# Extracción eficiente de fibras cerebrales que conectan dos regiones cerebrales

*Stalin Garrido P., Carlos Kublik M. y Pamela Guevara A.*
*Depto. de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de Concepción*
*stgarrido@udec.cl, carlkublik@udec.cl, pguevara@udec.cl*

***Resumen*—Para estudiar la conectividad cerebral a partir de fibras cerebrales calculadas mediante tractografía, es necesario calcular intersecciones entre las fibras y regiones de interés (ROI) del cerebro. Las ROI están dadas por imágenes etiquetadas de la corteza o núcleos subcorticales. Para ello, se implementó un algoritmo de segmentación de fibras cerebrales que conectan dos regiones cerebrales. El método consiste en dos etapas. En primer lugar, se realiza una segmentación según el área cercana a la región de interés y posteriormente, se aplica un refinamiento espacial, para detección más precisa de la intersección. Esta implementación se desarrolló en los lenguajes de programación Python y C++, para poder comparar el rendimiento de ambos. Si bien ambas implementaciones entregan el mismo resultado, los tiempos de ejecución en C/C++ son mucho menores, alcanzando una aceleración de más de 600 veces.**

***Palabras clave*—Resonancia Magnética de Difusión, Tractografía, Fibras cerebrales, Segmentación, Tiempo de ejecución.**

## I. Introducción

El estudio de la conectividad del cerebro es de gran interés para comprender su función. De esta manera, es posible además estudiar distintas patologías.

La Resonancia Magnética de Difusión (**dMRI**) [1], [2] permite obtener imágenes de la estructura del tejido cerebral. A partir de la información de cada vóxel es posible estimar la distribución de las direcciones principales de los axones neuronales, por medio de un modelo 3D de difusión local. Con la distribución de las fibras en cada vóxel, se puede aplicar un algoritmo de tractografía [3], con el cual se calcula una estimación de las trayectorias de las fibras cerebrales. Los conjuntos de tractografía contienen una lista de fibras cerebrales compuestas por una secuencia de puntos 3D. Para el estudio de la conectividad cerebral, es necesario combinar la información de las fibras cerebrales con información anatómica, proveniente de imágenes de Resonancia Magnética T1. Estas imágenes representan la anatomía del cerebro, permitiendo distinguir los distintos tejidos. Existen métodos de segmentación de estas imágenes que entregan imágenes T1 etiquetadas con las distintas regiones cerebrales (o ROI, de regiones de interés). Es necesario entonces, contar con algoritmos que segmenten o extraigan las fibras cerebrales que conectan un par de ROIs, de forma eficiente.

En este trabajo se presentan dos implementaciones de un algoritmo que extraen las fibras que conectan un par de ROIs, una en Python y la otra en C/C++. Se comprueba que el algoritmo en C++ es mucho más eficiente.

## II. Materiales y métodos

### *II-A. Materiales*

Para este trabajo se utilizó de un conjunto de datos de tractografía determinística de cerebro completo, a partir de la base de datos ARCHI [4]. En el análisis, se utilizaron solamente las fibras del hemisferio izquierdo de un sujeto, con un total de 264.641 fibras. Además se tiene la imagen anatómica T1, etiquetada mediante el software FreeSurfer[1], en base a un atlas de las regiones de la corteza cerebral [5]. Las dimensiones de la imagen son de (160, 240, 256) vóxeles en x,y,z, y las dimensiones de cada vóxel son de (1.1 x 1.0 x 1.0) mm. A partir de esta imagen es posible extraer cada región a través de su respectiva etiqueta.

### *II-B. Método de segmentación de fibras*

El algoritmo se diseñó para identificar las fibras que conectan dos regiones de interés de la imagen T1, a partir de un conjunto de tractografía. La imagen se encuentra en formato NIFTI (*.nii*), un formato estándar de imágenes cerebrales que cuenta con bibliotecas para su lectura en Python y C/C++. Las fibras están en formato *.bundles*, formato que es menos conocido, pero para el cual también tenemos bibliotecas en Python y C/C++ para su lectura.

El algoritmo se divide en dos etapas, con la finalidad de reducir la cantidad de fibras a analizar en cada una de ellas y acelerar el procesamiento.

Cada región se identifica por su etiqueta (ver Tabla II del Anexo), es decir, están compuestas por todos los vóxeles que tienen la etiqueta de la región.

En la primera etapa se propone seleccionar de forma rápida las fibras cercanas a cada una de las dos regiones de interés, y descartar todas aquellas fibras que con seguridad no conectan ambas regiones. Para ello se calcula la caja englobante de cada región, es decir el paralelepípedo en la imagen, de menor tamaño, que contenga a todos los vóxeles de la región. Se analiza cada extremo de la fibra, para saber si está cercano a cada una de las cajas englobantes. Para decidir si una fibra pertenece o no a la caja de la región, consideramos los 3

[1] https://surfer.nmr.mgh.harvard.edu

primeros y los 3 últimos puntos de la fibra. Para determinar si un punto pertenece a la caja, se comparan sus coordenadas con las coordenadas de la caja. Si al menos uno de estos puntos pertenecen a la caja, se guarda la fibra completa para un posterior análisis. Para optimizar el tiempo de procesamiento, primero se seleccionan todas las fibras que conectan la primera región. Luego, se analizan sólo estas fibras, para saber si su otro extremo conecta la otra región. Al final de esta etapa, sólo se almacenan las fibras cuyos extremos conectan a las las regiones deseadas.

En la segunda etapa del algoritmo, se aplica un refinamiento a las fibras seleccionadas en la etapa anterior, para detectar las intersecciones con mayor precisión. En esta etapa se analiza la cercanía de los extremos de cada fibra con cada vóxel de las regiones. En esta etapa, igualmente, primero se analiza la cercanía con respecto a una región, para para desacatar las fibras que no pertenezcan, y luego analizar las fibras restantes con respecto a la segunda región. Para cada vóxel de una región, se determinan sus coordenadas y se calcula su centroide. Las coordenadas de los centroides de todos los vóxeles de cada región son almacenas para el análisis. Al igual que la etapa anterior, para cada fibra, se comparan los 3 primeros puntos y los 3 últimos puntos, con el centroide de cada vóxel de la región a analizar. En este caso se usa un umbral de distancia máxima $dmax$.

Los algoritmos descritos fueron implementados en los lenguajes de programación Python y C/C++. Python proporciona bibliotecas y estructuras de datos fáciles de utilizar, sin embargo el costo computacional es alto para este tipo de análisis, dada la cantidad de iteraciones y comparaciones. Por su lado, C/C++ resulta un poco más complejo de implementar, pero gracias a un mejor manejo de la memoria y algunos procesos, permite tiempos de ejecución mucho menores. La primera parte de ambas implementaciones contempla la lectura del archivo de fibras y de la imagen con las regiones anatómicas etiquetadas. Es importante mencionar que los ejes coordenados de la imagen están almacenados de la forma (z, y, x), como se puede visualizar en la Figura 1. Como se describió, para ambos casos se cuenta con bibliotecas para lectura/escritura de los archivos.

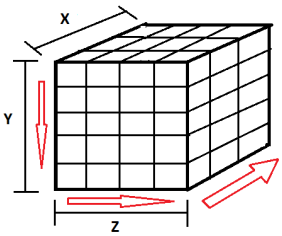



Figura 1: Visualización espacial de la lectura.

### *II-C. Implementación en Python*

Para la lectura de la imagen usamos la biblioteca **NiBabel**[2], la que permite leer imágenes en formato **NIFTI**. La imagen es leída es almacenada en una matriz tridimensional, permitiendo que sea fácil el proceso de lectura de ésta. Las fibras leídas son obtenidas en forma de una lista de fibras, donde cada fibra está dada por una arreglo *numpy* de Nx3, donde N es el número de puntos de la fibra, para los que se almacenan sus 3 coordenadas (x,y,z).

Para cada región de interés, dada por su etiqueta, se obtienen los índices de los vóxeles asociados a esta en la imagen anatómica. A partir de los índices y del tamaño de cada vóxel, se calculan las coordenadas de la caja englobante a partir de los valores mínimos y máximos en (x, y, z) de los vóxeles de la región. Luego, para cada fibra en la lista, se realiza la evaluación de si pertenece o no a una caja englobante, y luego a la otra. Terminada esta etapa, se almacenan las fibras seleccionadas, para aplicar el refinamiento por distancia con respecto a los vóxeles de cada región de interés. De esta forma de obtiene el conjunto final de fibras que conectan las dos regiones cerebrales de interés.

### *II-D. Implementación en C/C++*

En C/C++ no está disponible la biblioteca **NiBabel** para la lectura de imágenes **NIFTI**, por lo que se utilizó la biblioteca **NifTI-1C**[3] . En este caso la imagen anatómica se obtiene como un arreglo de bytes, por lo que se debió convertir a una matriz 3D, con las coordenadas (x, y, z), a partir de las dimensiones entregadas por la misma biblioteca. Para la lectura de fibras se ++usó una biblioteca en C que devuelve las fibras en una estructura. Esta estructura contiene punteros a las fibras, donde cada fibra es almacenada en una estructura, que a su vez almacena los puntos mediante punteros. Estas estructuras son totalmente dinámicas pero requieren reserva y liberación de la memoria.

En ambas implementaciones se obtiene un archivo nuevo de fibras en formato *bundles*.

## III. Resultados

Una vez implementado el algoritmo en ambos lenguajes, se procedió a comparar los resultados, comprobando que se obtienen idénticos resultados, es decir, se segmentan las mismas fibras. Luego, se procedió a realizar mediciones del tiempo de procesamiento para ambas implemetaciones. Estas pruebas se realizaron usando un computador con una CPU de propósito general (I5-4460 3,4GHz), mediante el programa Microsoft Visual Studio.

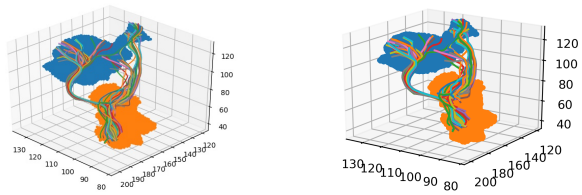
La Tabla I muestra la cantidad de fibras segmentadas para cada par de regiones de prueba, así como la distancia utilizada, y los tiempos de ejecución (en segundos) para ambas implementaciones. Las regiones del cerebro se identifican con su número de etiqueta, según nombre listado en la Tabla II). Se observa que el tiempo de ejecución en Python es mucho mayor que en C/C++, siendo en promedio de 2.4 s, para las regiones analizadas, con respecto a un promedio de 1500 s en C/C++. Esto da una aceleración de más de 600 veces más rápido en C/C++ que en Python. Se observa además que el tiempo de ejecución depende del tamaño de las zonas, y de la cantidad de fibras que las conectan, siendo mayor, a medida que estos valores aumentan.

[2]https://nipy.org/nibabel/

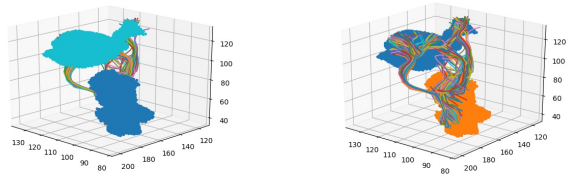
[3]https://nifti.nimh.nih.gov/

| Regiones del cerebro | Distancia de voxel | Número de fibras | | Tiempo de ejecución (s) | | Aceleración C/C++ c/r Python |
|---|---|---|---|---|---|---|
| | | En C/C++ | En Python | En C/C++ | En Python | |
| 7 a 25 | 0.5 | 100 | 100 | 1.122 | 527.810 | 470.4 |
| 7 a 28 | 0.5 | 2 | 2 | 0.958 | 350.596 | 365.9 |
| 7 a 29 | 0.5 | 455 | 455 | 1.447 | 920.979 | 636.5 |
| 27 a 30 | 0.5 | 16 | 16 | 6.068 | 4183.212 | 689.4 |
| 7 a 25 | 1 | 289 | 289 | 1.161 | 604.739 | 520.9 |
| 7 a 28 | 1 | 2 | 2 | 0.980 | 404.082 | 412.3 |
| 7 a 29 | 1 | 590 | 590 | 1.514 | 1011.837 | 668.3 |
| 27 a 30 | 1 | 43 | 43 | 6.249 | 4701.486 | 752.4 |

Tabla I: Rendimiento para distintas zonas.

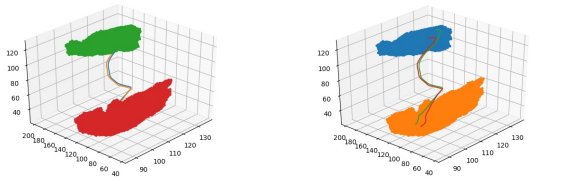


(a)

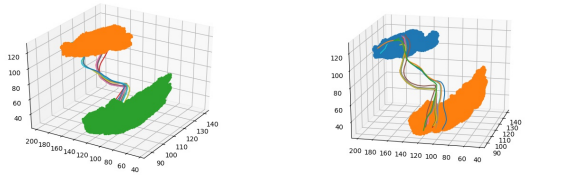


(b)

Figura 2: Fibras segmentadas que conectan las regiones 7 y 25, para (a) d = 0.5 mm y (b) d = 1 mm. (izq: vista exterior, der: vista interior)

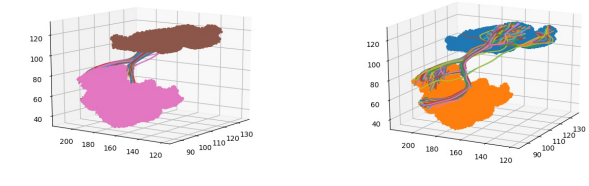


(a)

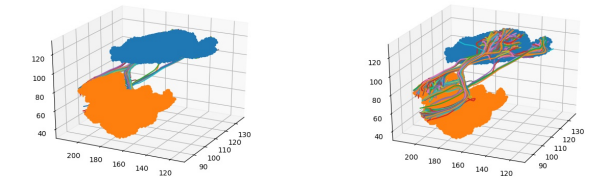


(b)

Figura 3: Fibras segmentadas que conectan las regiones 7 y 28, para (a) d = 0.5 mm y (b) d = 1 mm. (izq: vista exterior, der: vista interior)

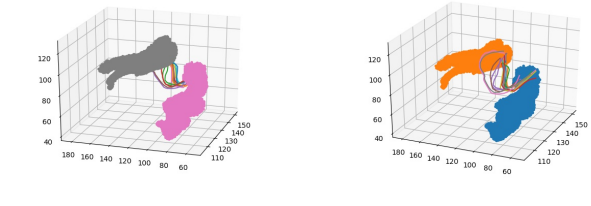


(a)

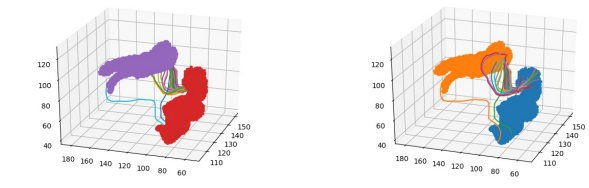


(b)

Figura 4: Fibras segmentadas que conectan las regiones 7 y 29, para (a) d = 0.5 mm y (b) d = 1 mm. (izq: vista exterior, der: vista interior)



(a)



(b)

Figura 5: Fibras segmentadas que conectan las regiones 27 y 30, para (a) d = 0.5 mm y (b) d = 1 mm. (izq: vista exterior, der: vista interior)

Las figuras 2, 3, 4 y 5 muestran un despliegue de las fibras que conectan distintos pares de regiones cerebrales, para una distancia máxima igual a 0.5 y 1 mm. Se incluye una vista exterior y una interior para ver mejor que la intersección es efectivamente con las regiones seleccionadas. Se observa que con distancia de 1 mm, se seleccionan un poco más de fibras, pero siempre cercanas a la región.

## IV. Conclusiones

Se ha logrado implementar un algoritmo de segmentación de fibras cerebrales, mediante los lenguajes de programación C/C++ y Python.

En cuanto a la calidad de los resultados obtenidos, se observa que son correctos. Esta calidad puede reducirse para distancias más grandes, pero se observa que un umbral de 1 mm da aún buenos resultados, ya que es cercano a las dimensiones de los vóxeles de la imagen utilizada.

El utilizar una primera etapa de filtrado, con una caja englobante ayudó a reducir el tiempo de búsqueda, descartando de forma rápida la gran mayoría de las fibras a analizar.

Se puede destacar los bajos tiempos de procesamiento que se pudo lograr en C/C++, mucho menores que en Python. Si bien era esperado un mejor rendimiento en esta implementación, esta es bastante superior a lo que se pensó. Esto se debe a las estructuras utilizadas en ambas implementaciones, donde en C/C++ se hace uso eficiente de la memoria y el acceso a ella.

De todas maneras, Python sigue siendo un excelente lenguaje para pruebas debido a su facilidad de uso y menor tiempo de implementación. Sin embargo, para algoritmos con uso de datos masivos, como son las fibras cerebrales, se aconseja lograr implementaciones estables en C/C++. Esto permite procesar datos de tractografía determinísticos, e incluso probabilísticos, con más de 3 millones de fibras, en tiempos bastante bajos.

Como trabajo futuro, se propone poder detectar la intersección de las fibras con todos los pares de regiones a la vez, y así realizar en una sola ejecución la segmentación de todas las fibras cerebrales de un cerebro.

## V. Agradecimientos

Este trabajo fue financiado por CONICYT FONDECYT 1190701, CONICYT PIA/Anillo de Investigación en Ciencia y Tecnología ACT172121 y CONICYT BASAL FB0008.

## Referencias

[1] J. Ahualli, "Aspectos generales de las secuencias de difusión de imagen en resonancia magnética," *Revista argentina de radiología*, vol. 74, no. 3, pp. 226–236, 2010.

[2] D. Le Bihan and M. Iima, "Diffusion magnetic resonance imaging: what water tells us about biological tissues," *PLoS Biology*, vol. 13, no. 7, p. e1002203, 2015.

[3] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi, "In vivo fiber tractography using dt-mri data," *Magnetic Resonance in Medicine*, vol. 44, no. 4, pp. 625–632, 2000.

[4] B. Schmitt, A. Lebois, D. Duclap *et al.*, "CONNECT/ARCHI: an open database to infer atlases of the human brain connectivity," in *ESMRMB conference*, 2012.

[5] R. Desikan, F. Segonne, B. Fischl, B. Quinn, B. Dickerson, D. Blacker, R. Buckner, A. Dale, R. Maguire, B. Hyman, M. Albert, and R. Killiany, "An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest," *NeuroImage*, vol. 431, pp. 968–980, 2006.

## Anexo

| n° de la zona | Nombre de la zona | Lóbulo |
|---|---|---|
| 0 | Banks of Superior Temporal Sulcus | - |
| 1 | Caudal Anterior Cingulate | Frontal |
| 2 | Caudal Middle Frontal | Frontal |
| 3 | Corpus Callosum | - |
| 4 | Cuneus | Occipital |
| 5 | Entorhinal | Temporal |
| 6 | Fusiform | Temporañ |
| 7 | Inferior Parietal | Parietal |
| 8 | Inferior Temporal | Temporal |
| 9 | Isthmus | Parietal |
| 10 | Lateral Occipital | Occipital |
| 11 | Lateral Orbitofrontal | Frontal |
| 12 | Lingual | Lingual |
| 13 | Medial Orbitofrontal | Frontal |
| 14 | Middle Temporal | Temporal |
| 15 | Parahippocampal | Temporal |
| 16 | Paracentral | Frontal |
| 17 | Pars Opercularis | Frontal |
| 18 | Pars Orbitalis | Frontal |
| 19 | Pars Triangularis | Frontal |
| 20 | Pericalcarine | Occipital |
| 21 | Postcentral | Parietal |
| 22 | Posterior Cingulate | Parietal |
| 23 | Precentral | Frontal |
| 24 | Precuneus | Parietal |
| 25 | Rostral Anterior Cingulate | Frontal |
| 26 | Rostral Middle Fronta | Frontal |
| 27 | Superior Frontal | Frontal |
| 28 | Superior Parietal | Parietal |
| 29 | Superior Temporal | Temporal |
| 30 | Supramarginal | Parietal |
| 31 | Frontal Pole | Frontal |
| 32 | Temporal Pole | Temporal |
| 33 | Transverse Temporal | Temporal |
| 34 | Insula | - |

Tabla II: Etiquetas de las regiones cerebrales en la imagen T1: número de etiqueta, circonvolución y lóbulo.