UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

ESTUDIO DE ESTRUCTURAS VORTICIALES DENTRO DE MODELOS DE ANEURISMAS CEREBRALES

MEMORIA PARA OPTAR AL TITULO DE
INGENIERO CIVIL MECÁNICO

FELIPE ANDRÉS CUEVAS RIVEROS

PROFESOR GUÍA:
ÁLVARO VALENCIA MUSALEM

PROFESOR CO-GUÍA:
WILLIAMS CALDERÓN MUÑOZ

COMISIÓN:
ALEJANDRO ORTIZ BERNARDIN

SANTIAGO DE CHILE
2021

RESUMEN DE MEMORIA PARA OPTAR AL TÍTULO DE: Ingeniero Civil Mecánico
POR: Felipe Andrés Cuevas Riveros
FECHA: Diciembre 2021
PROFESOR GUÍA: Álvaro Valencia Musalem

ESTUDIO DE ESTRUCTURAS VORTICIALES DENTRO DE MODELOS DE ANEURISMAS CEREBRALES

Un aneurisma cerebral es una enfermedad cerebrovascular donde la debilidad en la pared arterial produce un abultamiento de sangre. Esta cavidad tiene riesgo de romperse y causar una hemorragia Subaracnoidea junto a diferentes complicaciones, incluso fatales.

Estudios han relacionado el riesgo de ruptura con diferentes parámetros, algunos respecto a la geometría del aneurisma y otros a parámetros fluidodinámicos. Estos han logrado cierto nivel de correlación, pero los autores consideran que los resultados aún no son concluyentes.

El objetivo de esta tesis es relacionar las estructuras vorticiales dentro del aneurisma con resultados medibles por CFD como los esfuerzos de corte. Para esto se simularán casos de aneurismas preparados por una tesis de doctorado de Nicolás Amigo en 2018 que estudió la hemodinámica en los modelos, luego se aislarán los resultados de la zona interna del aneurisma, después se ubicarán y caracterizarán los vórtices para finalmente realizar el análisis solo en la zona de interés.

Para aislar los resultados de la zona del aneurisma se segmenta el modelo CAD con una adaptación del método de ramificación de bifurcaciones (propuesto por Luca Antiga en [1]) a aneurismas cerebrales. La identificación de vórtices se computa con los criterios λ_2, Q, $\mho$ (Omega), y derivados normalizados y adimensionales de estos.

Se identifica la etapa crítica del trabajo el proceso de aprendizaje y desarrollo de los procedimientos con los módulos VTK y VMTK de Kitware.

A mi madre, padre y hermano. Gracias por todo

Agradecimientos

El desarrollo de esta memoria tuvo muchos altos y bajos. Desde un principio el tema siempre me interesó y me mantuvo motivado, en ocasiones el desarrollo se complicó y hubo que realizar repetidas modificaciones a la metodología. Algunos momentos el final se veía muy lejano y a poco tiempo aumentaba la cuesta arriba, la que sin el apoyo de muchas personas muy importantes para mí no podría haber sobrellevado y agradezco el soporte dado para poder terminar mi trabajo.

Partir por agradecer a mi familia: mi madre, padre y hermano quienes incondicionalmente me han apoyado siempre.

Parte importante del trabajo y el inicio de este se lo debo a mi profesor guía Álvaro Valencia, quien me introdujo a esta área de investigación y me apoyó durante el desarrollo de la tesis.

Quiero agradecer a mi polola Coni quien me dio el apodo “el suelto” con el que me conoció la mayoría del departamento de mecánica, y a su familia por el apoyo brindado durante los últimos meses de mi trabajo dándome ánimos y celebrando todos los pequeños pasos logrados hasta terminar.

También quiero agradecer a mi amigo Franco por todas las experiencias compartidas, el apoyo e interés por todos mis proyectos y estudios, a Felipe Corrales con quien compartí la mayoría del tiempo que estuve en la Universidad en las buenas y en las malas, y a Lisa por su fiel y sincera amistad.

Agradezco además a Oliwi mi primer grupo de amigos de la Universidad: Carlos, Magda, Vale, Domi, Mariana, Caro y Natalia quienes me acompañaron desde el primer año compartiendo juntas de estudio, carretes, cumpleaños y salidas ocasionales haciendo más llevadera la introducción al mundo universitario.

No puede faltar tampoco la comunidad del departamento de mecánica “la Gente Bonita del Cuarto”, con quienes compartí mis últimos años estudiando, carreteando y haciendo los jueves de oncecita.

También agradezco a la secretaria del departamento Claudia y las funcionarias María Eugenia y Maricarmen por toda la ayuda, conversaciones y tiempo compartido en el departamento de ingeniería mecánica.

Tabla de Contenido

Agradecimientos ..... **iii**

Índice de Tablas ..... **vii**

Índice de Figuras ..... **viii**

1 Introducción **1**
1.1 Motivación 1
1.2 Objetivos 2
1.3 Estructura de la Memoria 2

2 Aneurismas Cerebrales **3**
2.1 Clasificación 3
2.2 Parámetros Morfológicos 4
2.3 Parámetros Hemodinámicos 5
2.4 Formación, crecimiento y ruptura 6

3 Flujo Sanguíneo **7**
3.1 Ecuaciones del Navier-Stokes 7
3.2 Modelos Viscosos 8
3.3 Condiciones de Borde 9
3.3.1 Condiciones de Borde de Velocidad 9
3.3.2 Condiciones de Borde de Presión 10

4 Visión Computacional y Visualización **12**
4.1 Introducción a la Visualización 12
4.1.1 Tipo de Datos 12
4.2 Transformaciones 14
4.2.1 Transformaciones geométricas 15
4.2.2 Algoritmo de los Cubos Marchantes 15
4.2.3 Triangulación 17
4.3 VTK 18
4.3.1 Objetos Data Set 18
4.3.2 Objetos de Proceso 19
4.3.3 Modelo Gráfico 20

5 Segmentación de mallas **22**
5.1.1 Algoritmo de Segmentación de Ramificaciones 24
5.1.2 Segmentación de mallas con campos dependientes de la concavidad 25

5.2 Materiales y Métodos 28
5.2.1 Recorte Previo 29
5.2.2 Computo de Centerlines 30
5.2.3 Selección de Ramificaciones a Recortar 31
5.2.4 Implementación 32
5.3 Resultados de segmentación 32
5.4 Discusión 38

6 Caracterización de Vórtices 39
6.1 Revisión General de Técnicas de Identificación de Vórtices 39
6.2 Métodos de 1° Gen.: Basados en la vorticidad 40
6.3 Métodos de 2° Gen.: Basados en valores propios 41
6.3.1 Criterio Q 41
6.3.2 Criterio $\boldsymbol{\lambda 2}$ 42
6.3.3 Comparación entre $\boldsymbol{\lambda 2}$ y $\boldsymbol{Q}$ 43
6.4 3° Gen. Métodos Omega, Liutex y Omega-Liutex 44
6.4.1 Método Omega 44
6.5 Materiales y Métodos 45
6.5.1 Malla de la geometría 46
6.5.2 Point y Cell Data de cada time-step 48
6.5.3 Localizador 49
6.5.4 Computador de Estructuras Vorticiales 51
6.6 Resultados 51
6.6.1 Performance de Algoritmos Localizadores 51
6.6.2 Computación de Estructuras Vorticiales 52
6.7 Discusión 54

7 Estudio de Estructuras Vorticiales en Aneurismas Cerebrales 55
7.1 Materiales y Métodos 55
7.1.1 Geometrías y simulaciones 55
7.1.2 Segmentación de Estructura Vorticial 56
7.1.3 Selección del Valores Umbrales 58
7.1.4 Visualización de Estructuras Vorticiales 58
7.1.5 Relación de Estructuras Vorticiales con Parámetros Medibles por CFD 59
7.1.6 Implementación 59
7.2 Resultados 60
7.2.1 Distribución de Criterios de Caracterización de Vórtices durante la Diástole 60
7.2.2 Distribución de Criterios de Caracterización de Vórtices durante la Sístole 64
7.2.3 Visualización de Estructuras Vorticiales 69
7.2.4 vVF y WSS en el ciclo Cardiaco 70
7.3 Discusión 72

8 Conclusión 73
8.1 Trabajo Futuro 74

Bibliografía..... 75

Anexos..... 78
Anexo-A. Tipos de Celdas lineales en VTK 78

Índice de Tablas

Tabla 5.1: Resultado de segmentación de geometrías probadas. 33
Tabla 5.2: Geometrías donde se logra la segmentación. 34
Tabla 6.1: Clasificación de algoritmos de detección de Vórtices según [24]. 40
Tabla 6.2: Comparación criterios $\lambda 2$ y Q 43
Tabla 6.3: Inputs del Algoritmo de caracterización de Vórtices. 45
Tabla 6.4: Tipos de Celdas de ANSYS. 47
Tabla 7.1: Grupos de Arterias Cerebrales. 55
Tabla 7.2: Geometrías con que se realizó estudio de estructuras vorticiales. 59

Índice de Figuras

figura 2.1: Vista inferior del Cerebro. 3
figura 2.2: Clasificación de Aneurismas según geometría. 4
figura 2.3: Clasificación de aneurismas por tipo. 4
figura 2.4: Dimensiones utilizadas en los parámetros morfológicos. 4
figura 2.5:Localizaciones más frecuentes de aneurismas cerebrales. 6
figura 3.1: Circuito RCR para modelo de Windkessel. 10
figura 3.2: Esquemas condición de Borde de Presión Windkesel. 11
figura 4.1: Proceso de Visualización. 12
figura 4.2: Tipos de datos atribuidos. 13
figura 4.3: Tipos de Set de Datos. 14
figura 4.4: 15 casos generalizados del estado topológico de la celda. 16
figura 4.5: Casos generalizados complementarios del algoritmo Asymptotic decider. 17
figura 4.6: Triangulación Delaunay y Teselación Voronoi para un mismo set de datos. 18
figura 4.7: Relación entre triangulación Delaunay y teselación de Voronoi. 18
figura 4.8: Objeto de Set de Datos de vtk. 19
figura 4.9: Diagrama del Objeto DataSet. 19
figura 4.10: Diagrama de heredabilidad de datos atribuidos del Set de Datos. 19
figura 4.11: Objetos básicos involucrados en el modelo gráfico de vtk. 20
figura 5.1: Segmentación según Superficie o Parte. 22
figura 5.2: Segmentación según criterio geométrico. 22
figura 5.3: Segmentación según información topológica o geométrica. 23
figura 5.4: Identificación de puntos de referencia. 24
figura 5.5: Descomposición de las ramificaciones propuesta por [1]. 25
figura 5.6: Campo de segmentación respecto a la concavidad de [23]. 27
figura 5.7: Ejemplo de $polopi$ a partir de un VD en 2D 27
figura 5.8: Ejemplos de localización de polos respecto al punto y la geometría. 28
figura 5.9: Diagrama del algoritmo de segmentación creado. 29
figura 5.10: Cubo de recorte generado 29
figura 5.11: Recorte de geometría con el cubo. 30
figura 5.12: Visualización del PolyData de vmtkcenterlines. 30
figura 5.13: Resumen de las interacciones del usuario en el algoritmo de segmentación. 31
figura 5.14: Segmentación de la geometría 25. 32
figura 5.15: Segmentación de la geometría 40. 32
figura 5.16: Segmentación geometría 43 utilizando 1 target point en el aneurisma. 33
figura 5.17: Segmentación geometría 43 utilizando 2 target point en el aneurisma. 33
figura 5.18: Segmentación geometría 47_48. 33
figura 5.19: Geometrías en las que no funcionó la segmentación. 38
figura 6.1: Ejemplo de uso de Streamlines para identificación de vórtices. 39
figura 6.2: Fenómeno de Vortex Breakdown al seleccionar valor umbral. 43
figura 6.3: Descomposición del vector vorticidad en R y S. 45

figura 6.4: Diagrama del Algoritmo de caracterización de Vórtices. 46
figura 6.5: Pseudo código para armar una UnstructuredGrid. 47
figura 6.6: Operaciones booleanas para ordenar nodos en las celdas. 48
figura 6.7: Pseudo código Método 1 de generar tabla LookUp. 50
figura 6.8: Pseudo código Método 2 de generar tabla LookUp. 50
figura 6.9: Pseudo código Método 3 de generar tabla LookUp. 50
figura 6.10: Performance de diferentes métodos del algoritmo indexador. 52
figura 6.11: Performance del algoritmo de indexación 3. 52
figura 6.12: Estructura vorticial $\lambda 2$ en geo11. 53
figura 6.13: Estructura vorticial Q en geo11. 53
figura 7.1: Perfil de velocidad y de Presión utilizado en las simulaciones. 56
figura 7.2: Diagrama del algoritmo de segmentación de estructuras vorticiales. 56
figura 7.3: Voxelización de geo20 para diferentes valores ds. 57
figura 7.4:Diferencias del uso de CountourFilter y MultiThreshold para computar las estructuras vorticiales. 58
figura 7.5: Distribución temporal de la UDF de Velocidad y time-steps de la simulación. 59
figura 7.6: Distribución criterios de caracterización de vórtices en geo11 durante la diástole. 60
figura 7.7: Distribución criterios de caracterización de vórtices en geo16 durante la diástole. 61
figura 7.8: Distribución criterios de caracterización de vórtices en geo19 durante la diástole. 62
figura 7.9: Distribución criterios de caracterización de vórtices en geo20 durante la diástole. 63
figura 7.10: Distribución criterios de caracterización de vórtices en geo30_34 durante la diástole. 64
figura 7.11: Distribución criterios de caracterización de vórtices en geo11 durante la sístole. 65
figura 7.12: Distribución criterios de caracterización de vórtices en geo16 durante la sístole. 66
figura 7.13: Distribución criterios de caracterización de vórtices en geo19 durante la sístole. 67
figura 7.14: Distribución criterios de caracterización de vórtices en geo20 durante la sístole. 68
figura 7.15: Distribución criterios de caracterización de vórtices en geo30_34 durante la sístole. 69
figura 7.16: Estructura vorticial de geo1 en un ciclo según $\mho = 0.51$ en diferentes instantes del ciclo cardiaco. 70
figura 7.17: vVF y WSS de geo11 a lo largo del ciclo cardiaco. 70
figura 7.18: vVF y WSS de geo16 a lo largo del ciclo cardiaco. 70
figura 7.19: vVF y WSS de geo19 a lo largo del ciclo cardiaco. 71
figura 7.20: vVF y WSS de geo20 a lo largo del ciclo cardiaco. 71
figura 7.21: vVF y WSS de geo30_34 a lo largo del ciclo cardiaco. 71
figura 8.1: Celdas lineales. 78

1 Introducción

1.1 Motivación

Un aneurisma cerebral es una enfermedad cerebrovascular que consiste en un abultamiento de sangre en una cavidad de arteria ocasionada por un debilitamiento de la pared de esta. Estos abultamientos de sangre tienen riesgo de ruptura y pueden causar diferentes complicaciones como vasopasmos, hidrocefalia, o isquemia, entre otras.

Se localizan generalmente en las arterias del círculo de Willis (zona inferior del cerebro). Alrededor del 2% de la población común tiene un aneurisma cerebral, muchos de ellos asintomáticos incluso (entre 50-80%) [2].

Se ha estudiado distintas formas de poder inferir el estado de la enfermedad o cuantificar el riesgo de ruptura estableciendo relaciones entre distintos parámetros. Algunas de estas cantidades relacionan la geometría del aneurisma (AR o Tamaño máximo) y otras comparan con propiedades mecánicas (Esfuerzos de Corte o Flujo dentro del aneurisma) [3].

Un trabajo que se está comenzando a realizar es relacionar las estructuras vorticiales con parámetros que puedan predecir el estado de la enfermedad. Avances de esto se ha logrado en [4] se utiliza el método propuesto por [5] que consiste en un algoritmo semiautomático para aislar la zona del aneurisma del modelo completo, de esta manera se pueden analizar exclusivamente las estructuras al interior del aneurisma.

Un vórtice corresponde a la región de un fluido donde el flujo sigue una trayectoria rotatoria alrededor de una línea axial que puede ser recta o curva. Generalmente son fáciles de identificar a simple vista, pero el panorama se complejiza al tratar de definir una estructura o definición matemática, lo que ha llevado a múltiples definiciones matemáticas de vórtices.

Además de la complejidad de identificar y caracterizar vórtices, existe la dificultad de visualizarlos en estructuras comprensibles, sobre todo cuando se trata de flujos en 3d.

1.2 Objetivos

El trabajo se centrará estudiar las estructuras vorticiales dentro del aneurisma, entiéndase cantidad de vórtices, forma e intensidad de ellos. Estas estructuras se pueden relacionar con los esfuerzos de corte en la pared del aneurisma.

El objetivo general es:

- Caracterizar las estructuras vorticiales dentro de modelos de aneurismas cerebrales y relacionarlas con resultados medibles por CFD.

Los objetivos específicos son:

- Implementar un algoritmo para segmentar la zona correspondiente al Aneurisma en la estructura de datos simulada en ANSYS Fluent.
- Implementar un algoritmo de identificación y de vórtices.
- Caracterizar y Visualizar los vórtices en la zona segmentada.
- Estudiar las estructuras vorticiales dentro del aneurisma en las diferentes fases del ciclo cardiaco.

1.3 Estructura de la Memoria

Esta Memoria de Título está compuesta por 8 Capítulos y N Anexos.

En el Capítulo 2 entrega una pequeña introducción a los aneurismas cerebrales, indicando sus clasificaciones, parámetros de caracterización, y una pequeña revisión del estudio sobre su formación, crecimiento y ruptura.

El Capítulo 3 contextualiza las propiedades del flujo sanguíneo y los modelos matemáticos que rigen la dinámica de la sangre al interior del organismo.

El Capítulo 4 introduce la visón computacional, Visualización de Datos y el módulo de visualización utilizado (VTK), para comprender los algoritmos presentados más adelante en el desarrollo de la metodología.

En el Capítulo 5 presenta metodologías para segmentación de mallas 3d y la implementación de un algoritmo a partir de descomposición de ramificaciones [1] adaptado a aneurismas cerebrales.

El Capítulo 6 expone una contextualización de identificación de vórtices y la implementación de los modelos para la identificación de Vórtices, específicamente los criterios de λ_2, Q y $\mho$ y su normalización.

El Capítulo 7 se enfoca en la aplicación de los métodos descritos en los 2 capítulos anteriores a aneurismas cerebrales y la visualización de las estructuras vorticiales.

El Capítulo 8 discute sobre los resultados y conclusiones de la memoria de manera general y propone trabajos complementarios a futuro.

2 Aneurismas Cerebrales

Un aneurisma cerebral es una enfermedad cerebrovascular que consiste en un abultamiento de sangre en una cavidad de arteria ocasionada por un debilitamiento de la pared de esta. Estos abultamientos de sangre tienen riesgo de ruptura causando una hemorragia subaracnoidea y diferentes complicaciones como vasopasmos, hidrocefalia, o isquemia, entre otras.

Aproximadamente entre un 2 y 3% de la población común sufre esta afección [6] pero se estima que entre el 50 y 80% de los casos no hay ruptura [2].

2.1 Clasificación

Los aneurismas generalmente se localizan en la zona posterior del cerebro, en el círculo de Willis, compuesto por la arteria carótida interna, arterias vertebrales, arteria basilar, arteria cerebral media, arteria cerebral anterior y la arteria comunicante anterior.

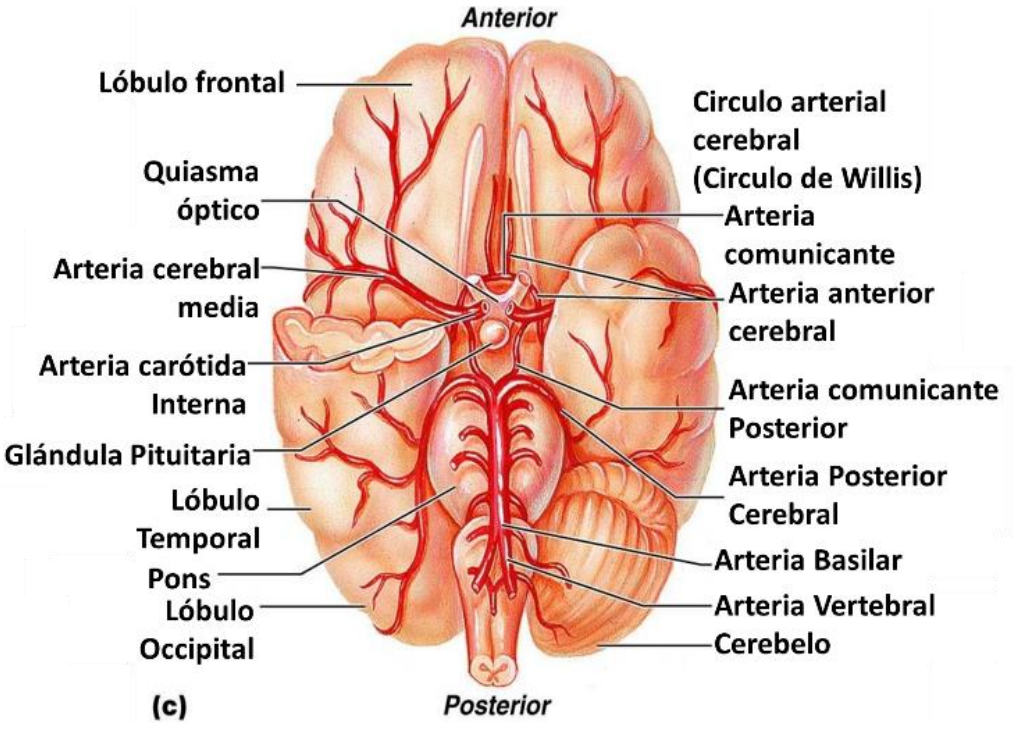


figura 2.1: Vista inferior del Cerebro.

Según el tamaño de los aneurismas, estos se pueden clasificar aproximando su geometría a una esfera. Serán **pequeños** (si su diámetro es menor a 11[mm]), **grandes** (diámetro entre 11 y 25[mm]) o **gigantes** (diámetro superior a 25[mm]).

Según su geometría se clasifican en Saculares, Fusiformes y Disecantes.

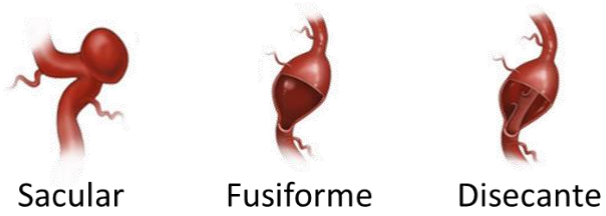


figura 2.2: Clasificación de Aneurismas según geometría.

Además de la geometría se pueden clasificar en aneurismas verdaderos: cuando se forman por dilatación de las 3 capas arteriales; y falsos cuando se forman con 1 o las 2 exteriores.

Otra clasificación importante es la localización en la arteria que se ubica, distingüendo entre Lateral, Lateral con bifurcación (cuando se ubica entre una arteria y arteriola) y Terminal (cuando se ubica entre la bifurcación de una arteria en 2 o más arterias).

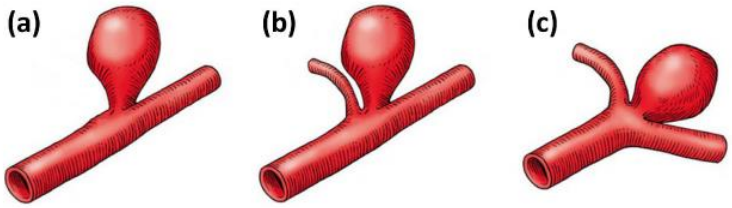


figura 2.3: Clasificación de aneurismas por tipo.
a) Lateral. b) Lateral con bifurcación. C) Terminal.

2.2 Parámetros Morfológicos

En este trabajo se considerarán los 3 parámetros morfológicos de los aneurismas más importantes según los resultados obtenidos por [7] y [8]. Estos parámetros se basan en las dimensiones que se pueden apreciar en la figura 2.4.

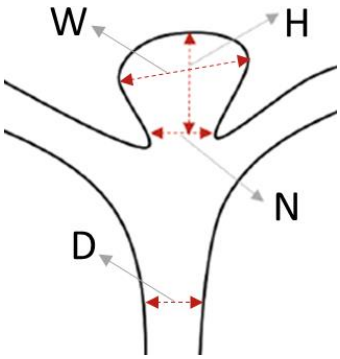


figura 2.4: Dimensiones utilizadas en los parámetros morfológicos.

La primera es la razón de aspecto (AR) que se calcula con el tamaño del Aneurisma (H), que se mide desde el cuello hasta el punto más alto, y con el diámetro del cuello (N).

$$AR = H/N \tag{2.1}$$

Otro parámetro es el factor de cuello de botella (BNF) que relaciona el ancho del aneurisma (W) con su cuello (N).

$$\mathrm{BNF} = \mathrm{W}/\mathrm{N} \tag{2.2}$$

El tercer parámetro morfológico es la relación entre el largo máximo del aneurisma (H) y el diámetro de la arteria alimentadora.

$$\mathrm{SR} = \mathrm{H}/\mathrm{D} \tag{2.3}$$

2.3 Parámetros Hemodinámicos

En cuanto a los parámetros sanguíneos, el más mencionado es el esfuerzo de corte en las paredes (WSS), del cual se puede definir el esfuerzo de corte promedio en las paredes del aneurisma.

$$WSS_a = \frac{1}{A_a}\int_{A_a} \overrightarrow{WSS}(\vec{x}, t)\, dA_a \tag{2.4}$$

Es de utilidad distinguir los valores del esfuerzo de corte promedio en la pared durante la diástole y durante la sístole, dado que los mecanismos de formación y ruptura de aneurismas se atribuyen a esfuerzos de corte altos y bajos [7].

$$DWSS = \frac{1}{A_a}\int_{A_a} \overrightarrow{WSS}(\vec{x}, t_{min})\, dA_a \tag{2.5}$$

$$SWSS = \frac{1}{A_a}\int_{A_a} \overrightarrow{WSS}(\vec{x}, t_{max})\, dA_a \tag{2.6}$$

Otros factores destacados en los estudios de parámetros hemodinámicos son el índice de oscilación de esfuerzo de corte (OSI) que cuantifica el cambio de magnitud de WSS y su oscilación tangencial a lo largo de un ciclo cardiaco.

$$\mathrm{OSI} = \frac{1}{2}\left(1 - \frac{\left|\int_0^T \overrightarrow{WSS}(\vec{x}, t)dt\right|}{\int_0^T \left|\overrightarrow{WSS}(\vec{x}, t)\right|dt}\right) \tag{2.7}$$

El esfuerzo de corte promedio en el tiempo (TAWSS) se calcula promediando el esfuerzo de corte durante un ciclo del pulso.

$$\mathrm{TAWSS} = \frac{1}{\mathrm{T}}\int_0^T \left|\overrightarrow{WSS}(\vec{x}, t)\right|dt \tag{2.8}$$

Además, se define el tiempo relativo de residencia (RRT) para cuantificar el estado del flujo turbulento.

$$\mathrm{RRT} = \frac{1}{(1 - 2 \cdot OSI) \cdot TAWSS} \tag{2.9}$$

2.4 Formación, crecimiento y ruptura

La formación, crecimiento y ruptura de aneurismas cerebrales se ha adjudicado a diferentes causas. Entre las más comunes se encuentran **causa inflamatoria**, que comienza con una disfunción del tejido endotelial, luego la respuesta inflamatoria desencadena anormalidad en las células de musculo liso desregulando la lámina elástica interna y la síntesis de colágeno, finalmente ocasionando la apoptosis de las células de músculo liso adelgazando la pared de la arteria que junto a fenómenos hemodinámicos ocasionan la ruptura [3]. Otra causa de los aneurismas cerebrales es su relación con **factores de riesgo** como la hipertensión y el tabaquismo [9]. También se atribuye la predisposición a desarrollar aneurismas cerebrales a **factores genéticos** como la incidencia de la aflicción si se tienen parientes de 1° y 2° grado, genes asociados a la elastina y colágeno, y enfermedades hereditarias como enfermedad poliquística renal, síndrome de Marfan, síndrome de Ehlers-Danlos tipo IV y displasia fibromuscular [3] [9].

La localización más frecuente de aneurismas se concentra en las arterias anterior comunicante (30%), Posterior Comunicante (25%) y Media Cerebral (20%). El detalle de la distribución se observa en la figura a continuación:

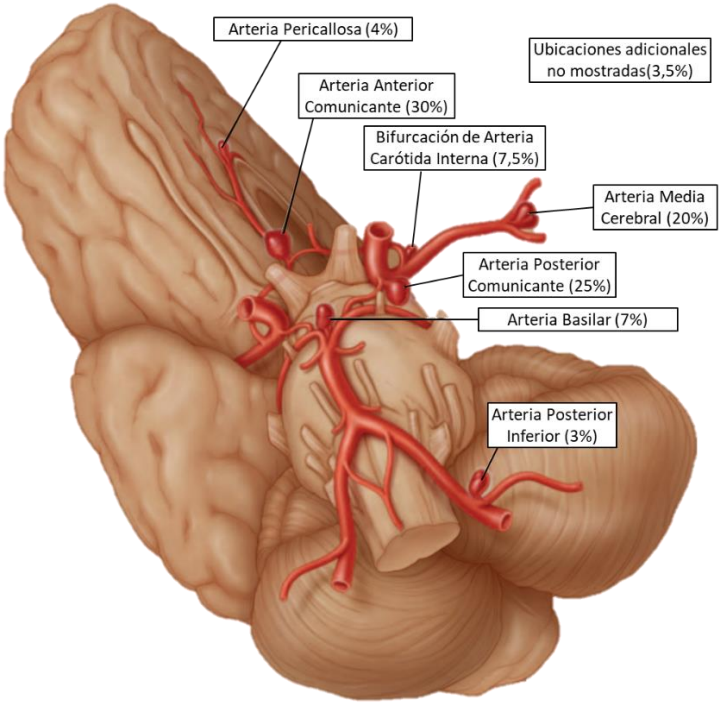


figura 2.5:Localizaciones más frecuentes de aneurismas cerebrales.
Fuente: Traducción de figura 1 de [9].

Respecto a parámetros que caracterizan el aneurisma la barrera observada es que no existe una estandarización o metodología para tomar las medidas de aneurismas, lo que origina variadas definiciones de una misma caracterización. Recientemente se encontró que el AR, BNF y HW estadísticamente muestran ser potenciales predictores del riesgo de ruptura independientemente de la localización del aneurisma [8].

En cuanto a propiedades mecánicas se identifica el WSS como parámetro clave para identificar el riesgo de ruptura del aneurisma [7].

3 Flujo Sanguíneo

La Sangre corresponde a aproximadamente el 8% del peso corporal. De ésta, el 55% es plasma sanguíneo y el 45% son elementos corpusculares. Las funciones de la sangre son transporte (oxigeno, dióxido de carbono, hormonas, nutrientes, calor y desechos), regulación (pH, temperatura corporal, y contenido de agua en las células) y protección (coagulación contra pérdida de sangre y transporte de glóbulos blancos, fagocitos y anticuerpos contra las enfermedades) [10].

Se han desarrollado diferentes modelos para simular el comportamiento de la sangre. Dentro del cuerpo humano debido a las condiciones en que se encuentra (presión, temperatura, velocidad, etc.) y su composición química se comporta como un fluido. A continuación, se enuncian las ecuaciones y modelos que describen su mecánica.

3.1 Ecuaciones del Navier-Stokes

Para un fluido incompresible, la ecuación de continuidad y la ecuación de Navier Stokes:

$$\nabla \cdot \vec{u} = 0 \tag{3.1}$$

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}\right) = -\nabla P + \nabla \tau \tag{3.2}$$

Donde la primera ecuación describe la continuidad de masa y la segunda la conservación de momentum en el fluido. En ambas ecuaciones $\vec{u}$ es el campo de velocidades en un punto en el espacio, P es presión, y τ el tensor de esfuerzos del fluido, que dependerá del modelo de viscosidad utilizado.

El tensor de esfuerzos puede ser escrito en función de la viscosidad dinámica (μ) y la tasa de deformación ($\dot{\gamma}$) como:

$$\tau = \mu\dot{\gamma} \tag{3.3}$$

Donde para flujos incompresibles se tiene:

$$\dot{\gamma} = 2\sqrt{\epsilon_{ij}\epsilon_{ji}} \tag{3.4}$$

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.5}$$

3.2 Modelos Viscosos

Para modelar la sangre como fluido no Newtoniano se pueden tomar diferentes modelos. En la bibliografía se ha tomado el modelo viscoso de Carreau [11] o el modelo de Casson [12]

Un estudio entre diferentes modelos no-Newtonianos de viscosidad validado in-vivo con pacientes de estenosis (en diferentes grados) demostró que los modelos que mejor aproximan el flujo en el centro de la arteria son el de Carreau, Casson Modificado y el modelo de Quemada [13].

Modelo de Carreu

Se trata de una ley de fluido Newtoniano generalizada, basada en un decrecimiento de la densidad ante altas deformaciones de corte.

$$\mu_{app} = \mu_\infty + (\mu_0 - \mu_\infty)\left(1 + K\dot{\gamma}_{ij}^2\right)^n \tag{3.6}$$

Aquí μ_0 y μ_∞ representan los valores asintóticos de viscosidad a bajas y grandes deformaciones de corte. K y n controlan la zona de transición. En la bibliografía [11] para la sangre se utiliza $\mu_\infty = 0.00345\ Ns/m^2$ asd $\mu_0 = 0.056\ Ns/m^2$, $K = 10.976$ y $n = -0.3216$.

Modelo de Casson y Casson modificado

El modelo toma la relación entre el esfuerzo y la tasa de corte con la forma:

$$\sqrt{\tau} = \sqrt{\mu_0 \dot{\gamma}} + \sqrt{\tau_0} \tag{3.7}$$

Con μ_0 la viscosidad newtoniana (para la sangre $\mu_0 = 0.0036[Pa]$). Luego reemplazando τ en la ecuación (3.3) se puede reescribir la viscosidad dinámica aparente:

$$\mu_{app} = \left(\sqrt{\frac{\tau_0}{\dot{\gamma}}} + \sqrt{\mu_0}\right)^2 \tag{3.8}$$

Como para $\dot{\gamma}$ muy cercanos a cero la expresión diverge, resulta útil reescribirla de la forma:

$$\sqrt{\mu_{app}} = \sqrt{\tau_0 \left(\frac{1 - e^{-m\dot{\gamma}}}{\dot{\gamma}}\right)} + \sqrt{\mu_0} \tag{3.9}$$

Con esta definición el parámetro m controla la viscosidad máxima cuando $\dot{\gamma}$ tiende a cero.

A partir del modelo original también surge el modelo de Casson modificado:

$$\mu_{app} = \left(\frac{\sqrt{\tau_0}}{\sqrt{\lambda} + \sqrt{\dot{\gamma}}} + \sqrt{\mu_0}\right)^2 \tag{3.10}$$

Donde fue validado in-vivo en [13] con los parámetros $\mu_0 = 0.00298\ [Pa \cdot s]$, $\tau_0 = 0.02876\ [Pa \cdot s]$ y el parámetro $\lambda = 4.02\ [s^{-1}]$

Modelo de Quemada

El modelo define la viscosidad aparente según:

$$\mu_{app} = \mu_0 \left(1 - \frac{k_0 + k_\infty \sqrt{|\dot{\gamma}|/\gamma_0}}{2\left(1 + \sqrt{|\dot{\gamma}|/\gamma_0}\right)} \phi \right)^{-2} \tag{3.11}$$

Donde $\mu_0 = 0.0012\ [Pa \cdot s]$, $k_0 = 4.33$, $k_\infty = 2.07$, $\gamma_0 = 1.88\ [s^{-1}]$ y $\phi = 0.4$

3.3 Condiciones de Borde

La sangre fluye por el cuerpo humano bombeada por el corazón, definiendo una naturaleza pulsátil, lo que nos lleva a establecer condiciones de borde con dependencia temporal. En 1955 se propone una solución hasta hoy en día usada para definir el perfil de velocidades en la entrada a partir de un gradiente de presión conocido [14].

3.3.1 Condiciones de Borde de Velocidad

Perfil de Womersley

Se considera que los flujos de caudal y presión son pulsantes y periódicos, por lo que se pueden resolver la presión y el caudal en el dominio de frecuencias, luego como serie de Fourier tendrán la forma:

$$P(t) = \sum_{n=0}^{N} P_n e^{in\omega t} \tag{3.12}$$

$$u_z(t) = \sum_{n=0}^{N} u_n(r) e^{in\omega t} \tag{3.13}$$

En el caso de arterias, una buena aproximación al perfil de velocidades es el perfil de velocidad es el perfil de Womersley en un tubo de pared elástica delgada. [15] de donde se deduce la solución a partir de la ecuación (3.2), que para la geometría propuesta tendrá la forma (en coordenadas cilíndricas):

$$\rho \frac{\partial u_z}{\partial t} = \mu \left[\frac{\partial^2 u_z}{\partial r^2} + \frac{1}{r} \frac{\partial u_z}{\partial r} \right] - \frac{\partial \mathrm{P}}{\partial \mathrm{z}} \tag{3.14}$$

Para un gradiente de presión constante (correspondería al termino $n = 0$), con un poco de algebra la solución es:

$$\mathrm{u}_0(\mathrm{r}) = \frac{2Q_0}{\pi a^2} \left[1 - \left(\frac{r}{a} \right)^2 \right] \tag{3.15}$$

Por otro lado, para un gradiente de presión dependiente del tiempo, las ecuaciones (3.12) y (3.13) definirían al reemplazar en (3.14) la ecuación solo para el término n-ésimo:

$$\frac{\partial^2 u_n}{\partial r^2} + \frac{1}{r}\frac{\partial u_n}{\partial r} + \frac{i^3 n\omega}{\nu} u_n = -\frac{P_n}{\mu} \tag{3.16}$$

La ecuación (3.16) corresponde a una EDO de Bessel. Trabajando con las condiciones de borde $\left.\frac{\partial u_n}{\partial r}\right|_{r=0} = 0$ (simetría radial respecto al centro del tubo), $u_n(r=a) = 0$ (Bordes sin deslizamiento) y utilizando $\alpha_n = a\sqrt{-in\omega/\nu}$ obtiene:

$$\mathrm{u_n}(r) = \frac{iP_n}{n\omega\rho}\left[1 - \frac{J_0(\alpha_n r/a)}{J_0(\alpha_n)}\right] \tag{3.17}$$

Integrando el perfil en el área de la sección se igual al caudal Q_n, de esta manera se relaciona Q_n y P_n obteniendo:

$$u_n(r) = \frac{Q_n}{\pi a^2}\left[\frac{1 - \frac{J_0(\alpha_n r/a)}{J_0(\alpha_n)}}{1 - \frac{2J_1(\alpha_n)}{\alpha_n J_0(\alpha_n)}}\right] \tag{3.18}$$

3.3.2 Condiciones de Borde de Presión

Modelo de Windkessel

El modelo se basa en la analogía del flujo de sangre a un circuito eléctrico. Este circuito puede varía según el modelo propuesto. Uno se basa en el circuito de la figura 3.1:

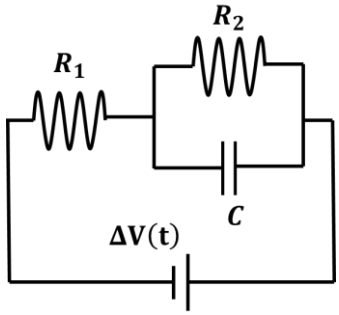


figura 3.1: Circuito RCR para modelo de Windkessel.
fuente: Elavoración propia a partir de [16].

Con este circuito usando el análogo donde $I = Q$ y $V = P$, el sistema tendrá la ecuación [16]:

$$\left(1 + \frac{R_1}{R_2}\right) Q(t) + CR_1 \frac{dQ}{dt} = \frac{P(t)}{R_2} + C\frac{dP}{dt} - P_v \tag{3.19}$$

Donde $R_T = R_1 + R_2$ es la Resistencia Terminal y C la Compliance Terminal . Los valores de R_1, R_2 y C se consiguen de manera experimental y varían entre cada sección del sistema circulatorio.

Otros circuitos que se proponen para el modelo consideran más componentes. Un ejemplo es el modelo híbrido Windkessel-Womersley que utiliza el perfil de Womersley tubular incorporando esta solución en reemplazo de los componentes en serie LR de un modelo de Windkessel-6 [17].

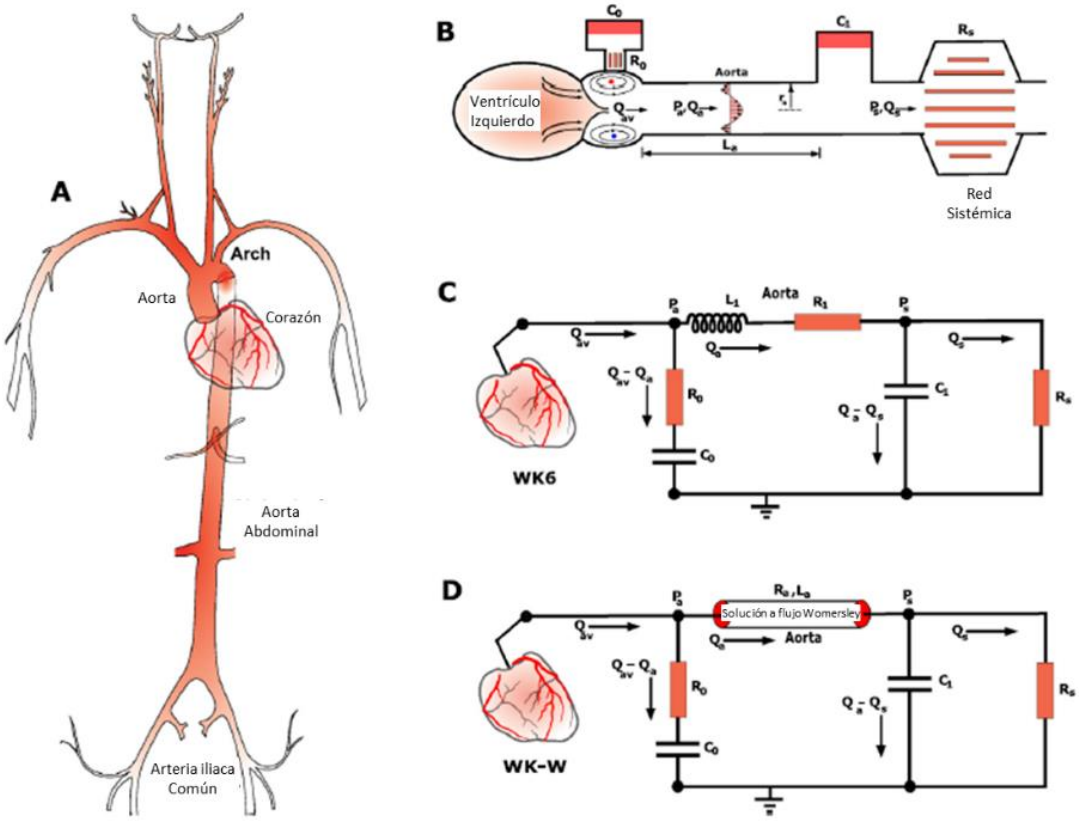


figura 3.2: Esquemas condición de Borde de Presión Windkesel.
A) Definición de problema de flujo en las arterias. B) Simplificación del subdominio de la arteria aorta izquierda para resolver perfil de Womersley. C) Modelo Windkessel de 6 elementos (WK6). D) Modelo híbrido propuesto.
Fuente: Adaptación de [17].

4 Visión Computacional y Visualización

4.1 Introducción a la Visualización

La visión es uno de los sentidos más importantes del ser humano. Nos sirve para ver el entorno en que nos encontramos y es una de las primeras interfases que utilizamos para interactuar.

La visualización es una técnica de cómputo, que transforma lo simbólico en geométrico. Esto permite observar el resultado de simulaciones y cómputos [18]. En otras palabras, ofrece un método para observar de manera concreta lo abstracto.

Cabe señalar que el proceso de visualización es diferente al procesamiento de imágenes (transformación, extracción de información y mejoramiento de imágenes 2d) y a la computación grafica (Crear imágenes 2d o 3d a través de técnicas de renderización) que son subprocesos a realizar dentro del contexto de visualización [19].

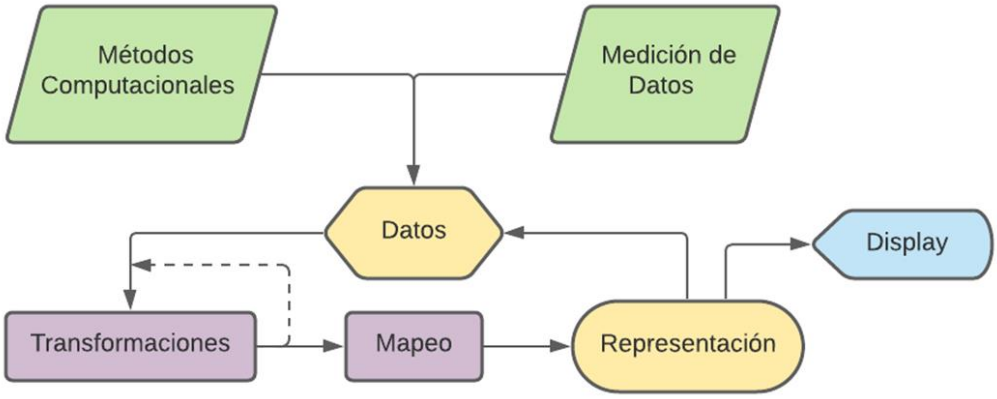


figura 4.1: Proceso de Visualización.
fuente: Traducción y modificación de [19].

4.1.1 Tipo de Datos

Topología y Geometría

El primer concepto a clarificar es la diferencia entre geometría y topología que representan al set de datos. La geometría corresponde a la región en que se ubican los datos ($\vec{x} \in \mathbb{R}^n$). Por otro lado, la topología corresponde a la conectividad y asociatividad del set de datos.

Una forma de asociar la geometría y la topología es establecer un sistema compuesto por puntos y celdas. Dada una geometría $\{U \subseteq R^n\}$:

- Un punto corresponde a la identidad que tiene asociado un vector $\vec{p}_i$ incluido en la geometría ($\vec{p}_i \in U$).
- Una Celda C_i corresponde a un set de puntos $\vec{p}_j \in U$ y su conectividad (relación) entre ellos. La cantidad de puntos está definida por la topología de la celda.

Generalmente la topología y la geometría estarán compuestas por puntos en 1d, 2d o 3d. En estos casos la topología de las celdas puede ser una del Anexo: **Tipos de Celdas lineales en VTK**.

En conjunto la geometría y la topología conforman una estructura.

Datos Atribuidos (Attribute Data)

Son información asociada a la estructura de un set de datos. Esta estructura se relaciona directamente con la geometría y topología. Para el caso 1d, 2d y 3d en la figura 4.2 se pueden ver los distintos tipos de datos atribuidos.

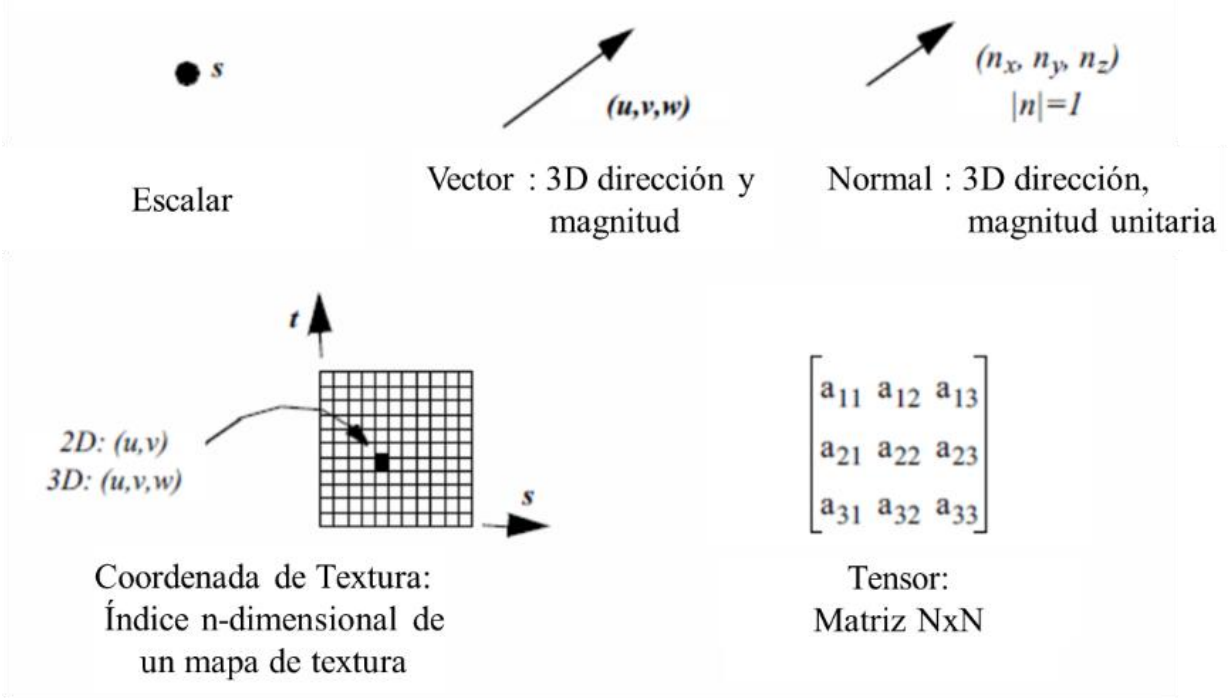


figura 4.2: Tipos de datos atribuidos.

Estos datos pueden ser escalares como temperatura, vectores como velocidad, tensores como el tensor de esfuerzos, etc.

Set de Datos

Un set de datos consiste en una estructura organizada y sus datos atribuidos. La estructura generalmente está dada por el tipo de celdas. Algunos de los tipos de set de datos se pueden observar en la figura 4.3:

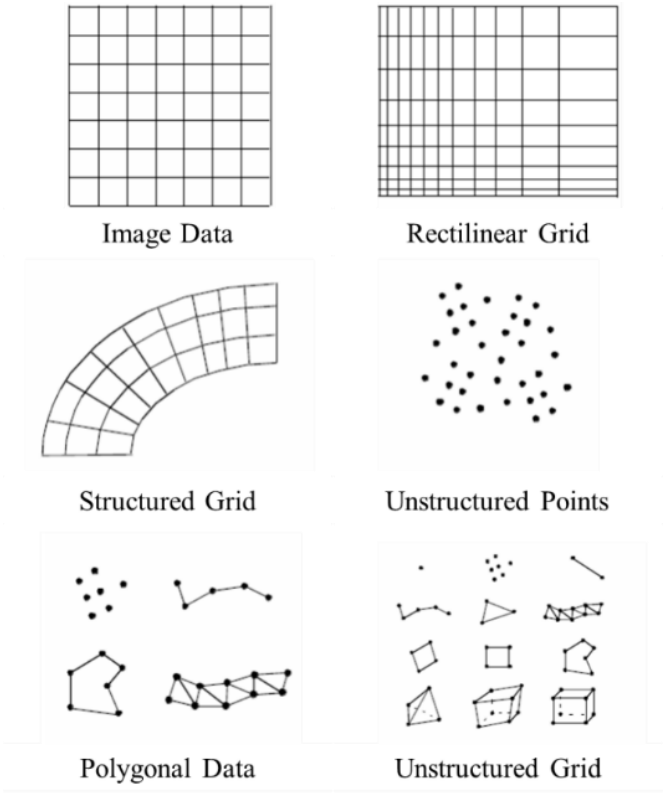


figura 4.3: Tipos de Set de Datos.

4.2 Transformaciones

Un proceso importante en la visualización son los algoritmos que transforman los datos que se quieren visualizar. Las transformaciones se pueden clasificar en:

- **Transformaciones Geométricas:**
  Afectan principalmente los puntos del set de datos. Por ejemplo, traslaciones, escalas, rotaciones, etc.

- **Transformaciones Topológicas:**
  Cambian la topología del set de datos. Por ejemplo, el tipo de celdas.

- **Transformaciones de Atributos:**
  Cambian los datos atribuidos o crean nuevos. Por ejemplo, computar magnitud de un vector, puntos de elevación a partir de la geometría, etc.

- **Transformaciones Combinadas:**
  Cambian la estructura y datos atribuidos. Por ejemplo, computar y definir líneas o superficies de contorno, crear bloques, etc.

Otra forma de clasificar las transformaciones es según el tipo de datos con los que trabaja cada algoritmo, estos son: Algoritmos Escalares, Vectoriales, Tensoriales y Algoritmos de Modelado (Generan la topología, geometría, Normales de superficies o texturas, de un Set de Datos).

4.2.1 Transformaciones geométricas

En computación gráfica se representan objetos 3d en una imagen 2d realizando métodos de proyección y perspectiva. Para facilitar el trabajo, se representan los puntos de un sistema cartesiano (x, y, z) en el denominado sistema homogéneo (x_h, y_h, z_h, w_h) donde:

$$x = \frac{x_h}{w_h} \quad y = \frac{y_h}{w_h} \quad z = \frac{z_h}{w_h} \tag{4.1}$$

De esta manera una traslación del punto (x, y, z) en un vector (t_x, t_y, t_z)se puede hacer definiendo $w_h = 1$ y aplicando la matriz:

$$T_T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Análogamente se puede escalar un objeto con la matriz:

$$T_S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

Aplicar Rotación en torno a los ejes x, y o z:

$$T_{Rx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

$$T_{Ry} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$T_{Rz} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

Para aplicar una serie de transformaciones geométricas basta con multiplicar secuencialmente estás al vector inicial. Hay que tener cuidado de que la aplicación de estas generalmente no es conmutativa, por ende, se debe tener cuidado en el orden en que estas son aplicadas para obtener el resultado deseado.

4.2.2 Algoritmo de los Cubos Marchantes

Sea un campo escalar $f(x, y, z)$ y un valor umbral v se desea encontrar la isosuperficie donde $f(x, y, z) = v$. Esto se puede lograr con el algoritmo de los cubos marchantes propuesto por Lorensen y Cline en [20].

Supongamos una topología voxelizada. El algoritmo consiste en verificar los valores de los vértices de cada celda. En función del valor de estos respecto al umbral v se define cuales pertenecen al interior y exterior del volumen generado por la isosuperficie. En conocimiento de esto se pueden definir las posibilidades en que la superficie atraviesa la celda.

Cada vértice tiene sólo 2 opciones (estar dentro o fuera), entonces en celda de 8 nodos se generan $2^8 = 256$ posibilidades. Usando simetrías de rotación y reflexión se puede reducir a 15 casos.

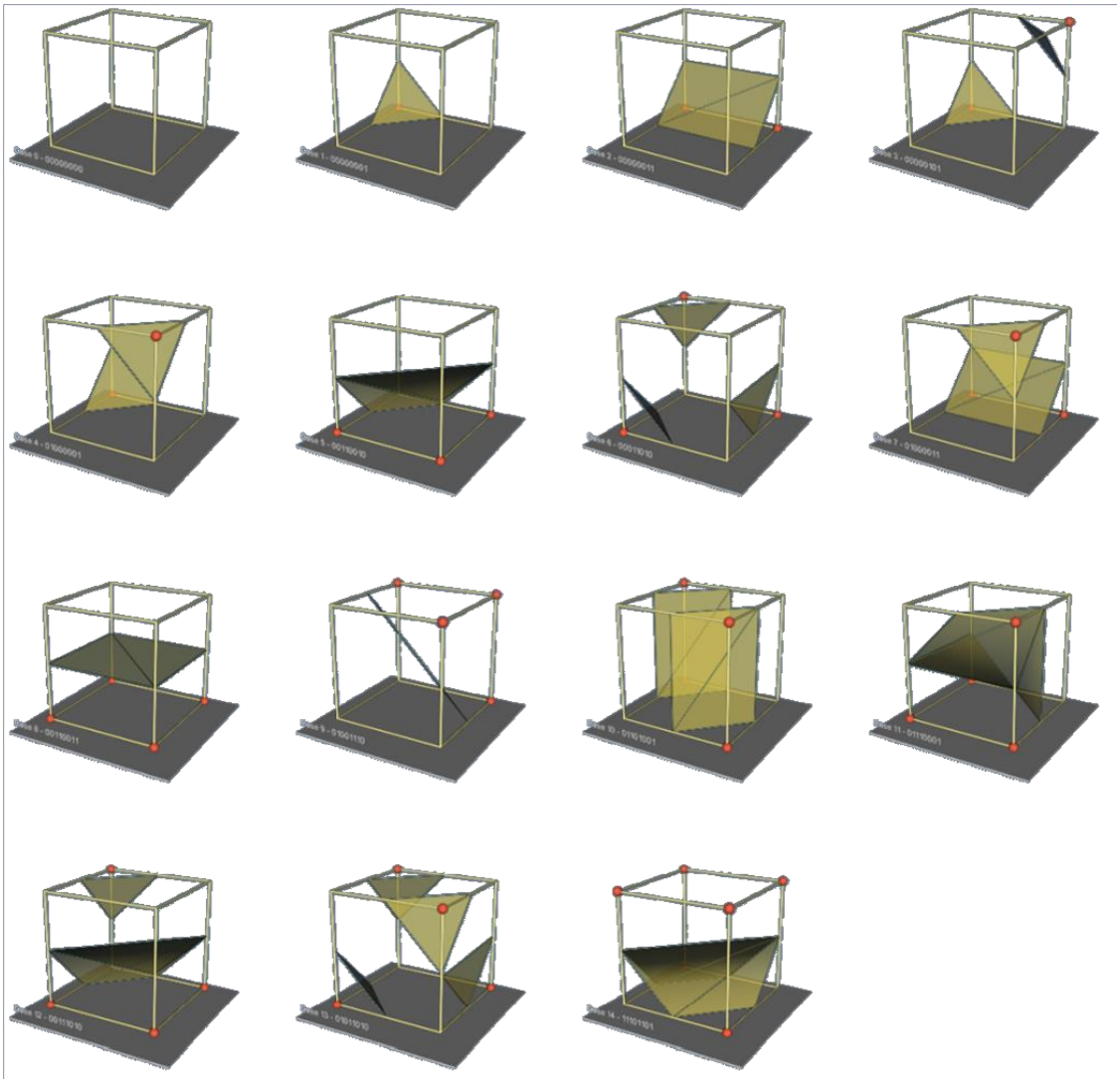


figura 4.4: 15 casos generalizados del estado topológico de la celda.
Los puntos en rojo indican los vértices al interior del volumen de la isosuperficie.
fuente: Figura 6.6 de [19].

De esta manera el algoritmo se resume en:

1. Seleccionar la celda.
2. Identificar que nodos de la celda están dentro y fuera.
3. Crear el índice a partir del vector binario de estado de los vértices de la celda.
4. Buscar el estado topológico de la celda con el índice en la tabla look-up de casos.
5. Interpolar la posición de los puntos de corte linealmente con los valores de los nodos.

Terminado el cómputo en la celda se prosigue con la siguiente hasta que todas las celdas son analizadas.

Como se puede observar, en algunos casos generalizados se pueden ocasionar ambigüedades del contorno al invertir la superficie o los puntos pertenecientes. Estas ambigüedades si no se seleccionan correctamente pueden generar superficies con agujeros no esperados. Para remediar esto Nielson y Hamann desarrollaron un algoritmo que decide la conexión de la superficie ambigua "*Asymptotic decider*" agregando 6 casos complementarios a los 15 generalizados [19].

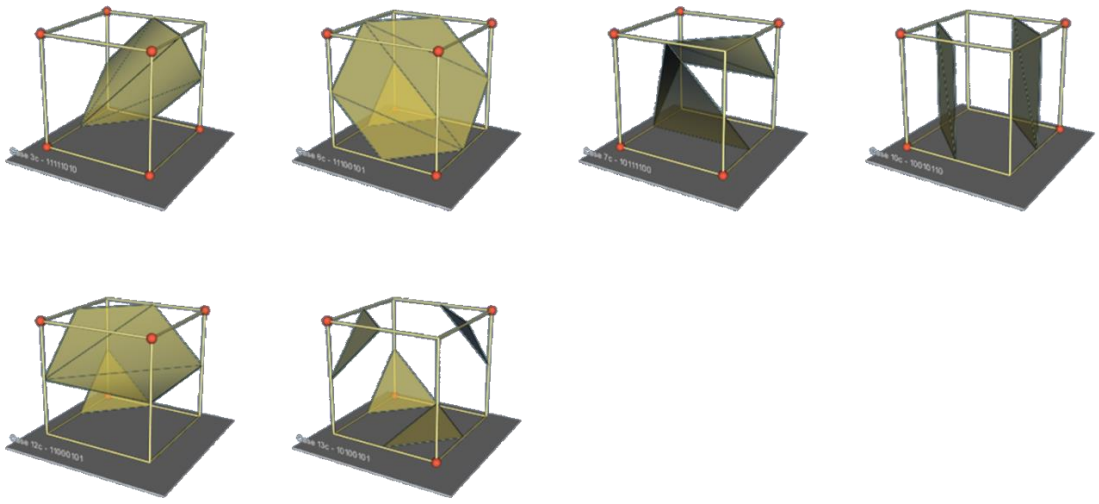


figura 4.5: Casos generalizados complementarios del algoritmo Asymptotic decider.
Fuente: figura 6.10 de [19].

Estudiando los casos de corte posibles y cómo manejar ambigüedades se puede extrapolar el algoritmo a cualquier topología de celda (tetraedros, cuña, pirámide, hexaedro, etc).

4.2.3 Triangulación

Los algoritmos de triangulación construyen la topología de un set de puntos. Estos construyen estructuras con topología triangular (caso 2D o superficie 3D) o tetraedros (Volumen 3D). Uno de los algoritmos de triangulación más populares es la Triangulación Delaunay.

El algoritmo tiene la característica de que para una celda n-dimensional contiene n+1 nodos en su interior y son exactamente los vértices de la celda, de esta manera no se produce superposición de celdas ni existen regiones vacías. Otra característica importante es que en el caso 2D está demostrado que es la triangulación óptima para un set de puntos dados, y de manera general los ángulos interiores de cada triángulo de Delaunay son mayores o iguales al mínimo ángulo de todas las triangulaciones posibles [19]. Los triángulos de Delaunay se construyen a partir de los 3 puntos más cercanos para cada punto. En el caso 3D la celda tetraédrica se construye con los 4 puntos más cercanos no coplanares.

Otro Algoritmo de Construcción topológica conocido es la teselación Dirichlet, también conocida como teselación Voronoi. Para cada punto p_i del set de datos, la celda de Voronoi corresponde al lugar geométrico (x, y, z) donde la distancia a p_i es menor que a cualquier otro punto del set de datos.

La Teselación de Voronoi y la Triangulación de Delaunay están directamente relacionadas, pudiéndose construir una a partir de la otra. Los centros de la esfera que inscribe un triángulo Delaunay corresponden a los vértices de las celdas de Voronoi.

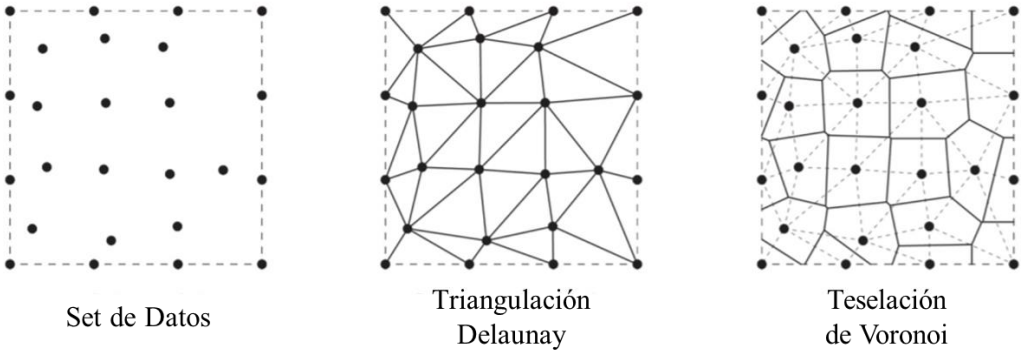


figura 4.6: Triangulación Delaunay y Teselación Voronoi para un mismo set de datos.

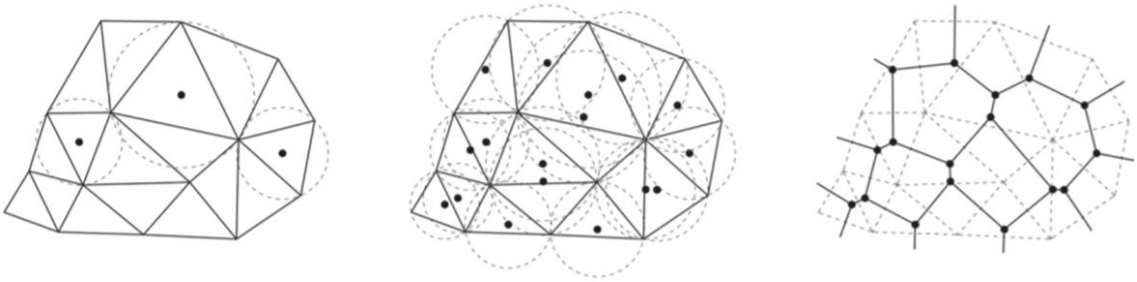


figura 4.7: Relación entre triangulación Delaunay y teselación de Voronoi.

4.3 VTK

Visualization ToolKit (vtk) es un sistema computacional basado en programación orientada al objeto, desarrollado por Kitware en C++ y con soporte para lenguajes interpretados como Java y Python, donde se aprovecha mucho mejor el potencial de esta herramienta.

El principio de funcionamiento del sistema (como tal, orientado al objeto) consiste en seccionar las complejas tareas a realizar en partes más pequeñas y simples denominadas objetos, lo que permite identificar de manera más fácil las abstracciones de cada objeto y el comportamiento de estos. De esta manera los datos y las operaciones que se pueden realizar con estos se incluyen dentro del mismo objeto.

Otra cualidad de la programación orientada al objeto es la heredabilidad. Se pueden definir clasificaciones y subclasificaciones, de esta manera un objeto que pertenece a una subclasificación de otro compartirá las mismas cualidades y metodologías que los del objeto base, pero no al revés, lo que permite modificar algún método sin afectar a las clasificaciones padres. Esto facilita el posterior desarrollo y extensión de la variedad de objetos del sistema [19].

4.3.1 Objetos Data Set

La estructura de Set de Datos en vtk son objetos compuestos de una estructura organizante (geometría y topología) y los datos atribuidos a cada punto. El objeto es vtkDataSet.

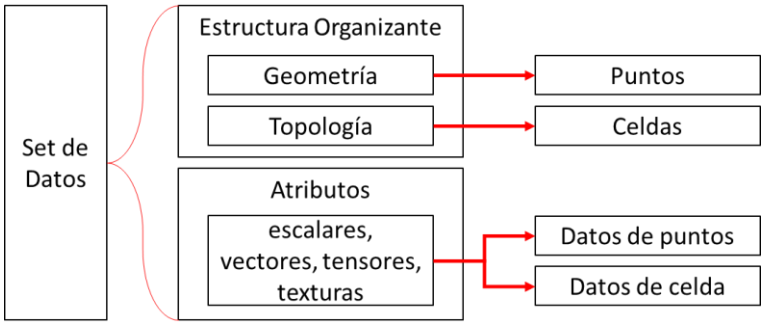


figura 4.8: Objeto de Set de Datos de vtk.

A su vez, los puntos y las celdas también son objetos de vtk (vtkPoints y vtkCell respectivamente). Un punto es un arreglo de 2 o 3 escalares (Caso 2D y 3D respectivamente) y la celda es un arreglo de objetos (puntos) que la conforman.

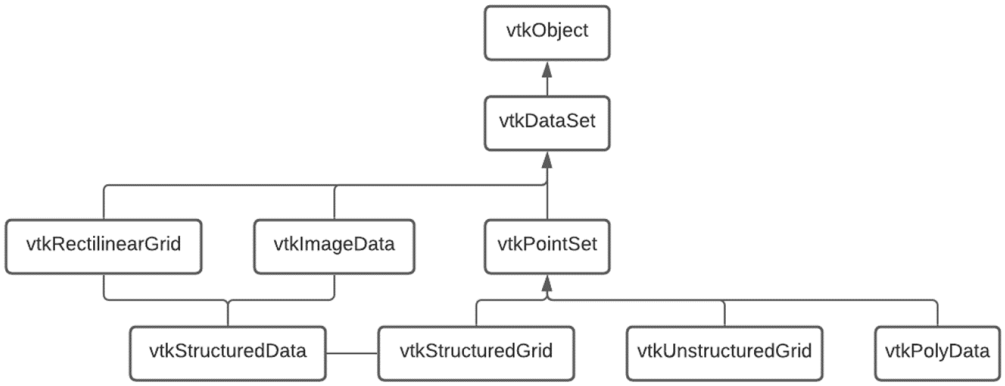


figura 4.9: Diagrama del Objeto DataSet.
Fuente: figura 5.14 de [19].

Los datos atribuidos por otro lado están conformados por la clase padre vtkFieldData, de la cual se desprenden posteriormente las clases vtkPointData y vtkCellData. Ambas clases contienen tuplas que representan escalares (1 dimensión), vectores (2 o 3 dimensiones), tensores de 2x2 (4 componentes) y tensores de 3x3 (9 componentes).

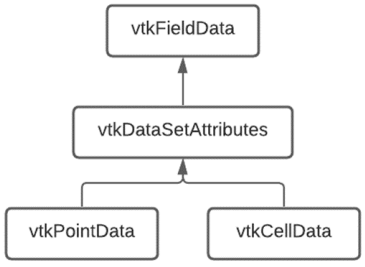


figura 4.10: Diagrama de heredabilidad de datos atribuidos del Set de Datos.
Fuente: figura 5.16 de [19].

4.3.2 Objetos de Proceso

En vtk se distinguen 3 objetos de procesos: Source, Filtros y Mappers. La principal diferencia entre estos es la cantidad de inputs y outputs que tiene cada uno.

Los Objetos de **Source** tienen sólo generan objetos como outputs. Establecen una interfase con datos externos al sistema vtk y/o generan parámetros locales. En el caso de los que establecen parámetros locales se denominan Objetos de procedimiento. Por otro lado, los Objetos Source que establecen una interfase externa distinguen entre lectores (reader) y canales o puertos de datos o fuentes de adquisición de datos externa.

Los **Filtros** por su parte requieren al menos un objeto de datos como input y generan al menos un objeto de datos como output. Se encargan de controlar y realizar las operaciones del proceso a través de transformaciones y algoritmos generalmente.

Por otro lado, los **Mapper** son objetos que terminan el proceso estableciendo la representación final. Ejemplo de estos son escalas de color, texturas, etc. A partir de un campo escalar, vectorial o tensorial.

4.3.3 Modelo Gráfico

Para Visualizar los set de Datos en bruto o procesados a través de Filtros y Mappers, vtk utiliza 7 objetos básicos para renderizar una escena. Existen muchos más con los que se puede personalizar o mejorar la visualización.

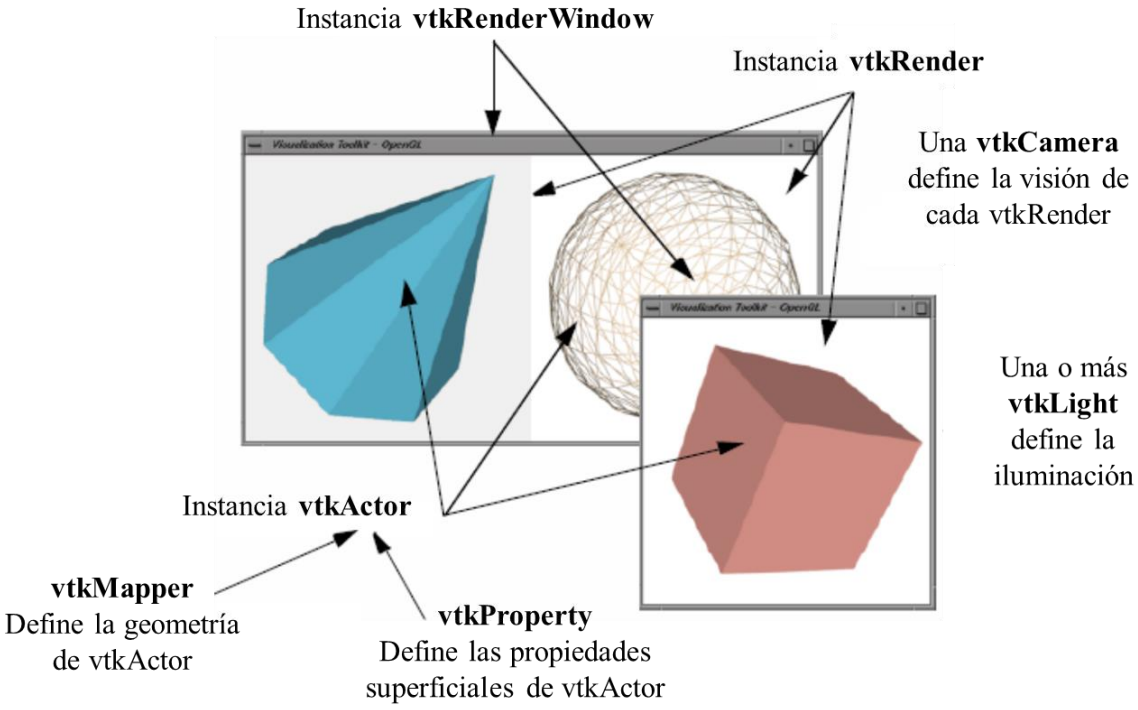


figura 4.11: Objetos básicos involucrados en el modelo gráfico de vtk.

vtkMapper y vtkProperty son objetos Mapper que sirven como input al objeto vtkActor. Este último corresponde a la identificación de cada objeto que se quiere visualizar, incluyendo su geometría, características y localización en el sistema de referencia. Cabe destacar que vtkActor es una subclase de vtkProp[1]. Para cada tipo de datos y propiedades existen vtkMapper y vtkProperty compatibles e incompatibles, por lo que se debe usar uno que corresponda a la estructura de los datos a procesar.

[1] Hay que destacar que vtkProp y vtkProperty son clases diferentes.

vtkRender es un objeto que describe una instancia a la que se adhieren actores para su posterior representación. Consiste en la escena que se va a representar.

vtkCamera y vtkLight definen el punto de vista y la iluminación de la escena a ver en la ventana inicializada por vtkRenderWindow.

En el caso de utilizar un lenguaje interpretado como Python se debe incluir también el objeto vtkRenderWindowInteractor() que permite interactuar con la escena. Si no se incluye este objeto, ocasiona un error que cierra la ventana de renderización y Python.

Un pseudo código para la renderización de una escena con los objetos Obj1 y Obj2 se aprecia a continuación:

```
1. # Mappers
2. Inicializar y configurar Mapper 1 → vtkMapper() → .SetInput(Obj1)
3. Inicializar y configurar Mapper 2 → vtkMapper() → .SetInput(Obj2)
4. # Actores
5. Inicializar y configurar Actor 1 → vtkActor()
6. Inicializar y configurar Actor 2 → vtkActor()
7. Act1.SetMapper(Mapper1)
8. Act1.GetProperty().SetProperty_N(Property N)
9. ::
10. Act2.SetMapper(Mapper2)
11. Act2.GetProperty().SetProperty_N(Property N)
12. ::
13. # Render y RenderWindow
14. Inicializar Render → vtkRender()
15. Inicializar Render Window → vtkRenderWindow()
16. Render.AddActor(ACt1)
17. Render.AddActor(ACt2)
18. # Interactor
19. Inicializar Estilo → vtkInteractorStyleSwitch ()
20. Inicializar Interactor → vtkRenderWindowInteractor()
21. Interactor.SetRenderWindow(RenderWindow)
22. Interactor.SetInteractorStyle(Estilo)
23. # Visualizar
24. RenderWindow.Render()
25. Interactor.Start()
```

5 Segmentación de mallas

La segmentación de una malla (2D o 3D) es el proceso de dividir la geometría de esta en diferentes regiones con sus respectivas etiquetas. Matemáticamente hablando la segmentación no tiene una definición específica. De aquí el problema reside en la definición requerida de segmentación, que dependerá del uso de esta, así como de las propiedades y funcionalidad de los componentes de la geometría a segmentar.

Se han desarrollado diferentes algoritmos de segmentación, haciendo uso de diferentes herramientas. Algunas de las clasificaciones de estos métodos se dan en [21]:

- **Según Objetivo (Superficie o Parte):**
  Una forma de clasificar un método de segmentación es según si esta distingue entre superficies o volúmenes. De manera gráfica en la

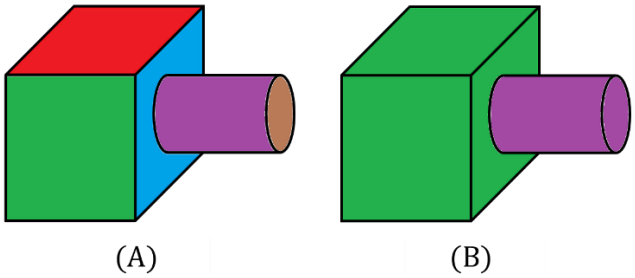


figura 5.1: Segmentación según Superficie o Parte.
(A) Segmentación de superficies. (B) Segmentación de Partes.

- **Según Criterio Geométrico o Bordes:**
  Los algoritmos basados en criterios de región computan la segmentación en base a la uniformidad de parámetros geométricos como largo, diámetro, etc. Por otro lado, los criterios basados en La información de borde segmentan la geometría a partir de curvaturas agudas.

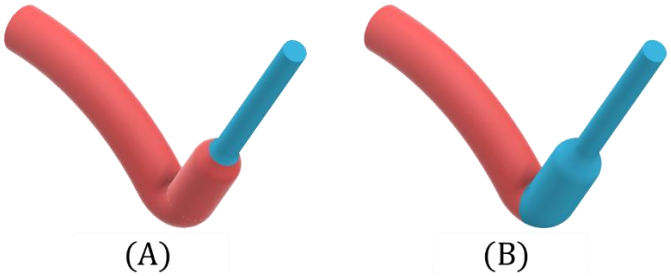


figura 5.2: Segmentación según criterio geométrico.
(A) Segmentación según criterios de región. (B) Segmentación basada en bordes

- **Según Grado de aprendizaje (basados en redes neuronales):**

  Para los métodos basados en redes neuronales, los métodos supervisados se entrenan con data sets previamente segmentados. El caso de los semi-supervisados, estas se entrenan con geometrías previamente segmentadas y otras en bruto. En el caso de los métodos no supervisados alude a las redes neuronales entrenadas con poca (o nula) información.

- **Según Participación del Usuario:**

  En la mayoría de los algoritmos de segmentación se requiere al menos un input. En el caso de los algoritmos con participación del usuario utilizan información proporcionada por el usuario (selección manual de algún subproceso, indicación de un punto como condición de borde, etc.). En cambio, los algoritmos automáticos no requieren interacción del usuario para generar la segmentación de la geometría.

- **Según Cantidad de Inputs:**

  Algunos criterios solo utilizan un input como dato, y a partir de este computan los criterios y la selección de las etiquetas en la segmentación. Existen otros algoritmos que utilizan más de un input, generando diferentes segmentaciones y estableciendo la definitiva correlacionando los resultados obtenidos.

- **Según información topológica o geométrica:**

  Similar a la primera clasificación, los algoritmos con información geométrica caracterizan una región en base a la uniformidad de su geometría. Por otro lado, los algoritmos conformados en información topológica , que incluye uniformidad geométrica, así como alguna clasificación adicional (por ejemplo, identificar tipo de componente).

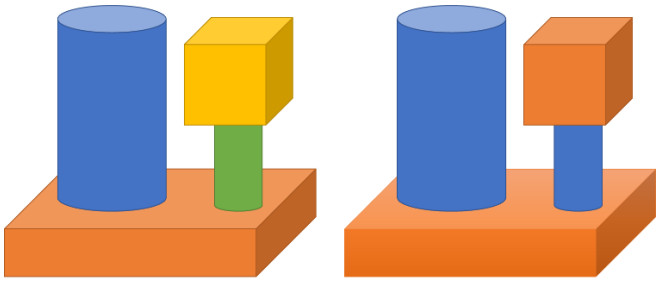


figura 5.3: Segmentación según información topológica o geométrica. (A) Información geométrica. (B) Información Topológica

- **Según Subclasificación:**

  Algunos métodos de segmentación tienen la capacidad de utilizar la jerarquía estructural de la geometría. De esta manera se asemeja a la segmentación natural de un ser humano, donde se subdivide una geometría en conjuntos, luego esos conjuntos en subconjuntos y sucesivamente. El algoritmo termina cuando se obtiene el nivel de segmentación deseado.

5.1.1 Algoritmo de Segmentación de Ramificaciones

El algoritmo propuesto por [1] trata de un método de segmentación para geometrías relativamente uniformes tubulares con bifurcaciones como vasos sanguíneos. En el caso de aneurismas saculares también tiene sentido el uso de este algoritmo.

Primero se define la centerline como una línea al interior de una geometría "tubular" que maximiza la distancia de esta a cualquier punto de la superficie. Formalmente, la centerline $c(\tau)$ parametrizada por el largo de arco $\tau \in [0, L]$ entre los puntos p_1 y p_2 está definido según Antiga et. al. [22] por:

$$c(\tau) = \arg \min_{\gamma \in MA(\Omega)} \left\{ \int_{\gamma^{-1}(p_1)=0}^{\gamma^{-1}(p_2)=L} \frac{d\tau}{R[\gamma(\tau)]} \right\} \tag{5.1}$$

Donde $\Omega \subseteq \mathbb{R}^3$es la geometría, $MA(\Omega)$ es el eje medio (también conocido como esqueleto topológico), γ es el camino entre los puntos p_1 y p_2 incluidos en $MA(\Omega)$ y $R[\mathbf{x}]$ el radio de la esfera más grande inscrita en la superficie con centro en $MA(\Omega)$.

Luego de la definición de las centerlines en la geometría se procede a definir el sistema de referencia de la bifurcación. Para esto en [1] se definen 2 puntos en cada centerline. A modo de ejemplo, en el caso de un vaso sanguíneo con 1 bifurcación, el punto C_1^A , perteneciente a la centerline $\tilde{\boldsymbol{c}}_1(\tau)$, se localiza en la intersección de esta con la superficie tubular correspondiente a la centerline $\tilde{\boldsymbol{c}}_2(\tau)$.

Una vez definidos los puntos de referencia, se posiciona un punto $\boldsymbol{x_o}$ en el centroide de las superficies de las esferas más grandes inscritas en los puntos de referencia. Análogo al cálculo de un centro de masas, con los radios de las esferas se puede obtener $\boldsymbol{x_0}$:

$$\mathrm{x}_0 = \frac{\sum_{ij}\left(r_i^j\right)^2 \vec{c}_{ij}}{\sum_{ij}\left(r_i^j\right)^2} \qquad \begin{matrix} i = \{1,2\} \\ j = \{A, B\} \end{matrix} \tag{5.2}$$

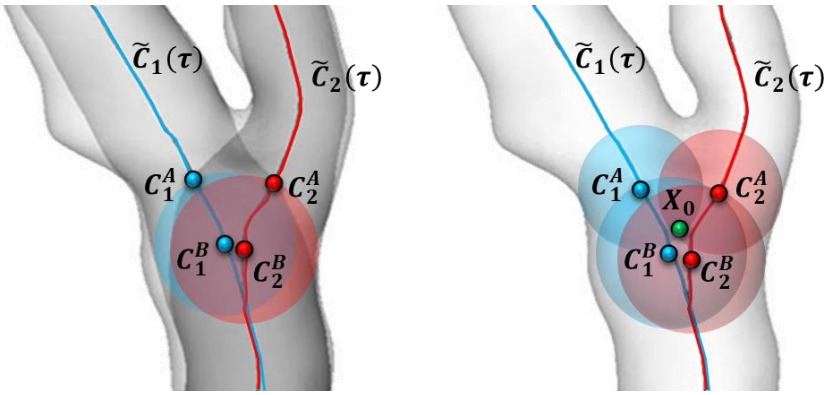


figura 5.4: Identificación de puntos de referencia.
Fuente: Modificación de figura 2 de [1].

La ventaja de definir el punto $\boldsymbol{x_0}$ en el centroide de estas esferas es que permite un ajuste automático del centro de segmentación para geometrías con bifurcaciones con diámetros muy diferentes.

El esquema descompone luego cada centerline $\tilde{\boldsymbol{c}}_{\boldsymbol{i}}(\tau)$ en los tramos $\tau \in [0, \tau_i^B]$ y $\tau \in [\tau_i^A, L]$ donde la brecha $\tau \in [\tau_i^B, \tau_i^A]$ es simplemente descartada. Se llamará a estos nuevos segmentos de centerlines ($\tilde{g}_i(\tau_i)$).

La región de cada ramificación $\psi_i \subset \Omega \subseteq \mathbb{R}^3$ se compone de todos los puntos $\vec{x}_i \in \mathbb{R}^3$que están mas cerca de $\tilde{g}_i(\tau_i)$ que de cualquier otro $\tilde{g}_j(\tau_i)$ [1]. Esto es:

$$\psi_i = \left\{ x \in \mathbb{R}^3 : |x - \tilde{g}_i(\tau_i)|^2 \leq \left| x - \tilde{g}_j(\tau_j) \right|^2, \qquad \forall j \neq i \right\} \tag{5.3}$$

Finalmente, la descomposición de la geometría Ω en ramificaciones está dada por intersecciones con ψ_i. Particularmente la ramificación $\Omega_i = \Omega \cap \psi_\mathrm{i}$. De esta manera la superficie de la ramificación se define como:

$$\partial\Omega_\mathrm{i} = \partial\Omega \cap \partial\psi_\mathrm{i} \tag{5.4}$$

En la figura 5.5 se puede observar el resultado del algoritmo en casos con bifurcaciones con diámetros similares y con diámetros diferentes.

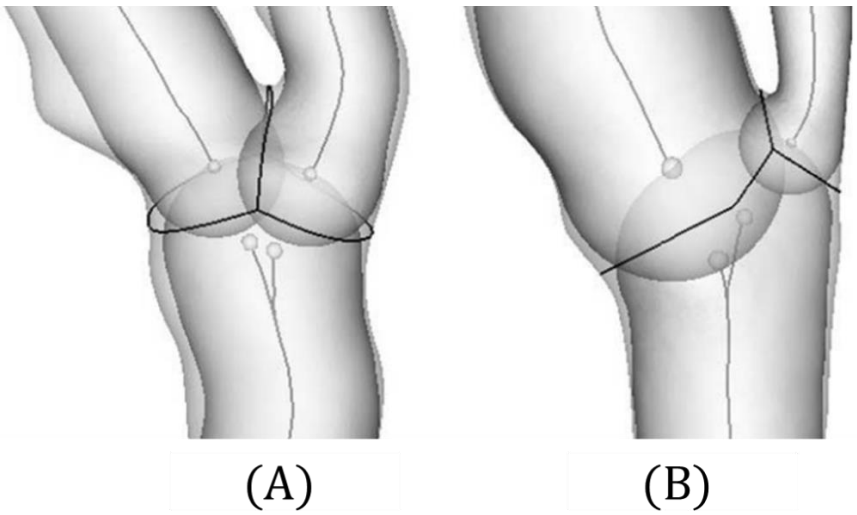


figura 5.5: Descomposición de las ramificaciones propuesta por [1].
(A) Caso con diámetros similares de cada bifurcación. (B) Caso en que una de las bifurcaciones tiene un diámetro considerablemente menor a las otras 2. Fuente: figura 3 de [1].

5.1.2 Segmentación de mallas con campos dependientes de la concavidad

Un buen objetivo en segmentación de mallas es estudiar las propiedades topológicas de la superficie de la geometría a segmentar. Un buen parámetro es utilizar información acerca de la concavidad de una región. Kin-Chung Au et al. en [23] se propone el uso del inverso de la curvatura gaussiana en cada punto para computar un campo dependiente de la concavidad de la superficie, donde se pueden utilizar las isolíneas de este campo para realizar la segmentación.

Se define el campo de segmentación como el campo harmónico de la superficie que se puede obtener a partir de resolver la ecuación de Laplace en la superficie de la geometría.

$$\Delta\phi = \frac{\partial^2}{\partial x^2}\phi + \frac{\partial^2}{\partial y^2}\phi + \frac{\partial^2}{\partial z^2}\phi = 0 \tag{5.5}$$

Discretizando para obtener el valor del campo en cada nodo de la superficie, el campo se puede encontrar a partir del sistema lineal de ecuaciones matriciales:

$$\begin{bmatrix} \boldsymbol{L} \\ \boldsymbol{C} \end{bmatrix} \phi = \begin{bmatrix} 0 \\ \boldsymbol{B} \end{bmatrix} \tag{5.6}$$

Donde **B** y **C** corresponden al vector y matriz de condiciones de borde y **L** es la Matriz Laplaciana que representa la conectividad e influencias de un nodo a otro de la malla. La matriz **L** se define en [23] como:

$$L_{ij} = \begin{cases} -1 & Si\ i = j \\ \dfrac{\omega_{ij}}{\sum_{k \in Ni} \omega_{ik}} & Si\ j \in Ni \\ 0 & \sim \end{cases} \tag{5.7}$$

Acá $(k \in Ni)$ es el conjunto de nodos que son vecinos del nodo i, y ω_{ij} es un peso dependiente de la concavidad que denota la influencia del nodo j sobre el nodo i de la malla, definido por la expresión:

$$\omega_{ij} = \frac{\left|e_{ij}\right| \cdot \beta_{ij}}{G_{ij} + \epsilon} \tag{5.8}$$

En la ecuación $\left|e_{ij}\right|$ es la longitud de la arista que une el nodo i con el nodo j, G_{ij} es la suma de la curvatura gaussiana absoluta en los vértices v_i y v_j, ϵ es una pequeña constante para prevenir la división por cero y β_{ij} es el peso de concavidad de la arista e_{ij}. Este último es una pequeña constante muy importante para definir bien el campo armónico. Una definición útil de β_{ij} es:

$$\beta_{ij} = \begin{cases} 0.01 & Si\ v_i\ o\ v_j\ son\ cóncavos \\ 1 & Cualquier\ otro\ caso \end{cases} \tag{5.9}$$

Como se está trabajando en un ambiente discreto, se define que un vértice v_i es discreto si existe algún vértice $v_j \in Ni$ que cumpla:

$$\frac{\left(\vec{v}_i - \vec{v}_j\right)}{\left\|\vec{v}_i - \vec{v}_j\right\|} \cdot \left(\hat{n}_j - \hat{n}_i\right) > 0 \tag{5.10}$$

Donde $\hat{n}_i$ es la normal del nodo i.

El sistema de la ecuación (5.6) puede ser resuelto usando el solver de gradiente conjugado..

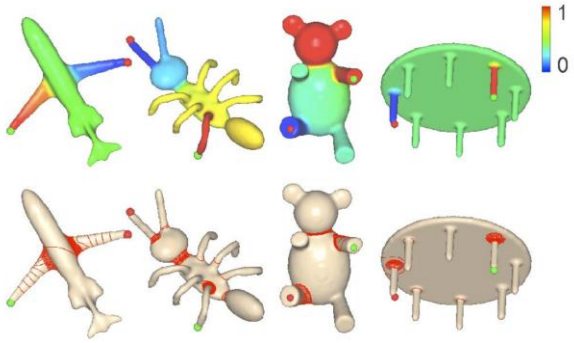


figura 5.6: Campo de segmentación respecto a la concavidad de [23].
Los puntos verde y rojo corresponden a condiciones de borde Dirichlet con valor 1 y 0 respectivamente.

Un año más tarde, Jiang y Strother [5] proponen una modificación y aplicación a aneurismas cerebrales del algoritmo de segmentación de campos dependientes de la concavidad, utilizando parámetros dependientes de las centerlines y el diagrama de Voronoi interno asociado a las centerlines.

Dada una superficie triangulada $\boldsymbol{ST}$, el diagrama de Voronoi interno $\boldsymbol{VD}$ se construye a partir de las esferas más grandes inscritas en la superficie. De esta manera para cada punto $p_i \in \boldsymbol{ST}$ se identifica el $polo(p_i)$ como el vértice más lejano al punto p_i dentro de la celda que lo contiene en el $\boldsymbol{VD}$ interno [22].

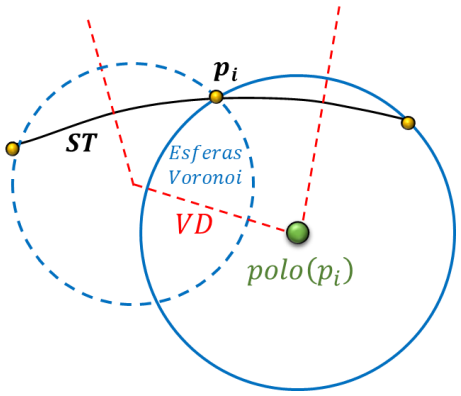


figura 5.7: Ejemplo de $polo(p_i)$ a partir de un VD en 2D

Se define la Distancia de Desviación Normalizada ($\boldsymbol{NDD}$) para cada punto $p_i \in \boldsymbol{ST}$ como:

$$NDD(p_i) = \frac{D[polo(p_i)]}{R[p_i]} \tag{5.11}$$

Donde $D[polo(p_i)]$ corresponde a la distancia desde $polo(p_i)$ a la centerline, y $R[p_i]$ es el radio de la sección transversal que contiene a p_i.

El parámetro $\boldsymbol{NDD}$ mide la uniformidad tubular de la geometría. Por ejemplo, si el radio de esta cambia suavemente el valor de $\boldsymbol{NDD}$ no se ve mayormente afectado, no así, en el caso de protuberancias o geometrías saculares a lo largo del vaso. Esto se puede apreciar gráficamente en la figura 5.8.

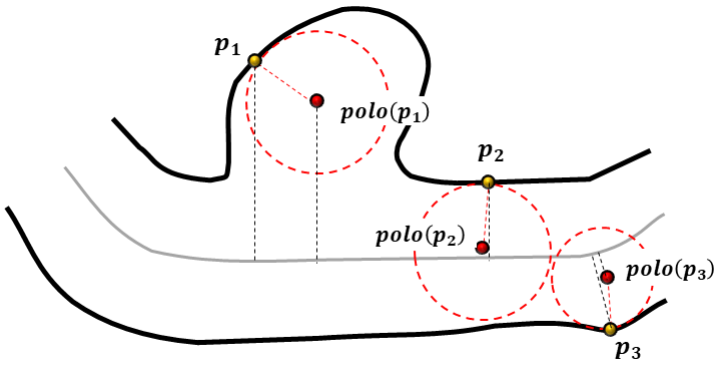


figura 5.8: Ejemplos de localización de polos respecto al punto y la geometría.

Con $\boldsymbol{NDD}$ definido para cada punto de la superficie $\boldsymbol{ST}$, Jiang y Strother modifican la ecuación (5.8):

$$\omega_{ij} = \frac{|e_{ij}| \cdot \alpha_i \cdot \beta_i}{G_{ij} + \epsilon} \tag{5.12}$$

Donde α y β se definen respecto a NDD:

$$\alpha = \begin{cases} 10^{-5} & Si\ t_1 \leq NDD \leq t_2\ \&\ v_i\ es\ cóncavo \\ 10^{-3} & Si\ t_1 \leq NDD \leq t_2\ \ solamente \\ 1 & En\ otro\ caso \end{cases} \tag{5.13}$$

$$\beta = \begin{cases} 0.01 & Si\ v_i\ es\ cóncavo \\ 1 & Si\ no \end{cases} \tag{5.14}$$

Al igual que en el algoritmo del campo dependiente de la concavidad, se deben incluir condiciones Dirichlet en el aneurisma ($\phi = 1$) y en el vaso parental ($\phi = 0$). En este método se sugieren 8 puntos de condiciones de borde, 4 en el aneurisma y 4 en el vaso parental.

Una vez resulto el sistema (5.6) restringido a las condiciones de borde, se segmenta el aneurisma del resto de la geometría usando isolíneas de ϕ.

5.2 Materiales y Métodos

Para aislar los aneurismas del resto de la geometría se utiliza como base el algoritmo de segmentación de ramificaciones e interacciones del usuario con los módulos VTK v8.1 y VMTK v1.4 programado en Python 3.6.1 para ser usado en Anaconda Prompt dentro de un ambiente con vmtk instalado.

Como input se utiliza la geometría en formato STL. Esta debe ser transformada a un objeto de vtk para poder trabajar con los módulos a utilizar. Al tratarse de una superficie 3d conviene leer el STL como PolyData, lo cual es posible haciendo uso de la clase vtkSTLReader(). Posteriormente se utiliza la clase vtkXMLPolyDataWriter() para guardar el PolyData que será usado en los algoritmos posteriores (tanto para segmentación, identificación de vórtices y análisis de resultados).

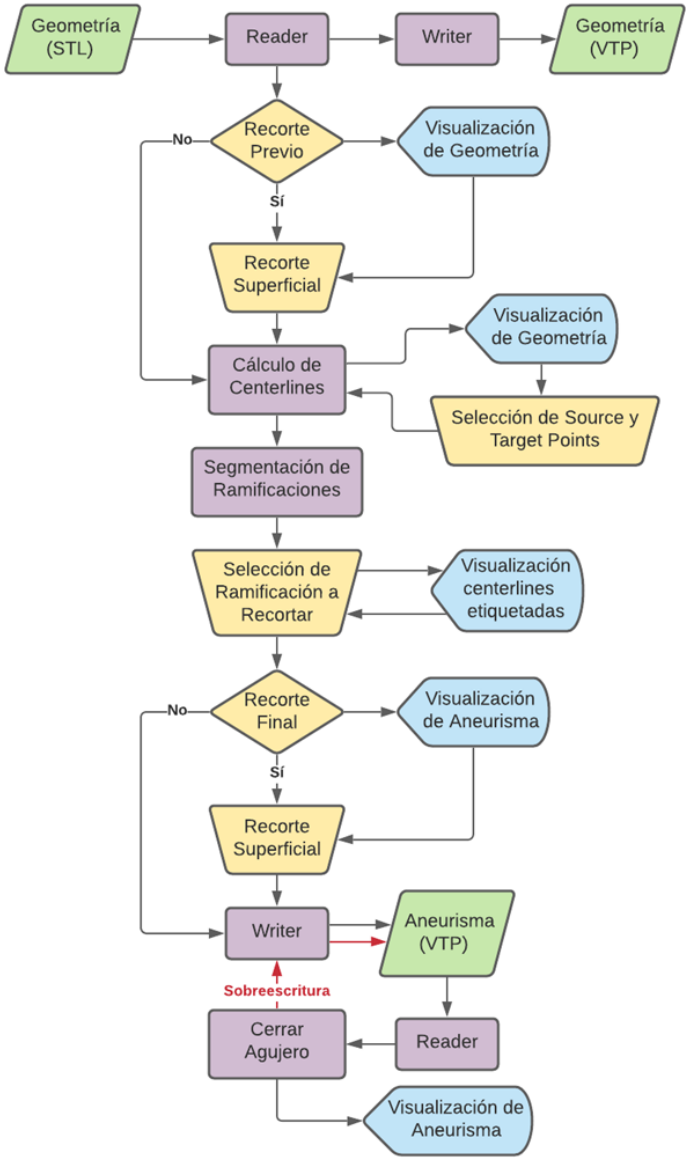


figura 5.9: Diagrama del algoritmo de segmentación creado.

5.2.1 Recorte Previo

Con el PolyData disponible, el primer paso del algoritmo es opcional, que consiste en recortar parte de la geometría al interior de un cubo interactivo definido por el usuario con el widget vmtkSurfaceClipper. Esto permite sobrellevar casos donde la entrada o salida del aneurisma están muy juntos, lo que disminuye la calidad de la segmentación.

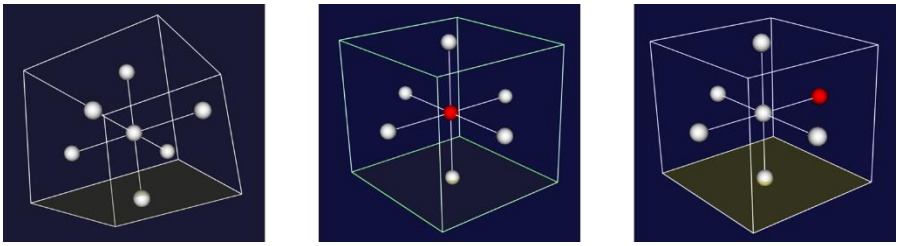


figura 5.10: Cubo de recorte generado

- Apretar “i” genera un cubo. Las caras de este son completamente invisibles salvo una que es amarilla y transparente que es muy útil cuando se quiere previsualizar la intersección del plano de corte con la geometría.
- Presionar el botón de la rueda del mouse estando sobre el cubo o hacer click izquierdo en la esfera central permite moverlo libremente.
- Con Click izquierdo en cualquier otra parte del cubo se puede rotar en las 3 direcciones.
- Al hacer click izquierdo sobre una de las esferas exteriores permite cambiar las dimensiones del cubo, empujando la cara correspondiente.
- Para realizar el corte se debe presionar “espacio”, lo que luego de realizar el corte a la superficie hace desaparecer el cubo. Se puede realizar más cortes generando más cubos
- Presionar “q” termina el algoritmo de corte y procede al siguiente.

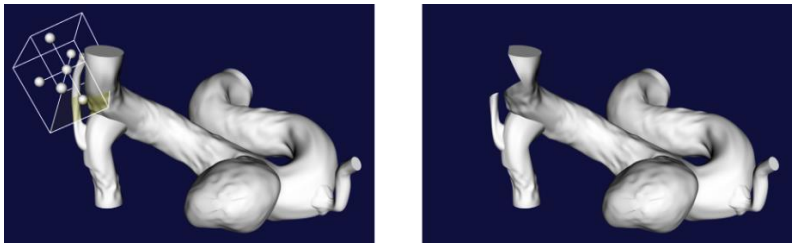
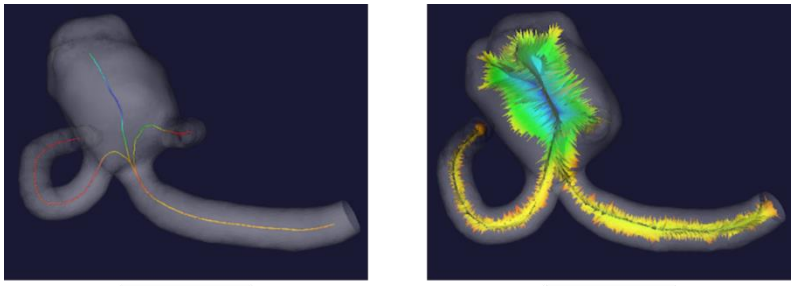


figura 5.11: Recorte de geometría con el cubo.

5.2.2 Computo de Centerlines

El cómputo de las centerlines utiliza el módulo vmtkCenterlines. Sin embargo, para que el método de segmentación de ramificaciones sea compatible con los aneurismas no se puede usar la versión automática, requiriendo que el usuario indique algunos puntos de referencia que son el source point y los target points.

Como el computo de las centerlines se puede acelerar con el diagrama de Voronoi [22], el algoritmo además de las centerlines también puede entregar el diagrama de Voronoi Interno de la geometría.



(A) (B)

figura 5.12: Visualización del PolyData de vmtkcenterlines.
(A) Visualización de Centerlines. (B) Visualización del Diagrama de Voronoi Interno.
Fuente: Tutoriales de VMTK, centerlines, página oficial.[2]

[2] http://www.vmtk.org/tutorials/Centerlines.html

El source point se selecciona preferiblemente el inlet de la geometría. En los casos donde la altura del aneurisma es muy pequeña respecto al radio de la arteria, conviene colocar el aneurisma como source point. Para otros casos donde el inlet está muy pegado al aneurisma, es complicado de cortar con el recorte previo (paso anterior) y existe sólo una salida conviene colocar el source point ahí.

5.2.3 Selección de Ramificaciones a Recortar

A partir de las centerlines, el módulo vmtkBranchExtractor realiza la segmentación propuesta por Antiga en [1] etiquetando cada segmento de la centerline. Por otro lado, existe el módulo vmtkBranchClipper que dada una geometría y sus centerlines segmentadas en ramificaciones asociada, permite elegir una etiqueta y eliminarla de la geometría. El problema es que no se conoce a priori cada etiqueta ni la cantidad de estas. Para resolverlo se realiza un --pipe intermedio con el módulo vmtkCenterlinesLabeler que dado un objeto de centerlines e indicar el array de los PointData que contiene las etiquetas cambia a voluntad estas.

Como el algoritmo de vmtkBranchClipper corta sólo la geometría asociada a 1 label para no realizar repetidas veces el recorte se repiten la etiqueta en todos los segmentos que se desean recortar. En la definición del algoritmo se asigna recortar la etiqueta 0, por lo que en el display de vmtkBranchClipper se asignará 1 a las etiquetas a mantener, en este caso el aneurisma, y 0 a todas las demás.

Finalizada la reasignación de etiquetas y el recorte de ramificaciones, se ejecuta nuevamente el módulo vmtkSurfaceClipper para eliminar algún remanente no deseado del resto de la geometría o para recortar un plano. Al igual que en el primer paso este es opcional.

Antes de terminar se muestra la geometría final, que tendrá un agujero en la región donde estaba conectada al resto de la geometría que se puede cerrar con el filtro vtkFillHolesFilter(). Como este no hace un --pipe directo con vmtk se guarda el PolyData de la geometría segmentada abierta, se lee, se filtra el PolyData y se sobreescribe el mismo PolyData guardado. Para verificar que se realizó correctamente la segmentación se vuelve a mostrar la geometría segmentada después de aplicar el filtro.

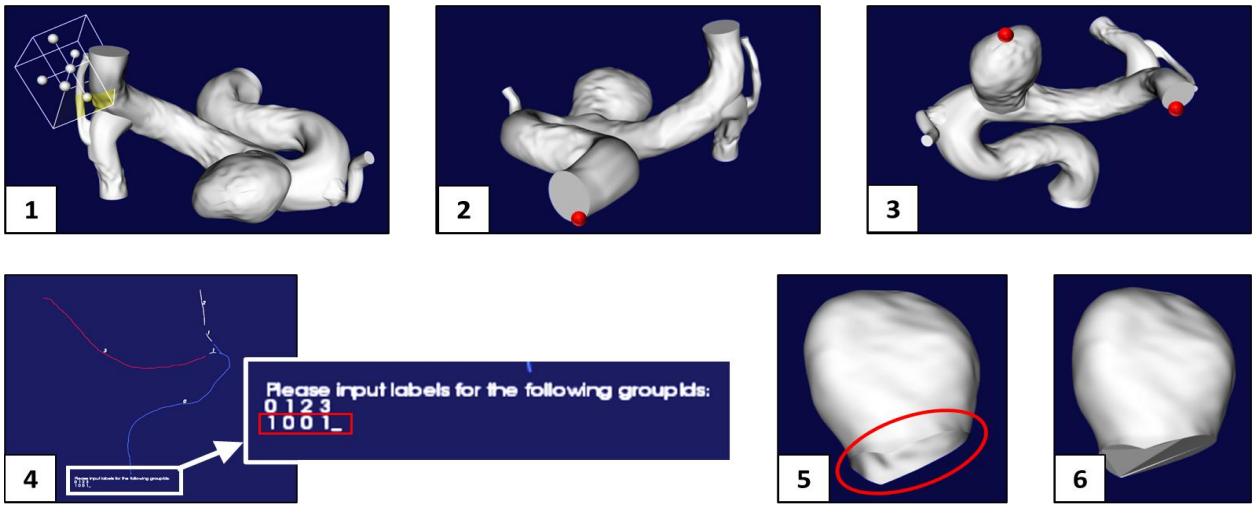


figura 5.13: Resumen de las interacciones del usuario en el algoritmo de segmentación.
(1) Recorte Inicial. (2) Selección del Source Point. (3) Selección de los Target Points. (4) Reasignación de etiquetas. (5) Visualización y Recorte final de Geometría Segmentada antes de cerrar. (6) Visualización de geometría Segmentada cerrada.

5.2.4 Implementación

El algoritmo se probó en 37 geometrías en formato STL de alta resolución, transformadas desde STP usando Autodesk Inventor 2020, con los siguientes parámetros:

- Formato ASCII
- Unidades en [m]
- Desviación máxima de superficie: 0.005%
- Desviación normal máxima : 10°
- Longitud de arista máxima : 100%
- Relación Altura/Anchura : 21.5

Cuando la geometría no entrega una segmentación exitosa en el primer intento, se procede a cambiar el orden y localización de los source y target points. Si esto no funciona se continúa limpiando mejor la geometría durante el primer paso.

Si independiente de los cambios no mejora, se deja de obtener una geometría vacía/nula, o termina de indefinirse el cómputo de las centerlines, se considera que no funcionó la segmentación.

Hay que destacar además que se debe evitar utilizar el recorte de superficies después de la segmentación de ramificaciones, salvo que se obtenga geometrías desconectadas a la geometría principal (como es el caso de la figura 5.14) .

5.3 Resultados de segmentación

A continuación, se muestran algunos casos con resultados interesantes a analizar.

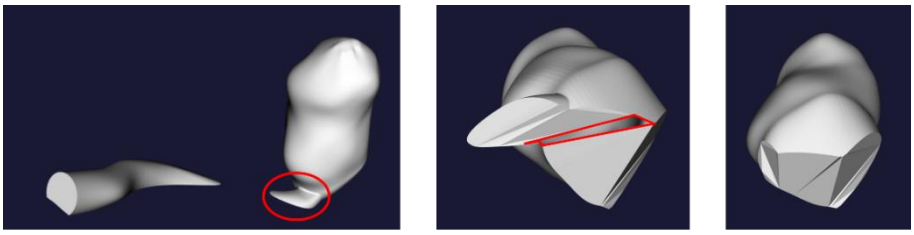


figura 5.14: Segmentación de la geometría 25.
(A) Segmentación sin eliminar los elementos indeseados. (B) Segmentación eliminando el elemento extra del caso A, sin recortar el borde del aneurisma. (C) Segmentación eliminando ambos elementos extras del caso A.

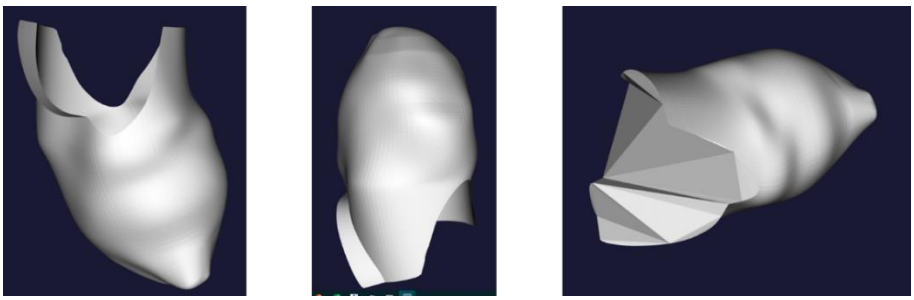


figura 5.15: Segmentación de la geometría 40.

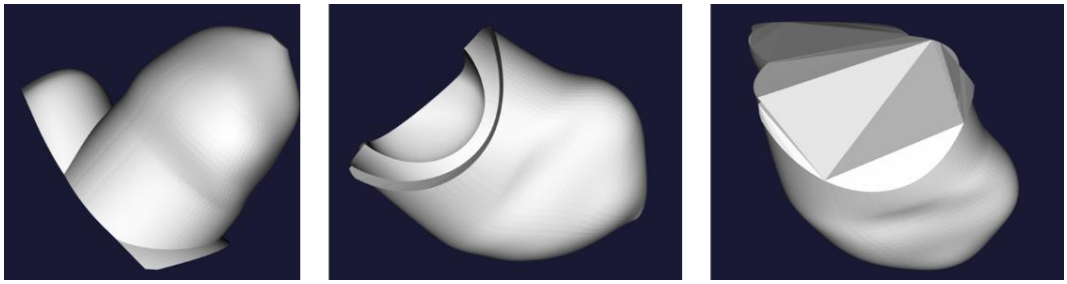


figura 5.16: Segmentación geometría 43 utilizando 1 target point en el aneurisma.

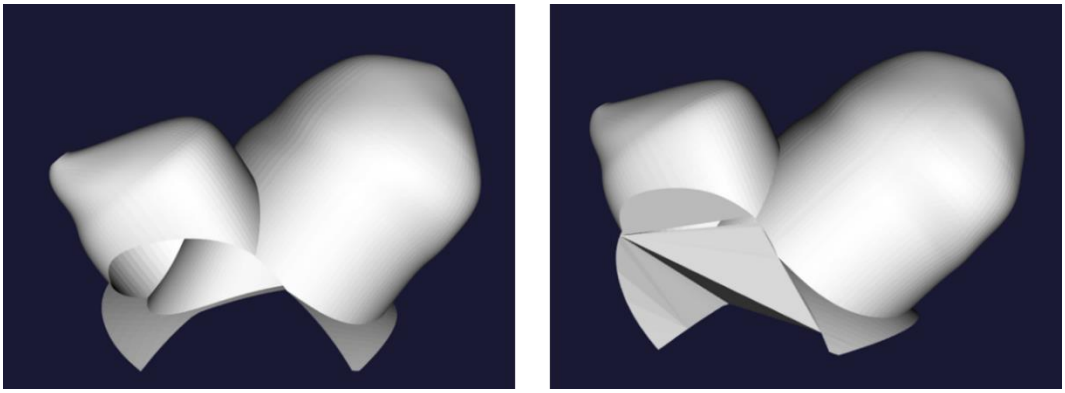


figura 5.17: Segmentación geometría 43 utilizando 2 target point en el aneurisma.

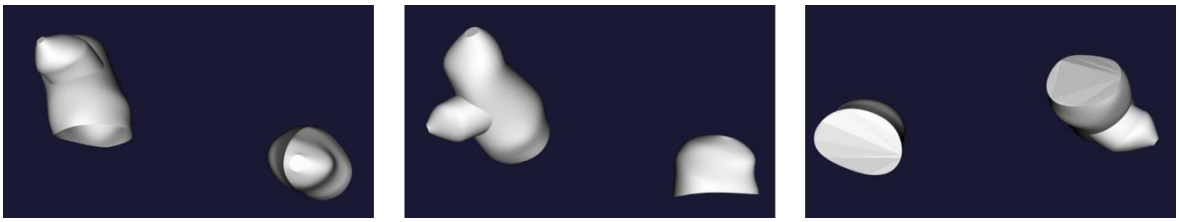


figura 5.18: Segmentación geometría 47_48.

El resumen de los resultados de segmentación:

Tabla 5.1: Resultado de segmentación de geometrías probadas.

Geometría	Segmentación
geo1	Si
geo3	Si
geo4	No se puede
geo5_31	No se puede
geo6	Si
geo7	No se puede
geo8	Si
geo9	Si
geo10	Si
geo11	Si

Geometría	Segmentación
geo24	Si
geo25	Si
geo27	No se puede
geo28	No se puede
geo30_34	Si
geo35	Si
geo38	Si
geo39	Si
geo40	Si
geo41	No se puede

Tabla 5.1 (Continuación)

Geometría	Segmentación
geo12	No se puede
geo13	No se puede
geo16	Si
geo17	No se puede
geo19	Si
geo20	Si
geo21_33	No se puede
geo22	No se puede
geo23	Si

Geometría	Segmentación
geo43	Si
geo44_45	Si
geo46	Si
geo47_48	Si
geo49	Si
geo50	Si
geo51	Si
geo52	Si

Tabla 5.2: Geometrías donde se logra la segmentación.

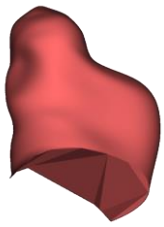
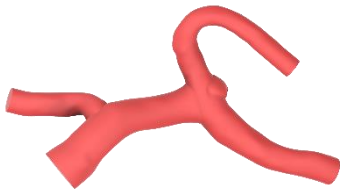
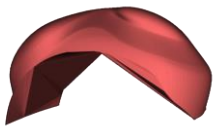
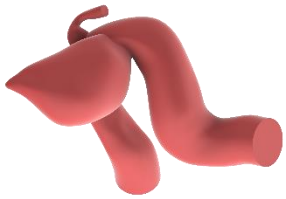
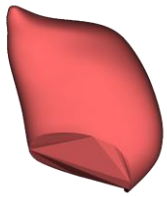
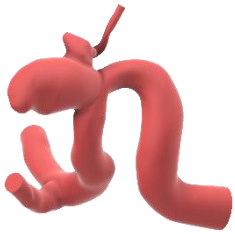
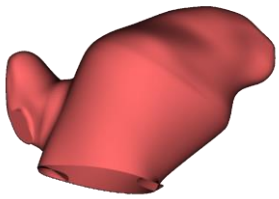
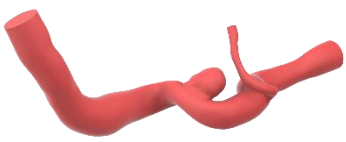
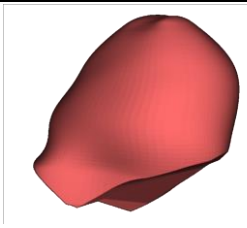
Geometría	Geometría Completa	Geometría Segmentada
geo1		
geo6		
geo8		
geo9		
geo10		

Tabla 5.2 (Continuación)

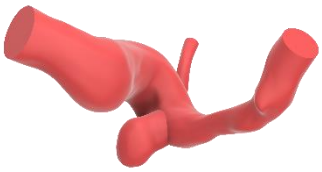
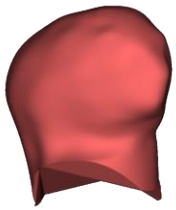
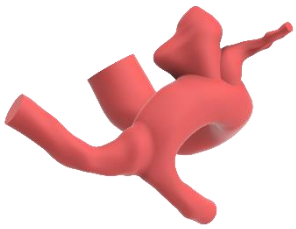
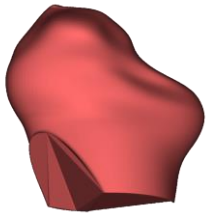
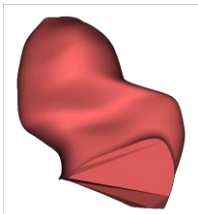
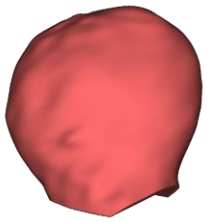
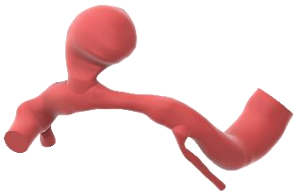
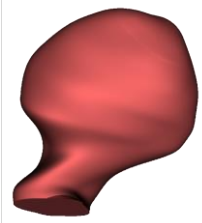
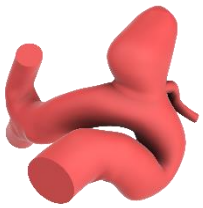
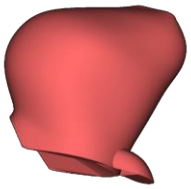
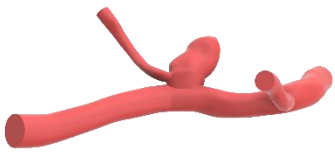
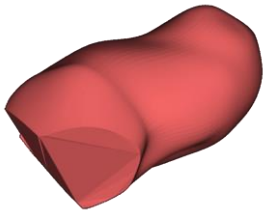
Geometría	Geometría Completa	Geometría Segmentada
geo11		
geo16		
geo19		
geo20		
geo23		
geo24		
geo25		

Tabla 5.2 (Continuación)

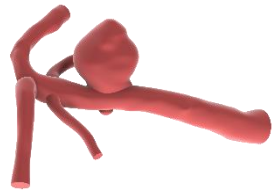
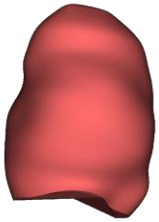
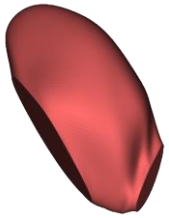
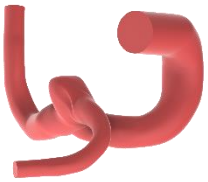
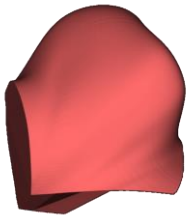
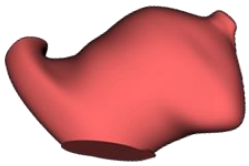
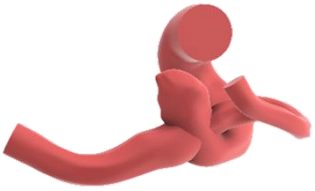
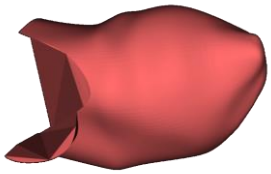
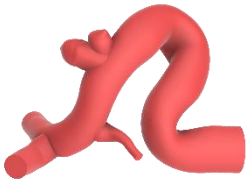
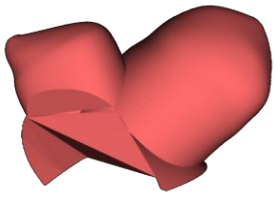
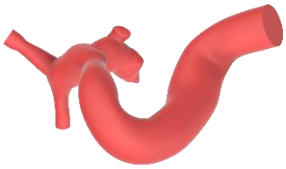
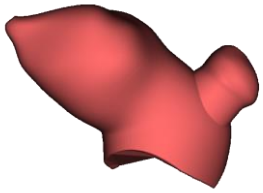
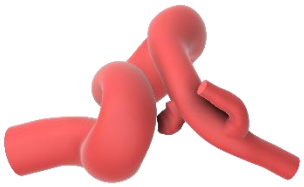
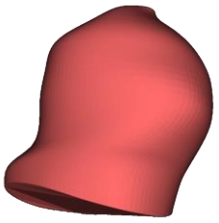
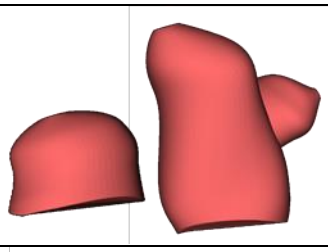
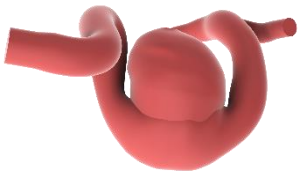
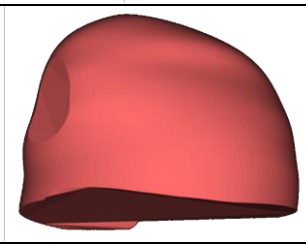
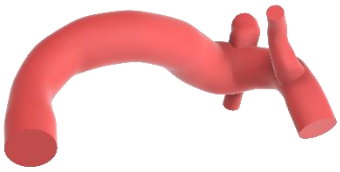
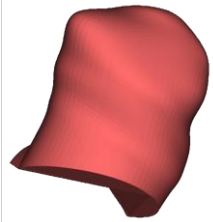
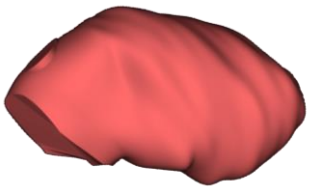
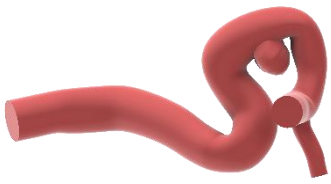
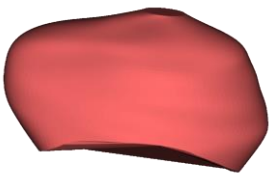
Geometría	Geometría Completa	Geometría Segmentada
geo30_34		
geo35		
geo38		
geo39		
geo40		
geo43		
geo44_45		

Tabla 5.2 (Continuación)

Geometría	Geometría Completa	Geometría Segmentada
geo46		
geo47_48		
geo49		
geo50		
geo51		
geo52		

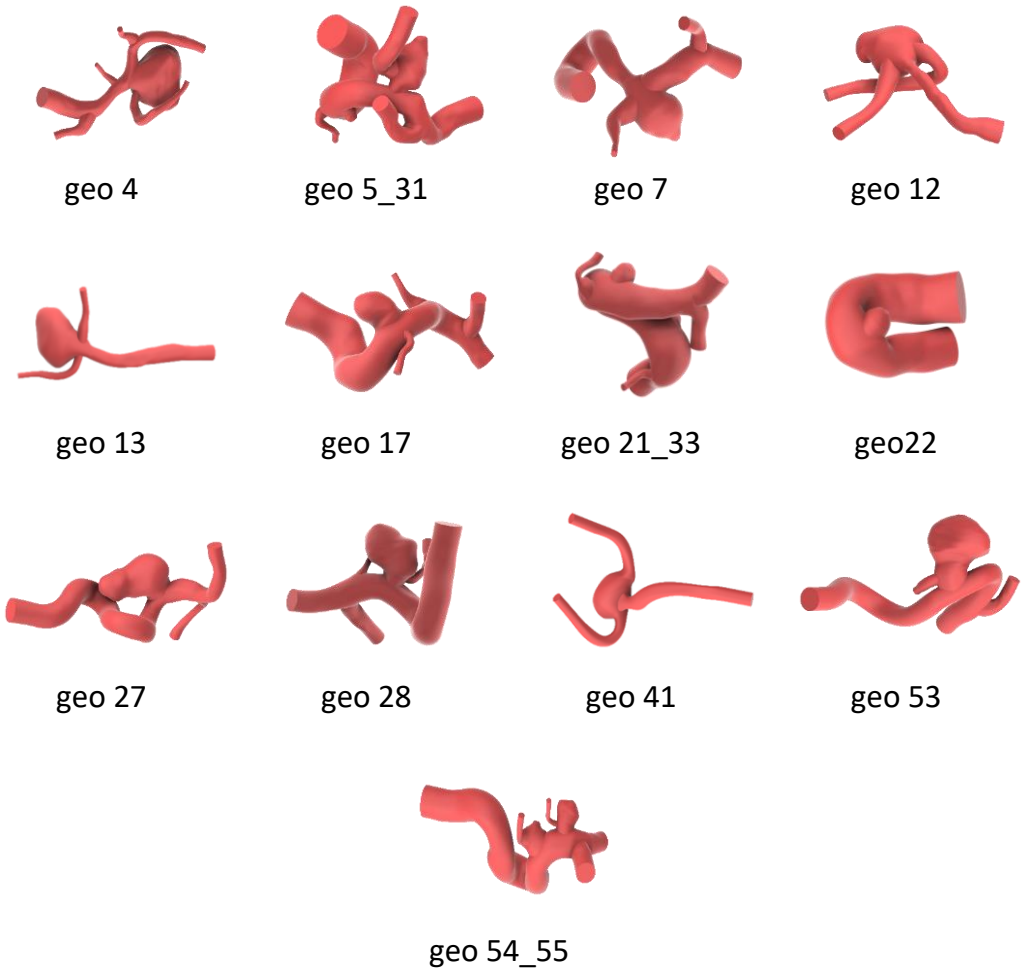


figura 5.19: Geometrías en las que no funcionó la segmentación.

5.4 Discusión

De 37 geometrías en que se probó el algoritmo en 25 se obtiene una segmentación exitosa lo que corresponde a un 67.57% del total de la muestra.

Como se indica en la metodología del algoritmo, durante el uso de este se observa que es sensible a algunos parámetros geométricos como el tamaño del aneurisma, el diámetro de la arteria en la entrada o salidas y la localización del aneurisma respecto a la arteria. A partir de la figura 5.19 se puede dar cuenta de que las geometrías en las que no funcionó el algoritmo comparten que tienen al menos un vaso con un radio muy pequeño respecto al resto de la geometría.

En la mayoría de las geometrías que el algoritmo no funciona el cómputo de las centerlines se indefine, lo que provoca un error en Python que termina cerrando el programa.

El otro incidente común es un cómputo erróneo de las centerlines lo que desemboca en una geometría vacía/nula. Esto ocurre también utilizando como inputs geometrías STL con calidad media, entregando ramificaciones muy separadas con muy pocos puntos. Debido a esto se podría atribuir el fallo del algoritmo en estos casos a la topología de la geometría y a la distribución de los vértices de cada triángulo de la superficie en los STL.

6 Caracterización de Vórtices

Cualitativamente se puede definir un vórtice como una masa de fluido que gira en torno a una región en común [24] . Matemáticamente por otro lado, no existe una definición muy clara de esto.

Algunos métodos se definen como sondeos condicionales, haciendo uso de las Pathlines, Streamlines o búsqueda de mínimos de presión en el fluido, los cuales fallan en flujos transientes y/o turbulento, incluida su dificultad en encontrar vórtices cerca de un borde [25]. Otros por otro lado se definen matemáticamente armando estructuras coherentes (a partir de propiedades basadas en las trayectorias del fluido) o estructuras coherentes lagrangianas (A partir de métodos construidos en un sistema lagrangiano, ie. dinámico local).

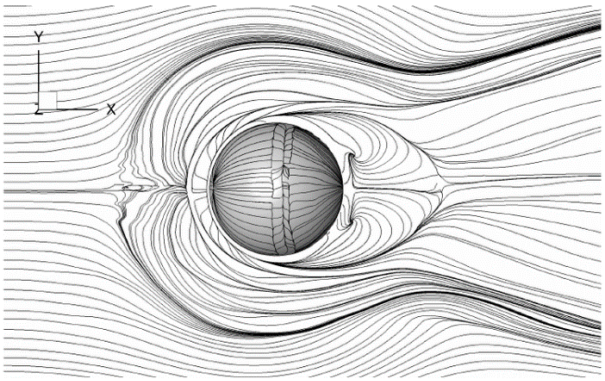


figura 6.1: Ejemplo de uso de Streamlines para identificación de vórtices. Los vórtices corresponden a trayectorias cerradas

6.1 Revisión General de Técnicas de Identificación de Vórtices

Los métodos para caracterizar vórtices se pueden clasificar de diferentes formas. En [26] se habla de métodos basados en Estructuras Coherentes, criterios basados en estructuras lagrangianas coherentes y métodos alternativos basados en sondeos o construcciones varias.

Basados en el Tensor del gradiente de Velocidad

- Criterio Q, Criterio Δ, Criterio λ_2, Criterio de Esfuerzo Rotacional (Swirling-strength Criterion), Criterio de Esfuerzo Rotacional Mejorado (Enhanced Swirling-strength Criterion), Decomposición triple del gradiente.

Basados en la Vorticidad

- Magnitud de Vorticidad ($|\boldsymbol{\omega}|$), Lineas de Vorticidad, Criterio N_k (Kinematic Vorticity Number).

Basados en Estructura Lagrangiana

- Exponentes directos de Lyapunov, Criterio M_z, Heliticidad.

Otros Métodos

- Streamlines cerradas o convergentes, funcional R, Predictor-Corrector.

Por otro lado, en [24] se incluyen otros métodos que clasifica según región/línea, invariancia galileana(lagrangiana) y dataframe (local/global), lo que se puede observar en la Tabla 6.1.

Tabla 6.1: Clasificación de algoritmos de detección de Vórtices según [24].

Método	Región/línea	Galileano	Local/Global
Heliticidad	Línea	No invariante	Local
Parámetro de Remolino	Región	No invariante	Local
Criterio λ_2	Región	Invariante	Local
Predictor-Corrector	Línea	Invariante	Global
Valores Propios	Línea	No invariante	Local
Vectores Paralelos	Línea	No invariante	Local
Máxima Vorticidad	Línea	Invariante	Local
Streamlines	Región	No invariante	Global
Combinatorio	Región	No invariante	Local

En publicaciones más recientes [27] se clasifican los métodos de caracterización de vórtices en 3 generaciones.

6.2 Métodos de 1° Gen.: Basados en la vorticidad

Estos métodos se basan en mediciones a partir de la magnitud de la vorticidad. Para un campo de velocidades $\vec{u}(x, y, z, t)$ se define la vorticidad como:

$$\boldsymbol{\omega} = \nabla \times \vec{u} \tag{6.1}$$

La idea principal propuesta por Helmholtz en 1858 propone 3 teoremas [28] sobre vórtices:

- El esfuerzo en un filamento vorticial es constante a través de su longitud.
- Un filamento vorticial puede terminar en medio del flujo, sino que debe extenderse hacia los bordes o formar un camino cerrado.
- En ausencia de fuerzas rotacionales externas, un flujo irrotacional debe continuar irrotacional.

Debido a los límites en la investigación de la época, Helmholtz generó la idea errónea de que los vórtices son equivalentes a zonas de gran vorticidad. Esto **es incorrecto** y se ha demostrado que un vórtice puede aparecer en áreas donde la magnitud de la vorticidad es menor que en sus áreas cercanas [29]. Por esta razón no representa un método adecuado para caracterizar vórtices.

6.3 Métodos de 2° Gen.: Basados en valores propios

En un esfuerzo por definir métodos para identificación de vórtices se aludía a definiciones en base a las Pathlines y Streamlines identificando así un centro y/o eje de rotación. El principal problema de esto era que ambas estructuras no son galileanas (lagrangianas) invariantes, lo que significa que la posibilidad de determinar un centro de rotación absoluto matemáticamente es sólo una idea vaga.

La idea de definir estructuras en base al gradiente de velocidad del flujo prometió resolver el problema dado que permite distinguir la componente de vorticidad y la de esfuerzos, además de que se pueden armar estructuras lagrangianas invariantes [28].

De manera específica estos métodos utilizan los valores propios de del gradiente de velocidad o sus invariantes. Algunos de estos métodos son el criterio Q, criterio Δ, criterio λ_{ci} y criterio λ_2.

Dado el campo de velocidades $\vec{u}(x, y, z, t)$ se define el gradiente (***J***) como el tensor:

$$J = \nabla\vec{u} = \begin{bmatrix} \partial_x u_x & \partial_y u_x & \partial_z u_x \\ \partial_x u_y & \partial_y u_y & \partial_z u_y \\ \partial_x u_z & \partial_y u_z & \partial_z u_z \end{bmatrix} \tag{6.2}$$

Convenientemente, se puede separar el gradiente del vector de velocidades en su parte simétrica (***S***) y su parte asimétrica (**Ω**):

$$S = \frac{J + J^T}{2} \tag{6.3}$$

$$\Omega = \frac{\mathrm{J} - \mathrm{J}^{\mathrm{T}}}{2} \tag{6.4}$$

6.3.1 Criterio Q

A partir del gradiente de velocidad del flujo, el segundo invariante de dicho tensor es:

$$Q = \frac{1}{2}\left(tr(J)^2 - tr(J^2)\right) = \frac{1}{2}(\|\Omega\|^2 - \|S\|^2) \tag{6.5}$$

Donde $\|\boldsymbol{\Omega}\|$ y $\|\boldsymbol{S}\|$ son las normas euclidianas[3] de los tensores, definida por:

$$\|\mathrm{M}\| = \sqrt{\Sigma_{i,j} M_{i,j}^2} \tag{6.6}$$

Es importante notar que Q es lagrangiano invariante, ie. no depende del sistema de referencia.

La componente$\|\boldsymbol{\Omega}\|$ representa la magnitud de la vorticidad y $\|\boldsymbol{S}\|$ representa la magnitud del tensor de deformación. Luego si la magnitud de la vorticidad domina se define esa zona como parte de un vórtice. En otras palabras, $Q > 0$ define un vórtice donde:

[3] En algunas publicaciones se hace alusión a la norma de Frobenius o Hilbert-Schmidt para matrices en lugar de la euclidiana. Cabe recordar que para matrices $A \in \mathcal{M}_{N\times N} \subseteq \mathbb{R}^n$ estas normas son similares.

Cabe mencionar que el criterio Q es más restrictivo que el criterio Δ que busca dónde los valores propios del tensor J son complejos [25].

6.3.2 Criterio λ_2

En una geometría plana, en el caso de una región con movimiento rotatorio, se genera un mínimo de presión en su centro debido al balance de fuerzas en el eje radial a la rotación. La fuerza Centrífuga se equipara con la fuerza generada por el gradiente de Presión. En el caso de régimen transiente, existencia de fuerzas viscosas o en un caso 3d esto no es cierto [30] . En otras palabras, un vórtice puede genera un mínimo de presión en su eje de rotación, pero un mínimo de presión no representa necesariamente un vórtice, además de que las fuerzas viscosas pueden "eliminar" dicho mínimo.

Para considerar estas restricciones en [25] se propone lo siguiente:

Sabemos que la información de los mínimos de presión (P) se encuentra en el Hessiano de la presión ($\nabla^2 P$). Tomando el Gradiente de la ecuación de Navier-Stokes (ecuación (3.2)) se tiene que:

$$\nabla a = -\frac{1}{\rho}\nabla^2 P + \nu\nabla^2 u \tag{6.7}$$

Donde ∇a es el tensor gradiente de aceleración $\left(\nabla\left(\frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u}\right)\right)$. Dado que $\nabla^2 P$ es simétrico, conviene descomponer ∇a en su parte simétrica y antisimétrica:

$$\nabla a = \underbrace{\left[\frac{D}{Dt}S + \Omega^2 + S^2\right]}_{\textit{parte simétrica}} + \underbrace{\left[\frac{D}{Dt}\Omega + \Omega S + S\Omega\right]}_{\textit{parte antisimétrica}} \tag{6.8}$$

Por otro lado, la ecuación de transporte de vorticidad para fluido incompresible, isotrópico y con fuerzas conservativas:

$$\frac{D}{Dt}\omega = (\omega\cdot\nabla)u + \nu\nabla^2\omega \tag{6.9}$$

Recordando que $(\nabla u = \Omega + S)$ y $(\omega = \nabla\times\vec{u})$ se puede despejar:

$$\frac{D}{Dt}S - \nu\nabla^2 S + \Omega^2 + S^2 = -\frac{1}{\rho}\nabla^2 P \tag{6.10}$$

El primer término de (6.10) corresponde a la componente transiente irrotacional, y el segundo termino es la componente viscosa. De esta manera se puede considerar sólo $(\Omega^2 + S^2)$ para determinar la existencia de un mínimo de Presión (Dado por $\nabla^2 P$). Esto es, que la región tiene 2 valores propios negativos.

Tomando en cuenta además que el tensor $(\Omega^2 + S^2)$ es simétrico (entonces $\lambda_i \in \mathbb{R}$), si sus valores propios $\lambda_1 \geq \lambda_2 \geq \lambda_3$ se requiere que $\lambda_2 < 0$ para identificar el punto como vórtice. Esto es donde se cumple:

$$\lambda_2(\Omega^2 + S^2) < 0 \tag{6.11}$$

6.3.3 Comparación entre λ_2 y Q

Recordando que Q es el segundo invariante del gradiente de velocidad, se puede reescribir en función de los valores propios del tensor $(\Omega^2 + S^2)$:

$$Q = -\frac{1}{2}(\lambda_1 + \lambda_2 + \lambda_3) \tag{6.12}$$

De esta manera, considerando que $\lambda_1 \geq \lambda_2 \geq \lambda_3$, Se pueden comparar los casos donde el criterio λ_2 y Q coinciden o discrepan en función de los valores propios del tensor $(\Omega^2 + S^2)$:

Tabla 6.2: Comparación criterios λ_2 y Q

Casos			*Criterio* λ_2	*Criterio Q*
$\lambda_1 > 0$	$0 \leq \lambda_2 < \lambda_1$	$-(\lambda_1 + \lambda_2) < \lambda_3 < \lambda_2$	*No hay Vortices*	*No hay Vortices*
$\lambda_1 > 0$	$0 \leq \lambda_2 < \lambda_1$	$\lambda_3 < -(\lambda_1 + \lambda_2)$	*No hay Vortices*	*Si hay Vortices*
$\lambda_1 > 0$	$\frac{-\lambda_1}{2} < \lambda_2 \leq 0$	$-(\lambda_1 + \lambda_2) < \lambda_3 < \lambda_2$	*Si hay Vortices*	*No hay Vortices*
$\lambda_1 > 0$	$\frac{-\lambda_1}{2} < \lambda_2 \leq 0$	$\lambda_3 < -(\lambda_1 + \lambda_2)$	*Si hay Vortices*	*Si hay Vortices*
$\lambda_1 > 0$	$\lambda_2 \leq \frac{-\lambda_1}{2}$	$\lambda_3 < \lambda_2$	*Si hay Vortices*	*Si hay Vortices*
$\lambda_1 < 0$	$\lambda_2 < \lambda_1$	$\lambda_3 < \lambda_2$	*Si hay Vortices*	*Si hay Vortices*

El principal problema de estos métodos es que para cada caso particular el valor umbral se debe configurar manualmente y el resultado puede ser muy sensible para discriminar incluso vórtices "fuertes" y "débiles". En métodos como el criterio Q una mala selección del valor umbral puede ocasionar el fenómeno "Vortex Breakdown".

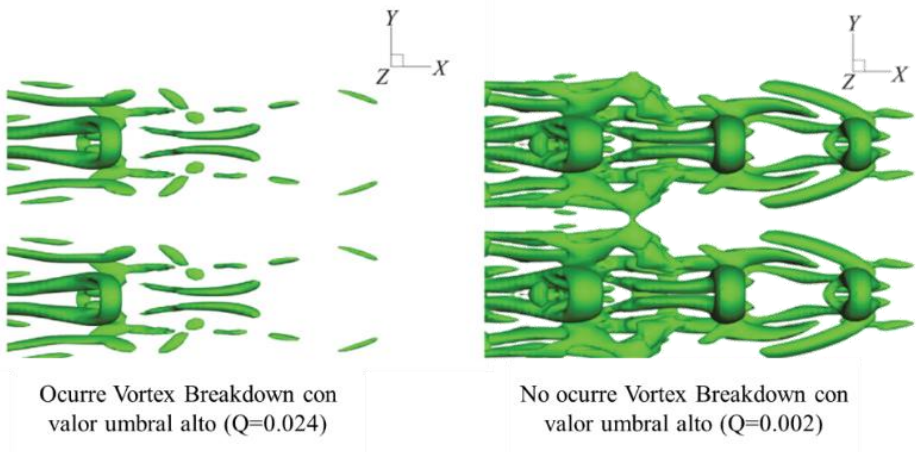


figura 6.2: Fenómeno de Vortex Breakdown al seleccionar valor umbral.
Fuente: Traducción a partir de [27].

Otro problema de estos métodos es que no permiten identificar ejes de rotación o dirección de los vórtices ni representan la magnitud del esfuerzo rotatorio.

6.4 3° Gen. Métodos Omega, Liutex y Omega-Liutex

6.4.1 Método Omega

Con la tarea de mejorar los algoritmos de identificación de vórtices, en 2016 se propone el método Omega. Dado que la vorticidad puede ser pequeña existiendo gran rotación del flujo, así como puede ser grande en ausencia de rotación en un flujo laminar y además el tensor de deformación juega un papel importante en fenómenos vorticiales , se define $\mho$ a partir de la relación entre el tensor de vorticidad ($\boldsymbol{\Omega}$) y el tensor de deformación ($\boldsymbol{S}$).[4]

$$\mho = \frac{\|\Omega\|_F^2}{\|\Omega\|_F^2 + \|S\|_F^2} \tag{6.13}$$

Donde $\|A\|_F$ corresponde a la norma de Frobenius (equivalente a la norma euclidiana, Hilbert-Schmidt y p=2 para matrices $A \in \mathcal{M}_{N\times N} \subseteq \mathbb{R}^n$). Para evitar la división por cero se introduce el pequeño parámetro ε:

$$\mho = \frac{\|\Omega\|_F^2}{\|\Omega\|_F^2 + \|S\|_F^2 + \varepsilon} \tag{6.14}$$

Elegir bien el valor de ε es importante para que el método funcione. Estudios sugieren que el valor correcto está fuertemente relacionado a la dimensionalidad del problema, el paso de tiempo, y el caso específico. Aquí es donde se sugiere el valor en función de los tensores de rotación y deformación [27] [28]:

$$\varepsilon = 0.001(\|\Omega\|_F^2 - \|S\|_F^2)_{max} \equiv 0.002 Q_{max} \tag{6.15}$$

Se tendrá entonces un vórtice si el tensor de rotación sobrepasa al de deformación, esto es cuando se cumpla $\mho \geq 0.5$

La principal ventaja de este método es que permite encontrar los vórtices fuertes y débiles, y no es sensible a variaciones pequeñas del valor umbral a sondear (en general valores de $\mho = 0.51$ y $\mho = 0.52$ entregan buenos resultados) , reduciendo la probabilidad de Vortex Breakdown.

Por otro lado, el método Liutex, inicialmente apodado Rortex[5] en 2018, [31] propone una solución que además de identificar los vórtices entrega el vector del eje de rotación del vórtice.

El método consiste en hacer un cambio de base del gradiente de velocidad (en el sistema global x,y,z) a un sistema coordenado (X,Y,Z) donde el eje de rotación del flujo está alineado al eje Z. Esta definición nace de la descomposición real de Shur, donde podemos descomponer el tensor de vorticidad en su parte rotacional rígida y la parte no rotacional antisimétrica, y el vector de vorticidad se puede descomponer en un vector rotacional rígido llamado vector Rortex y su vector no rotacional llamado vector de cizalle, en otras palabras:

[4] Para evitar confusión entre el método Omega y la parte antisimétrica del tensor gradiente de velocidad se ha escrito como $\mho$ al método Omega en este documento.

[5] Se hace énfasis en que Liutex y Rortex son lo mismo.

$$\nabla \times \vec{u} = \vec{R} + \vec{S} \tag{6.16}$$

En [31] se demuestra la existencia y unicidad de una serie de transformaciones para armar el cambio de base de (x,y,z) a (X,Y,Z), define un algoritmo rápido para el cómputo del vector Rortex y define:

- El vector Rortex $\vec{R} = R\vec{r}$ es la parte rotacional rígida de un flujo en un punto dado.
- Un vórtice es la región conectada donde $R \neq 0$.

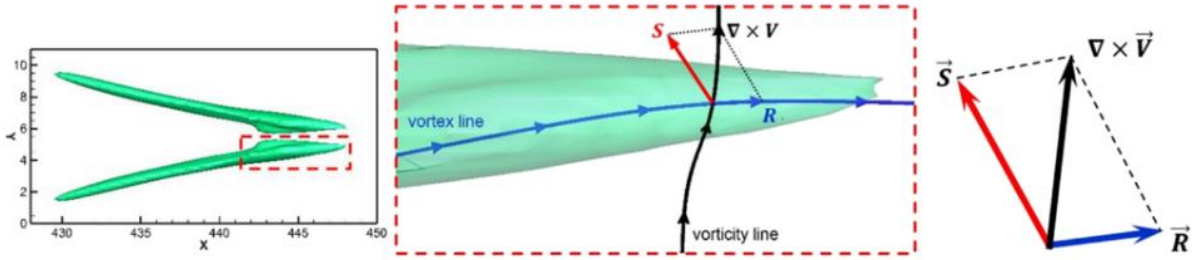


figura 6.3: Descomposición del vector vorticidad en $\vec{R}$ y $\vec{S}$.

6.5 Materiales y Métodos

Se utilizarán los criterios λ_2, Q y Ω para caracterizar vórtices, los cuales se programaran en un código basado en los módulos vtk v9.0.1 y Numpy v1.20.2 en Python 3.6.1. A diferencia del módulo de segmentación, como no utiliza el módulo vmtk este puede ser utilizado también en Spyder o Jupyter Notebook donde es más fácil testear. Los input del algoritmo se detallan en la tabla a continuación:

Tabla 6.3: Inputs del Algoritmo de caracterización de Vórtices.

N°	Archivo	Tipo	Descripción	Carácter
1	Geometría.vtu	vtkUnstructuredGrid()	Malla de la geometría	Obligatorio
2	Locator	Tabla LookUp	Indexación entre nodos de la malla y el export de ANSYS	Opcional
3	expNNNN.csv	Tabla csv	Export del time-step NNNN de la simulación de ANSYS con las coordenadas, velocidades y gradientes de cada nodo.	Obligatorio
			Puede incluir también otros valores como densidad, presión, WSS,etc	Opcional
4	-	Información de la simulación	Información de la Simulación como la versión de ANSYS, regiones y time-steps disponibles.	Opcional

El diagrama del algoritmo se puede apreciar en el diagrama de flujo de la figura 6.4, donde se observa la relación y transformaciones de la malla y time-steps de ANSYS a objetos de vtk.

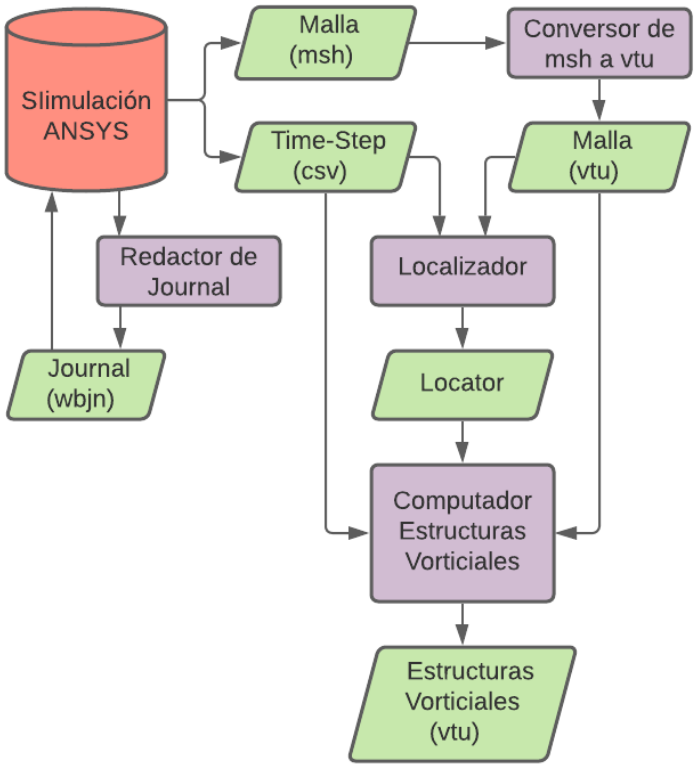


figura 6.4: Diagrama del Algoritmo de caracterización de Vórtices.

6.5.1 Malla de la geometría

La malla de la geometría se debe exportar en formato msh en codificación ASCII desde ANSYS Meshing. Esto se configura en Tools >> Options >> Meshing >> Export.

Después de exportarla se debe transformar a un objeto vtk para poder utilizarla en el algoritmo. Esto se resuelve programando un lector de mallas de ANSYS que sigue la indexación y lee la malla en base a la implementación de ANSYS Meshing.

El lector de malla toma el archivo .msh y lo lee línea por línea, según el balance de paréntesis "(" y ")", el tipo de información (comentario, encabezado, nodos, caras, caras periódicas o celdas), guardando las coordenadas de cada nodo en la lista Nodes, la lista de ids de nodos de cada cara y a qué celdas pertenecen en la lista Faces, y los tipos de celdas en la lista Cells.

$$Nodes = \begin{bmatrix} [n_{1x} & n_{1y} & n_{1z}] \\ [n_{2x} & n_{2y} & n_{2z}] \\ \vdots & \vdots & \vdots \\ [n_{Nx} & n_{Ny} & n_{Nz}] \end{bmatrix} \tag{6.17}$$

$$Faces = \begin{bmatrix} [3 & n_{id1} & n_{id2} & n_{id3} & c_1 & c_2] \\ [4 & n_{id1} & n_{id2} & n_{id3} & n_{id4} & c_1 & c_2] \\ & & \vdots & & & \\ & & \vdots & & & \end{bmatrix} \tag{6.18}$$

$$Cells = [ct_1 \quad ct_2 \quad \dots \quad ct_{Nc}] \tag{6.19}$$

Donde $(n_{ix}\ n_{iy}\ n_{iz})$ corresponden a las coordenadas del nodo i, En la lista de caras el primer número es la cantidad de nodos que tiene la cara (3 para triangular y 4 para cuadrada), n_{idk} corresponde al id del nodo número k que pertenece a la cara, c_1 y c_2 son los ids de las celdas a las que corresponde la cara. En caso de que la cara es un borde (por ende, pertenece sólo a una celda) c_1 o c_2 será 0. En la lista de celdas ct_i es la topología de la celda i.

Tabla 6.4: Tipos de Celdas de ANSYS.

ct_i	*Topología*
1	*Triangular*
2	*Tetraedro*
3	*Cuadrilátero*
4	*Hexaedro*
5	*Pirámide*
6	*Cuña*

Una vez almacenada en listas la información del archivo .msh se debe armar el objeto vtkUnstructuredGrid según el siguiente pseudo código:

```
 1. # Construcción de nodos
 2. Inicializar Points → vtkPoints()
 3. Points.InsertPoint(id , xi , yi , zi)
 4.
 5. # Mesh
 6. Inicializar Mesh → vtkUnstructuredGrid()
 7.
 8. # Construccción de topología
 9. Inicializar CellArray → vtkCellArray()
10. for cell in Cells:
11.         Inicializar cell → vtkCell(Tetraedro, Hexaedro, Pirámide, Prisma )
12.         Cell.InsertPoint(Sub id, Point Id)     # Deben ir ordenados según el tipo
                                                     de celda
13.         Mesh.InsertCell(cell)
```

figura 6.5: Pseudo código para armar una UnstructuredGrid.

Hay que recalcar que la indexación de ANSYS parte desde 1 y las de vtk, Numpy y Python parten en 0, por lo que al momento de establecer el nexo entre un id se debe corregir (entiéndase restar 1) para ubicarlo y colocarlo en los objetos.

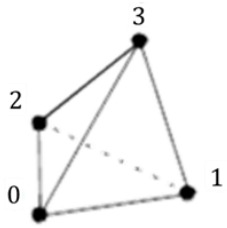
Como se detalla en la línea 12. de la figura 6.5, es de suma importancia que los nodos de cada celda se coloquen en orden según el tipo de celda. Para solucionar esto, a partir de la lista Faces se crean 2 listas.

$$Cells\_Nodes = \begin{bmatrix} [n_{id1c1} & n_{id2c1} & \cdots & n_{idNc1}] \\ [n_{id1c2} & n_{id2c2} & \cdots & n_{idNc2}] \\ & \vdots & & \\ [n_{id1cN} & n_{id2cN} & \cdots & n_{idNcN}] \end{bmatrix} \tag{6.20}$$

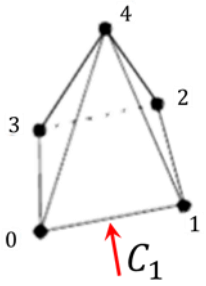
$$Cells\_Faces = \begin{bmatrix} [F_{1c1} & F_{2c1} & \cdots & F_{Nc1}] \\ [F_{1c2} & F_{2c2} & \cdots & F_{Nc2}] \\ & \vdots & & \\ [F_{1cN} & F_{2cN} & \cdots & F_{NcN}] \end{bmatrix} \tag{6.21}$$

Donde n_{idNcK} es el id del nodo N perteneciente a la celda K y F_{NcK} es la lista de la cara N (que incluye el número de nodos e ids de los nodos que la componen) que pertenece a la celda K.

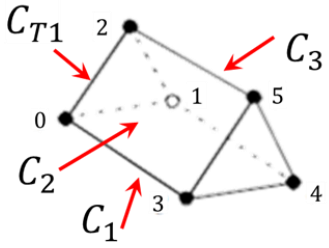
Para ordenar los nodos se debe hacer un análisis entre las caras de la celda y qué nodos contienen, lo que se puede realizar fácilmente con operaciones booleanas según corresponda al tipo de celda. El orden se detalla en el anexo **Tipos de Celdas lineales en VTK.** El análisis de caras se detalla en la :



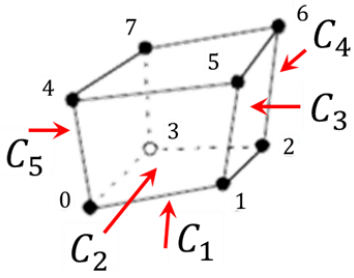
TETRAEDRO
No importa el orden de los nodos, la celda siempre será convexa.



PIRÁMIDE
Buscar cara con 4 nodos.
Incluir nodo que falta (p_4)



CUÑA
Elegir cara triangular C_{T1} (Puede ser cualquiera de las 2
La cara cuadrada que no tiene a p_2 es C_1
La cara cuadrada que no tiene a p_1 es C_2
La cara cuadrada que no tiene a p_0 es C_3
p_3 no es p_0 y pertenece a las caras C_1 y C_2
p_4 no es p_1 y pertenece a las caras C_1 y C_3
p_5 no es p_2 y pertenece a las caras C_2 y C_3



HEXAEDRO
Elegir C_1 (Puede ser cualquiera)
C_2 no es C_1 y contiene a p_0 y p_1
C_3 no es C_1 y contiene a p_1 y p_2
C_4 no es C_1 y contiene a p_2 y p_3
C_5 no es C_1 y contiene a p_0 y p_3
p_4 no es p_0 y pertenece a las caras C_2 y C_5
p_5 no es p_1 y pertenece a las caras C_2 y C_3
p_6 no es p_2 y pertenece a las caras C_3 y C_4
p_7 no es p_3 y pertenece a las caras C_4 y C_5

figura 6.6: Operaciones booleanas para ordenar nodos en las celdas.

6.5.2 Point y Cell Data de cada time-step

A partir de una simulación en ANSYS, se exportan los time-steps con la información de la malla y las velocidades y gradientes de velocidad para cada nodo en una tabla csv. Esto se puede hacer mientras se calcula la simulación en ANSYS definiendo un export en la sección “calculation

activities" o directamente desde el post processing seleccionando cada intervalo de tiempo, la región de interés y las variables a exportar.

La diferencia es que en el primer método el proceso es automático, pero si son muchos nodos puede aumentar considerablemente el tiempo de simulación. Por otro lado, el segundo método es más tedioso, salvo que en ansys existen los scripts llamados Journal.

Un Journal de ANSYS es un script que se puede grabar similar a los macros de Excel copiando las acciones en un código que después se puede editar, así como se puede programar desde cero. Son de gran utilidad cuando se trata de tareas similares que se deben repetir un gran número de veces. En el caso de las simulaciones de los aneurismas se trata de 100 time-steps por geometría.

A partir de información como la versión de ANSYS, la región de interés (en el caso de las simulaciones utilizadas, tienen el nombre part part N) y los time-steps deseados se programa en python un algoritmo que repite en un script Journal de ANSYS la tarea de seleccionar las variables y exportarlas en un archivo llamado expNNNN.csv para cada time step NNNN deseado. De esta manera se exportan todos los time-steps que se quieren de manera automática en menor tiempo que realizarlo manualmente. Para mejorar aún más el rendimiento del script, se desactivan la visualización de datos y gráficos en el Post Processing de ANSYS (Esto se debe hacer manualmente antes de correr el Journal).

6.5.3 Localizador

Con las tablas csv disponibles se debe verificar la correspondencia de los nodos de la malla de la geometría con los nodos de la malla del export. Debido a la cantidad de nodos, realizar esta tarea puede tomar mucho tiempo, por lo que es conveniente utilizar una tabla LookUp.

Método 1

La primera alternativa para armar la tabla LookUp es a partir de un algoritmo de fuerza bruta, que revisa la diferencia entre las coordenadas de cada nodo de la malla (mesh.vtu) con los nodos del export de ANSYS. Esto se realiza con la lista de las coordenadas de los nodos de la malla (VTU), la lista de las coordenadas de los nodos el export de ansys (CSV) y una copia de CSV (CSV_LOC) donde para cada coordenada en VTU se busca la coordenada correspondiente en CSV_LOC[6] (nodo) . Encontrado el nodo, se ingresa en el Indexador la referencia del nodo en CSV, donde luego para que no se repita un índice se elimina el nodo de la lista CSV_LOC.

```
1. # Inicializar listas
2. Inicializar VTU → vtkXMLUnstructuredGridReader(mesh.vtu)
3. Inicializar CSV → read(expor.csv de ANSYS)
4. CSV_LOC = CSV.copy()
5.
6. # Iterar
7. for coord in VTU:
8.         for nodo in CSV_LOC:
9.                 if distancia(coord,nodo) < error:
10.                         Locator.append(CSV.index(nodo))
11.                         CSV_LOC.pop(nodo)
12.                         break
```

[6] Como se trabaja con valores de punto flotante la coincidencia se refiere a que la distancia no supera un error definido (generalmente 10^{-6} o inferior.)

figura 6.7: Pseudo código Método 1 de generar tabla LookUp.

Método 2

Utilizando el módulo Numpy se puede optimizar el método anterior, eliminando la necesidad de un loop for trabajando con arrays en lugar de listas.

```
 1. # Inicializar listas
 2. Inicializar VTU → vtkXMLUnstructuredGridReader(mesh.vtu) → np.array()
 3. Inicializar CSV → read(expor.csv de ANSYS) → np.array()
 4. CSV_LOC = CSV.copy()
 5.
 6. # Iterar
 7. for coord in VTU:
 8.         search    = np.array(len(CSV_LOC)*[coord)
 9.         distancia = np.sum(np.abs(search-CSV_LOC)**2,axis=-1)**(1./2)
10.         dist = distancia.copy().sort()
11.         nodo = CSV_LOC[distancia.index(dist[0])]
12.         Locator.append(CSV.index(nodo))
13.         CSV_LOC.pop(nodo)
```

figura 6.8: Pseudo código Método 2 de generar tabla LookUp.

Método 3

A partir de la estructura del algoritmo del método 1, claramente se puede ver que el tiempo de ejecución de este tendrá una cota inferior (caso en que las listas están igualmente ordenadas) y una cota superior(caso en que las listas están ordenadas al revés).

Para acercarse al caso más favorable se puede introducir una lista adicional VTU_LOC que luego se ordenan junto a CSV_LOC. Cuando una lista está compuesta por listas el algoritmo sort() de Python ordena los elementos en base al primer componente de cada sub-lista , en caso de haber coincidencias continua con los siguientes. De esta manera todas las coordenadas de VTU_LOC y CSV_LOC estarán ordenadas según x, y, z. Con esto se logra el que segundo loop for no debería iterar muy lejos del origen de CSV_LOC actual.

```
 1. # Inicializar listas
 2. Inicializar VTU → vtkXMLUnstructuredGridReader(mesh.vtu)
 3. Inicializar CSV → read(expor.csv de ANSYS)
 4.
 5. # Listas Ordenadas
 6. VTU_LOC = VTU.copy().sort()
 7. CSV_LOC = CSV.copy().sort()
 8.
 9. # Iterar
10. for coord in VTU_LOC:
11.         for nodo in CSV_LOC:
12.                 if distancia(coord,nodo) < error:
13.                         Locator[VTU.index(coord)] = CSV.index(nodo))
14.                         CSV_LOC.pop(nodo)
15.                         break
```

figura 6.9: Pseudo código Método 3 de generar tabla LookUp.

Como no se está usando una malla dinámica, la ventaja de utilizar la tabla LookUp es que el cómputo de la indexación se debe realizar sólo una vez, luego esta puede ser utilizada para cualquier otro time-step de la misma simulación.

El performance de cada método se puede ver en la sección de resultados del capítulo.

6.5.4 Computador de Estructuras Vorticiales

Para caracterizar las estructuras vorticiales se utilizan las ecuaciones (6.5), (6.11) y (6.14) para calcular Q, λ_2 y $\mho$ en cada nodo. Adicional a esto se calculan las versiones normalizadas de λ_2 y Q [4]:

$$\overline{\overline{\lambda_2}} = \frac{\lambda_2}{\|\vec{v}\|} \tag{6.22}$$

$$\bar{\bar{Q}} = \frac{Q}{\|\vec{v}\|} \tag{6.23}$$

Y las versiones adimensionales unitarias de λ_2, Q y $\mho$:

$$\lambda_{2A} = \frac{\lambda_2}{\min(\lambda_2)} \tag{6.24}$$

$$Q_A = \frac{Q}{\max(Q)} \tag{6.25}$$

$$\mho_A = \frac{\mho - 0.5}{0.5} \tag{6.26}$$

Las versiones normalizadas permiten la comparación entre una simulación y otra. Por otro lado, las versiones adimensionales definen un vórtice para λ_{2A}, Q_A y $\mho_A$ donde pertenezcan al intervalo $(0,1)$ por lo que permite la comparación entre métodos

Como $\mho$ es adimensional y normalizado ($\mho \in (0,1)$) no tiene mucho sentido definir una versión normalizada para el método.

6.6 Resultados

A continuación, se presentan los resultados de los algoritmos Localizadores y del computador de estructuras vorticiales.

6.6.1 Performance de Algoritmos Localizadores

Se prueba el performance del método 1 con el caso favorable y el más desfavorable para medir el tiempo que tardaría la indexación en función de la cantidad de nodos de la malla. Esto se midió con un procesador AMD Ryzen 9 4900HS de 16 CPU y 16 Gb de RAM.

Para testear los algoritmos se crea una lista de N puntos ordenada y otra equivalente ordenada al revés, lo que pasarían a representar VTU y CSV en los algoritmos Localizador.

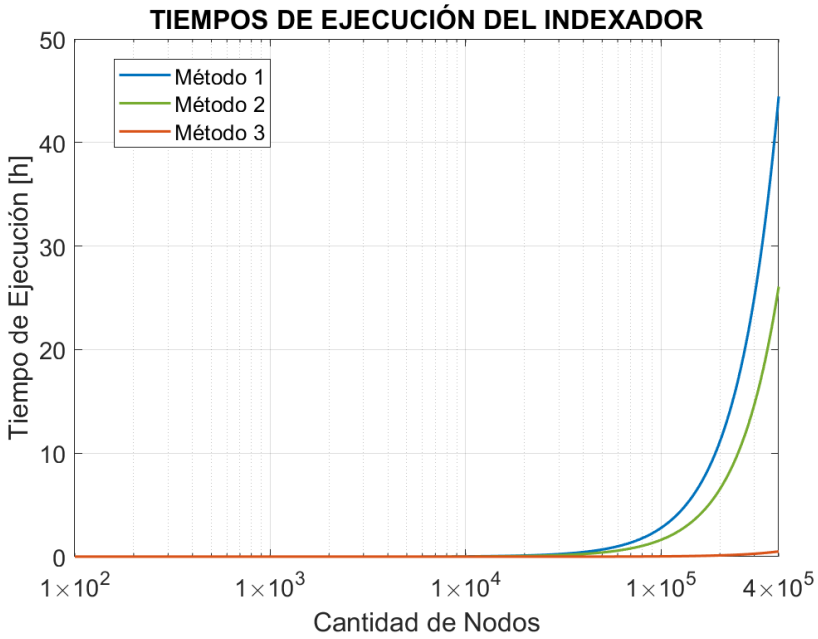


figura 6.10: Performance de diferentes métodos del algoritmo indexador.

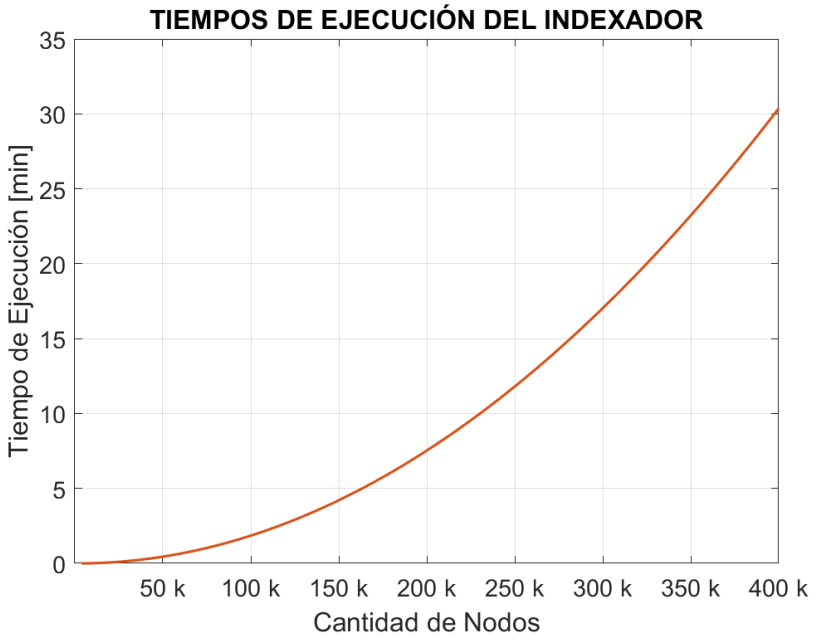


figura 6.11: Performance del algoritmo de indexación 3.

6.6.2 Computación de Estructuras Vorticiales

ANSYS tiene implementado el cálculo de los criterios λ_2 y Q, los cuales se exportan y son comparados con los valores calculados. El error medio no llega a superar el 1% para ambos métodos.

Para la geometría geo11 se grafica una IsoSuperficie (detalle del procedimiento en el capítulo 7, en la sección de Materiales y Métodos) con el valor de $\lambda_2 = -84666\ [s^{-2}]$ y λ_2=$-190499[s^{-2}]$ la cual es después comparada con la IsoSuperficie del mismo valor generado por el algoritmo trabajado.

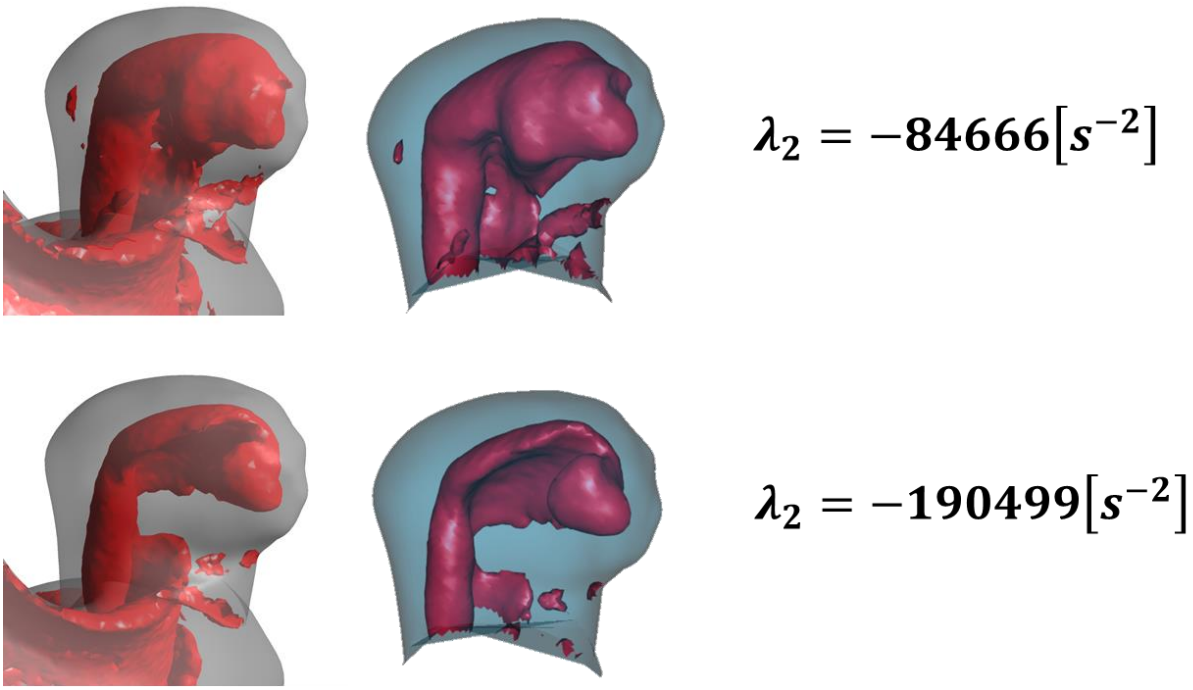


figura 6.12: Estructura vorticial λ_2 en geo11.
A la izquierda resultado de ANSYS Results. A la derecha resultado del algoritmo propio.

De igual manera se realizó con el criterio Q, utilizando los valores

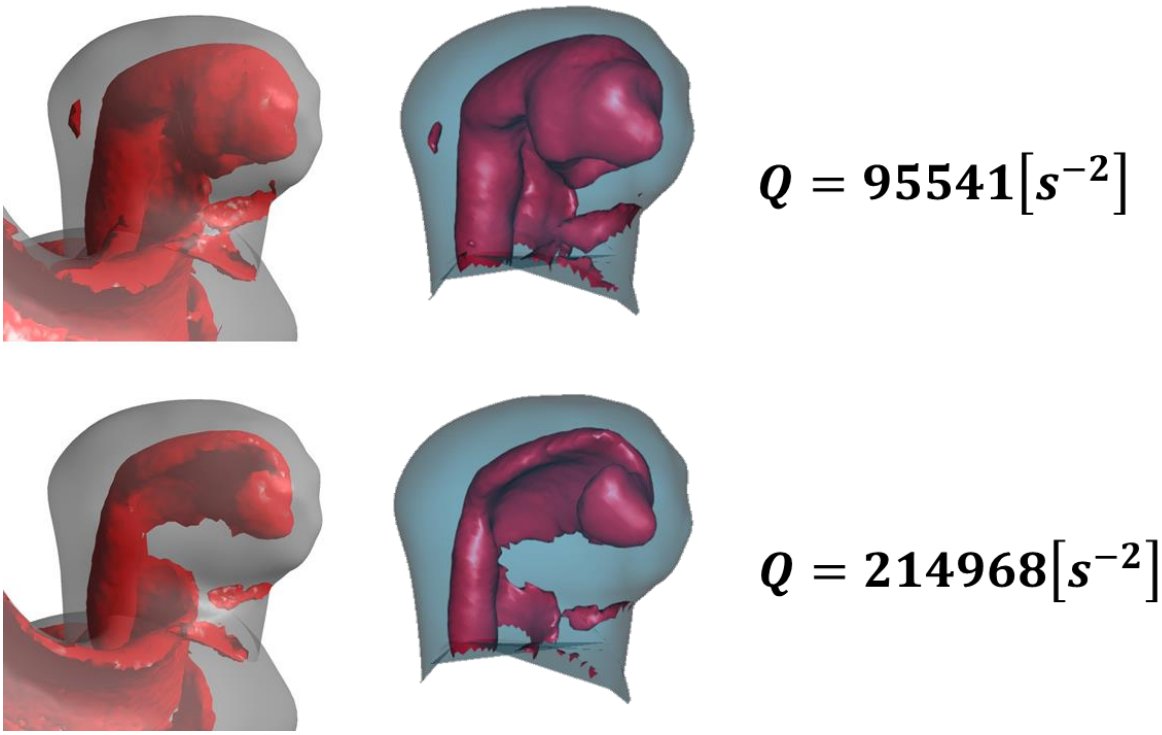


figura 6.13: Estructura vorticial Q en geo11.
A la izquierda resultado de ANSYS Results. A la derecha resultado del algoritmo propio.

6.7 Discusión

La diferencia de tiempo de cómputo de los algoritmos localizadores en la figura 6.10 deja en claro que el algoritmo a utilizar en el método general será el método 3, donde se reduce el tiempo del orden de horas a sólo minutos cuando la cantidad de nodos es superior a 30.000.

En cuanto al performance del cómputo de vórtices en las geometrías que se probó muestra excelentes resultados para los criterios λ_2 y Q respecto a los resultados de ANSYS. Sin embargo, esto valida el método bajo la asunción de que la simulación está correctamente realizada. Como se detalla en el siguiente capítulo, se realizó un análisis de sensibilidad de malla respecto a conservación de masa y a la variación de los valores de SWSS y DWSS [16].

En las figura 6.12 y figura 6.13 también queda en evidencia la precisión del algoritmo respecto a los vórtices débiles como los que se observan en ambas al extremo inferior, tanto para ANSYS como para el algoritmo desarrollado.

También se observa que la estructura vorticial del criterio Q es un poco más suave que con el criterio λ_2.

7 Estudio de Estructuras Vorticiales en Aneurismas Cerebrales

7.1 Materiales y Métodos

En conjunto con las simulaciones realizadas por Nicolas Amigo en 2018, las herramientas mencionadas en los capítulos anteriores y las herramientas de visualización y análisis de datos a mencionar en este capítulo se procede con el estudio de las estructuras vorticiales en puntos particulares (como la sístole y la diástole) y a lo largo de todo el ciclo cardiaco.

7.1.1 Geometrías y simulaciones

La base de datos cuenta con 88 aneurismas en 75 modelos computacionales (Esto debido a que existen geometrías con más de un aneurisma) . De estos emergen 64 simulaciones CFD en ANSYS Fluent con un total de 71 aneurismas simulados en la tesis de Doctorado [16].

Las geometrías están clasificadas por Localización (ICA, ACA, MCA y VBA, detallados en la Tabla 7.1), Tipo (Lateral, Lateral con bifurcación, Terminal) y Ruptura (Si / No):

Tabla 7.1: Grupos de Arterias Cerebrales.

Grupo	Arterias
ICA	Carótida interna, oftálmica, comunicante posterior, coroidal anterior
ACA	Cerebral anterior, comunicante anterior, pericallosa
MCA	Cerebral media
VBA	Vertebral, basilar, cerebral posterior

El mallado de las geometrías se realizó con el software ANSYS Meshing utilizando elementos tetraédricos. En la tesis de Nicolás Amigo [16] se realiza el análisis de sensibilidad de malla con densidades de 800, 1500, 2000 y 3000 [celdas/mm^3]. Estudiando la variación de DWSS, SWSS, el tiempo de simulación y la conservación de masa, se selecciona la densidad de 1500[celdas/mm^3] para mallar todas las geometrías.

Condiciones de Borde

La base de datos cuenta con los perfiles de velocidad de la arteria carótida obtenidos mediante ultrasonido Doppler. Para cada uno de estos perfiles junto a la información de la geometría se saca el perfil del caudal promedio al cual se le saca la transformada de Fourier. De esta manera se obtiene el perfil de velocidad promedio que se usa en todas las geometrías.

Con el perfil de caudal Amigo A. utiliza el modelo del circuito Windkessel (3.19) con pequeños ajustes a parámetros sugeridos en [32]. Con el fin de ajustar el perfil de presión entre 80 y 120 [mmHg] utiliza R_1=10.5[mmHg s/cm^3], R_2=12.5[mmHg s/cm^3] y C=3.1[cm^3/mmHg] para obtener la presión a partir del perfil temporal del caudal.

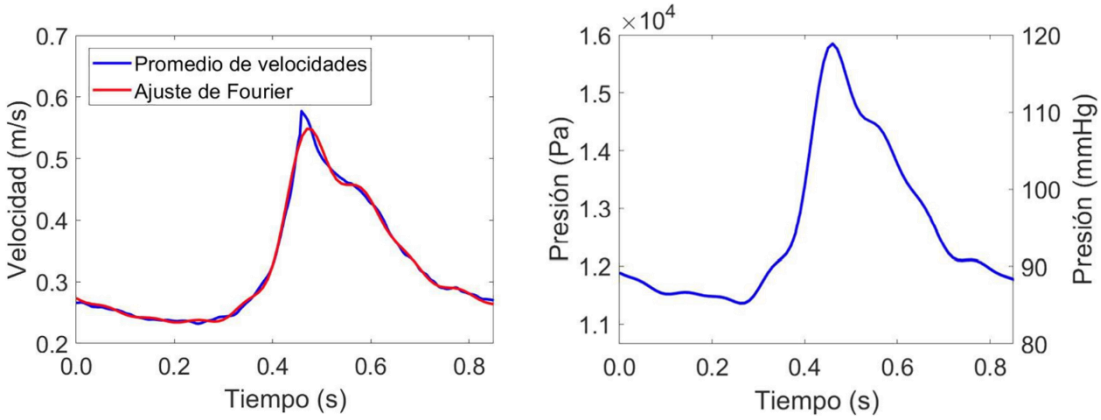


figura 7.1: Perfil de velocidad y de Presión utilizado en las simulaciones.
Fuente: Tesis Amigo Ahumada (2010) [16].

Respecto al modelo Viscoso, N. Amigo utilizó el modelo de Casson con las constantes τ_0=0.009[Pa] μ_0=0.0036[Pa·s] y m=100.

7.1.2 Segmentación de Estructura Vorticial

Utilizando las geometrías segmentadas y las estructuras vorticiales se pueden segmentar estas en la región del aneurisma. Utilizando los módulos vtk v9.0.1 y Numpy v1.20.2 en Python 3.6.1 se programa un algoritmo que permite elegir si re-samplear los puntos de la malla, y luego obtener las estructuras vorticiales sólo en la zona segmentada. El resumen del algoritmo se detalla en la

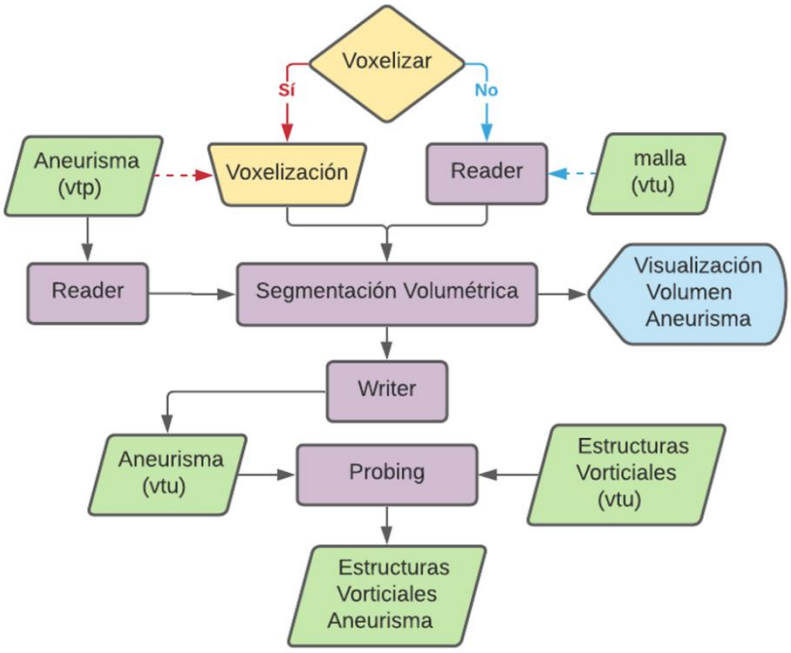


figura 7.2: Diagrama del algoritmo de segmentación de estructuras vorticiales.

Voxelización

Para poder comparar entre una geometría u otra este algoritmo permite re-samplear la malla en puntos cartesianos equiespaciados. Para esto se busca el cubo cartesiano en el cual está inscrita la geometría del aneurisma (Aneurisma.vtp) que luego se subdivide en voxeles de lado **ds**.

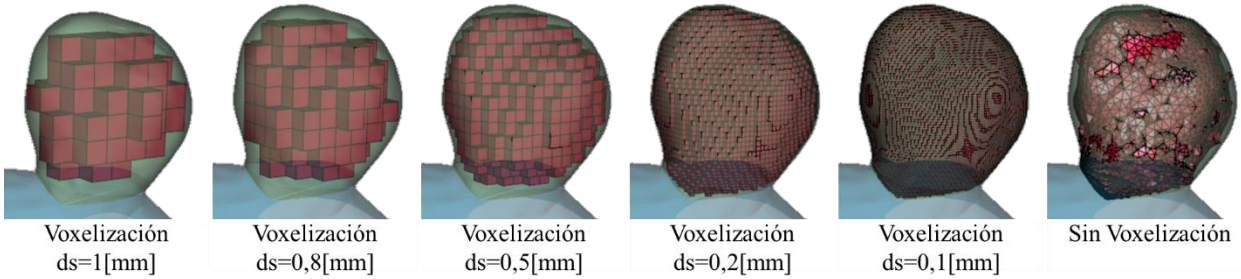


figura 7.3: Voxelización de geo20 para diferentes valores ds.

Si bien a menor **ds** mejor es la resolución esto aumenta la cantidad de nodos y celdas en la región segmentada. Si el **ds** es muy pequeño esto puede ocasionar que la región tenga muchos más nodos y celdas que la malla original completa. Si se trata sólo de la geometría esto no incrementa considerablemente el tamaño del archivo, no así, al momento de sondear las estructuras vorticiales esto si aumenta considerablemente el tamaño del archivo, por lo que se debe seleccionar con cuidado un valor de ds que permita una buena resolución y que no contenga muchos más nodos que la geometría sin Voxelización.

Segmentación Volumétrica

Para realizar la segmentación volumétrica se utilizan los filtros vtkSelectEnclosedPoints() y vtkMultiThreshold(). El primero recibe una superficie (vtp) y un volumen (vtu) y a cada punto del volumen asigna el valor de 1 si está dentro de la superficie y 0 si no lo está. Luego transforma el PointData a CellData.

Por otro lado, vtkMultiThreshold() revisa los valores de Point, Field y Cell Data y filtra las celdas según condiciones a configurar. Para el caso de las celdas al interior de la superficie lo que se debe hacer es buscar en que celdas el valor de todos sus nodos es 1.

Si se quisiera las celdas del borde se debe buscar que celdas tienen al menos un nodo con valor 0 y otro con valor 1. Para el caso de las celdas fuera de la superficie se debe buscar las celdas que contengan a todos sus nodos con valor 0.

Luego de tener el volumen segmentado este se exporta en una UnstructuredGrid (Aneurisma.vtu) que servirá de puntos de sondeo de las estructuras vorticiales del modelo completo.

Probing

En la última etapa del algoritmo se utiliza el filtro vtkProbeFilter(). Este toma los nodos de un **Input** y sondea todos los Field, Point y Cell Data en la malla **Source** realizando interpolación basada en la topología de **Source**. De esta manera, utilizando el volumen segmentado Aneurisma.vtu como input y la estructura vorticial de cada time-step se puede obtener las estructuras vorticiales exclusivas del aneurisma.

7.1.3 Selección del Valores Umbrales

Para caracterizar las estructuras vorticiales se debe seleccionar el valor umbral a utilizar en cada método. Una formade hacerlo es estudiando la distribución de los valores que toman λ_2, Q, $\mho$ y sus versiones normalizadas y adimensionales en diferentes instantes de tiempo. Generalmente tomar el valor umbral en el punto de inflexión en la identificación del punto como vórtice o flujo corriente entrega un resultado muy ruidoso, producto de vórtices extremadamente pequeños y valores que podrían no ser vórtices debido a la precisión de punto flotante (valores del orden de 10^{-6} o inferior).

Junto a la distribución a partir de histogramas de los valores se estudia también el % de vórtices a partir de la frecuencia acumulada, de derecha a izquierda en el caso de criterios de la forma "$x \leq$" y de izquierda a derecha en el caso de criterios de la forma "$x \geq$", asignando 100% al valor de inflexión en la decisión y 0% al valor extremo (máx o mín según corresponda).

7.1.4 Visualización de Estructuras Vorticiales

Para visualizar los vórtices se puede utilizar el filtro vtkContourFilter(), que utiliza el algoritmo de cubos marchantes generando IsoSuperficies para un valor dado de un Array especificado. El input de este filtro es una vtkUnstructuredGrid con al menos un Array de Point o Cell Data. Entre las variables de inicialización se indicará la cantidad y de cuál array se desea generar las IsoSuperficies, incluido el valor que genera cada una de estas. El output de vtkContourFilter() es una superficie PolyData con la(s) IsoSuperficies configuradas.

Otra forma de forma de visualizar las estructuras vorticiales es utilizando el filtro vtkMultiThreshold() con un procedimiento similar a a la segmentación Volumétrica. Para esto se debe transformar primero los valores de caracterización de vórtices desde PointData a CellData con el filtro vtkPointDataToCellData(). Luego se debe configurar los condicionales de vtkMultiThreshold() según el criterio (λ_2, Q o $\mho$) y el valor umbral a seleccionar.

La ventaja de utilizar MultiThreshold es que si se realizó la Voxelización de la geometría segmentada se conoce a priori el volumen de cada celda, luego con la cantidad de celdas que corresponden al vórtice se conocerá el volumen de la estructura.

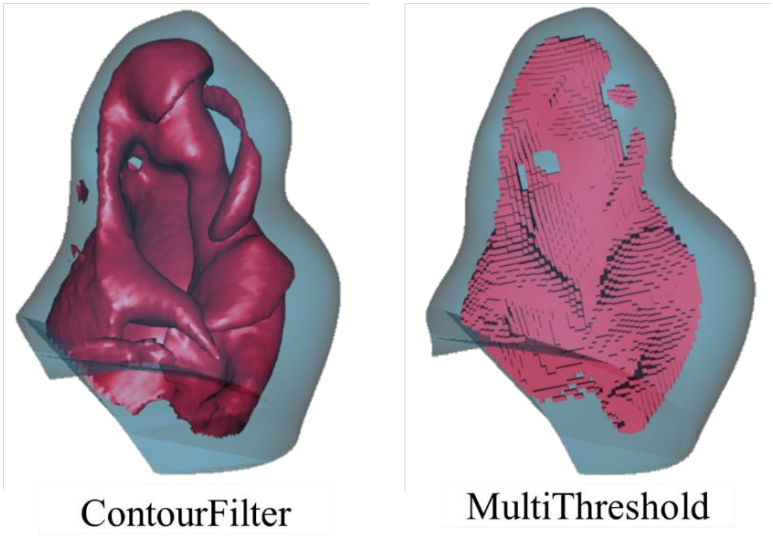


figura 7.4:Diferencias del uso de CountourFilter y MultiThreshold para computar las estructuras vorticiales.

7.1.5 Relación de Estructuras Vorticiales con Parámetros Medibles por CFD

Con el volumen del vórtice se puede calcular el porcentaje de volumen del aneurisma que corresponde a vórtices a lo largo del tiempo. Nicole Varble et. al. definen en [33] este parámetro con el criterio Q, sin embargo, es fácil de extrapolar a cualquiera de los otros dos criterios utilizados.

$$vVF_n = \frac{Volumen\ Vórtices\ Criterio\ n}{Volumen\ del\ Aneurisma} \tag{7.1}$$

Este parámetro de los vórtices se compara con los esfuerzos de corte máximos y mínimos en el ciclo cardiaco, para estudiar la posibilidad de relación entre las estructuras vorticiales y el impacto del flujo en la pared del aneurisma.

7.1.6 Implementación

A partir de las UDF de condiciones de borde implementadas en las simulaciones se seleccionan los time-step correspondientes al último ciclo cardiaco, donde la diástole ocurre en el time-step 2840 y la sístole en el time-step 3240.

En la figura 7.5 se pueden ver la componente temporal del perfil de velocidad implementado, la distribución temporal de los time-step almacenados durante la simulación y algunos time-step de interés para los resultados.

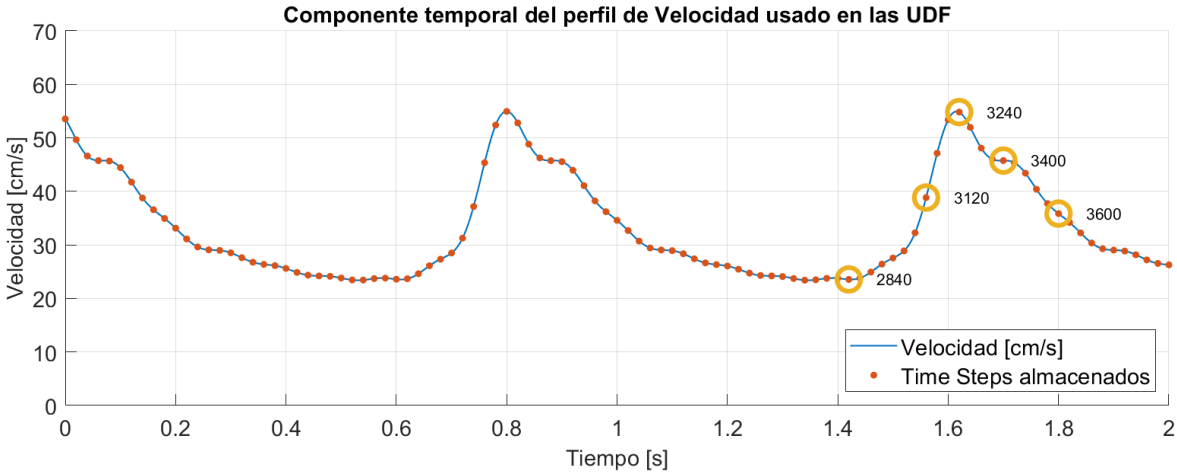


figura 7.5: Distribución temporal de la UDF de Velocidad y time-steps de la simulación.

El estudio se hace con 5 modelos de las geometrías que se logró segmentar:

Tabla 7.2: Geometrías con que se realizó estudio de estructuras vorticiales.

Geometría	Ruptura	Localización	N° Nodos	N° Elementos
geo11	Si	ACA - Terminal	227.539	1.273.124
geo16	Si	ICA - Lateral	85.997	454.373
geo19	Si	ACA - Lateral	18.293	87.926
geo20	No	ICA - Lateral	239.438	1.266.253
geo30-34	No	VBA - Lateral	126.953	687.320

7.2 Resultados

7.2.1 Distribución de Criterios de Caracterización de Vórtices durante la Diástole

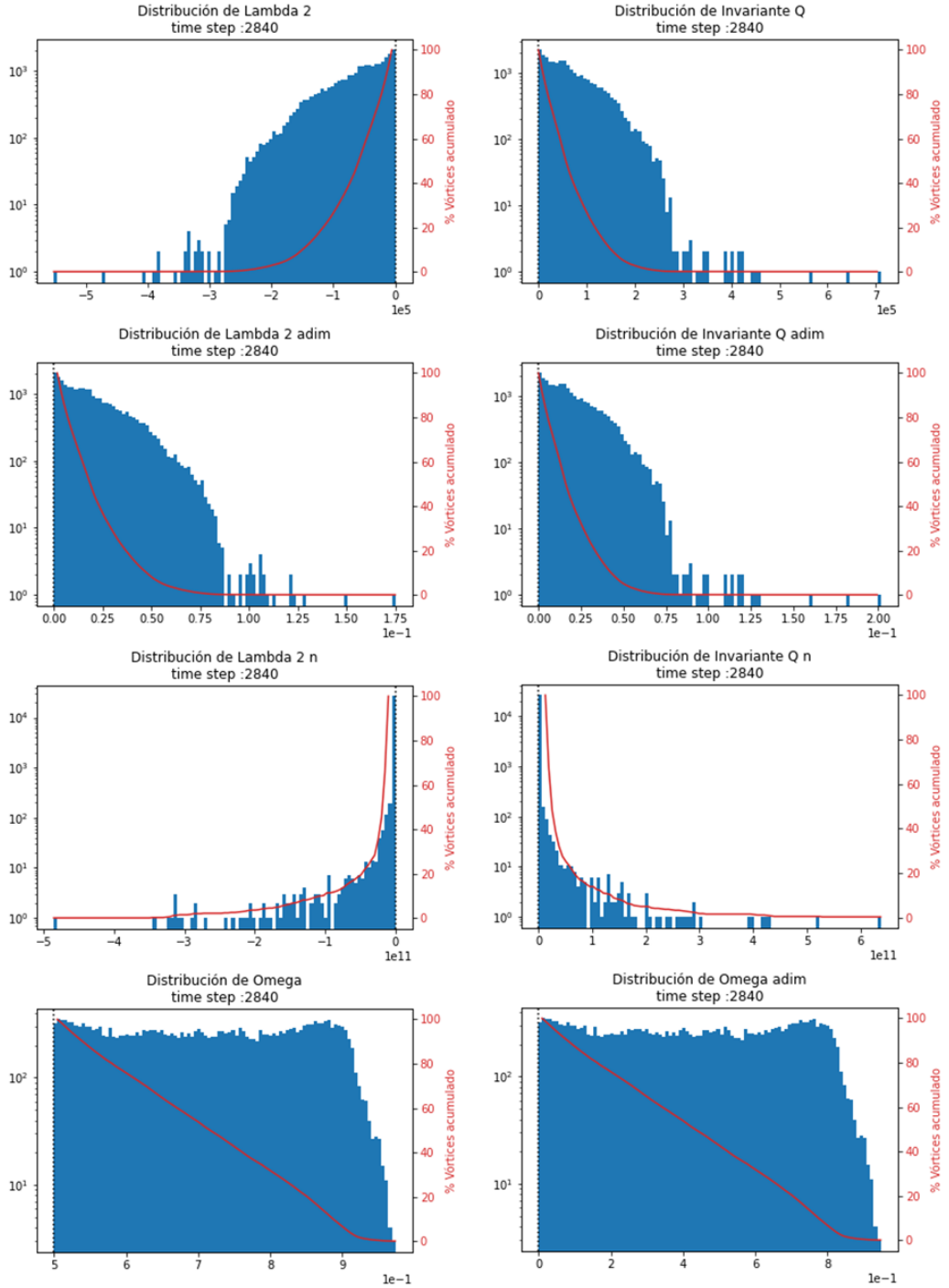


figura 7.6: Distribución criterios de caracterización de vórtices en geo11 durante la diástole.

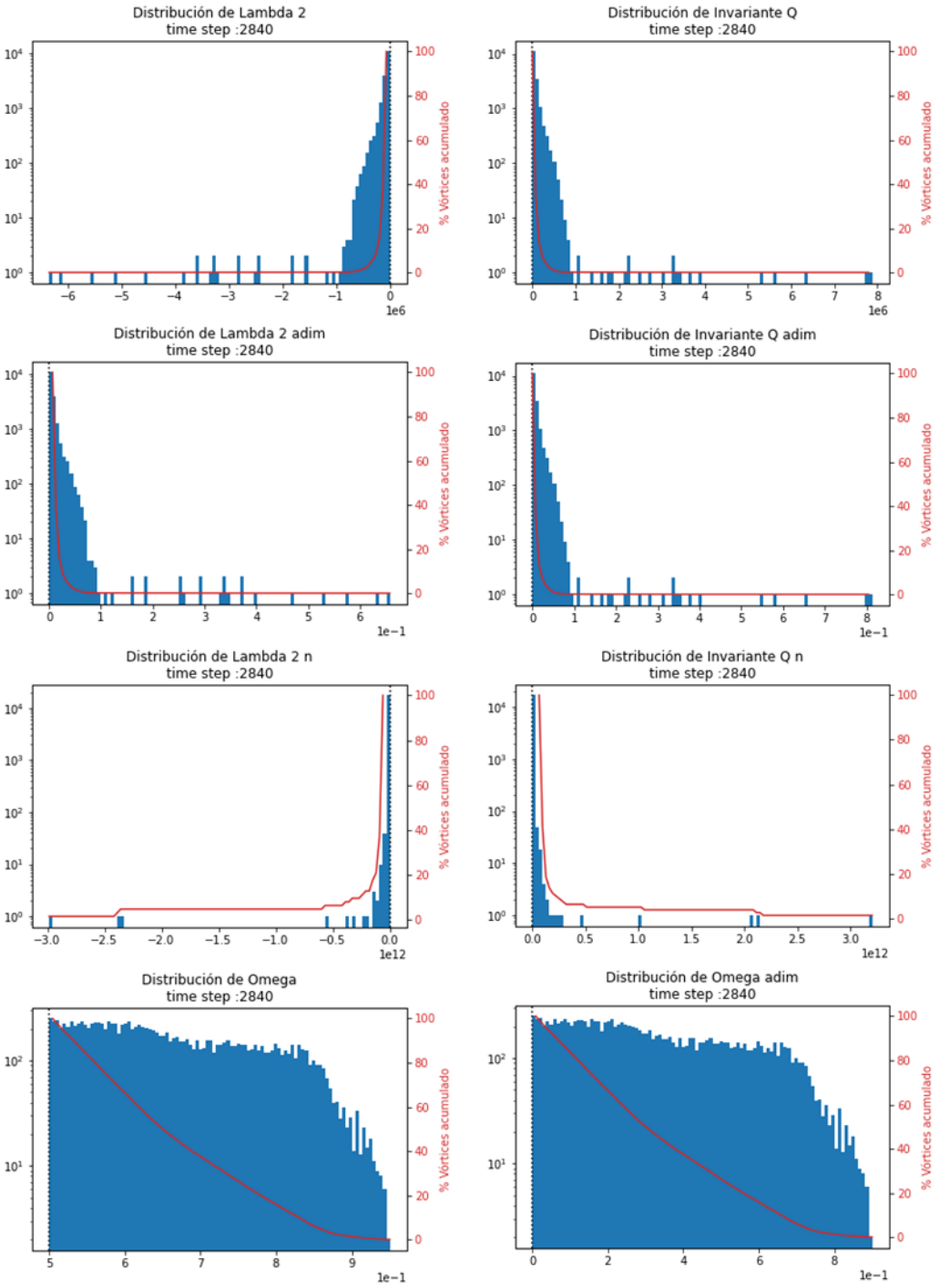


figura 7.7: Distribución criterios de caracterización de vórtices en geo16 durante la diástole.

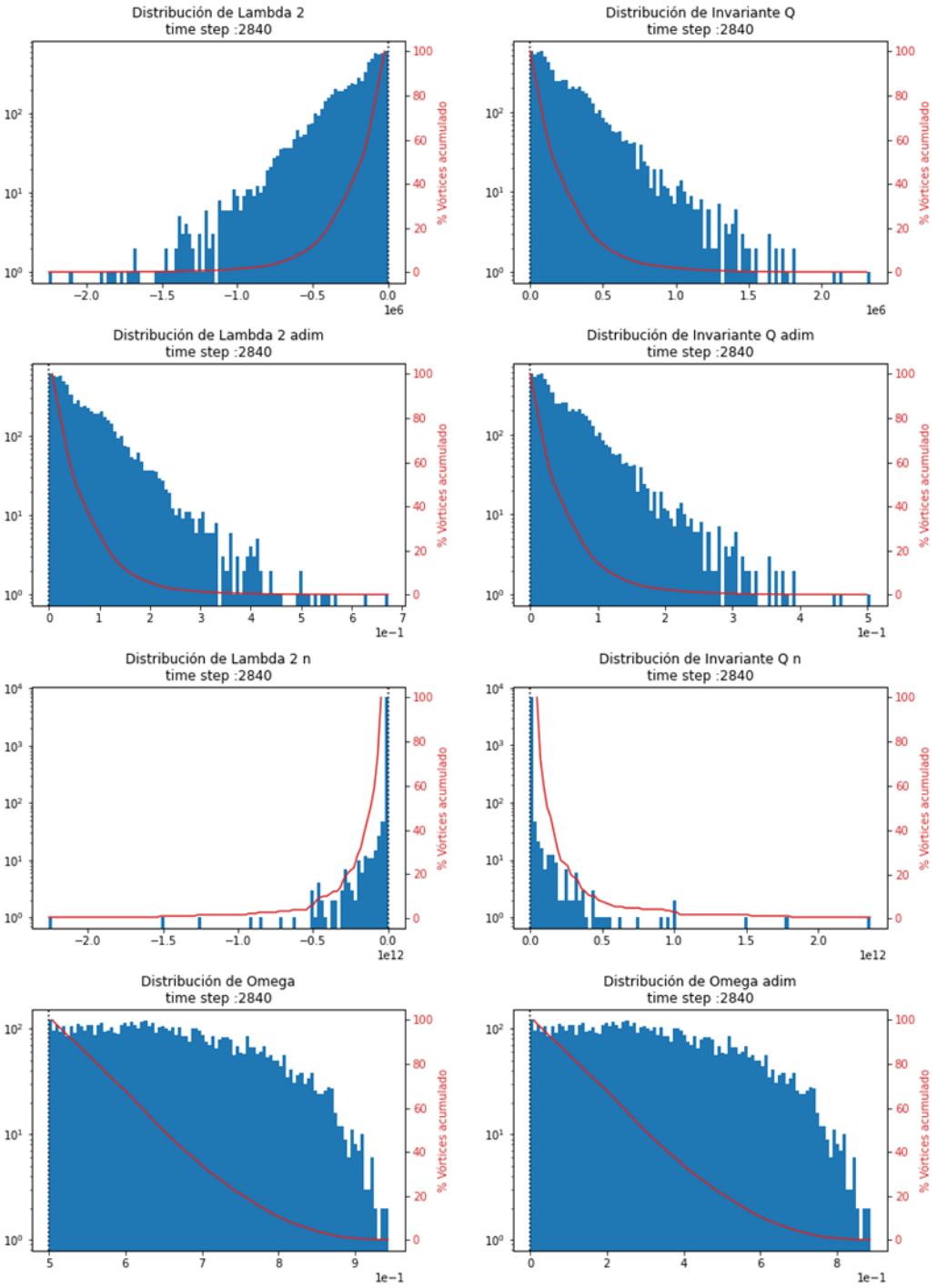


figura 7.8: Distribución criterios de caracterización de vórtices en geo19 durante la diástole.

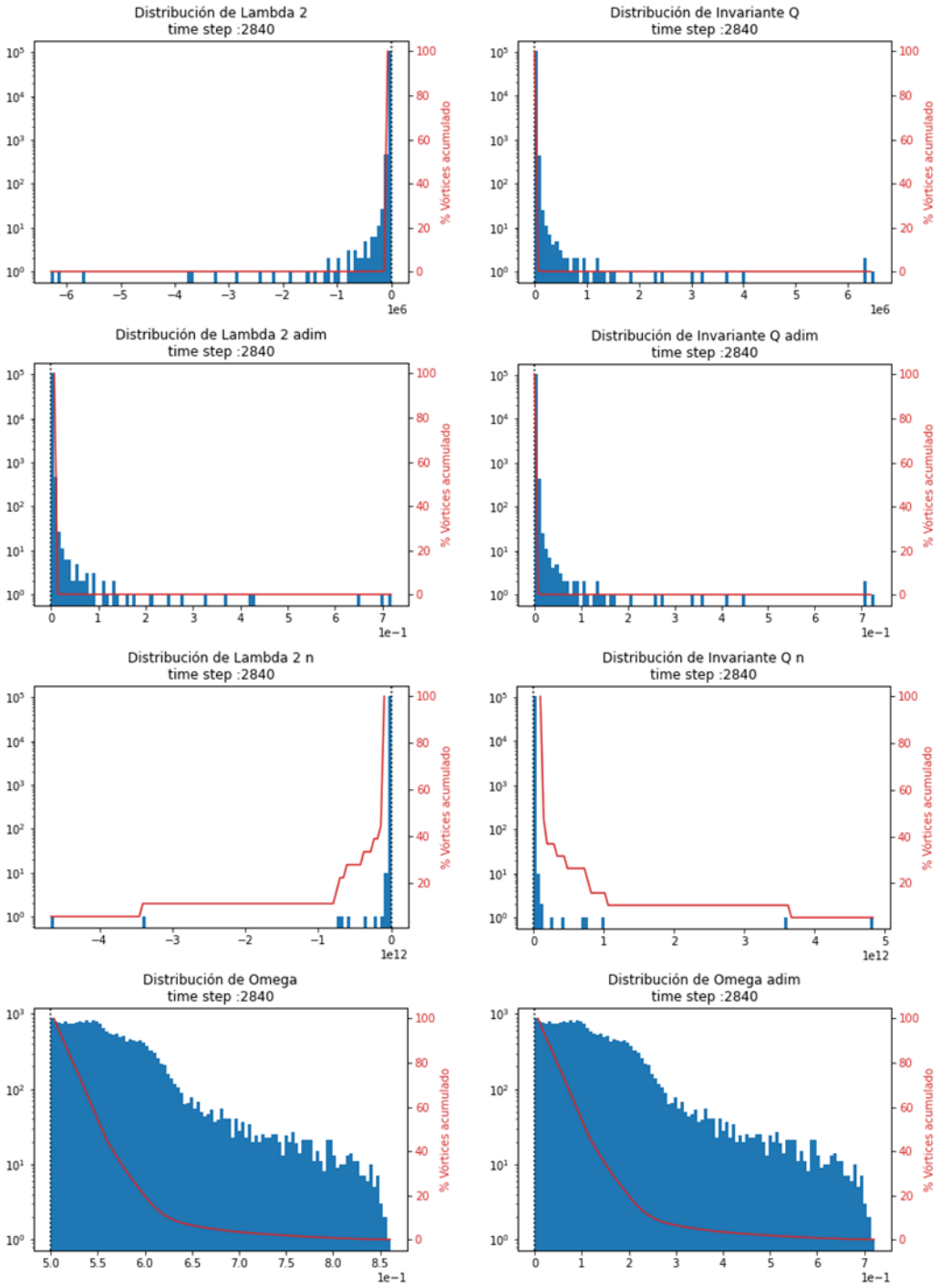


figura 7.9: Distribución criterios de caracterización de vórtices en geo20 durante la diástole.

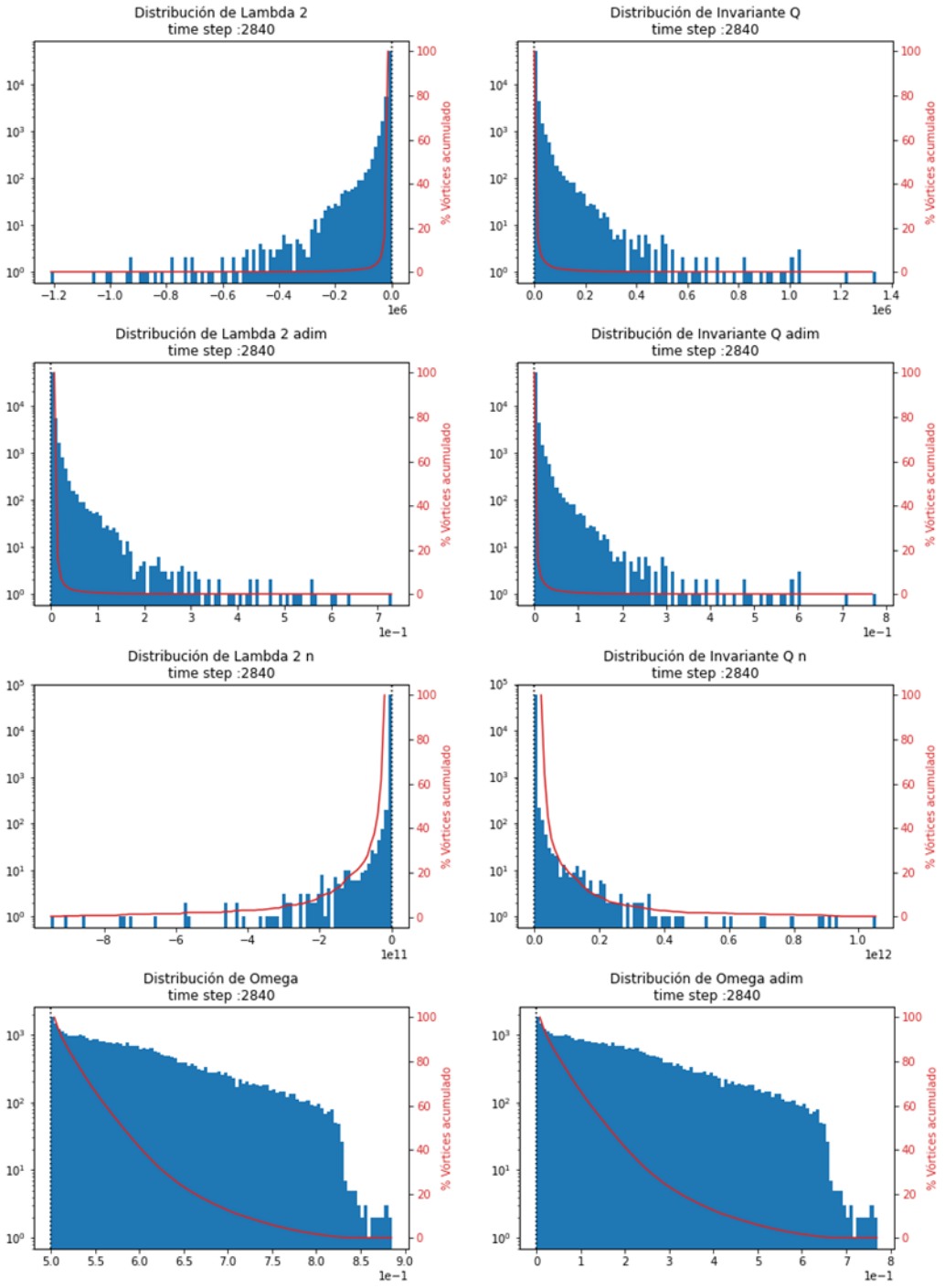


figura 7.10: Distribución criterios de caracterización de vórtices en geo30_34 durante la diástole.

7.2.2 Distribución de Criterios de Caracterización de Vórtices durante la Sístole

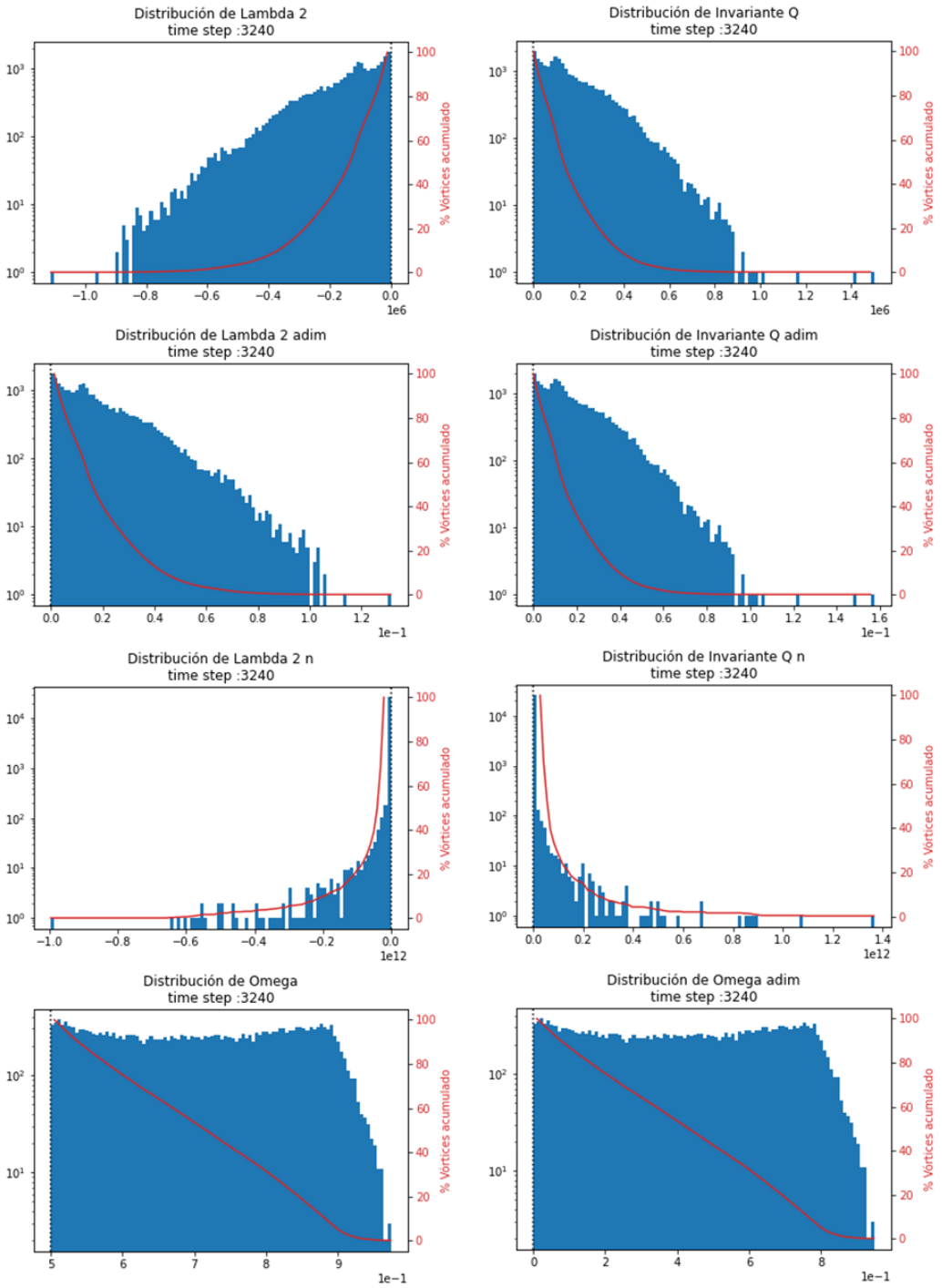


figura 7.11: Distribución criterios de caracterización de vórtices en geo11 durante la sístole.

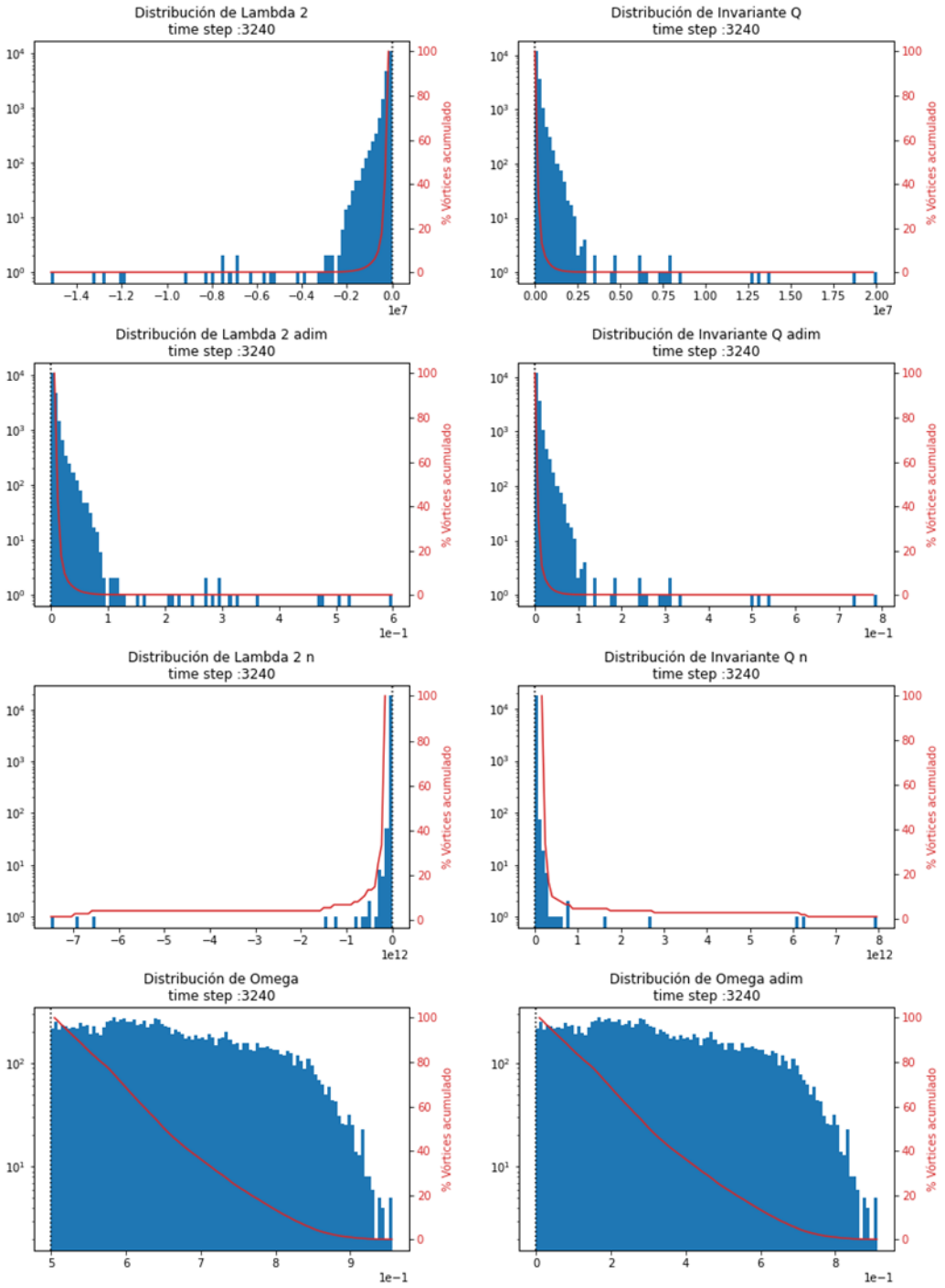


figura 7.12: Distribución criterios de caracterización de vórtices en geo16 durante la sístole.

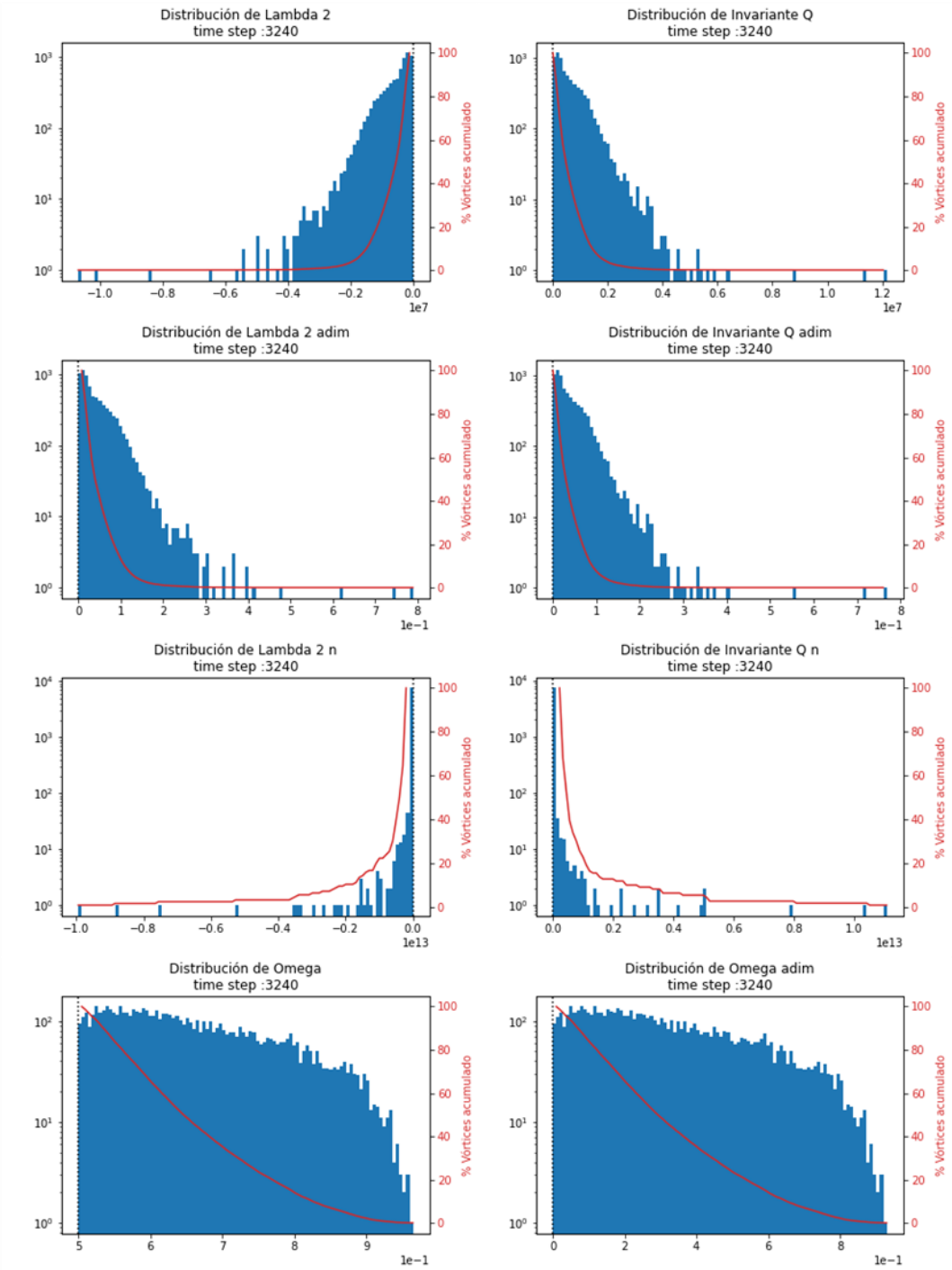


figura 7.13: Distribución criterios de caracterización de vórtices en geo19 durante la sístole.

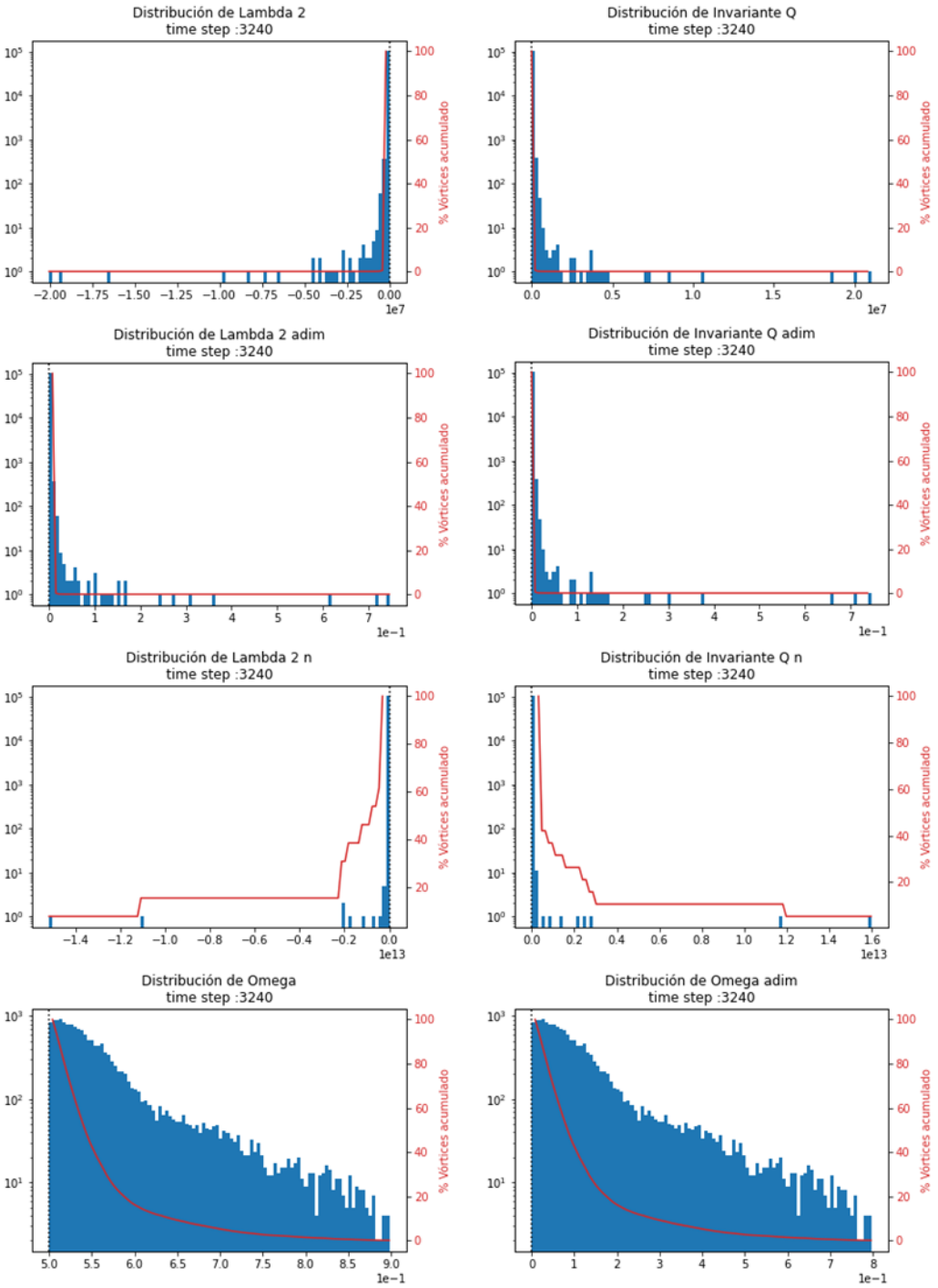


figura 7.14: Distribución criterios de caracterización de vórtices en geo20 durante la sístole.

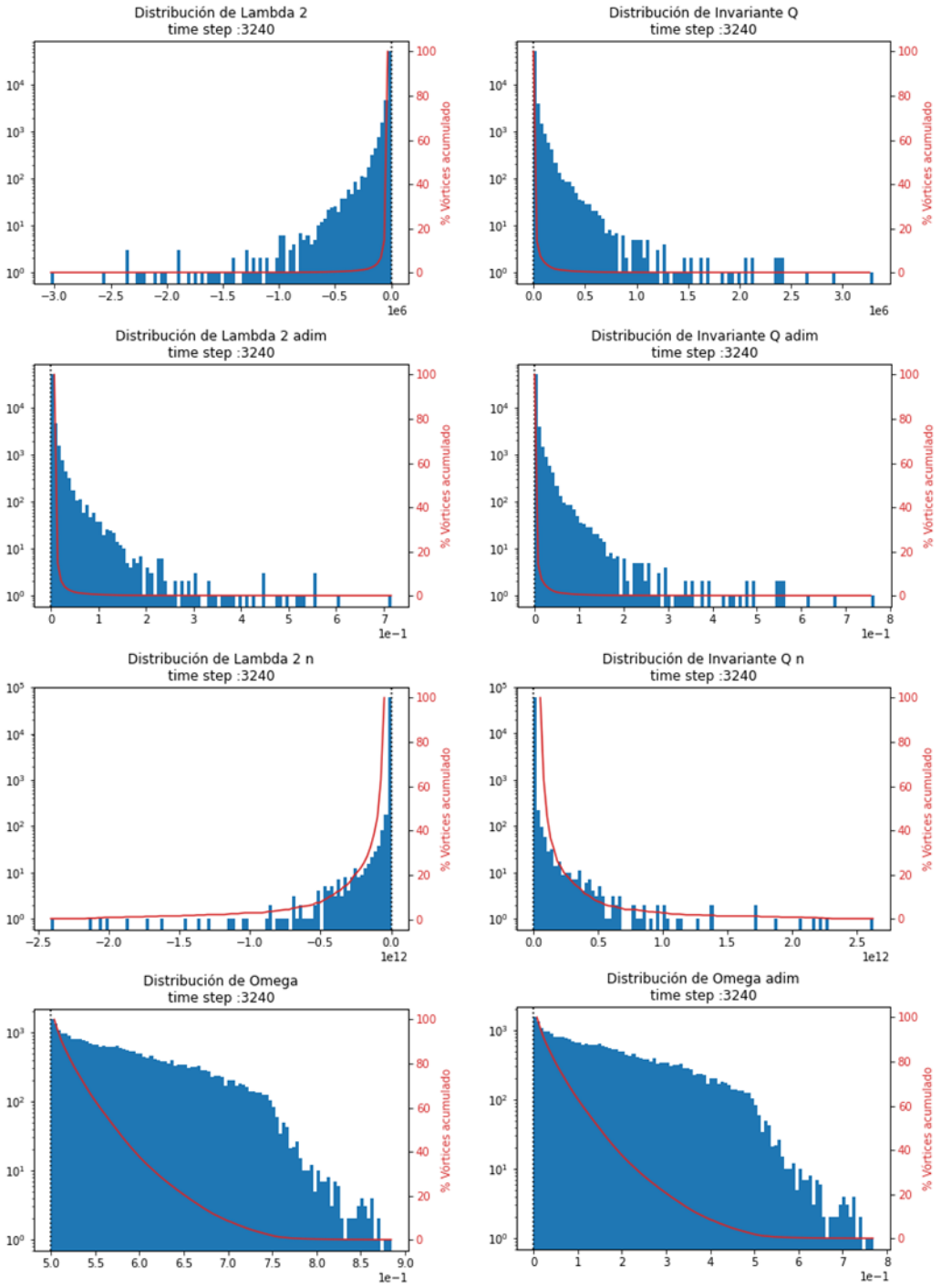


figura 7.15: Distribución criterios de caracterización de vórtices en geo30_34 durante la sístole.

7.2.3 Visualización de Estructuras Vorticiales

Para efectos de visualización de estructuras vorticiales se obtiene una superficie más continua utilizando el método de isosuperficie. A modo de ejemplo en el ciclo cardiaco se grafica en 3D la isosuperficie de la geometría "geo1" con el criterio $\mho \geq 0.51$ en diferentes instantes del ciclo cardiaco.

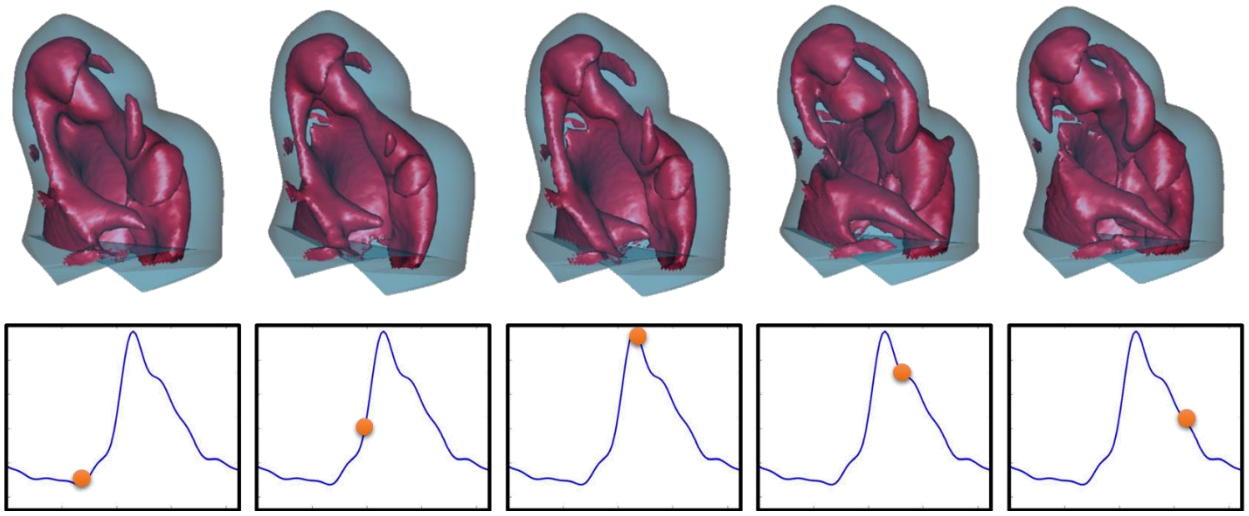


figura 7.16: Estructura vorticial de geo1 en un ciclo según $\mho = 0.51$ en diferentes instantes del ciclo cardiaco.

7.2.4 vVF y WSS en el ciclo Cardiaco

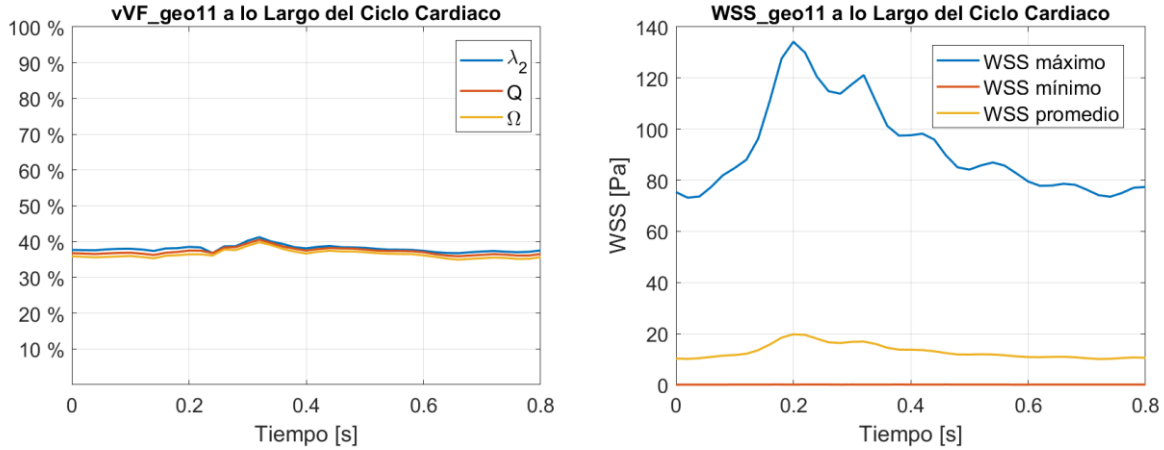


figura 7.17: vVF y WSS de geo11 a lo largo del ciclo cardiaco.

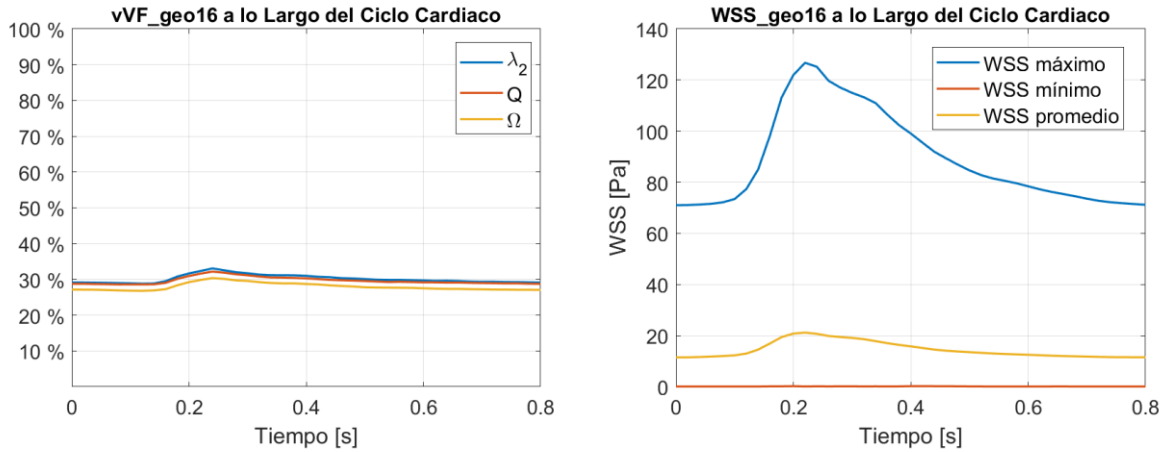


figura 7.18: vVF y WSS de geo16 a lo largo del ciclo cardiaco.

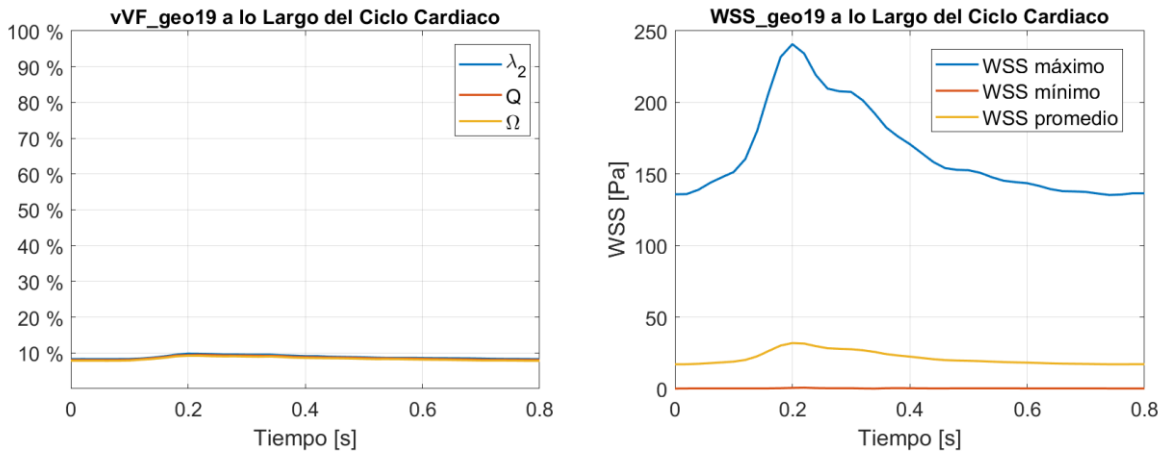


figura 7.19: vVF y WSS de geo19 a lo largo del ciclo cardiaco.

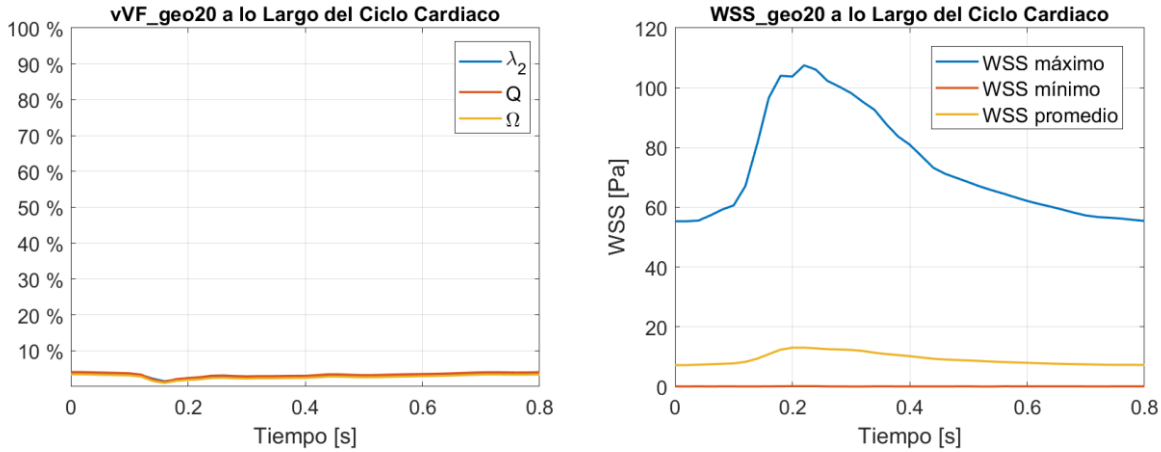


figura 7.20: vVF y WSS de geo20 a lo largo del ciclo cardiaco.

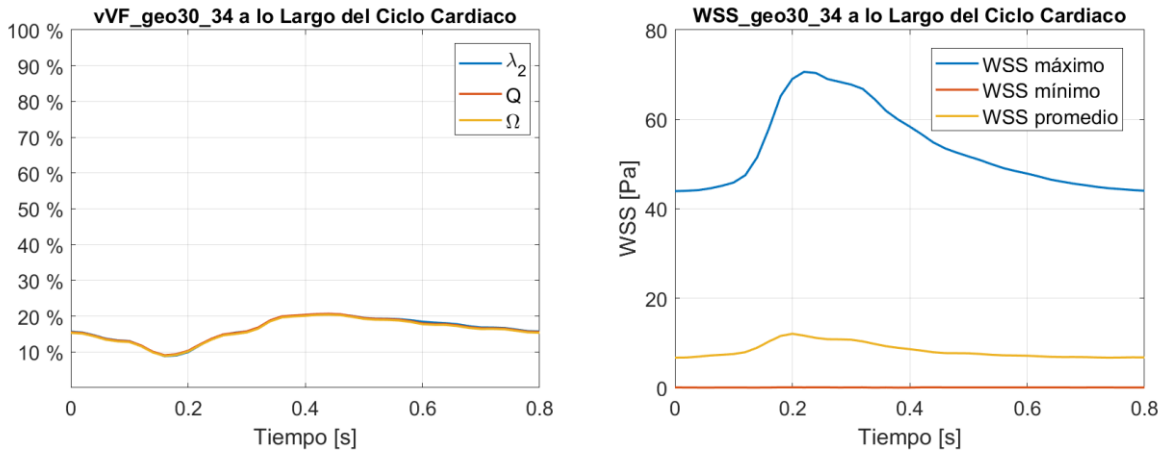


figura 7.21: vVF y WSS de geo30_34 a lo largo del ciclo cardiaco.

7.3 Discusión

Retomando la base de las simulaciones, los modelos fueron mallados con una densidad de 1500[celdas/mm^3] lo que significa que el **ds** promedio de cada celda es de 0.0873[mm], luego el valor ds = 0.1[mm] y ds=0.2 [mm] corresponden a una buena elección, destacando que el aumento de tiempo de cómputo de la Voxelización con ds=0.2[mm] es del orden de 1 a 2 [min], por otro lado, el caso ds=0.1[mm] es del orden de 4 a 10 [min].

La distribución de los valores de los criterios de cómputo de estructuras vorticiales λ_2 y Q entregan resultados similares para los casos estudiados. Para la versión original y normalizada se aproxima a una reflexión entre un criterio y el otro, por otro lado, para la versión adimensional unitaria sería difícil distinguir entre uno u otro.

El criterio $\mho$ en cambio presenta una distribución notablemente diferente a los otros dos criterios, donde en lugar de concentrar los valores en el umbral de selección se distribuyen de manera mucho más uniforme en el espectro.

Las versiones normalizadas (propuestas por [4]) ecualizan la distribución de λ_2 y Q, concentrando los vórtices más cercano al valor umbral de selección. Hay que destacar que esto no asemeja al criterio $\mho$ donde por el contrario se distribuye de manera más homogénea los valores. En otras palabras, lo que hace es ecualizar agudizando la curva de % de vórtices acumulados (la curva roja en los histogramas).

Cuando se trata de visualizar las estructuras vorticiales el algoritmo de IsoSuperficies representa una mejor opción, obteniendo un resultado mucho más continuo y uniforme, en comparación al algoritmo de MultiThreshold que muestra una versión voxelizada de la estructura vorticial que está directamente relacionada con la densidad del algoritmo de Voxelización.

En la figura 7.17 se observa que, a diferencia de las otras geometrías, la geo11 presenta un vVF mayor que las otras estudiadas, y el perfil de WSS máximo tiene una oscilación mucho más agitada. En cambio, en los otros casos presentan un perfil temporal de WSS más suave.

También existe una diferencia respecto al rango de valores de WSS máximo, donde en las geometrías con ruptura (geo11, geo 16 y geo 19) es mayor al caso de las geometrías sin ruptura (geo 20 y geo 30_34), lo que concuerda con lo dicho en [3] respecto al mecanismo por inflamación donde los WSS terminan rompiendo la pared en la zona debilitada.

8 Conclusión

En esta tesis se utilizaron algoritmos para segmentación de mallas 3d aplicados a aneurismas cerebrales para aislar los resultados obtenidos en simulaciones CFD para analizar exclusivamente las estructuras vorticiales.

El principal contratiempo y lo que tomó la mayoría de este trabajo es la comprensión y la metodología de ensayo y error para utilizar correctamente los módulos VTK y VMTK mayoritariamente en los fundamentos y estructura de los objetos de estos.

El algoritmo base para la segmentación utilizado corresponde a una adaptación del modelo propuesto en [1] a aneurismas cerebrales. Lamentablemente el algoritmo no es muy eficiente en las geometrías probadas resultando en segmentaciones exitosas en el 67.57% de los casos. Entre los problemas más comunes encontrados con la segmentación se hayan deficiencias dimensionales, donde en la mayoría de los casos donde el algoritmo falló existía un vaso sanguíneo muy pequeño respecto al resto de la geometría. De esta manera la falla se podría atribuir a errores en la precisión del cálculo, donde al estar trabajando en [mm] el orden de magnitud de la medida entre vértices del STL de 10^{-5}.

El algoritmo de identificación de vórtices entrega buenos resultados en comparación a los resultados entregados por ANSYS. Aun así, una validación del modelo de manera independiente comprobaría la robustes del mismo.

De los 3 métodos para caracterización de vórtices el criterio $\mho$ muestra ser el más rápido de utilizar, dado que no es necesario buscar un valor umbral a lo largo del espectro, pues con valores de $\mho = 0.51$ y $\mho = 0.52$ se obtienen los resultados esperados. Por otro lado, el criterio λ_2 es el que presenta mejores resultados después de realizar la inspección del espectro de valores, caracterizando vórtices más robustos con menor cantidad de discontinuidades que el criterio Q.

En conjunto con los histogramas y el porcentaje acumulado de vórtices denotan la poca efectividad de las versiones normalizadas de λ_2 y Q. Se puede notar en las figuras 7.6 a 7.10 que la curva de vórtices acumulados tiene una subida más abrupta respecto a la escala en que se muestra el histograma. Tomando en cuenta por otro lado la escala del eje x (del orden de 10^{11} en comparación a los otros 10^{5}) asemeja un poco al criterio $\mho$ la búsqueda de un umbral por inspección, donde cambios pequeños en el valor seleccionado no generan grandes diferencias en el resultado encontrado.

De manera general se logra el objetivo principal de estudiar las estructuras vorticiales en los modelos de aneurismas cerebrales y se establecen herramientas computacionales funcionales para estudiarlos de manera general.

Al igual que lo reportado en el estudio con 204 aneurismas en [33], el valor medio y el rango de vVF no representa una tendencia clara en la ruptura del aneurisma. Sin embargo, con una muestra mayor de perfiles se podría estudiar la forma del perfil, ya que para el resultado de este trabajo se observa una disminución del vVF durante la sístole en los aneurismas no rotos en comparación a los rotos donde el vVF presenta un leve aumento en la sístole. Esto podría formular

la estrategia del tratamiento estableciendo mecanismos que promuevan o disminuyan la cantidad de vórtices durante la sístole.

8.1 Trabajo Futuro

La principal optimización al trabajo actual es aumentar la cantidad de estadística obtenida. Con un poco más de desarrollo de los algoritmos se podría optimizar el cálculo de los resultados, así como el tiempo que tardan estos.

También es necesario mejorar el algoritmo de segmentación. Implementar otros algoritmos como el de campos dependientes de la concavidad podría mejorar los resultados obteniendo cortes más continuos y suaves. De esta manera se podrían tener valores más reales de los parámetros morfológicos del aneurisma, abriendo también la posibilidad de mapear en una superficie genérica los esfuerzos de corte en la superficie de este.

También implementar el criterio de caracterización de vórtices Liutex [31] podría generar más tipo de estadísticas y resultados, dado que además de la localización entrega información respecto a magnitud y dirección de los vórtices.

Un estudio más profundo al perfil de vVF y relacionarlo además con parámetros morfológicos y hemodinámicos podrían en conjunto entregar otro tipo de predicciones o correlaciones.

Bibliografía

[1] L. Antiga y D. A. Steinman, «Robust and Objective Decomposition and Mapping of Bifurcating Vessels,» *IEEE TRANSACTIONS ON MEDICAL IMAGING,* vol. 23, nº 6, pp. 704-713, 2004.

[2] S. Balocco, M. A. Zuluaga, G. Zahnd, S.-L. Lee y S. Demirci, «Arterial Flow Impact on Aneurysmal Hemodynamics,» de *Computing and Visualization for Intravascular Imaging and Computer-Assisted Stenting,* Academic Press, 2016, pp. 253-285.

[3] P. Texakalidis, A. Sweid, N. Mouchtouris, E. C. Peterson, C. Sioka, L. Rangel-Castilla, J. eavey-Cantwell y P. Jabbour, «Aneurysm Formation, Growth, and Rupture: The Biology and Physics of Cerebral Aneurysms,» *World Neurosurg,* vol. 130, pp. 277-284, 2019.

[4] K. Sunderland, C. Haferman, G. Chintalapani y J. Jiang, «Vortex Analysis of Intra-Aneurismal Flow in Cerebral Aneurysms,» *Computational and Mathematical Methods in Medicine,* 2016.

[5] J. Jiang y C. M. Strother, «Interactive Decomposition and Mapping of Saccular Cerebral Aneurysms Using Harmonic Functions; Its First Application With Patient-Specific Computational Fluid Dynamics (CFD) Simulations,» *IEEE TRANSACTIONS ON MEDICAL IMAGING,* vol. 32, 2013.

[6] P. M. Munarriz, P. A. Gómez, I. Paredes, A. M. Castaño-Leon, S. Cepeda y A. Lagares, «Basic Principles of Hemodinamycs and Cerebral Aneurysm,» *World Neurosurgery,* vol. 88, pp. 311-319, 2016.

[7] M. A. A. Sheikh, A. S. Shuib y M. H. H. Mohyi, «A review of hemodynamic parameters in cerebral aneurysm,» *Interdisciplinary Neurosurgery,* vol. 22, 2020.

[8] D. Kocur, N. Przybyłko, M. Niedbała y A. Rudnik, «Alternative Definitions of Cerebral Aneurysm Morphologic Parameters Have an Impact on Rupture Risk Determination,» *World Neurosurg,* pp. 157-164, 2019.

[9] J. L. Brisman, J. K. Song y D. W. Newell, «Cerebral Aneurysms,» *The New England Journal of Medicine,* pp. 928-939, 2006.

[10] G. J. Tortora y B. Derryckson, «Aparato Circulatorio: La Sangre,» de *Principios de la Anatomía y fisiología,* 11 ed., Medica Panamericana, 2006, pp. 670-698.

[11] A. Valencia, D. Ledermann, R. Rivera, E. Bravo y M. Galvez, «Blood flow dynamics and fluid–structure interaction,» *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS,* vol. 58, pp. 1081-1100, 2008.

[12] A. Apostolidis, A. Moyer y A. Beris, «Non-Newtonian effects in simulations of coronary arterial blood flow,» *Journal of Non-Newtonian Fluid Mechanics,* vol. 233, pp. 155-165, 2016.

[13] M. Abbasian, M. Shams, Z. Valizadeh, A. Moshfegh, A. Javadzadegan y S. Cheng, «Effects of different non-Newtonian models on unsteady blood flow hemodynamics in patient-specific arterial models with in-vivo validation,» *Computer Methods and Programs in Biomedicine,* vol. 186, p. 105185, 2020.

[14] J. R. Womersley, «Method for the calculation of velocity, rate of flow and viscous drag in arteries when the pressure gradient is know.,» *Philosophical Magazine,* vol. 127, pp. 553-563, 1955b.

[15] J. R. Womersley, «Oscillatory motion of a viscous liquid in a thin-walled elastic tube—I: The linear approximation for long waves,» *Philosophical Magazine,* pp. 199-221, 1955a.

[16] N. R. Amigo Ahumada, Á. Valencia Musalém, W. Calderón Muñoz, A. Guzmán Cuevas, E. Finol y J. Ortega Palma, Caracterización morfológica y estudio de la hemodinámica de aneurismas cerebrales humanos mediante simulaciones computacionales, (Tesis para optar al grado de doctor en ciencias de la ingeniería mención fluidodinámica). Santiago: Universidad de Chile, 2018.

[17] Y. Aboelkassem y Z. Virag, «A hybrid Windkessel-Womersley model for blood flow in arteries,» *Journal of Theoretical Biology,* vol. 462, pp. 499-513, 2019.

[18] 118 Contributors, Visualization Handbook, C. D. Hansen y C. R. Johnson, Edits., Butterworth-Heinemann, 2005.

[19] W. Schroeder, K. Martin y B. Lorensen, The Visualization Toolkit. An Objetc-Oriented Approach To 3D Graphics, 4.1 ed., 2018.

[20] W. E. Lorensen y H. E. Cline, «Marching Cubes: A High Resolution 3D Surface Construction,» *Computer Graphics,* nº 21.3, pp. 163-169, 1987.

[21] P. Theologou, I. Pratikakis y T. Theoharis, «A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation,» *Computer Vision and Image Understanding,* vol. 135, pp. 49-82, 2015.

[22] L. Antiga, B. Ene-Iordache y A. Remuzzi, «Computational geometry for patient-specific reconstruction and meshing of blood vessels from MR and CT angiography,» *IEEE Transactions on Medical Imaging,* vol. 22, nº 5, pp. 674-684, 2003.

[23] O. Kin-Chu Au, Y. Zheng, M. Chen, P. Xu y C.-L. Tai, «Mesh Segmentation with Concavity Aware Fields,» *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS,* vol. 18, nº 7, pp. 1125-1134, Julio 2012.

[24] M. Jiang, R. Machiraju y D. Thompson, «Detection and Visualization of Vortices,» de *Visualization Handbook*, C. D. Hansen y C. R. Johnson, Edits., Butterworth-Heinemann, 2005, pp. 295-309.

[25] J. Jeong y F. Hussain, «On the Identification of a Vortex,» *Journal of Fluid Mechanics,* vol. 285, pp. 69-94, 1995.

[26] V. Holmén, A. Sopasakis, J. Revstedt y K. Åström, *Methods for Vortex Identification,* Lund: Lund University, Mathematics (Faculty of Engineering), 2012.

[27] L. Chaoqun, G. Yi-sheng, D. Xiang-rui, W. Yi-qian, L. Jian-ming y Z. Yu-ning, «Third generation of vortex identification methods: Omega and Liutex/Rortex based systems,» *Journal of Hydrodynamics,* vol. 31, nº 2, pp. 205-233, 2019.

[28] C. Liu, H. Xu, X. Cai y Y. Gao, Liutex and Its Applications in Turbulence Research, Academic Press, 2021.

[29] Y. Wang, Y. Yang, G. Yang y C. Liu, «DNS Study on Vortex and Vorticity in Late Boundary Layer Transition,» *Communications in Computational Physics,* vol. 22, nº 2, pp. 441-459, 2017.

[30] R. Cucitore, M. Quadrio y A. Baron, «On the effectiveness and limitations of local criteria for the identification of a vortex,» *European Journal of Mechanics - B/Fluids,* vol. 18, nº 2, pp. 261-282, 1999.

[31] C. Liu, Y. Gao, S. Tian y X. Dong, «Rortex—A new vortex vector definition and vorticity tensor and vector decompositions,» *Physics of Fluids,* vol. 30, 2018.

[32] I. E. Vignon-Clementel, C. A. Figueroa, K. E. Jansen y C. A. Taylor, «Outflow boundary conditions for 3D simulations of non-periodic blood flow and pressure fields in deformable arteries,» *Computer Methods in Biomechanics and Biomedical Engineering,* vol. 13, nº 5, pp. 625-640, 2010.

[33] V. N, T. G, X. J, S. K y M. H, «Identification of vortex structures in a cohort of 204 intracranial aneurysms,» *Journal of the Royal Society,* vol. 14, p. 20170021, 2017.

Anexos

Anexo-A. Tipos de Celdas lineales en VTK

Para evitar confusiones se mantendrá en inglés el nombre de las celdas.

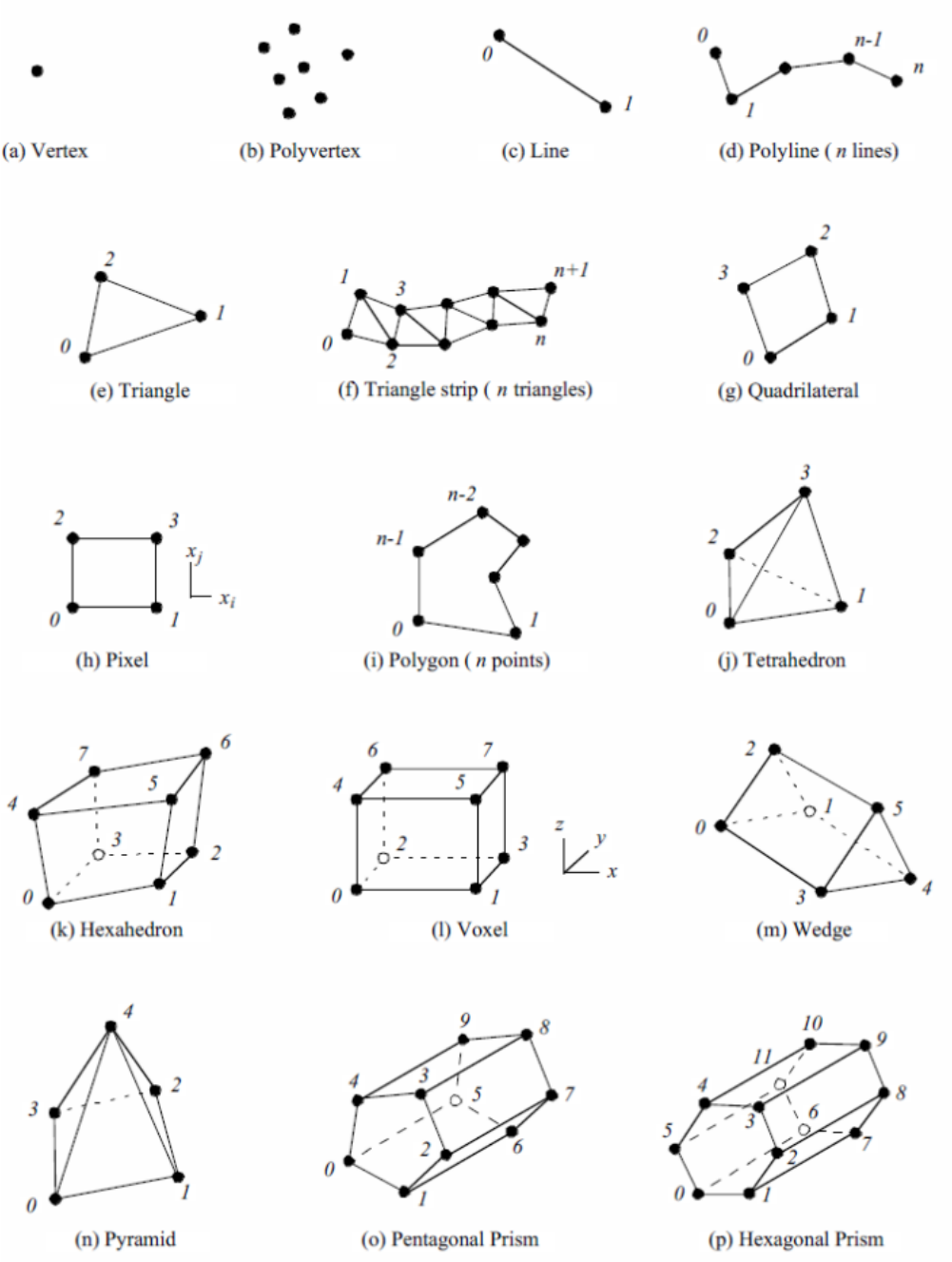


figura 8.1: Celdas lineales.

Vertex: Celda cero-dimensional, conformada por 1 punto.

PolyVertex: Celda cero-dimensional compuesta, conformada por un numero arbitrario de puntos.

Line: Celda unidimensional definida por 2 puntos.

PolyLine: Celda unidimensional compuesta, definida por una lista de n+1 puntos que forman n líneas.

Triangle: Celda bidimensional, conformada por 3 puntos. El orden de los puntos genera el vector normal contrario a las agujas del reloj.

TriangleStrip: Celda bidimensional compuesta, conformada por más de 1 triangulo. Estos no deben estar necesariamente en un mismo plano. Los triángulos se forman por puntos consecutivos (p_i, p_{i+1}, p_{i+2}).

Quadrilateral: Celda bidimensional definida por 4 puntos en un mismo plano. Sus aristas no se cruzan y el vector normal se define con la regla de la mano derecha del orden de los puntos. También debe ser convexa.

Pixel: Celda bidimensional definida por 4 puntos con aristas perpendiculares. El orden de los puntos se define en base a la malla, donde el índice crece en función con las coordenadas (primero en x, luego en y, después en z).

Polygon: Celda bidimensional formada por al menos 3 puntos en un mismo plano. Sus vértices no se deben cruzar.

Tetrahedron: Celda tridimensional, conformada por 4 puntos que no pertenecen a un mismo plano. Sus caras y aristas no se intersectan y es de volumen convexo.

Hexahedron: Celda tridimensional, conformada por 6 puntos que no pertenecen a un mismo plano. Sus caras y aristas no se intersectan y es de volumen convexo.

Voxel: Celda topológicamente equivalente al Hexahedron, pero con algunas restricciones. Sus caras son perpendiculares a algún eje coordenado, sus aristas son paralelas al eje coordenado, y el índice de los puntos crece en función con las coordenadas (primero en x, luego en y, después en z).

Wedge: Celda tridimensional compuesta por 3 caras cuadradas, 2 triangulares, 9 aristas y 6 vértices. Debe ser convexo.

Pyramid: Celda tridimensional compuesta por 4 caras triangulares, 1 cara cuadrada, 8 aristas y 5 vértices. Debe ser convexo.

PentagonalPrism: Celda tridimensional compuesta por 5 caras cuadradas, 2 pentagonales, 15 aristas y 10 vértices. Debe ser convexo.

HexagonalPrism: Celda tridimensional compuesta por 6 caras cuadradas, 2 hexagonales, 18 aristas y 12 vértices. Debe ser convexo.