Extracción Automática de la Línea Central de Estructuras Tubulares: Implementación Matricial

B. Rodríguez Vila[1,2], F. Gayá Moreno[1,2], P. Sánchez-González[1,2],E.J. Gómez Aguilera[1,2]

[1] Grupo de Bioingeniería y Telemedicina, ETSI de Telecomunicación, Universidad Politécnica de Madrid, Madrid, España, {brvila,fgaya,egomez}@gbt.tfo.upm.es

[2] Centro de Investigación Biomédica en Red en Bioingeniería, Biomateriales y Telemedicina, CIBER-BBN

Resumen

Se propone una nueva implementación matricial de un algoritmo para la extracción automática de la línea central de estructuras tubulares. El algoritmo seleccionado calcula la línea central de estructuras complejas sin la necesidad de interacción con el usuario. En el trabajo se explica detalladamente cómo llevar a cabo la implementación matricial utilizando el lenguaje de computación de Matlab. La implementación matricial permite el cálculo de la línea central en pocos segundos, mejorando en varios grados de magnitud la implementación disponible en ITK.

1. Introducción y estado del arte

Con la disponibilidad de imágenes médicas 3D de alta resolución (TC o RM) existe un creciente interés en la reconstrucción y modelado de estructuras tubulares como el árbol bronquial, el tracto grastrointestinal y los vasos sanguíneos [1]-[4]. La extracción manual de la línea central requiere mucho tiempo, por lo que métodos automáticos y robustos facilitan enormemente este proceso.

La línea central es un potente descriptor de la forma de estas estructuras tubulares. Este preprocesado resulta en una representación espacial más simple, mientras que al mismo tiempo se conserva la topología de la estructura original. Esta información puede ser usada en muchas tareas de procesado de imagen como registro (usando la línea central como puntos de referencia) [5], segmentación, cuantificación de la estenosis [3] o visualización y planificación de operaciones quirúrgicas [2][4]. Sin embargo, aunque el concepto de qué es una línea central es más o menos intuitivo, su definición matemática no es única.

Se han propuesto muchos métodos en la literatura para el cálculo de líneas centrales. Éstos se pueden clasificar teniendo en cuenta la información de entrada.

La primera categoría de métodos son los basados en imágenes de intensidades de gris [6][7]. Esta categoría usa las derivadas de primer y segundo orden de la imagen para definir la línea central. Para ello se define una variable denominada “medialness” (de difícil traducción al español) que mide, en un determinado punto y escala, el grado de pertenencia del punto al eje central de la estructura. La segunda categoría se basa en imágenes binarias segmentadas de las estructuras tubulares (vasos sanguíneos, bronquios, etc.) y extrae la línea central como el esqueleto de esta imagen. Existen diversos métodos para el cálculo del esqueleto de una estructura (“skeletonization” en inglés) basados en teoría de grafos [8], basados en la superficie 3D de la segmentación [3], o en algoritmos de adelgazamiento topológico [4].

Dentro de esta última categoría destaca el algoritmo propuesto por Bouix y Siddiqi [9]. Este método presenta ciertas ventajas sobre otros de la literatura:

- La selección de puntos de anclaje no se hace de forma heurística, sino basada en un algoritmo cuyas propiedades teóricas ya han sido justificadas [10].
- Calcula todos los caminos de la línea central en estructuras tubulares complejas, mientras otros métodos están diseñados para hallar un único camino de la línea central cada vez.
- La aproximación puede hacerse completamente automática y sin interacción del usuario. Otros métodos requieren la selección, al menos, de los puntos finales de un determinado camino.

Debido a estas ventajas este algoritmo ha sido incluido en una librería tan conocida como Insight Toolkit (ITK) [11].

Sin embargo, y aunque los tiempos de ejecución que se presentan en el trabajo original son de unos pocos segundos, en el desarrollo disponible en ITK la ejecución lleva mucho más tiempo. Por esta razón se presenta una implementación del método seleccionado que permite hacer uso de la potente librería de cálculo vectorial utilizada por Matlab® para reducir considerablemente los tiempos de la implementación de ITK.

2. Descripción del método

El método consiste en la combinación de tres algoritmos: el primero calcula la información del flujo medio dentro de la estructura, el segundo realiza el adelgazamiento topológico usando la información del flujo medio (Average Outward Flux, AOF) para hallar los puntos de anclaje, y el tercer algoritmo realiza un refinamiento del esqueleto podando las ramas de menor tamaño que el radio de la estructura tubular. En las siguientes subsecciones se explican cada uno de los algoritmos.

2.1. Flujo medio

El flujo medio es una manera efectiva de distinguir entre puntos pertenecientes a la línea central y puntos que no, puesto que los primeros presentan valores de AOF

claramente negativos, mientras los segundos tienen valores cercanos a cero.

$$AOF(x) = \frac{1}{n}\sum_{i=1}^{26} < \widehat{N_i}, \nabla D(x_i) >$$

Donde x_i es el vecino en conectividad 26 de x, N_i es el vector que une x y x_i, y $D(x_i)$ es la transformación de distancia del objeto en el punto x_i.

2.2. Algoritmo de adelgazamiento topológico

La estrategia básica es guiar el adelgazamiento del objeto a partir de la transformación de distancia, teniendo cuidado de eliminar únicamente puntos simples. El proceso termina cuando todos los puntos supervivientes son no simples o tienen un valor de AOF (negativo) menor que un umbral establecido.

Un punto es simple si su eliminación no cambia la topología del objeto. Ésta clasificación se basa en dos variables que miden la topología de cada punto [12]. Un punto es simple si $C^* = 1$ (su eliminación no desconecta componentes conectados) y $\bar{C} = 1$ (su eliminación no crea agujeros o cavidades), donde:

C^*: el número de componentes 26-conectados adyacentes a x en $O \cap N_{26}^*$,

$\bar{C}$: el número de componentes 6-conectados adyacentes a x en $\bar{O} \cap N_{18}$.

Aquí O es el objeto con conectividad 26, N_{26}^* es el vecindario en conectividad 26 de x, excluyendo x, y N_{18} es el vecindario en conectividad 18 de x, incluyendo a x.

El algoritmo propuesto evalúa, uno a uno, los puntos de la frontera del objeto, y si son simples los inserta en una pila usando el valor de la transformación de distancia D para la ordenación de la pila. Si un punto es simple y no tiene AOF < umbral, el punto se elimina del objeto y se analizan todos sus puntos vecinos.

2.3. Algoritmo de poda

Puesto que el objetivo es extraer diversos caminos de la línea central (ej.: bronquios o vasos sanguíneos), éste algoritmo se encarga de eliminar únicamente las ramas del esqueleto que estén conectadas a un punto final y cuya longitud sea menor que un umbral. Un punto final se define como aquel punto de la línea central que tiene un único vecino en conectividad 26.

2.4. Automatización del método

El método presentado hasta ahora tiene 2 parámetros: el umbral para el AOF y el umbral de longitud para las ramas espurias. Los autores recomiendan el uso de un valor de umbral de AOF tal que el 25% de los valores se encuentren por debajo del umbral. En el caso de las ramas espurias, se recomienda usar el valor de radio de la estructura tubular de mayor interés. De esta manera los dos parámetros pueden ser calculados automáticamente a partir de la imagen de entrada.

3. Desarrollo matricial del método

En este apartado se presentan los desarrollos matriciales (donde sea posible), de los algoritmos presentados en el apartado anterior.

3.1. Flujo medio

La fórmula del AOF puede ser simplificada como la divergencia del gradiente del mapa de transformación de distancia del objeto. Las implementaciones de la divergencia y el gradiente en Matlab son matriciales, mientras que el cálculo de la transformación de distancia (bwdist( )) se realiza voxel a voxel. Debido a esto, el cálculo del mapa de distancias conlleva más de la mitad de todo el tiempo de procesado.

$$Flujo = divergence\left(gradient\left(bwdist(\sim imagen)\right)\right)$$

3.2. Adelgazamiento topológico

Éste es el apartado en el que se puede hacer una mayor aportación con un correcto uso de la biblioteca de cálculo vectorial.

La implementación matricial evalúa al mismo tiempo (y elimina, en el caso de que sean simples) todos los puntos que tengan asociado un mismo valor del mapa de distancias D. Cada conjunto de puntos con un mismo valor de D se divide en 8 subconjuntos de puntos, dependiendo de si sus coordenadas x,y,z son par o impar, para que la eliminación de un punto no afecte a ninguno de sus vecinos en la misma iteración. Para cada valor de distancia *dist* se evalúa (si los puntos son simples o no) secuencialmente las capas desde D = 1 hasta *dist*, teniendo en cuenta que cada capa está compuesta por 8 subconjuntos.

```
function evalua(dist)
   while (n° de puntos simples en dist > 0)
        for i=1:dist-1
                evalua(i)
```

Para poder realizar esto hay que ser capaz de calcular los valores de C^* y $\bar{C}$ (basados en el análisis topológico de entornos 3D alrededor de cada punto) para varios puntos al mismo tiempo, haciendo uso de matrices de 4 dimensiones.

Para el cálculo de C^* en una lista de puntos (caracterizados por su índice dentro de la imagen 3D) se hace uso de la matriz de posiciones relativas de los vecinos para obtener el índice a la imagen de los vecinos. Se calcula el valor de la imagen (0 o 1, puesto que es una imagen segmentada binaria) en las posiciones de los vecinos, poniendo el valor del punto en estudio a cero. Posteriormente se realiza un etiquetado de regiones usando la matriz de conectividad conn1.

$$conn1 = false(3,3,3,3);$$

$$conn1(2,:,:,:) = true;$$

$$R = bwlabeln(\,índiceVecinos, conn1);$$

La matriz obtenida tiene la forma de la matriz de vecinos, pero contiene los índices a las distintas regiones

etiquetadas. Se transforma esta matriz en una de forma Nx27, y se calcula el número de etiquetas distintas para cada uno de los N vecindario. Si este número es 2 (el fondo y un única etiqueta), entonces $C^* = 1$, y la eliminación del punto no desconecta componentes conectados.

Variable	Dimensiones
Índice a la imagen	Nx1
Posición relativa de los vecinos	1x3x3x3
Índice a la imagen de los vecinos	Nx3x3x3
Valor de la imagen en los vecinos [1]	Nx3x3x3
Matriz de conectividad [2]	3x3x3x3
Etiquetado de regiones [3]	Nx3x3x3
[3] = bwlabeln([1],[2]);	Nx27

Tabla 1. *Tabla resumen de las variables utilizadas para el cálculo de* C^*

La condición de $\bar{C} = 1$ es equivalente a que el número de Euler del vecindario a 26 permanezca inalterado con la eliminación del punto en estudio [13]. Para ello se divide en entorno 3x3x3 del punto en 8 octantes de 2x2x2, cada una con 7 vecinos del punto. Así, cada punto tiene

$$8\ octantes * 7\frac{vecinos}{octante} = 56\ vecinos$$

De esta manera un vecindario de 27 vecinos pasa a ser un vecindario de 56 vecinos.

Mediante una *look-up table* se puede codificar los $2^7 = 128$ valores de cambio de número de Euler para cada octante, teniendo en cuenta todas las posibles combinaciones entre los 7 vecinos del octante. Finalmente, se considera que la supresión de un punto no afecta al número de Euler del vecindario si la suma de los valores de los 8 cuadrantes es igual a cero [13].

Variable	Dimensiones
Índice a la imagen	Nx1
Índice a la imagen de los vecinos	Nx27
Valor de la imagen en los vecinos	Nx27
Valor de la imagen en los vecindarios con octantes.	Nx56
Valor de la imagen en los octantes	8Nx7
Índice a la look-up table	8Nx1
Variación del número de Euler	Nx1

Tabla 2. *Tabla resumen de las variables utilizadas para el cálculo de* $\bar{C}$.

Así, para una lista de puntos a analizar, se calculan los índices de sus vecinos y el valor de la imagen en éstas posiciones. Estos vecindarios de 27 elementos se transforman en vecindarios de 56 elementos relativos a los octantes. En este punto tenemos 8N octantes de 7 elementos, que son analizados usando la look-up table. Si la suma de los valores obtenidos para cada vecindario es igual a cero, entonces $\bar{C} = 1$.La tercera condición para eliminar un punto es que éste no sea un punto terminal. Esto es, que únicamente tenga un vecino y su valor de AOF sea menor que el umbral. Esto es fácilmente calculable usando el valor de la imagen en los vecinos (*) y el valor de AOF en las posiciones de interés.

4. Resultados y discusión

En la Figura 2 se presentan los resultados obtenidos en 3 imágenes sintéticas con línea central conocida. La línea central calculada automáticamente se muestra en azul, y la línea conocida se muestra en negro. Como se puede observar en el segundo tubo sinusal y en la hélice, el algoritmo tiende a suavizar las grandes variaciones de la línea central, pero manteniendo siempre una posición centrada. Las dos implementaciones ofrecen exactamente el mismo resultado en todos los casos.

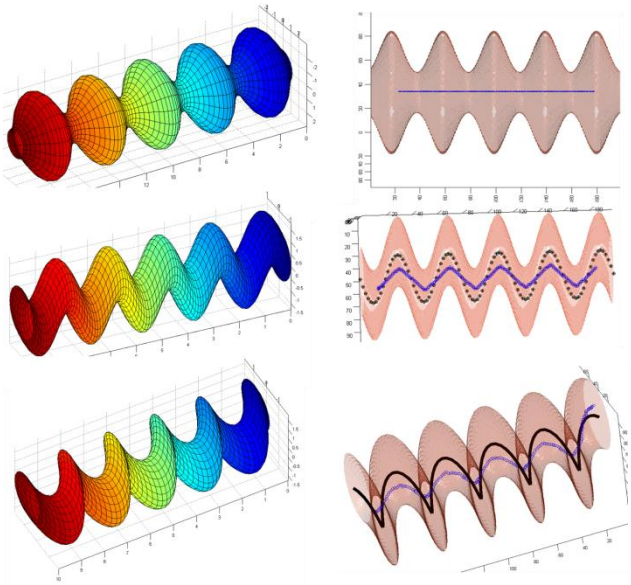

Figura 1.*Reconstrucción 3D de las imágenes sintéticas utilizadas (izq), y resultados de las líneas centrales obtenidas.*

En la figura 3 se muestran los resultados en imágenes médicas de una aorta abdominal y de un phantom de silicona de una aorta torácica. Se puede ver que el método es capaz de extraer líneas centrales complicadas, compuestas por muy diversas ramas.

En la Tabla 3 se pueden ver los tiempos de cálculo para cada una de las imágenes. En cada caso se muestra las dimensiones de la imagen, el número de vóxeles del objeto (así como el número de distancias que se evalúan en la implementación matricial) y el tiempo de ejecución de la implementación matricial.

Todas las imágenes usadas para la evaluación requirieron un mínimo de 2 horas para su cálculo usando la implementación de ITK en un ordenador con un procesador i7 y 8GB de memoria RAM. Sin embargo, el tiempo de ejecución en imágenes de 20x20x40 no conlleva más de 8 segundos, demostrando la gran dependencia de la implementación ITK frente al tamaño de la imagen.

Como se puede observar estudiando la tabla, el tiempo de ejecución de la implementación matricial no es sólo proporcional al número de vóxeles del objeto, como en el caso de la implementación original, sino que es proporcional al número de valores diferentes del mapa de distancias del objeto. Es paradigmático el caso de las imágenes sintéticas, que tienen el mismo tamaño en todos los casos. La imagen sinus2, que es la que tiene un mayor número de vóxeles a evaluar, tarda menos tiempo en calcularse debido a que tiene muchas menos distancias a evaluar que sinus1.

Imagen	Dimensiones	Voxeles	distancias	Tiempo(s)
TA	389x220x384	1125515	602	59
AA	262x204x174	463196	473	21
Sinus1	100x100x200	754355	446	19
Sinus2	100*100*200	847993	186	15
Hélice	100*100*200	484565	109	8

Tabla 3. *Tiempo de ejecución para las distintas imágenes usadas en la evaluación.*

La mejora en tiempo de ejecución de la implementación propuesta frente a la implementación disponible en ITK es de al menos 3 o 4 órdenes de magnitud. Además, la ejecución del algoritmo de ITK necesita un gran uso de memoria RAM, no pudiendo llevar a cabo la ejecución de la imagen TA.

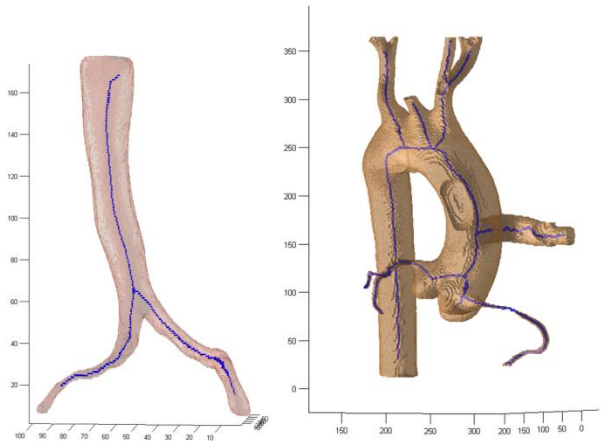

Figura 2.*Resultados del método en una imagen de aorta abdominal (izq) y una imagen de un phantom de una aorta torácica.*

5. Conclusiones

Se ha propuesto una nueva implementación matricial de un método de extracción automática de la línea central de líneas tubulares capaz de calcular el esqueleto de estructuras complejas como los árboles vasculares.

Se ha explicado detalladamente la implementación matricial que hace uso de todo el potencial de la librería de cálculo vectorial disponible en Matlab.

Ésta nueva implementación tiene un tiempo de ejecución proporcional al número de distancias diferentes del mapa de distancias del objeto, en vez de ser proporcional al número de vóxeles del objeto, como era el caso del método original.

Gracias a estas modificaciones, la implementación propuesta obtiene la línea central en tiempos varios órdenes de magnitud menores que la implementación disponible en ITK.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto europeo SCATh Smart Catheterization, G.A. 248782

Referencias

[1] Wink O, Niessen WJ, Viergever MA. Multiscale Vessel Tracking. IEEE Transactions on Medical Imaging 23 (1), 130–133, 2004.

[2] Sabry Hassouna M, Farag AA. PDE-based three dimensional path planning for virtual endoscopy. Information Processing in Medical Imaging. Lecture Notes in Computer Science Vol. 3565, 2005, pp 529-540.

[3] Antiga,L. Patient-specific modeling of geometry and blood flow in large arteries. PhD Thesis.

[4] Ge Y, Stelts DR, Wang J, Vining D. Computing the Centerline of a Colon: a Robust and Efficient Method Based on 3D Skeletons. Journal of Computer Assisted Tomography 23 (5), 786–794, 1999.

[5] Fontanilla, P, Rodríguez-Vila, B, Gómez, EJ. Real-time non-rigid registration of a parametric aorta model for a VR-based catheterization guidance system. Proceedings of the 2012 SCATh Joint Workshop on New Technologies for Computer/Robot Assisted Surgery

[6] Xu M, Pycock D. A scale-space medialness transform based on boundary concordance voting. Journal of Mathematical Imaging and Vision, 11:277–299, 1999.

[7] Krissian K, Malandain G, Ayache N, Vaillant R, Trousset Y. Model based detection of tubular structures in 3d images. Computer Vision and Image Understanding, 80(2):130{171, Nov. 2000.

[8] Bitter I, Kaufman AE, Sato M. Penalized-Distance Volumetric Skeleton Algorithm. IEEE Transactions on Visualization and Computer Graphics 7 (3), 195–206, 2001.

[9] Bouix S, Siddiqi K, Tannenbaum A. Flux driven automatic centreline extraction. Technical report SOCS-04.2, School of Computer Science, McGill University, 2004.

[10] Siddiqi K., Bouix S, Tannenbaum A, Zucker SW. Hamilton-Jacobi Skeletons. International Journal of Computer Vision 48 (3), 215–231, 2002.

[11] Mellado X, Larrabide I, Hernandez M, Frangi A. Flux driven medial curve extraction. Insight Journal 2007, http://hdl.handle.net/1926/560.

[12] Malandain G, Bertrand G, Ayache N. Topological Segmentation of Discrete Surfaces. International Journal of Computer Vision 10 (2), 183–197, 1993.

[13] Lee TC, Kashyap RL. Building skeleton models via 3D medial surface/axis thinning algorithms. Computer Vision, Graphics, and Image Processing, 56(6), pp. 462-478, 1994.