.-L. (2008). *Mesh Generation: Application to Finite Elements - 2nd ed.* ISTE-Wiley.
- Gamma, E., Helm, R., Johnson, R., y Vlissides, J. M. (1998). *Design Patterns CD: Elements of Reusable Object-Oriented Software*, pages 87–97,315–330. Addison-Wesley.
- García, F., Palacio, C., y García, U. (2009). Unstructured mesh generation for numerical models implementation. *Dyna rev.fac.nac.minas*, **76**(157), 17–25.
- Hahn, H., Preim, B., Selle, D., y Peitgen, H.-O. (2001). Visualization and interaction techniques for the exploration of vascular structures. In *Visualization, 2001. Proceedings VIS '01*, pages 395 –578.
- Kraevoy, V. y Sheffer, A. (2004). Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph.*, **23**, 861–869.
- La Cruz, A., Straka, M., Kochl, A., Sramek, M., Groller, E., y Fleischmann, D. (2004). Non-linear model fitting to parameterize diseased blood vessels. In *Proceedings of the conference on Visualization '04*, pages 393–400, Washington, DC, USA. IEEE Computer Society.
- Li, J., Regli, W. C., y Sun, W. (2007). Mathematical representation of the vascular structure and applications. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 373–378, New York, NY, USA. ACM.
- Martinez-Perez, M. E. y Espinosa-Romero, A. (2004). Retinal blood vessel 3D reconstruction from two views. In *4th Indian Conference on Computer Vision and Graphics*, pages 258–263.
- Martinez-Perez, M. E., Hughes, A. D., Thom, S. A., Bharath, A. A., y Parker, K. H. (2007). Segmentation of blood vessels from red-free and fluorescein retinal images. *Medical Image Analysis*, **11**(1), 47 – 61.
- Rivera Loaiza, C. (2000). *Utilización de redes de Petri para la elaboración de una interfaz de usuario*. Master's thesis, Universidad Michoacana de San Nicolás de Hidalgo, Facultad de Ingeniería Eléctrica, Michoacán, México.
- Schröder, P. (2000). Subdivision for modeling and animation. In *Course notes of 23th International conference on Computer Graphics and Interactive Techniques*. ACM Siggraph.
- Yim, P., Cebral, J., Mullick, R., Marcos, H., y Choyke, P. (2001). Vessel surface reconstruction with a tubular deformable model. *IEEE Transactions on Medical Imaging*, **20**(12), 1411 –1421.