UPV

# SOFTWARE PARA EL ESTUDIO DEL VOLUMEN DE ESTRUCTURAS CORTICALES EN IMÁGENES DE RMN CEREBRAL

Proyecto final de carrera

**Autor: José Enrique Romero Gómez**

**Director: José Vicente Manjón Herrera**

Titulación: Ingeniería Informática

**Fecha: 26/09/2011**

En este proyecto se ha desarrollado una aplicación para la separación interhemisférica del cerebro en imágenes de RMN basado en la comparación de regiones contra una biblioteca de casos segmentados manualmente.

**Índice**

## 1. Resumen

El estudio del volumen intracraneal requiere del uso de herramientas que permitan objetivar el diagnóstico y ofrezcan un rendimiento y precisión elevados. La segmentación[(11)] automática del volumen cerebral es el primer paso hacia un estudio más completo del cerebro y supondrá una herramienta versátil en el estudio de diversas patologías. *Propósito:* Mejorar un método ya implementado que segmenta el volumen cerebral con una calidad aceptable pero en un tiempo de ejecución elevado basado en comparación de regiones contra una biblioteca de casos de ejemplo segmentada manualmente. *Método:* El método recorre uno a uno todos los voxels del cerebro a segmentar extrayendo la región que lo envuelve y comparándola con regiones de los casos de ejemplo de la biblioteca. Esto era ineficiente así que se han introducido mejoras que van desde la carga y preselección de los casos más semejantes para usarlos en la segmentación hasta introducir una estimación pre calculada del etiquetado de los voxels que no suelen variar para evitar tener que procesarlos. *Resultados:* Se parte de un método que obtiene segmentaciones con una 98% de fiabilidad y en un tiempo de ejecución de 160 segundos y se ha mejorado hasta una fiabilidad del 99% en un tiempo inferior a 40 segundos.

**Palabras clave:** Resonancia magnética nuclear, RMN, segmentación automática, comparación de parches.

## 2. Introducción

El estudio del volumen cerebral es una tarea aún pendiente en la práctica clínica ya que no existen herramientas que lo hagan posible.

Hoy en día la tecnología nos ofrece imágenes de resonancia magnética digitales de alta resolución lo cual hace obligatorio servirse de las tecnologías de la información para explotar todo el potencial que nos ofrecen.

Para afrontar este problema contamos con diversas estrategias ya formuladas. El plano medio-sagital pretende separar los hemisferios cerebrales y está definido como la mejor aproximación a la fisura inerhemisférica mediante un plano [8]. También el plano el cual maximiza la simetría bilateral ([9], [10] y [11]). Los métodos de segmentación compartimental dividen el cerebro en hemisferios derecho e izquierdo, cerebelo y bulbo raquídeo. Estos métodos se pueden clasificar en dos tipos [18]: métodos basados en la reconstrucción de estructuras y métodos basados en segmentación y búsqueda de superficies. El primer tipo consiste en identificar voxels semilla, a menudo en la materia blanca, que representan los compartimentos a segmentar. El siguiente paso en estos métodos es reconstruir dichos compartimentos partiendo de las semillas hasta llegar a los límites. Las diferencias entre los métodos de este tipo radican en el enfoque empleado para encontrar semillas. Para realizar la segmentación se pueden utilizar dos planos como en FreeSurfer [12] y BrainVoyager [13]: se utiliza un plano sagital que corta el cuerpo calloso para dividir los hemisferios izquierdo y derecho y otro plano horizontal cortando el cerebro medio separando los hemisferios del cerebelo y bulbo raquídeo. En BrainVisa [14] se aplica el algoritmo de formas en cuello de botella para desconectar ambos hemisferios entre si y separar también el conjunto que forman cerebelo y bulbo raquídeo en los cuellos de botella. Hata [16] identificó semillas compartimentales a través de la materia blanca y la gris mediante el conocimiento anatómico local de zonas difusas de los hemisferios izquierdo y derecho, el cerebelo y el bulbo raquídeo.

El segundo grupo de métodos aborda el problema buscando los límites de las superficies entre compartimentos usando un criterio de optimización. Para una mejor segmentación es posible detectar superficies compartimentales deformables para los hemisferios izquierdo y derecho y el conjunto cerebelo y bulbo raquídeo utilizando un criterio basado en los límites ([17] y [18]) o bien segmentación no lineal utilizando plantillas pre segmentadas [19].

Recientemente Zhao [7] propusio un método llamado Adaptive disconnection para la segmentación del volumen intracraneal en hemisferios izquierdo y derecho cerebrales, hemisferios izquierdo y derecho del cerebelo y bulbo raquídeo basado en el algoritmos de formas en cuellos de botella [15] utilizando ecuaciones diferenciales parciales (PDE) basadas. Este méodo segmenta los compartimentos desconectando las formas en cuello de botella. El algoritmo the formas en cuello de botella basado en PDE detecta las formas en cuello de botella mediante conocimiento previo simulando una transferencia de información entre compartimentos mediante PDE. El método presenta errores en el cereberlo debido ah que su morfología hace difícil encontrar el límite entre las mitades izquierda y derecha. Otra desventaja de este método es un coste computacional elevado.

Por eso, el objetivo de este proyecto ha sido mejorar un método ya implementado para conseguir mayor calidad en los resultados, un uso menor de memoria así como un tiempo de ejecución lo más reducido posible.


*Figura 1: Pasos del pre proceso y la segmentación.*

Para conseguirlo partimos de un método ya implementado con el objetivo de mejorarlo en cuanto a calidad de sus resultados y sobretodo en cuanto a tiempo de ejecución.

Este método se basa en la comparación de regiones pequeñas del cerebro contra una biblioteca de ejemplos segmentados manualmente cuyo objetivo es abarcar cuanta más variabilidad mejor aunque esto no es objetivo del proyecto.

Las imágenes con las que trabaja el método son imágenes 3D en las que cada punto es un voxel, que es la unidad mínima en 3D así como el pixel es la unidad mínima en 2D.

Estas imágenes contienen la representación de un cerebro en tonos de gris y se someten a un proceso de filtrado y corrección como se ve en la figura 1 para obtener el volumen intracraneal sin interferencias y cumpliendo una serie de requisitos para poder analizarlos.

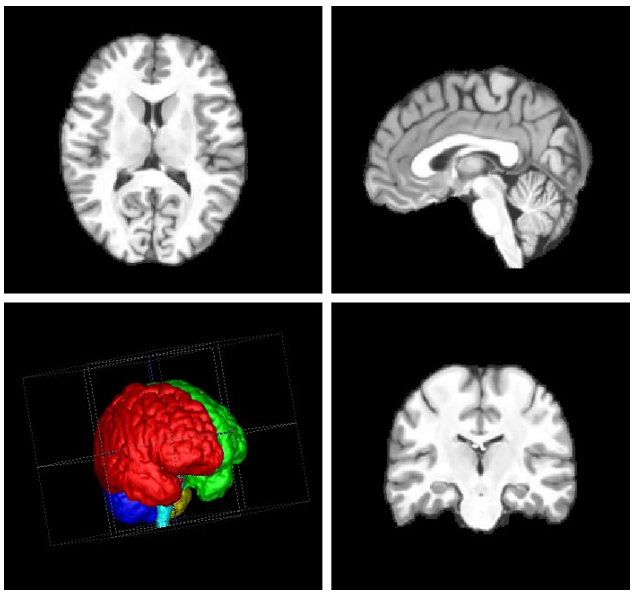

*Figura 2: Imagen de un cerebro en 3 cortes y representación 3D (solo parénquima).*

El método inicial obtiene ya de por si unos resultados con bastante calidad pero es muy ineficiente. Carga un gran volumen de información en memoria y lleva a la CPU al 100% de utilización.

El método para recorrer la imagen es muy pesado ya que se recorren todos los voxels uno a uno a pesar de que existen zonas sobre las que no sería necesario pasar.

## 3. Antecedentes

Como se explica en la introducción el objetivo de este proyecto es mejorar un método ya existente así que en primer lugar explicaremos en que consiste este método, como está implementado y que resultados obtiene para establecer el punto de partida.

El método trabaja segmentando un cerebro representado con una imagen 3D de RMN pre procesada comparando pequeñas regiones en forma de cubo con regiones de imágenes de ejemplo contenidas en una biblioteca de imágenes igualmente pre procesadas y segmentadas manualmente. De esta forma obtiene las regiones de la librería que más se parecen a la imagen objetivo para utilizar su segmentación manual [1].

### 3.1. Pre-procesado de las imágenes

En primer lugar debemos decir que las imágenes de MRN se puede agrupar en tres grupos: $T1_{(13)}$, $T2_{(14)}$ y $DP_{(2)}$. Estas difieren en el método de obtención y tienen utilidades diferentes a la hora de interpretarlas. En nuestro caso nos centramos en las imágenes T1 al ser este grupo el que mejor diferencia sustancia blanca, sustancia gris y líquido cefalorraquídeo.

Como ya se ha explicado nuestro método trabaja con imágenes de RMN pero estas son muy variadas y dado que nuestro objetivo es desarrollar un método robusto es necesario un pre-proceso para asegurarnos de que las imágenes se ajustan a un patrón.

Filtrado: Como es sabido, el ruido va asociado al uso de aparatos eléctricos e influencias externas lo cual termina afectando al resultado y en nuestro caso eso significa imágenes resultantes con ruido aleatorio, con puntos que alteran su brillo y para solucionarlo se les aplica un filtrado [3].

Corrección de la Inhomogeneidad: El funcionamiento de un aparato de RMN se basa en la influencia de un campo magnético muy intenso. Este campo no es completamente homogéneo sino que es más intenso en el centro del área sobre la que actúa y las imágenes presentan mayor brillo en el centro que en los extremos lo cual impide una interpretación objetiva de los nivele de intensidad por lo que es necesario corregirlo [5].

Normalización de la Intensidad: Con intensidad nos referimos al brillo de cada punto de la imagen y estos valores no son estándar, es decir, diferentes aparatos de RMN producirán diferentes niveles de intensidad por tanto debemos normalizar la intensidad de las imágenes [2].

Extracción intracraneal: Como ya hemos dicho nuestro método centra su análisis en el volumen cerebral y por tanto el resto de elementos presentes en la imagen como el cráneo o las vértebras no nos interesan. La solución es utilizar un método automático para extraer el volumen intracraneal [4].

Registro al espacio MNI: Los puntos, además de un valor de intensidad, están situados en unas coordenadas concretas que tampoco son estándar. Esto supone un problema debido a que, adelantando detalles, nuestro método se basa en la localidad de las zonas a analizar. Por tanto hay que registrar las imágenes al espacio normalizado del Montreal Neurological Institute ($MNI_{(8)}$) [5].

### 3.2. Biblioteca de ejemplos

Antes de describir el método en sí tenemos que hablar de la biblioteca de imágenes de ejemplo en la que se apoya puesto que repercute directamente en la calidad del resultado final por dos razones: En primer lugar la segmentación de los casos de ejemplo es en definitiva la que acabará asignándose a la imagen objetivo y en segundo lugar la calidad del resultado será mejor cuanto mayor sea la representatividad del conjunto de ejemplos.

Puesto que el objetivo de este proyecto es mejorar el método no vamos a especular sobre la cantidad y variabilidad de los casos de ejemplo pero si en la calidad de sus segmentaciones.

Para realizar este proyecto disponemos de 20 imágenes obtenidas del IXI Dataset [6]. Estas imágenes proceden de sujetos normales sanos de diversos hospitales de Londres:

Hammersmith Hospital - Philips 3T
Guy's Hospital - Philips 1.5T
Institute of Psychiatry - GE 1.5T

Las imágenes que hemos elegido de entre las 600 que forman el IXI Dataset son imágenes T1, como ya hemos dicho, y han sido pre procesadas como explica el apartado anterior.

Las 20 imágenes están segmentadas automáticamente pero contienen errores por lo que habrá que corregirlas en parte manualmente y en parte automáticamente para obtener un conjunto de segmentaciones válido.

En primer lugar partimos de que nuestras imágenes están normalizadas en intensidad por lo que podemos descartar los voxels cuyo valor está por debajo de un umbral. En nuestro caso consideramos que 100 es el valor mínimo para la sustancia gris por tanto podemos procesar todas las imágenes y $etiquetar_{(3)}$ como cero todos aquellos voxels que tengan un valor inferior a 100. Además dilataremos las segmentaciones para que cubran todos los voxels con un valor igual o superior a 100. Esto reduce el trabajo drásticamente ya que a partir de aquí no es necesario plantearse que voxels son cerebro y cuales no y solo hay que centrarse en si el etiquetado es el correcto.

El siguiente paso es revisar manualmente todas las imágenes pero la cantidad de errores que contienen hace que la tarea sea muy costosa por lo que se opta por revisar los tres primeros casos y utilizarlos como biblioteca inicial para segmentar el resto. Aplicando este método repetidas veces se eliminó una cantidad importante de errores.

### 3.3. Descripción del método

A grandes rasgos el método realiza los siguientes pasos:

- Seleccionar los diez casos de la biblioteca de ejemplos que más se parecen al cerebro a segmentar.
- Confeccionar una máscara[(5)] que indica que voxels se van a procesar.
- Recorrer el cerebro a segmentar voxel a voxel extrayendo la región que lo envuelve y comparándola con regiones de los casos de ejemplo.
- Escoger la región más parecida y usar la etiqueta del voxel de ejemplo como etiqueta del voxel destino.

La selección de los diez casos más parecidos se realiza restando la imagen de cada ejemplo al cerebro a segmentar voxel a voxel obteniendo así una nueva imagen 3D en la que el valor de los voxels será la diferencia entre el voxel de la imagen objetivo y su correspondiente voxel en una imagen de ejemplo.

$$Resta(i,j,k) = Objetivo(i,j,k) - Ejemplo(i,j,k) \quad (1)$$

Dado este conjunto de valores se calcula su valor medio que no dirá como de diferentes son en promedio las imágenes restadas y una vez obtenidos los valores para cada una de las imágenes de la biblioteca elegiremos los menores, es decir, los que presenten menos diferencias con nuestra imagen a segmentar.

Confeccionar la máscara que nos indicará que voxels debemos procesar consiste en realizar una operación lógica sobre el cerebro a segmentar en la cual se evalúa para cada voxel la expresión x >= 100 y cuyo resultado se almacena (ver figura 3).

$$Máscara(i,j,k) = Objetivo(i,j,k) > 100 \quad (2)$$

Así obtenemos una imagen 3D formada por 1s y 0s.

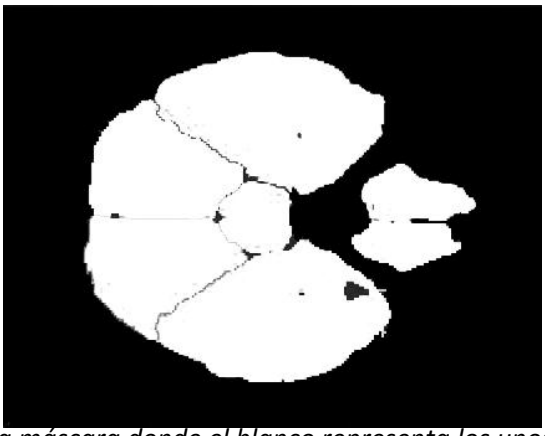

*Figura 3: Corte 50 de la máscara donde el blanco representa los unos y el negro los ceros.*

Posteriormente se pasa a segmentar la imagen objetivo. El cerebro se recorre voxel a voxel comprobando si la máscara contiene un 1 para dicho voxel. En tal caso se extrae el $parche_{(9)}$ que lo contiene como se ve en la figura 6. Este parche es un cubo de 5 x 5 x 5 voxels. Además se calculan las medias y varianza correspondientes que se usaran más tarde.

A continuación se recorren los diez cerebros de ejemplo extrayendo parches en ubicaciones contenidas en un radio de búsqueda determinado (esto es posible gracias a que todas las imágenes están situadas en el mismo espacio MNI).

El área de búsqueda es un parámetro del método que queda ajustado a 5 como valor óptimo (se verá más adelante) e implica recorrer un área de 11 x 11 x 11 voxels en el cerebro de ejemplo y para cada uno de ellos se realiza el proceso de extracción y comparación de parches descrito anteriormente.

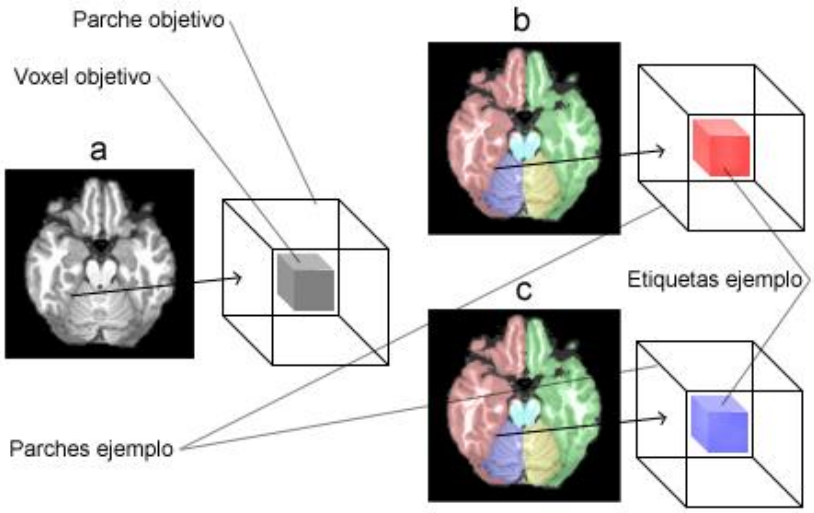


*Figura 4: Extracción de parches donde a es el cerebro a segmentar y b y c dos ejemplos de la librería.*

Para cada parche se comprueba si es mejor que el anterior en primer lugar con una comparación previa de las medias y varianzas obteniendo un valor que debe superar un umbral mínimo de 0,97. Este umbral es viable y decrece en caso de encontrar ningún parche de ejemplo para el parche objetivo.

Para obtener el valor de cada parche de ejemplo se utiliza la siguiente fórmula:

$$Umbral = \frac{2 * Med * TMed}{Med^2 + TMed^2} * \frac{2 * \sqrt{Var} * \sqrt{TVar}}{Var + TVar} \quad (3)$$

Donde Med y Var son las medias y varianzas de la imagen objetivo y TMed y TVar son las medias y las varianzas de la imagen de ejemplo.

Posteriormente se calcula la distancia euclidea al parche de la imagen objetivo mediante una suma de diferencias cuadradas (SSD) y si resulta ser mejor que el parche anterior (o si es el primero) se almacena junto con la etiqueta del voxel central.

Una vez realizado todo esto para cada cerebro de ejemplo tenemos una serie de parches candidatos con su etiqueta y distancia asociadas de los cuales deberemos elegir el mejor para asignar el valor al voxel del cerebro a segmentar (en la figura 4 se puede observar una representación simbólica de lo que sería un parche correspondiente a un voxel de una imagen dada). Este proceso de estimación de etiquetas está contemplado en el método Nonlocal means label fusion [1].

Este método consiste en calcular una media ponderada $v(X_i)$ de las etiquetas candidatas para cada voxel dentro del área de búsqueda $V_i$:

$$v(x_i) = \frac{\sum_{s=1}^{N} \sum_{j \in V_i} w(x_i, x_{s,j}) y_{s,j}}{\sum_{s=1}^{N} \sum_{j \in V_i} w(x_i, x_{s,j})} \quad (4)$$

Siendo $y_{s,j}$ la etiqueta del voxel ejemplo y $w(x_i, x_{s,j})$ el peso para dicha etiqueta obtenido por comparación de parches. Para ello se obtiene la distancia euclíde (almacenado anteriormente) al parche objetivo y se procesa para obtener $w(x_i, x_{s,j})$:

$$w(x_i, x_{s,j}) = \begin{cases} exp^{\frac{-\|P(x_i) - P(x_{s,j})\|_2^2}{h}}, & si\ ss > umbral \\ 0, & si\ no \end{cases} \quad (5)$$

Donde $P(x_i)$ representa el parche centrado en $x_i$, $||.||2$ es la normal L2 normalizada por el número de elementos y $P(x_{s,j})$ representa el parche de ejemplo *s* (dentro del área de búsqueda) en el cerebro *j*.

### 3.4. Implementación

La implementación del método es un tema importante a discutir ya que en la fase de desarrollo es muy importante utilizar herramientas que nos ofrezcan un abanico de posibilidades para realizar experimentos.

En nuestro caso hemos elegido desarrollar el método sobre Matlab[6] ya que integra una gran cantidad de herramientas para realizar análisis posteriores a los experimentos como puedan ser la medición del tiempo, tratado de imágenes o análisis estadísticos. No obstante, el proceso de segmentación tiene un coste computacional muy alto e implementarlo en Matlab no es razonable ya es una lenguaje interpretado y esto supone mucha sobrecarga. La solución pasa por utilizar MEX[7], una herramienta que ofrece Matlab para integrar funciones en C dentro de un script de Matlab.

De esta forma tenemos un script de Matlab que realiza tareas previas a la segmentación y una función MEX escrita en C que realiza el grueso de la segmentación.


*Figura 5: Diagrama de llamadas.*

### 3.5. Resultados iniciales

Los resultados iniciales del método son bastante buenos en cuanto a calidad del resultado pero muy mejorables en cuanto tiempo de ejecución y uso de memoria.

Cabe decir que las pruebas iniciales se han realizado con tan solo 3 casos.

La exactitud en promedio obtenida por el método inicial es del 98.3 % y un tiempo de ejecución de 190 segundos por caso.

### 3.6. Método de evaluación

El método de evaluación elegido es una validación cruzada leave-on-out con todos los casos disponibles en la biblioteca, esto es, seleccionar un caso como objetivo a segmentar y utilizar los restantes como librería, es decir, si tuviéramos 10 casos disponibles en nuestra librería los segmentaríamos todos uno por uno utilizando los 9 restantes como casos de ejemplo.

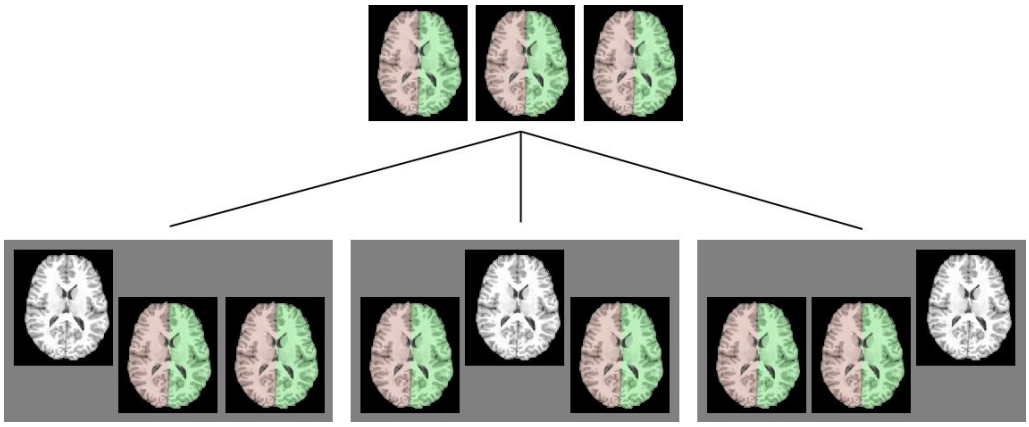

*Figura 7: Validación cruzada leave-one-out con 3 casos.*

Las medidas a tomar son el tiempo de ejecución que se emplea para la segmentación de cada caso y la similitud de la segmentación obtenida con la segmentación manual con una consideración: Solo tenemos en cuenta los voxels dentro de la máscara, es decir, descartamos el espacio que rodea al cerebro ya que es una zona no procesada la cual siempre es cero y distorsionaría los resultados aumentando el porcentaje de exactitud.

Esta medida de similitud la calculamos obteniendo un índice Kappa:

$$Kappa(A,B) = \frac{2 * (A \cap B)}{A \cup B} \tag{6}$$

Donde A representa las etiquetas obtenidas por el método y B las etiquetas hechas a mano.

## 4. Mejoras

Hasta aquí hemos descrito los datos que vamos a utilizar y la implementación inicial del método así como la dirección en la que van a ir las mejoras que se le apliquen. También hemos explicado el método de evaluación que vamos a utilizar en todos los experimentos y que criterios de medida hemos elegido que son el tiempo de ejecución y el índice Kappa de semejanza.

Solo queda mencionar que el equipo utilizado en todos los experimentos es el mismo y las pruebas se han realizado en las mismas condiciones, es decir, sin interferencias por parte de otras aplicaciones o datos ajenos al método en memoria.

### 4.1. Biblioteca de ejemplos (Aumento del número de ejemplos)

Nuestra biblioteca actual está formada por 20 casos de ejemplo segmentados manualmente. Esta es una cifra aceptable a la hora de obtener resultados realistas y pese a que hemos dicho que no íbamos a entrar a mejorar la calidad de la base de datos sería bueno analizar como mejora el método a medida que crece la biblioteca.

Para esto vamos a aprovechar que el cerebro es simétrico entre hemisferios, cerebelo y bulbo raquídeo y vamos a duplicar el número de ejemplos dándole la vuelta a todos los casos y sus segmentaciones teniendo en cuenta que estas habrá que corregirlas ya que al invertirlas estaremos cambiando de sito los hemisferios derecho e izquierdo y del mismo modo con ambas partes del cerebelo.

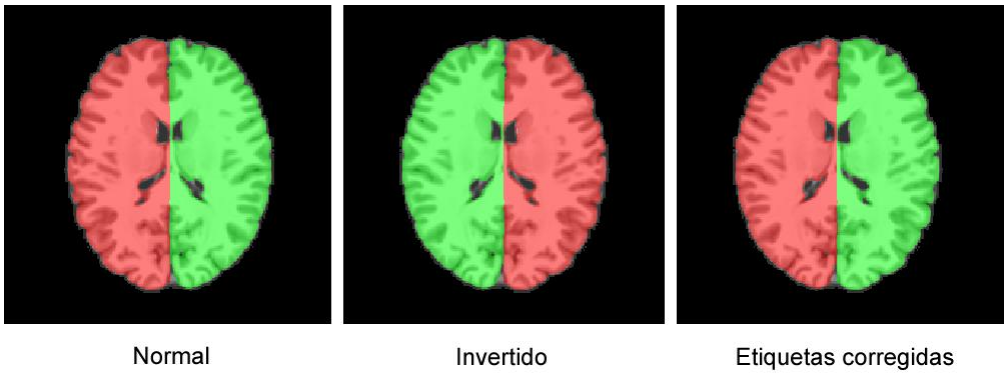


*Figura 8: Proceso de invertido de un cerebro.*

### 4.2. Preselección

Como se detalla en el punto 2.3, antes de iniciar la segmentación se hace una preselección de los 10 casos de la biblioteca que más se parecen ya que utilizar un número mayor de casos no supone una mejora significativa de la calidad de resultado pero si un aumento del tiempo de ejecución.

Con 20 casos en la biblioteca el coste computacional no era elevado pero con 40 casos lleva demasiado tiempo hacer la preselección por lo que se añade a la librería una versión de baja resolución para realizar dicha preselección mucho más rápido.

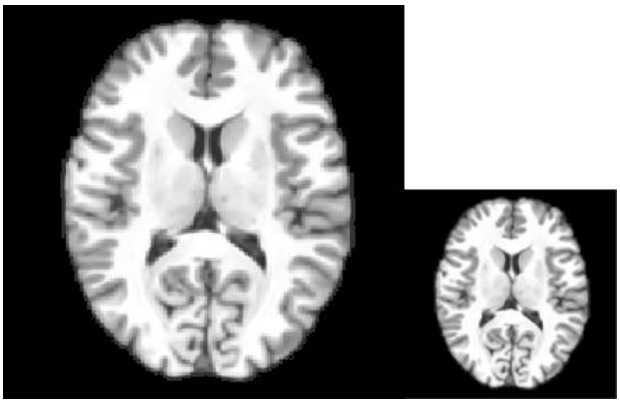

*Figura 9: Caso de ejemplo normal y reducido respectivamente.*

### 4.3. Máscara

La mejora de la máscara no solo implica cambios en la máscara. Se trata de un cambio bastante importante en el método para ajustarlo a una máscara que añade un nuevo concepto.

La máscara actual nos indica sencillamente que voxels debemos procesar y cuales quedan fuera de la masa cerebral por lo que el método visita todos los voxels del cerebro objetivo y esto es muy ineficiente ya que hay áreas en las cuales conocemos por su localización cual va a ser el valor de la etiqueta como por ejemplo los voxels que se encuentran en la zona central de los hemisferios.

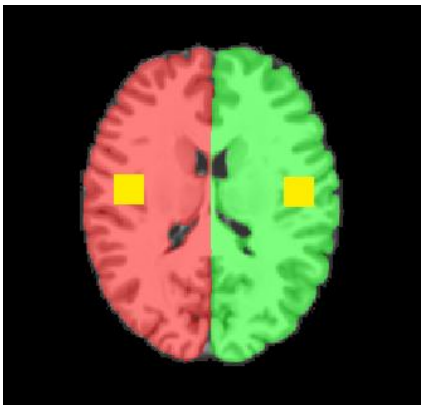

*Figura 10: Ejemplo de áreas cuya etiqueta es siempre la misma.*

Por tanto vamos a realizar dos cambios: En primer lugar vamos a reducir drásticamente el número de voxels a visitar restringiendo el área a procesar a las zonas que presentan diferencias de un cerebro a otro y en segundo lugar vamos a rellenar el resto de espacio con una estimación del valor de las etiquetas obtenida de la biblioteca.

En el proceso para generar la nueva máscara empezamos por obtener la estimación de las etiquetas calculando la moda de todas las etiquetas de la biblioteca. Después utilizaremos

estos valores para obtener las áreas de mayor variabilidad calculando las fronteras entre etiquetas. Para ello separamos las etiquetas en capas diferentes las dilatamos y obtenemos su intersección.

Solo queda dar un valor fijo que difiera de todas las etiquetas (por ejemplo 100) y solaparlo con la propia moda.

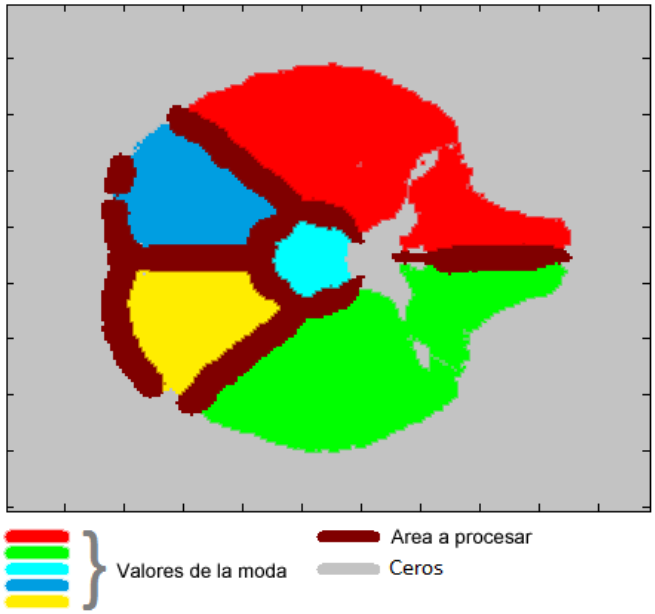


*Figura 11: Mascara mejorada.*

Pero como explico al principio del apartado este cambio no solo afecta a la máscara. También tendremos que modificar el método. Ahora cuando el valor de la máscara sea inferior a 100 se debe usar ese mismo valor y asignarlo como etiqueta.

De esto último se deduce que la máscara no se podrá utilizar sin más sino que habrá que adaptarla a cada cerebro a segmentar ya que como se muestra en la figura hemos estimado zonas que serán siempre idénticas y zonas que no lo serán como el contorno exterior de los hemisferios.

De este modo, antes de iniciar la segmentación, tendremos que etiquetar como cero todos los voxels del cerebro que tengan una intensidad por debajo de 100 con lo que se redefinirán los contornos. Además de esto, en el cerebro a segmentar pueden quedar zonas que no estén cubiertas por las etiquetas por ello habrá que hacer crecer todas las etiquetas a través de los voxels sin cubrir. Por último, puede haber áreas sin cubrir que estén aisladas del grueso del cerebro y que son imposibles de etiquetar automáticamente. Estas zonas se etiquetan para que sean visitadas por el método.

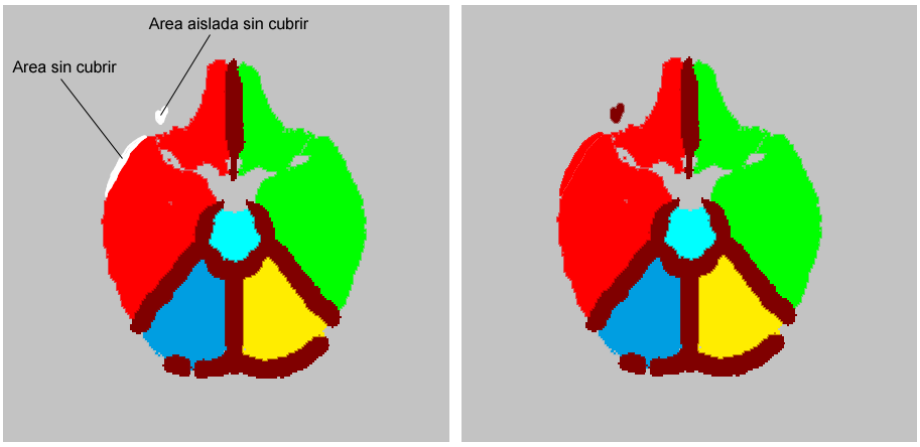


*Figura 12: Áreas sin cubrir y adaptación.*

Una característica muy a tener en cuenta de la máscara es que generarla tiene un coste computacional importante debido al cálculo de la moda sobre 40 casos. Por esta razón es más eficiente pre calcularla y posteriormente adaptarla al caso a segmentar.

### 4.4. Método multiescala

Como ya hemos visto, el método inicial se base en la localidad de los voxels para comparar regiones del cerebro objetivo con los cerebros de la biblioteca pero podemos introducir un cambio para ajustar más la localización de los parches a comparar utilizando un segundo parche más grande que el principal pero tomando voxels alternos para no sobrecargar en exceso.

Con esto se consigue caracterizar mejor el área que envuelve al voxel procesado y se obtienen parches mejor situados en los ejemplos de la biblioteca y como resultado se mejora la calidad de las segmentaciones.

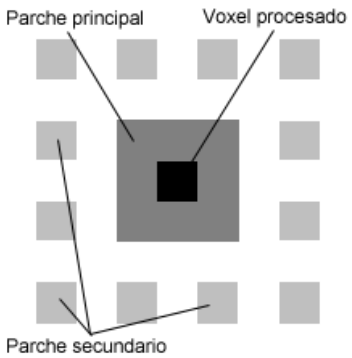


*Figura 13: Voxel central, parche principal y parche secundario.*

### 4.5. Etiquetado de múltiples voxels

Si analizamos el método y más concretamente los cálculos necesarios para etiquetar un solo voxel podemos ver que estamos desperdiciando información. Para encontrar la etiqueta adecuada para una voxel se comparan todos los voxels que forma su parche con otro parche proveniente de un cerebro de la biblioteca y tras todos los cálculos solo etiquetamos el voxel central cuando realmente hemos obtenido todo el área que es similar al área del parche que rodea al voxel objetivo.

Como consecuencia de esta observación se modifica el método para que una vez elegido un parche se utilicen las etiquetas de todos los voxels que lo forman teniendo la precaución de no escribir fuera de los límites del are a procesar definida en la máscara.

Si aplicamos esta mejora sin tener en cuenta nada más nos encontraremos con un problema y es que cada vez que utilicemos las etiquetas de un parche estaremos solapando parte de las empleadas por el parche anterior. Por esto se hace necesario implementar un método de almacenamiento y posterior selección de cada etiqueta.

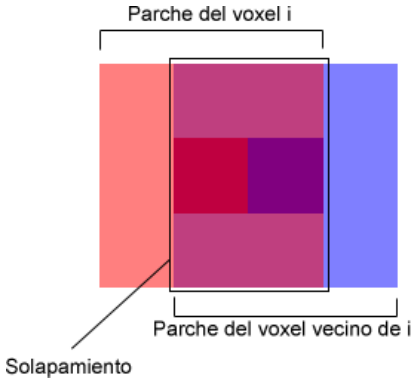


*Figura 14: Solapamiento entre parches de voxels vecinos.*

La solución pasa por almacenar todas las etiquetas que vamos obteniendo de cada parche en una variable con tamaño suficiente para contener un $peso_{(10)}$ para cada posible valor de etiqueta. Cada vez que obtengamos un parche incrementaremos el peso de sus etiquetas con la distancia entre el parche que la contiene y el parche objetivo.

Finalmente esto debe procesarse para asignar a cada voxel la etiqueta que ha obtenido mejor puntuación.

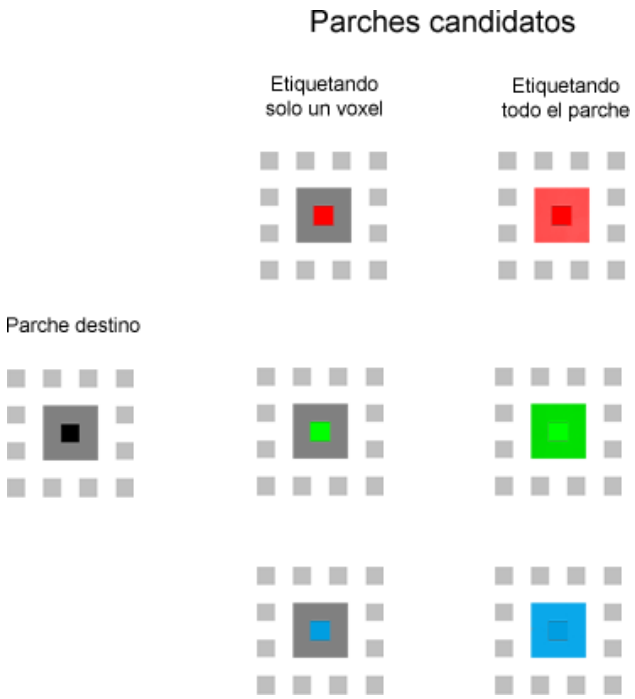


*Figura 15: Etiquetas que se aprovechan antes y después de la mejora.*

Con esta modificación se no se mejora sustancialmente el resultado de las segmentaciones sobre las cifras pero visualmente se obtiene una segmentación mucho más limpia en las zonas frontera entre etiquetas lo cual equivale a unos resultados mucho más presentables como se muestra en la figura 16.

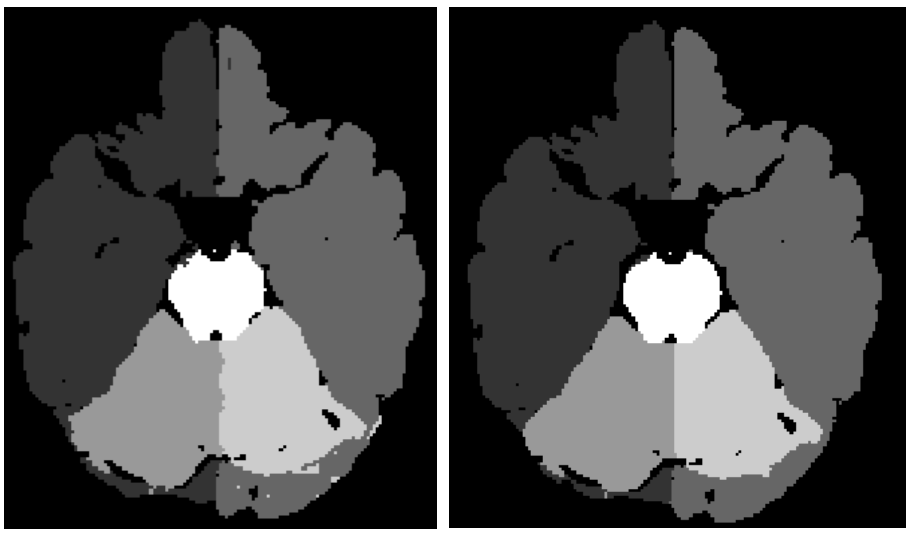

*Figura 16: Diferencia entre etiquetado de un solo voxel y etiquetado de todo el parche respectivamente.*

### 4.6. Procesado de voxels alternos

La mejora explicada en el apartado anterior introduce un concepto nuevo en el método. Ahora un voxel puede ser etiquetado varias veces debido a que se usan las etiquetas de todos los voxels del parche produciendo un solapamiento.

Dada esta situación podemos aprovecharla para etiquetar voxels sin tener que visitarlos ya que van quedar cubiertos por el área que ocupan los parches. No obstante vamos a dejar un solapamiento mínimo para que se almacenen varias alternativas para cada voxel y así escoger la mejor como ya hemos explicado en el apartado 3.5.

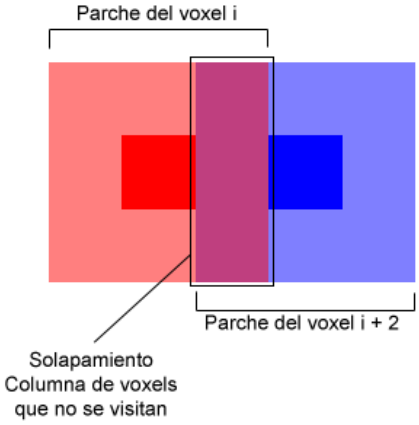


*Figura 17: Visitando voxels alternos.*

De esta forma, saltando una posición en X, Y y Z reducimos el tiempo de ejecución a la octava parte.

### 4.7. Uso de memoria y CPU

En el ámbito de la imagen médica el uso de memoria para ejecutar una segmentación se dispara debido al alto volumen de los datos.

Con la biblioteca de 20 casos se utilizaban casi 8 GB de memoria, es decir, cerca del límite del equipo de pruebas. Por tanto con una base de 40 casos era necesario mejorar la carga de los mismos.

El método para almacenarlos y cargarlos consistía en un fichero .mat[4] que almacenaba todos los casos de la biblioteca y dos ficheros más que contenían las medias y varianzas asociadas a los casos.

Estos ficheros tenían que cargarse por completo para poder trabajar con los casos por lo que la primera medida a tomar es separar cada caso en un fichero.mat independiente así como las medias y varianzas.

Además del uso de memoria también había que revisar el uso de CPU. Como ya se ha explicado la función MEX que realiza la segmentación está programada en C utilizando ejecución mediante hilos. Inicialmente, con el fin de aprovechar al máximo el rendimiento de la CPU se lanza un hilo por cada $corte_{(1)}$ de la imagen, es decir, 181 hilos.

Esto es ineficiente ya que el equipo de pruebas trabaja con un procesador de 8 núcleos y solo puede ejecutar 8 hilos simultáneamente por lo que lanzar 181 genera una gran cantidad de cambios de contexto además de llevar a la CPU al 100% de utilización lo cual no es favorable ya que hacemos el método sensible a la carga computacional que introduce el propio sistema operativo.

La solución consiste en repartir el trabajo entre 8 hilos de ejecución pero no es tan sencillo como asignar el mismo número de cortes a cada hilo ya que no todos los cortes contienen el mismo número de voxels a procesar.

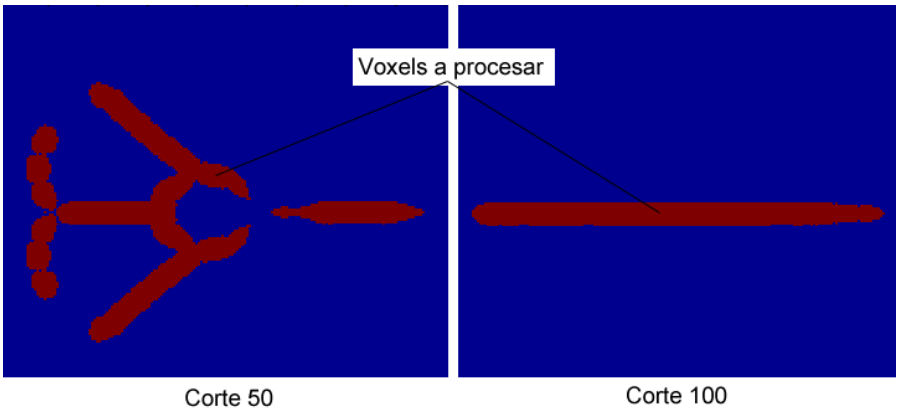


*Figura 18: Voxels a procesar en los cortes 50 y 100 de una caso.*

Por tanto se debe realizar un estudio para estimar la cantidad de voxels a procesar en cada corte.

La mejor forma de hacer es utilizar la máscara ya que esta es un resumen de los 40 casos de la biblioteca y contar por cada corte de la máscara cuantos voxels hay con valor 100 (que son los voxels a procesar). Una vez tenemos el número de voxels a procesar totales en un cerebro podemos dividirlo entre 8 y hacer 8 grupos de cortes de forma que cada grupo contenga una cantidad similar de voxels.

## 5. Ajuste de parámetros

En la descripción del método se mencionan una serie de parámetros comentando que papel desempeñan en el método pero no se entra en detalle sobre cuál es su valor óptimo ni como se ha llegado a él ya que se hizo en un trabajo previo [1].

No obstante es un punto muy importante ya que buscamos maximizar la calidad de las segmentaciones y minimizar el tiempo de ejecución y todos estos parámetros van a ser la clave para llegar a una solución de compromiso entre calidad y velocidad.

### 5.1. Número de ejemplos para la segmentación

El número de ejemplos que se utilizan para realizar la segmentación es quizás el parámetro más influyente en la calidad del resultado. Para obtener el valor adecuado hemos realizado un experimento lanzando segmentaciones consecutivas con diferentes valores para el número de casos y el resultado ha sido el siguiente:

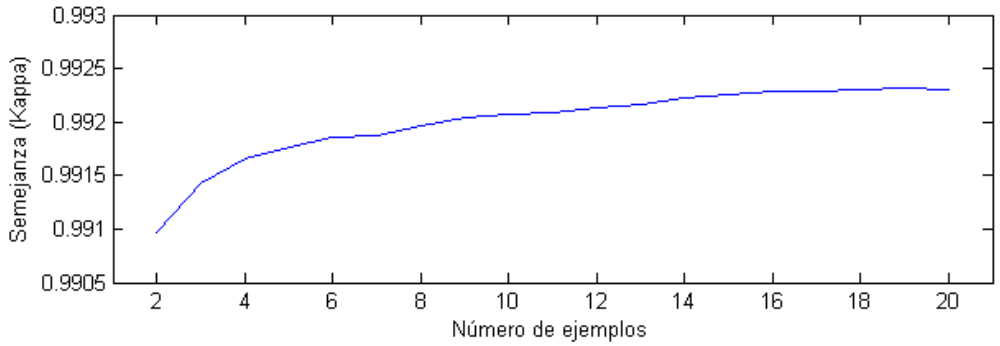


*Figura 19: Semejanza en función del número de casos de ejemplo.*

En las figuras se puede apreciar que el aumento de la calidad de los resultados es mínimo a partir de 10 casos pero el aumento del tiempo de ejecución es lineal por lo que el óptimo es utilizar alrededor de 10 casos para cada segmentación.

### 5.2. Tamaño del área de búsqueda

Como ya sabemos el método se basa en la localidad de los parches que compara pero esta condición no es estricta. Se trabaja sobre un área de búsqueda descrita por la distancia máxima en voxels sobre la que nos podemos desplazar para buscar en torno al voxel central.

Este parámetro afecta mucho al tiempo de ejecución ya que incrementarlo implica alejarse más del voxel centrar y tener que comparar muchos más parches.

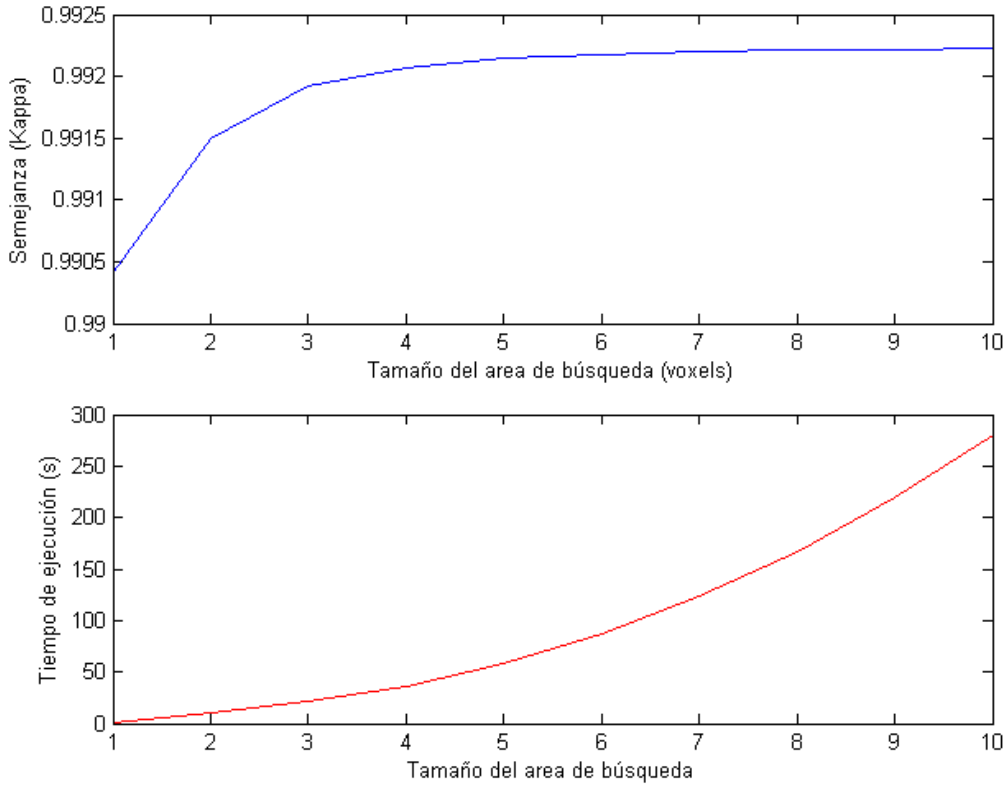


*Figura 20: Semejanza y tiempo de ejecución en función el tamaño del área de búsqueda.*

Como vemos en los gráficos a partir de 5 ya no se mejora la calidad de las segmentaciones, sería necesario aumentarlo mucho más para obtener resultados pero la perdida de eficiencia sería excesiva ya que el tiempo de ejecución crece exponencialmente por lo que el valor óptimo para este parámetro es 5.

### 5.3. Cálculo de distancia Euclidea entre parches

En el cálculo de la distancia entre parches utilizamos un valor normalizador. Este parámetro interviene a la hora de asignar pesos ponderados con la distancia a cada etiqueta para cada voxel y dado que es un valor que no influye en el flujo de ejecución no tiene ningún efecto sobre el tiempo de ejecución.

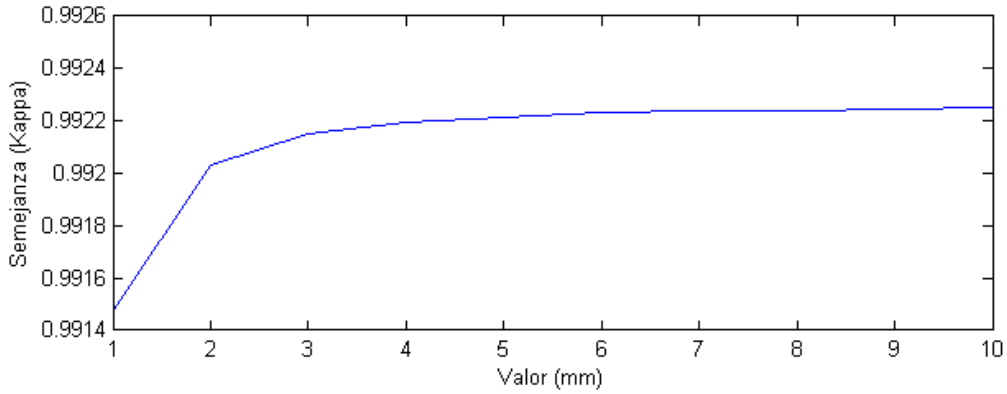


*Figura 21: Semejanza en función del valor en milímetros.*

En el grafico se observa que a partir de 5 ya no hay crecimiento luego este es su valor óptimo.

## 6. Resultados

En este apartado vamos a hablar de la evolución de los resultados obtenido al introducir las mejoras sobre el métodos y finalmente de los resultados finales con los parámetros óptimos.

La primera modificación que se hizo sobre el método fue sobre la máscara. Esto ha supuesto una reducción drástica del tiempo de ejecución. Pasamos de 166 segundos por caso a 30 segundos por caso ya que estábamos reduciendo enormemente la cantidad de voxels procesados por el algoritmo.

Además también se obtuvo un incremento en la calidad de las segmentaciones ya que el valor estimado por la moda junto con la posterior adaptación a cada caso obtiene más aciertos que procesar los voxels.

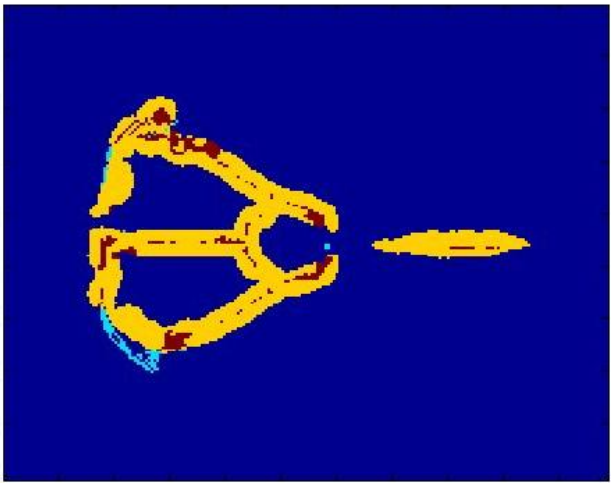

*Figura 22: Amarillo y marrón representan aciertos y fallos de la zona a procesar respectivamente, azul oscuro y claro representan aciertos y fallos de la moda respectivamente.*

Como muestra la figura la estimación previa de las etiquetas produce muy pocos errores.

La siguientes modificaciones fueron el parche multiescala y la escritura de todo el parche simultáneamente. Al introducir las modificaciones sin más aumentábamos el tiempo de ejecución significativamente de los 30 segundos por caso que habíamos obtenido a 230 segundos por caso ya que el parche multiescala añadía operaciones a la obtención del parche y cálculo de las distancias. Además al haber solapamiento de etiquetas era necesario procesarlas posteriormente lo cual suponía una carga importante ya que se procesaban de nuevo todos los voxels para obtener las etiquetas con más votos.

Además esta mejora no tuvo efecto sobre la calidad de las segmentaciones al menos en cuanto a cifras como se ve en la siguiente figura 23 aunque ya hemos explicado que los resultados con esta mejora son mucho mejores visualmente.

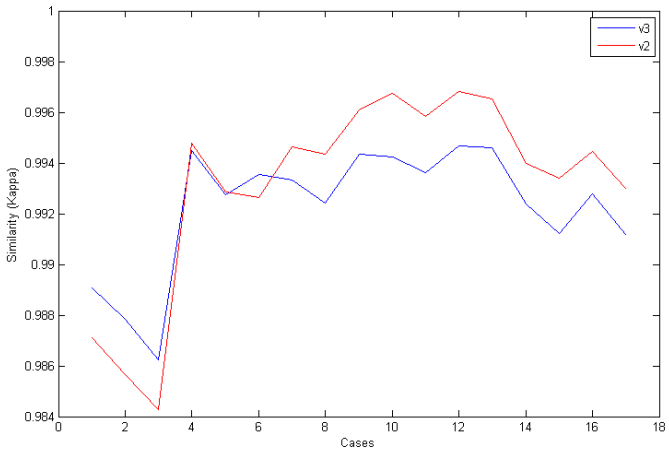


*Figura 23: Semejanza obtenida por el método sin multiescala (rojo) y con multiescala(azul).*

Por suerte esta modificación nos permitió introducir la siguiente mejora que consiste en aprovechar el solapamiento para procesar el área definida por la máscara alternando voxels y con esto bajamos el tiempo de ejecución a 78 segundos por caso saltando voxels en X e Y y a 50 segundos por caso saltando voxels en Z también y como era de esperar esta modificación no empeoró la calidad de las segmentaciones.

Las mejoras sobre el uso de memoria no repercutieron en ninguna mejora sobre el tiempo de ejecución ya que el equipo de pruebas no está sometido a ninguna carga adicional salvo la del sistema operativo así que los beneficios de esta modificación se quedan en la teoría de que se reduce la sensibilidad a cargas adicionales sobre la máquina.

Finalmente el aumento de casos de ejemplo de la biblioteca no afectó significativamente a la calidad del método lo cual no es un resultado del todo concluyente ya que todas las imágenes pertenecen a sujetos normales y sanos pero sí que obtuvimos una reducción del tiempo de ejecución a 37 segundos al vernos obligados a mejorar el proceso de selección de los 10 casos para la segmentación.

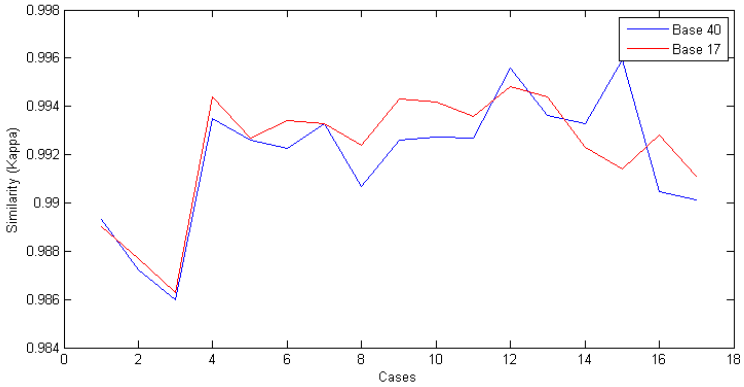


*Figura 24: Diferencias de semejanza con una librería de 17 casos y una de 40.*

## 7. Comparativa

Para ver en qué medida se han cumplido los objetivos de este proyecto vamos a realizar un estudio comparativo con un método actual el cual funciona bastante bien.

El método escogido es Adaptive disconnection [7] del cual hablamos a continuación.

### 7.1. Descripción del método Adaptive disconnection

Una característica a destacar de este método es que a diferencia del nuestro se basa en la estimación del volumen parcial. El volumen parcial es algo inherente a la tecnología de RMN e implica que algunos voxels contengan información de más de un tejido.

Esto es así debido al carácter discreto de las imágenes con las que trabajamos en las que cada voxel no equivale a la unidad mínima de materia en el cerebro sino que sintetiza el contenido de una pequeña región.

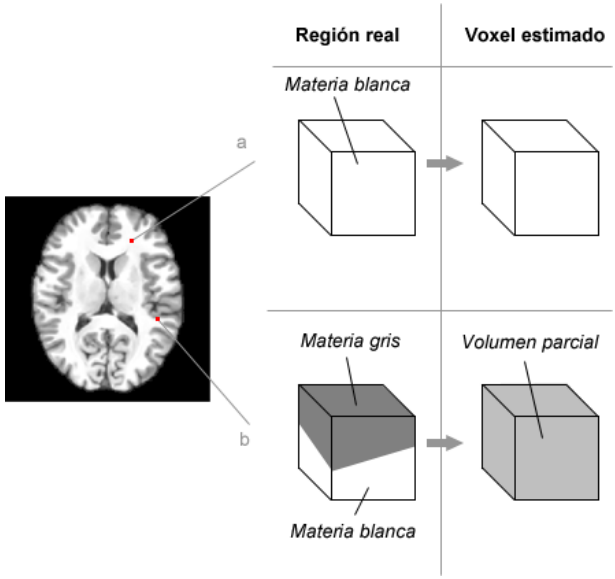


*Figura 25: Explicación del volumen parcial donde a representa una región de un solo tejido y b una región entre dos tejidos.*

Respecto al algoritmo podemos decir que se divide en dos grandes etapas: (1) la descomposición compartimental del volumen del cerebro en los hemisferios cerebrales (HC), el cerebelo (CB) y el bulbo raquídeo (BR), (2) segmentación hemisférica de HC y CB.

El primer paso es un proceso basado en reconstrucción estructural. La región formada por la materia blanca (MB) y el volumen parcial de materia blanca y materia gris (MB/MG) se divide en semillas$_{(12)}$ compartimentales para HC, CB y BR con el algoritmo de formas de cuello de botella basado en ecuaciones en derivadas parciales (PDE) y la clasificación basada en voxel mediante el valor potencial de la información (IPV).

De acuerdo con el conocimiento anatómico previo de la adyacencia entre HC, CB y BR, sólo BR está conectado con HC y CB, y no hay conexión directa entre el HH y CB. Sin embargo, las conexiones espurias entre CH y CB causadas por el efecto del volumen parcial (PVE) siempre existen en el dominio de la MG, pero se observan rara vez en el dominio de MB. Por lo tanto, se selecciona la región MB ∪ MG / MB como fuente de semillas compartimentales porque esta zona es la más grande de la región interior del tejido cerebral que probablemente podría mantener la adyacencia compartimental simple.

Debido a que el volumen parcia (VP) entre líquido cefalorraquídeo (LCR) y MG existe principalmente en las fronteras de HC, CB y BR las formas de la HC, CB y BR se reconstruyen hacia la zona de LCR / MG, en lugar de la interfaz de MG-LCR, utilizando un algoritmo de región simultánea en crecimiento, que extiende a las semillas de acuerdo a un criterio de cierre de frontera.

Después de descomponer el volumen del cerebro en HC, CB y BR, los voxels de LCR / MG, contienen una cierta cantidad de líquido cefalorraquídeo, se descartan de la HC y del

volumen de CB para asegurarse de que las formas de cuello de botella interhemisféricas son detectables por el algoritmo de formas de cuellos de botella basado en PDE.

Por último, los hemisferios izquierdo y derecho de HC y CB son desconectados en los cuellos de botella detectados en zonas interhemisféricas.

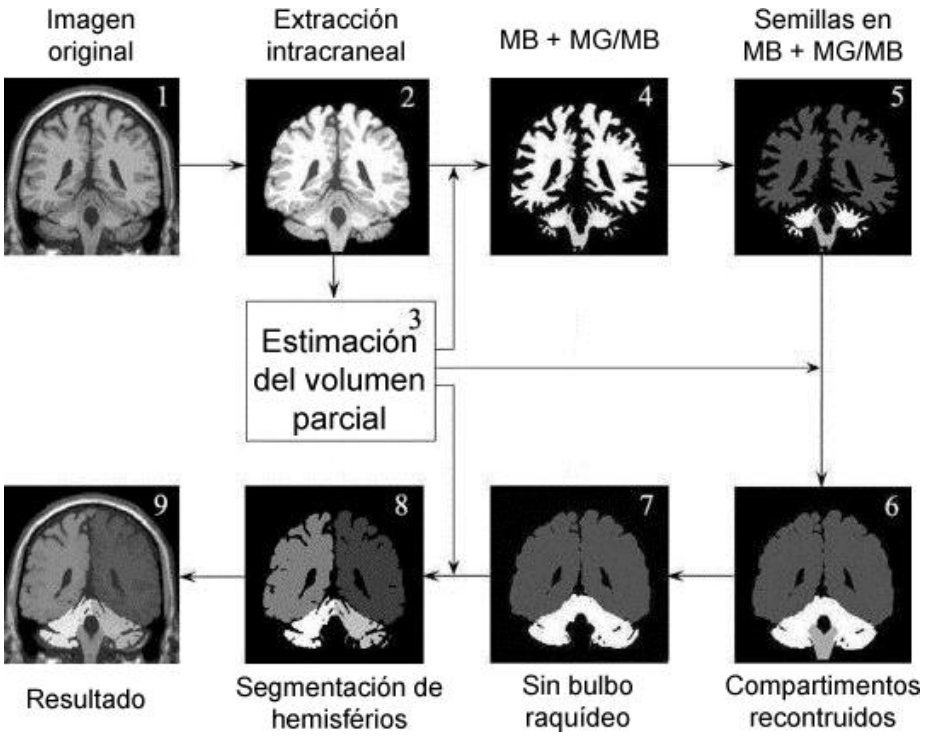


*Figura 26: Pasos de Adaptive disconnection.*

### 7.2. Método de evaluación

No podemos evaluar los resultados de este método de la misma forma que los nuestros ya que parte de premisas diferentes. En nuestro caso aplicamos una normalización en intensidad y descartamos los valores por debajo de cien pero en Adaptive disconnection esto no es así.

Por esto vamos a evaluar las diferencias solo en las zonas frontera entre las diferentes partes del cerebro usando para ello nuestra propia máscara ya que se ajusta perfectamente a este propósito.

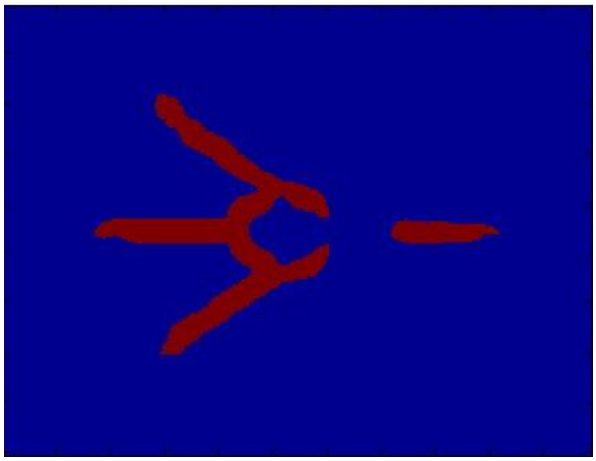

*Figura 27: Corte sagital de la máscara que cubre las fronteras entre partes del cerebro.*

### 7.3. Comparación de resultados

Al aplicar está máscara y reducir la evaluación a las zonas más conflictivas del cerebro ambos *métodos muestran diferencias significativas*. Los resultados obtenidos son los siguientes:

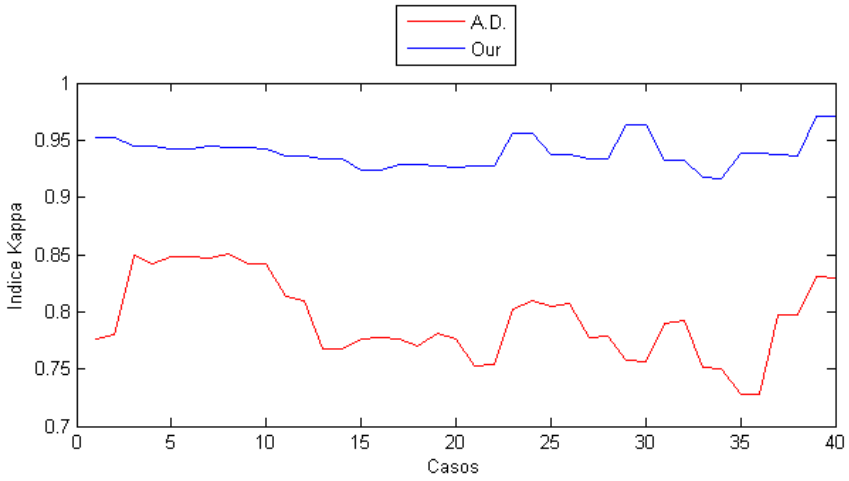


*Figura 28: Grafico comparativo de similitud obtenida por ambos métodos en 40 casos donde A.D. hace referencia a Adaptive disconnection y Our hace referencia a nuestro método.*

La exactitud media de los resultados obtenidos con Adaptive disconnection es del 79,35 % mientras que la obtenida con nuestro método es de 93.94 % (solo para la zona de mascara comentada). Para verificar que las diferencias entre los datos son significativos realizamos un gráfico Box Whisker y un test de comparación de medias con una confianza del 95%.

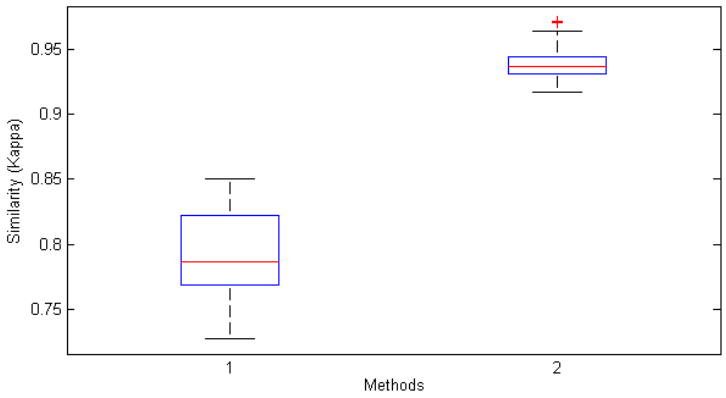


*Figura 29: Diagrama Box Whisker donde 1 se corresponde con Adaptive disconnection y 2 con nuestro método.*

En cuanto a las áreas de mejora en la figura 9 podemos ver las zonas de intersección entre los fallos de Adaptive disconnection y los aciertos de nuestro método.

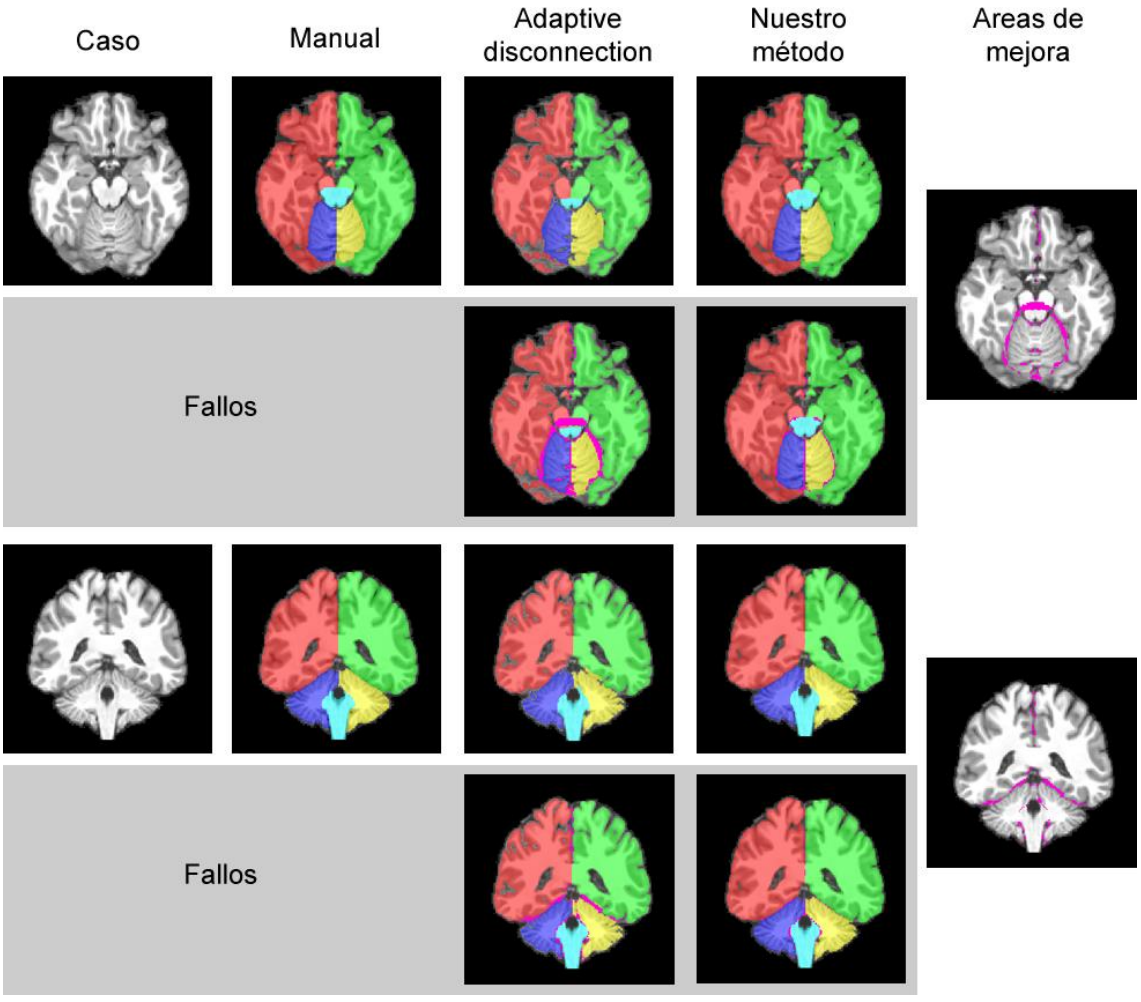


*Figura 31: Comparativa de resultados entre Adaptive disconnection y nuestro método.*

En cuanto al tiempo de ejecución la diferencia es sustancial pues Adaptive disconnection necesita en promedio unos 400 segundos por caso mientras que nuestro método tarda unos 40 segundos.

## 8. Conclusiones

En este proyecto se han desarrollado una serie de mejoras sobre un método de segmentación automática de los hemisferios cerebrales así como del cerebelo y bulbo raquídeo que han supuesto una gran diferencia en los resultados obtenidos. En cuanto a calidad de los resultados se han obtenido una mejora superior al 1% lo cual puede parecer insignificante pero no lo es si pensamos en que la mayor parte del volumen cerebral no presenta dificultad a la hora de decidir que etiqueta asignar. En cuanto al tiempo de ejecución se ha reducido al 25% lo cual es una mejora muy significativa.

En definitiva se ha obtenido un método que mejora el estado del arte que además presenta grandes ventajas en cuanto a velocidad.

## 9. Discusión

Queda abierta una vía de desarrollo en referencia a la mejora de la biblioteca de ejemplos sobre la que trabaja este método no solo en cuanto a ampliarla para obtener mejores resultados sino en cuanto a adaptarla para trabajar con patologías concretas como pueda ser el Alzheimer.

## 10. Agradecimientos

Agradecer al profesor Jose V. Manjón su interés y esfuerzo en la dirección de este proyecto que han superado con creces el máximo exigible en una tarea como esta.

## 11. Glosario

1. Corte: Conjunto de voxels que forman uno de los planos que cortan al cerebro.
2. DP: Densidad protónica.
3. Etiqueta: Valor asignado a un voxel que lo clasifica como miembro de una parte del cerebro.
4. Fichero .mat: Formato de ficheros empleados en Matlab para almacenar variables.
5. Máscara: Conjuntos de valores análogos a la representación del cerebro que se usa para guiar el proceso de segmentación y aportar información estimada.
6. Matlab: Programa de carácter científico que ofrece numerosas herramientas para la investigación.
7. MEX: Complemento de Matlab que permite integrar funciones programas en C dentro de fragmentos de código en el propio lenguaje Matlab.
8. MNI: Estandar del Montreal Neurological Institute que define la distribución de los voxels de un cerebro en el espacio tridimensional.
9. Parche: Conjunto de voxels en forma de cubo pertenecientes a una región de una imagen de RMN.
10. Peso: Valor que asignamos a una etiqueta candidata para determinar su calidad.
11. Segmentación: Proceso de clasificación de todos los voxels de una imagen de RMN. También hacemos referencia con este término al resultado de dicho proceso.
12. Semilla: Llamamos semilla compartimental a aquel voxel se toma como referencia para representar una parte del cerebro en la cual se encuentrá.
13. T1: Respuesta en t1.
14. T2: Respuesta en t2.

## 12. Bibliografía


1. Pierrick Coupe, Jose V. Manjón, Vladimir Fonov, Jens Pruessner, Montserrat Robles, D. Louis Collins. Patch-based Segmentation using Expert Priors: Application to Hippocampus and Ventricle Segmentation. NeuroImage, 54(2): 940-954, 2011.

2. José V. Manjón, Jussi Tohka, Montserrat Robles. Improved Estimates of Partial Volume Coefficients from Noisy Brain MRI Using Spatial Context. Neuroimage, 53(2), 480-490, 2010.

3. José V. Manjón, Pierrick Coupé, Luis Martí-Bonmatí, Montserrat Robles, Louis Collins. Adaptive Non-Local Means Denoising of MR Images with Spatially Varying Noise Levels. Journal of Magnetic Resonance Imaging, 31,192-203, 2010.

4. VBM8
   http://dbm.neuro.uni-jena.de/

5. SPM8
   http://www.fil.ion.ucl.ac.uk/spm/
   Statistical Parametric Mapping: The Analysis of Functional Brain Images . Edited By William D. Penny, Karl J. Friston, John T. Ashburner, Stefan J. Kiebel & Thomas E. Nichols Academic Press Title, 2007, SBN: 978-0-12-372560-8

6. IXI dataset http://www.brain-development.org/

7. L. Zhao, U. Ruotsalainen, J. Hirvonen, J. Hietala and J. Tohka . Automatic cerebral and cerebellar hemisphere segmentation in 3D MRI: adaptive disconnection algorithm. *Medical Image Analysis* , 14(3): 360 - 372, 2010 .
   http://www.cs.tut.fi/~jupeto/software.html

8. M. Brummer, Hough transform detection of the longitudinal fissure in tomographic head images. *IEEE Transactions on Medical Imaging*, **10** 1 (1991), pp. 74–81

9. Y. Liu, R. Collins and W. Rothfus, Robust midsagittal plane extraction from normal and pathological 3D neuroradiology images. *IEEE Transactions on Medical Imaging*, **20** 3 (2001), pp. 175–192.

10. S. Prima, S. Ourselin and N. Ayache, Computation of the mid-sagittal plane in 3D brain images. *IEEE Transactions on Medical Imaging*, **21** 2 (2002), pp. 122–138.

11. C. Sun and J. Sherrah, 3D symmetry detection using the extended Gaussian image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19** 2 (1997), pp. 164–168.

12. A. Dale, B. Fischl and M. Sereno, Cortical surface-based analysis: I. Segmentation and surface reconstruction. *NeuroImage*, **9** (1999), pp. 179–194.

13. N. Krigeskorte and R. Goebel, An efficient algorithm for topologically correct segmentation of the cortical sheet in anatomical MR volumes. *NeuroImage*, **14** 2 (2001), pp. 329–346.

14. J.-F. Mangin, D. Rivière, A. Cachia, E. Duchesnay, Y. Cointepas, D. Papadopoulos-Orfanos, P. Scifo, T. Ochiai, F. Brunelle and J. Régis, A framework to study the cortical folding patterns. *NeuroImage*, **23** (2004), pp. 129–138.

15. Mangin, J.-F., Régis, J., Frouin, V., 1996. Shape bottlenecks and conservative flow systems. In: IEEE Work. MMBIA, San Francisco, CA, pp. 319–328.

16. Y. Hata, S. Kobashi, S. Hirano, H. Kitagaki and E. Mori, Automated segmentation of human brain MR images aided by fuzzy information granulation and fuzzy inference. *IEEE Transactions on Systems, Man and Cybernetics Part C – Applications and Reviews*, **30** 3 (2000), pp. 381–395

17. P. Marais and J. Brady, Detecting the brain surface in sparse MRI using boundary models. *Medical Image Analysis*, **4** (2000), pp. 283–302.

18. L. Liang, K. Relly, R. Woods and D. Rottenberg, Automatic segmentation of left and right cerebral hemispheres from MRI brain volumes using the graph cuts algorithm. *NeuroImage*, **34** 3 (2006), pp. 1160–1170.

19. F. Maes, K. Van Leemput, L. DeLisi, D. Vandermeulen and P. Suetens, Quantification of cerebral grey and white matter asymmetry from MRI. *Lecture Notes in Computer Science*, **1679** (1999), pp. 348–357.