Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques Universitat de Barcelona

ALGORITMO SUPERVISADO DE SEGMENTACIÓN AUTOMÁTICA PARA IMÁGENES DE RESONANCIA MAGNÉTICA

Andreu Hinarejos Gimenez

Directora: Laura Igual

Realizado en: Departament de Matemàtica Aplicada i Anàlisi. UB

Barcelona, 20 de junio de 2014

Abstract

An exponential improvement in computation capacity throughout the last decades has allowed the use of further computationally demanding algorithms to solve many kinds of problems in real time. In this project both computer vision and machine learning techniques are applied to support a neuroimage study. In particular, a fully automatic method to segment the caudate nucleus of the brain from a magnetic resonance image (MRI). Studies have shown that neuroanatomical abnormalities in the caudate nucleus are strongly related to pediatric attention-deficit/hiperactivity disorders (ADHD). Therefore, providing an automatic subjectiveless tool to segment its volume not only improves the diagnose, but it also speeds up the process, freeing the experts from the arduous segmenting task. In order to achieve this purpose Graphcut unsupervised was developed. In this project a supervised term is added to the currently existing envoirment. Particularly a support vector machine classifier is trained with a set of MRI slices, which is used to refine the segmentation algorithm results.

Resumen

Un crecimiento exponencial en la capacidad de computación a lo largo de las ultimas décadas ha permitido la utilización de algoritmos que precisan un mayor coste computacional para resolver multitud de problemas en tiempo real. En este proyecto técnicas tanto de visión por computador como de aprendizaje automático se aplican para prestar apoyo a un estudio de neuroimagen. En particular se presenta un método completamente automático para segmentar el núcleo caudado del cerebro humano a partir de una imagen por resonancia magnética (MRI). Estudios han demostrado que ciertas anormalidades neuroanatómicas en el núcleo caudado están relacionadas con el trastorno de déficit de atención e hiperactividad (TDAH) en niños. Por este motivo, ofrecer una herramienta automática y objetiva para segmentar el volumen del caudado no solo mejora el diagnóstico sino que también acelera el proceso, liberando a los médicos expertos de la árdua tarea de segmentación manual. Con la finalidad de alcanzar estos objetivos se desarrolló el algoritmo Graphcut unsupervised. En este proyecto, un término supervisado se añade a este entorno. En concreto, se entrena un clasificador 'SVM' con un conjunto de imágenes MRI. Seguidamente este clasifiador se utiliza para refinar los resultados de segmentación

Índice de contenido

1 Introducción

1.1 Ámbito del problema

Las áreas de investigación de este proyecto son la visión artificial y el aprendizaje automático aplicado a la medicina. Los avances en visión por computador y el aprendizaje automático, campos en constante innovación y crecimiento, han abierto las puertas a la aplicación de estas tecnologías para ofrecer apoyo y automatizar multitud de procesos en distintos contextos.

Por otro lado, en el ámbito de la neuroimagen, se ha demostrado que existe una relación entre ciertas características neuroanatómicas del núcleo caudado del cerebro humano y el trastorno por déficit de atención e hiperactividad (TDAH). Para poder utilizar estas características en su diagnóstico es necesario segmentar esta región. Uno de los resultados más replicados en estudios clínicos concluye que el volumen de esta región en los pacientes es significativamente menor que en el grupo de control.

Existen diversos algoritmos de segmentación automática, entre los mas conocidos se hallan la detección de contornos, el thresholding y el clustering. Todos ellos tienen en común que se basan en la información de color contenida en la imagen. No obstante, el ser humano utiliza mucha más información que esta al segmentar, cuenta con conocimientos previos obtenidos a partir de un proceso de aprendizaje. Esto ha motivado la aplicación de técnicas de aprendizaje automático para mejorar los resultados de la segmentación.

1.2 Antecedentes

Este trabajo se enmarca dentro de un proyecto de investigación en el que colaboran profesores de la Universitat de Barcelona y el Grupo de Neurociencia Cognitiva del Hospital del Mar. En este proyecto se está desarrollando una librería, *BrainSegmentation,* de métodos de segmentación de estructuras subcorticales del cerebro humano a partir de imágenes MRI mediante distintas técnicas.

El algoritmo CaudateCut fue desarrollado para la segmentación del núcleo caudado en imágenes por resonancia magnética. Las aplicaciones de este algoritmo ya han sido presentadas en dos publicaciones [2,3]. No obstante para que este algoritmo pueda ser utilizado de un modo práctico, por médicos y personal no experto en informática, en el diagnóstico de nuevos pacientes, es imprescindible introducir cambios. Estos cambios son los que se realizarán en este proyecto, para adaptar CuadateCut a la librería *BrainSegmentation.*

1.3 Objetivos

El objetivo de este proyecto es continuar con el desarrollo de la librería BrainSegmentation, incluyendo un nuevo algoritmo. En particular, se pretende adaptar el código del algoritmo CaudateCut a la librería con la finalidad de proveer a los usuarios finales las herramientas necesarias para el correcto uso y actualización del software, si procede.
Para llevar a cabo esta tarea es imprescindible realizar un proceso de ingeniería del software, cuya finalidad es estructurar y adaptar el código al nuevo sistema.
Asimismo, se sigue un proceso de experimentación y análisis empírico de los resultados, junto a un proceso de calibración para obtener la mejor segmentación posible mediante este método, conocido como Graphcut supervised.
Se sigue un proceso de diseño modular y de bajo acoplamiento. El software se ha diseñado para

satisfacer las necesidades no únicamente de investigadores sino también de personal clínico, y es imprescindible este modelo de diseño para facilitar futuras modificaciones.

En concepto de objetivos no funcionales del trabajo se pretende obtener un compromiso entre tiempo y rendimiento razonable, cuyos parámetros puedan ser modificados con facilidad, ya sea por una mejora en las tecnologías, o en los datos de entrenamiento.
Además, se compararán los resultados obtenidos mediante cada método y se realizará un análisis exhaustivo de los mismos.

1.4 Motivación

El desarrollo de software para investigación con fines clínicos es sin duda uno de los estandartes de este sector en auge. Mejorar la calidad de la atención clínica, el diagnóstico de los pacientes, o incluso su calidad de vida, supone una gratificación inconmensurable en sí misma.

A título personal, como hijo de paciente y su médico que soy, siempre me ha llamado la atención la medicina. A lo largo de mis estudios por vocación en informática no he parado de ver infinidad de aplicaciones en este ámbito, y no descarto la posibilidad de dedicar mi carrera a ello. Gracias a mis estudios recientemente he conseguido trabajo en una importante empresa multinacional que se encarga de la fabricación de maquinaria de análisis clínico y de software de laboratorio.
La realización de este proyecto me ha ofrecido una nueva perspectiva, me ha permitido evaluar la dimensionalidad de un problema real, y aprender mucho sobre la materia. Asimismo estoy satisfecho de haber podido aportar mi grano de arena a un proyecto con aplicaciones prácticas[3], y no meramente teóricas.

1.5 Organización de la memoria

En este apartado se detalla la organización del resto del documento, que por conveniencia ha sido dividido como se estipula a continuación:

Análisis del problema
En esta sección se detalla el contexto médico de trabajo, se profundiza en terminología y conceptos avanzados para la correcta comprensión del proyecto realizado. Asimismo, se ofrece una descripción en detalle del algoritmo Braincut unsupervised[5], ya presente en *BrainSegmentation*, pues resulta imprescindible tener una visión global del funcionamiento del mismo para poder enmarcar correctamente las modificaciones añadidas en este proyecto.

Desarrollo
Este apartado describe en detalle el trabajo que se ha realizado, tanto el marco teórico como a nivel de diseño e implementación, para desarrollar la versión supervisada de BrainCut unsupervised[5]. También se describen las librerías utilizadas, de carácter imprescindible, y los requisitos del sistema para el correcto funcionamiento del software.

Experimentos y resultados:
En esta sección se especifican las pruebas realizadas, a la vez que se completa un análisis de los resultados obtenidos tanto a nivel cuantitativo como cualitativo, así como una evaluación de los tiempos de ejecución requeridos.

Conclusiones

Finalmente se realiza un análisis crítico del trabajo realizado, se valora la calidad del producto final, los objetivos alcanzados, y se estipulan las futuras líneas de desarrollo para BrainSegmentation.

2 Análisis del problema

A lo largo de esta sección, se describen las características específicas del problema en cuestión, inicialmente se explican una serie de conceptos imprescindibles para el correcto entendimiento del trabajo realizado. Seguidamente, en el apartado número 2, se detalla el funcionamiento de una primera adaptación de CaudateCut, un algoritmo conocido como Braincut unsupervised. En esta sección se detalla cada módulo que lo compone, y las interacciones entre éstos.

2.1 Contexto de trabajo

2.1.1 Imagen por resonancia magnética (IRM)

Las imágenes por resonancia magnética[8,9] son una práctica muy común en el diagnóstico médico. Se utiliza un campo magnético para alinear los átomos, generalmente de hidrógeno, de la zona que se quiere analizar. Seguidamente se emite un pulso de ondas de radio a una frecuencia de resonancia particular*, que al rebotar contra el tejido a dibujar es leído por un sensor. Estos datos son postprocesados en un ordenador, que aplicando la transformada de Fourier** se recomponen en una imagen bidimensional de la zona de interés. El formato de obtención de las imágenes consiste en una serie de cortes longitudinales, que al ser apilados forman un volumen del órgano o de la estructura analizada. En la Figura 1 pueden apreciarse una serie de ejemplos de este tipo de imágenes.

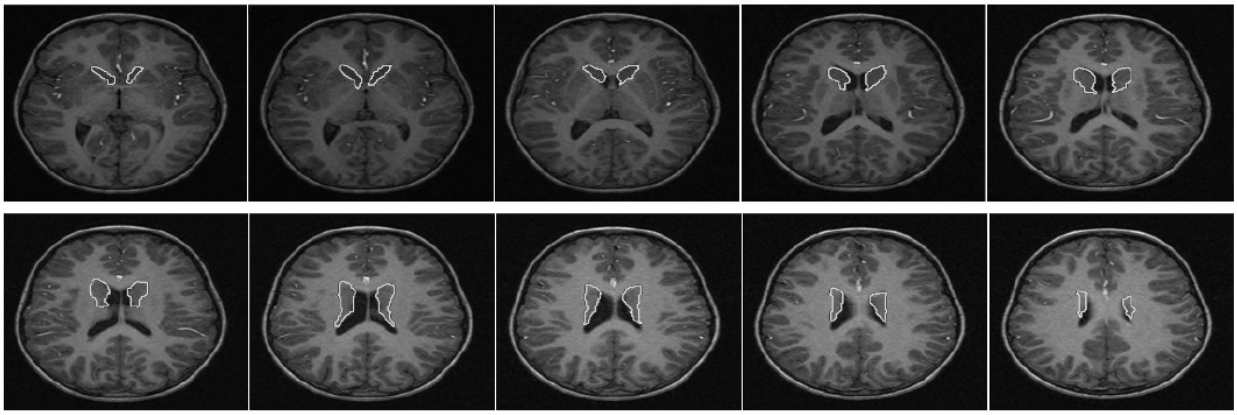
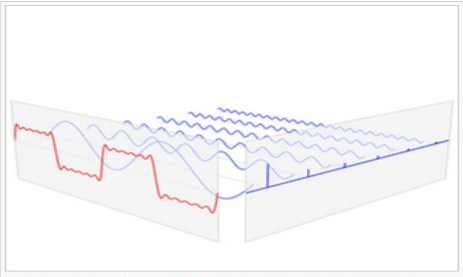


Figura 1 Imágenes por resonancia magnética en perspectiva axial. Núcleo caudado en blanco.

*La frecuencia de resonancia se calcula a partir de la fuerza del campo magnético, en Teslas, y de la composición del tejido.

** Función matemática descrita por Joseph Fourier (1768-1830), que permite la conversión entre el dominio espacial o temporal y el frecuencial[9]



The Fourier transform relates the function's time domain, shown in red, to the function's frequency domain, shown in blue. The component frequencies, spread across the frequency spectrum, are represented as peaks in the frequency domain.

2.1.2 Núcleo Caudado

El núcleo caudado es una de las tres estructuras básicas que componen los ganglios basales, junto con el putamen y el globus pallidus .Como puede verse en la Figura 2 se halla cerca de la base del cerebro, en el interior del telencéfalo. Esta región cerebral se encarga de integrar la información espacial con la respuesta motora, y se le atribuye un papel importante tanto en la memoria como en el aprendizaje. Estudios recientes han confirmado que variaciones en el volumen de esta región están estrechamente vinculadas con ciertos desordenes cerebrales, como el trastorno de déficit de atención e hiperactividad.

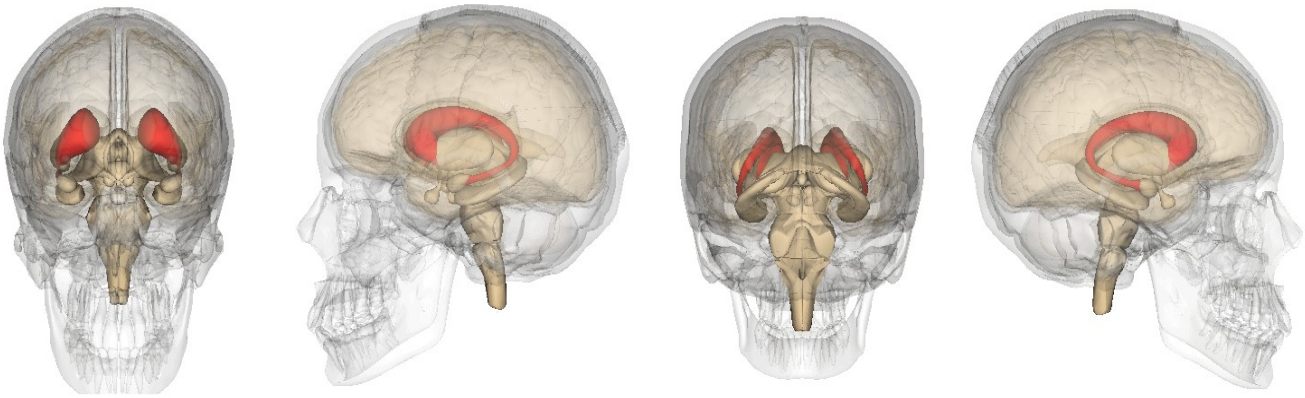


Figura 2 Posición del núcleo caudado (rojo) en el cráneo. Vista frontal, lateral izquierda, trasera y lateral derecha.

2.1.3 Visión Artificial

Este campo se encarga del estudio de métodos para el procesamiento, el análisis y el entendimiento de imágenes. Esta disciplina aplica sus teorías para construir un sistema de visión por ordenador, que puede aplicarse a la navegación, la interacción entre hombre y máquina o el modelado de objetos y entornos, entre otros campos.

2.1.3.1 Segmentación

La segmentación es un procedimiento característico de la visión artificial. Consiste en dividir una imagen en distintos conjuntos de píxeles, de forma que los píxeles de cada grupo comparten una serie de características comunes que los distinguen de los otros grupos. Esta característica puede ser su nivel de gris, su textura o posición, o una combinación de ellas. Aplicado al problema que intentamos resolver, queremos separar los píxeles que pertenecen al núcleo caudado de los que no.

2.1.4 Aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial que concierne todo lo relativo al estudio y la implementación de sistemas que pueden aprender a partir de datos. Arthur Samuel la

definió en 1959 como "campo de estudio que otorga a los ordenadores la capacidad de aprender sin ser explícitamente programados", o en otras palabras, se encarga de realizar predicciones a partir de propiedades conocidas aprendidas de los datos.

Existen distintas técnicas de aprendizaje automático, entre las que destacan el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo.

En el aprendizaje no supervisado[1] se desconoce a priori la estructura de la información, y se carece de etiquetas, es decir ground truth. Por este motivo aplican algoritmos como por ejemplo el clustering.
Por otro lado, en el aprendizaje por refuerzo se premian y se penalizan ciertas acciones del software en un entorno definido y concreto. El entorno suele definirse como un proceso de decisión de Markov[1]. Suele buscarse un equilibrio entre exploración del entorno y explotación del conocimiento acumulado a través de acciones previas.

Ya que en este proyecto se ha realizado una implementación supervisada, en el siguiente apartado se detallan sus características principales.

2.1.5 Aprendizaje supervisado

El aprendizaje supervisado[1] es un procedimiento mediante el cual se deduce una función a partir de los datos de entrenamiento, este hecho lo distingue del aprendizaje no supervisado, y permite aproximaciones que pueden aplicarse en una mayor variedad de contextos.
A priori se conoce información referente a los datos, sus etiquetas, o en otras palabras su ground truth. Las etiquetas permiten la extrapolación de una función que puede utilizarse para clasificación o regresión, en nuestro caso, queremos utilizar una máquina de vectores soporte para clasificar píxeles de un modo binario, es decir si son caudado, o no.

2.2 Braincut Unsupervised

Braincut Unsupervised es un algoritmo para la segmentación automática del núcleo caudado a partir de una imagen por resonancia magnética, que almacena el conjunto de cortes en perspectiva axial que conforman un modelo virtual del volumen cerebral del paciente. Este algoritmo está basado en el proyecto de investigación CaudateCut, y está detallado en profundidad en [5]

El actor principal es un médico cuyo objetivo es realizar una segmentación, para ello introduce el volumen de RMI y ejecuta el programa. Al finalizar, el médico obtiene el resultado de la segmentación como salida.
Braincut unsupervised además permite a un investigador introducir otra imagen con una segmentación manual para comparar el resultado, pudiendo por lo tanto evaluar el algoritmo, y modificarlo a conveniencia. Pueden consultarse los detalles de este algoritmo en [5].

Braincut unsupervised es una primera adaptación de CaudateCut, no obstante carece de ciertas

funcionalidades que tenía éste, por lo que los resultados obtenidos no son tan satisfactorios. Aun así, ofrece una base firme sobre la que desarrollar este proyecto, adaptar la versión completa de CaudateCut, conocido como Braincut supervised, o Braincut.

Este sistema está organizado en distintos módulos, como se puede apreciar en la Figura 4, cada uno de ellos cumple con una función particular que se detalla a continuación.

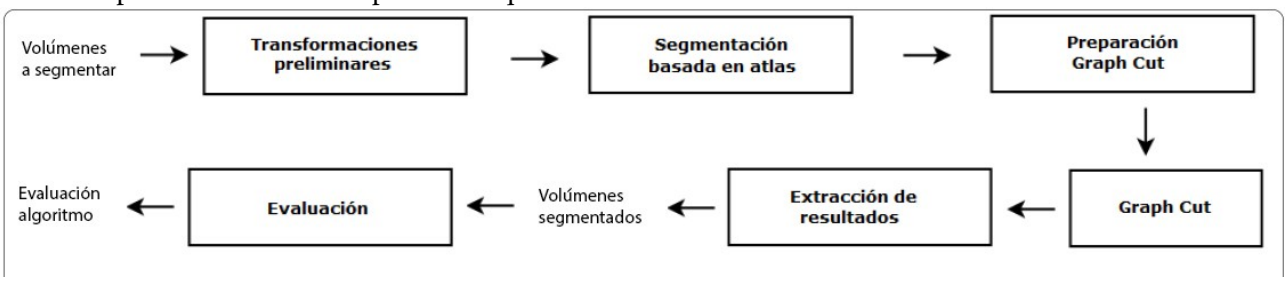


Figura 4) Módulos que componen Barincut Unsupervised. Procedimiento secuencial en cascada.[5]

A continuación se detalla el contenido de cada módulo.

2.2.1 Transformaciones preliminares

En esta etapa inicial se anonimiza a los sujetos, para preservar su intimidad y prevenir la vulneración de la confidencialidad del secreto profesional.
También se unifican los formatos de entrada, para permitir la máxima independencia de los siguientes módulos.

2.2.2 Segmentación basada en atlas

Braincut se ha estructurado acorde al procedimiento que sigue Caudatecut. En este proyecto inicialmente es menester segmentar las imágenes mediante un algoritmo basado en atlas que se explicará a continuación, para seguidamente refinar el resultado obtenido mediante Graph Cut.

En el contexto de segmentación de imágenes médicas podemos entender un atlas como un mapeo punto a punto de todas las estructuras cerebrales, compuesto por etiquetas que identifican cada región. No obstante no hay dos cerebros iguales, por lo que resulta impracticable la aplicación directa de la máscara. Por ello es imprescindible realizar los pasos que se detallan a continuación. Además están descritos en el diagrama de flujo de la Figura 5.

1. Se separa la materia gris de la materia blanca y del fluido cerebroespinal.

2. Se deforma la materia gris en el espacio original mediante el mapeo de un conjunto de puntos clave de ésta con una plantilla estándar, en adelante espacio normalizado, obteniendo un campo de deformación. Se realiza una primera segmentación en el espacio normalizado.

3. El campo de deformación se invierte y se aplica para devolver la segmentación al espacio original.

4.El resultado de la segmentación se combina con la máscara original de materia gris, mediante un valor de frontera o threshold, refinando el resultado.

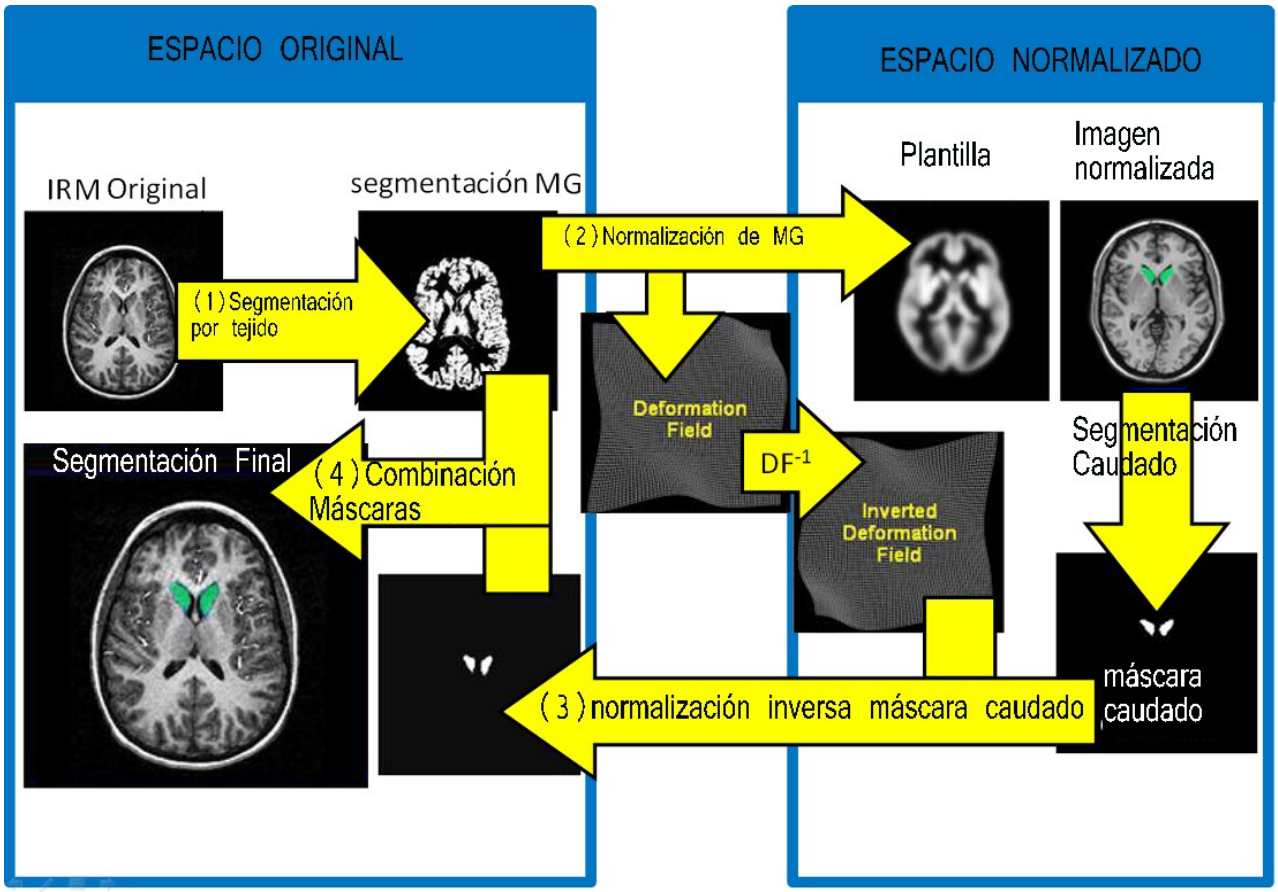


Figura 5) Etapas del algoritmo de segmentación basado en atlas.

2.2.3 Preparación para Graph Cut

En esta fase a partir de la segmentación mediante atlas se preparan dos argumentos de entrada que requiere Graph Cut.

Por un lado se realiza una erosión de los píxeles marcados como caudado, reduciendo el área del mismo y considerando los no erosionados una semilla de caudado, es decir un conjunto de píxeles que seguro pertenecen a la estructura. El proceso es análogo para determinar aquellos píxeles que seguro no son caudado, se realiza una dilatación de la estructura y se establece el área resultante fuera de la estructura como semilla de fondo.

Asimismo, para acelerar ciertos procesos posteriores se determinan las coordenadas de corte de una sección rectangular de la imagen original que seguro contiene la totalidad del caudado, Figura 6. En otras palabras, se reduce la dimensionalidad espacial del problema.

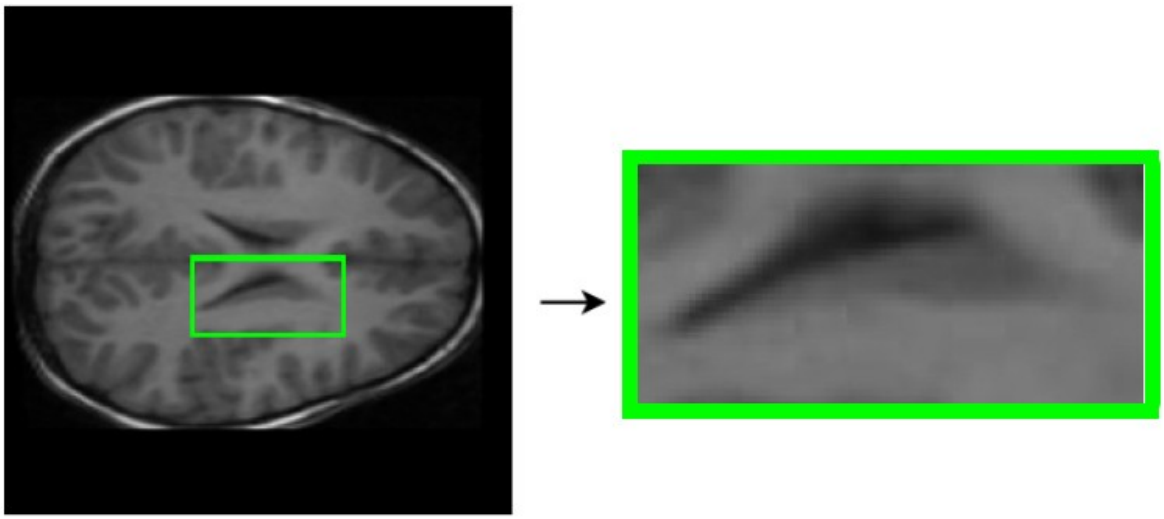


Figura 6) A la izquierda imagen original, en verde marcada el área que seguro contiene la totalidad del núcleo caudado, coordenadas obtenidas mediante segmentación basada en atlas. A la derecha perspectiva ampliada

2.2.4 Graph Cut unsupervised

Graph Cut afronta la segmentación como un problema de minimización de energía, para ello se define la función de penalización para cada píxel *p, E* (*p*):

$$E(p) = PU(p) + \delta PF(p),$$

Donde δ determina la importancia relativa de los dos términos de penalización, el potencial unario (PU) y el potencial de frontera (PF)

2.2.4.1 Potencial unario

Para calcular este potencial se utilizan modelos de caudado y fondo que se basan en información del nivel de gris de los píxeles pertenecientes a las semillas. Se inicializa el potencial para cada píxel *p* como:

$$PUp(\text{"caudado"}) = -\ln(Pu(Lp = \text{"caudado"}))$$
$$PUp(\text{"fondo"}) = -\ln(Pu(Lp = \text{"fondo"}))$$

Lp indica la etiqueta ('label') de p, que puede ser "caudado" o "fondo". Se obtiene por lo tanto para cada píxel p un potencial de fondo, y uno de caudado. La probabilidad de que sea marcado como "caudado", $Pu(Lp = \text{"caudado"})$ se calcula utilizando el histograma de niveles de gris de las semillas. La probabilidad de que sea marcado como "fondo" se calcula utilizando la probabilidad inversa, es decir $Pu(Lp = \text{"fondo"}) = 1 - Pu(Lp = \text{"caudado"})$. En la Figura 7 se puede apreciar el mapa de las probabilidades resultantes respecto al 'ground truth' (segmentación manual realizada por los médicos).

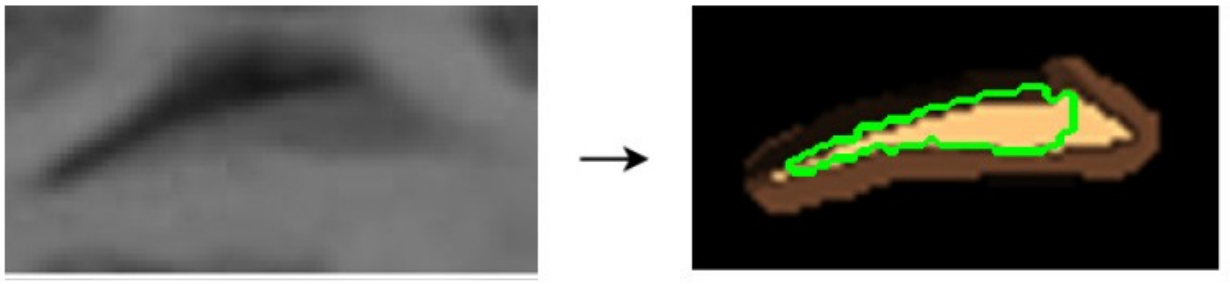


Figura 7) Imagen original a la izquierda. Potencial unario respecto a ground truth(verde)

2.2.4.2 Potencial de frontera

El potencial de frontera se calcula a partir de dos variables, el gradiente y la información geométrica entorno al píxel. En particular, por un lado se penalizan discontinuidades entre píxeles con intensidad de nivel de gris similar, y por otro discontinuidades entre píxeles con gradiente similar.
Por lo tanto, para un píxel *p*, y sus vecinos *q* en un sistema de 4-vecindad obtenemos que:

$$PF(p) = \alpha Pgra\{p,q\} + (1 - \alpha) Pgri\{p,q\}$$

Donde *Pgra* especifica la penalización por variación de gradiente, y *Pgri* la penalización por variación de la intensidad de gris. El valor del parámetro α se determina empíricamente mediante validación cruzada. En la Figura 8 se pueden apreciar visualmente estos potenciales.

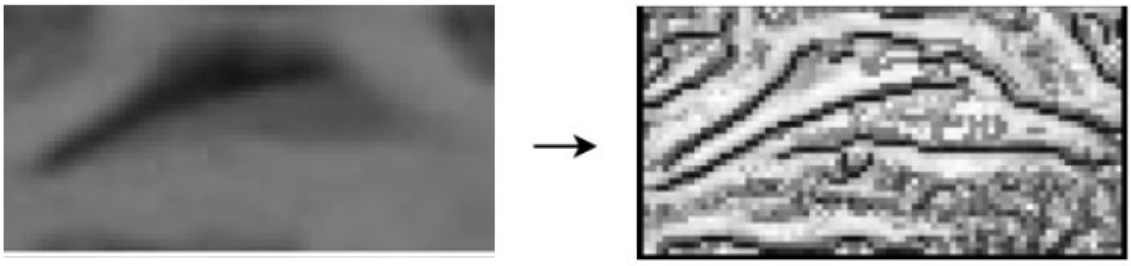


Figura 8) Imagen original a la izquierda. Representación visual de potencial de frontera a la derecha[2]

2.2.4.3 Corte Mínimo

Utilizando el valor de energía *E* (*p*) obtenido a partir de los potenciales descritos previamente se representa la imagen como un grafo y se calcula la segmentación como el camino que minimiza la energía global en el grafo. Este proceso queda reflejado en la Figura 9.

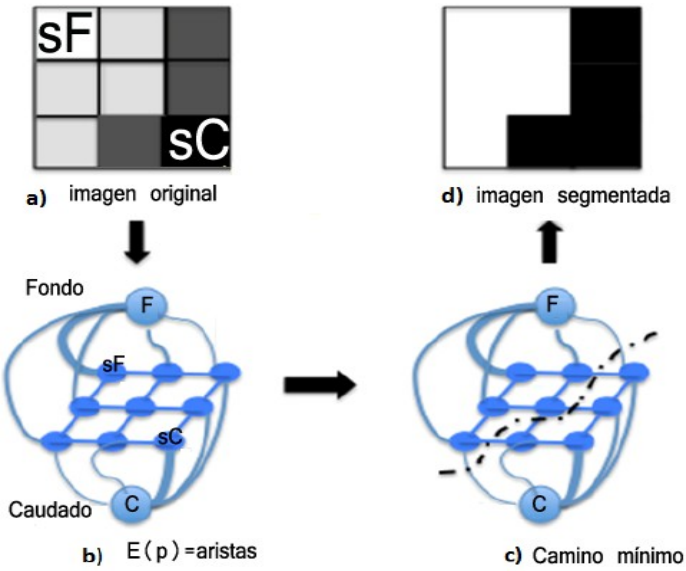


Figura 9) Etapas de Graph Cut. a) imagen original, sC : semilla caudado sF: Semilla fondo. b) Representación en forma de grafo, energía representada como grosor de arista. c) Camino que minimiza globalmente la energía. d) Resultado de segmentación.

2.2.5 Extracción de resultados

En esta fase del proceso se dibujan los volúmenes segmentados por los distintos métodos sobre las imágenes por resonancia magnética originales, para poder visualizar los resultados.

2.2.6 Evaluación

Finalmente, si se proporciona una segmentación manual como ground truth, el sistema determina mediante distintas métricas cuánto se asemeja cada segmentación a ésta. Por conveniencia las fórmulas utilizadas son descritas en el capítulo 4: Evaluación y resultados.

3 Desarrollo

Una vez expuesto el contexto de trabajo, en este apartado se detalla la propuesta de adaptación implementada en este proyecto. Como ya se ha comentado previamente el objetivo es introducir un término supervisado al cómputo.

Se requieren dos procedimientos completamente independientes para poder satisfacer los objetivos marcados en este proyecto, por un lado es imprescindible un proceso de entrenamiento que permita obtener un clasificador a partir de un conjunto de datos. Análogamente, se tiene que poder probar en aplicaciones reales, para un nuevo sujeto de test. Es decir hay que desarrollar un código que permita utilizar el clasificador.

3.1 BrainCut Supervisado

BrainCut Supervisado es el nombre que damos al algoritmo desarrollado. Las dos fases que lo componen se organizan del siguiente modo.
Por un lado se ha definido una fase de entrenamiento, este procedimiento permite a partir de un conjunto de volúmenes de entrenamiento que cuentan con ground truth (segmentación del caudado manual) obtener un clasificador de píxeles que pertenecen al caudado o al fondo de la imagen.
Por otro lado, se ha definido una fase de prueba. Esta fase permite utilizar el clasificador obtenido para realizar la segmentación de un nuevo volumen. En la Figura 10 pueden verse los distintos módulos implementados, junto con los argumentos de entrada y salida de cada fase.

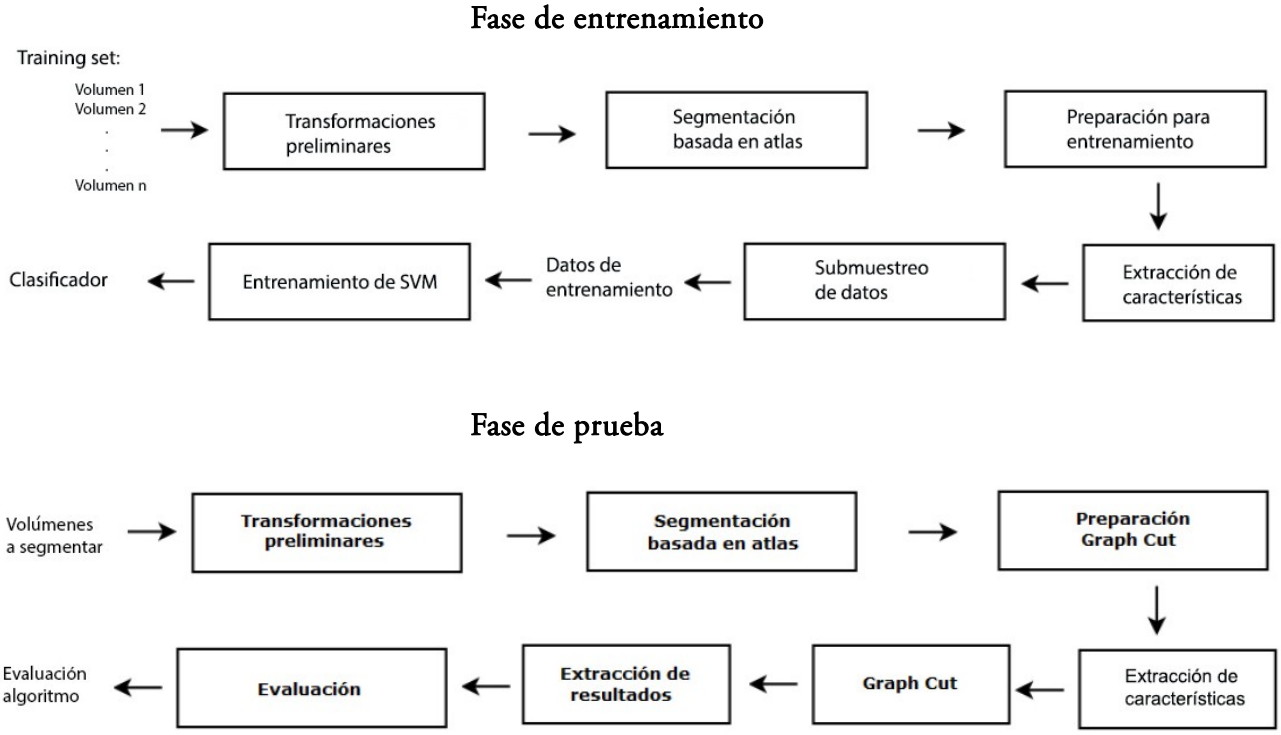


Figura 10) Diagrama de módulos de la fase de entrenamiento y de la fase de prueba

3.1.1 Fase de entrenamiento

Para esta fase se han reutilizado los módulos de 'transformaciones preliminares' y 'segmentación basada en atlas' de Braincut. El campo de 'preparación para entrenamiento' es análogo a su 'preparación Graph Cut'. Las nuevas etapas implementadas se describen en detalle a continuación. La extracción de características de cada pixel es fundamental para que el resultado obtenido del algoritmo de clasifiación sea satisfactorio. Lo que se pretende conseguir con estas características es que sean distintivas entre los píxeles que contienen caudado y los que contienen fondo, de forma que permitan al clasificador diferenciarlos, y por lo tanto obtener una clasificación correcta. Esta etapa se debe realizar del mismo modo tanto en la fase de entrenamiento como en la de prueba.

3.1.1.1 Extracción de características

Para la extracción de las características se ha utilizado una estructura de correlogramas circulares entorno al píxel tratado, que captura la información estructural de los píxeles entorno a éste. Esta implementación está detallada y justificada en [4]. En la Figura 11 puede verse una representación de esta estructura sobre el caudado.

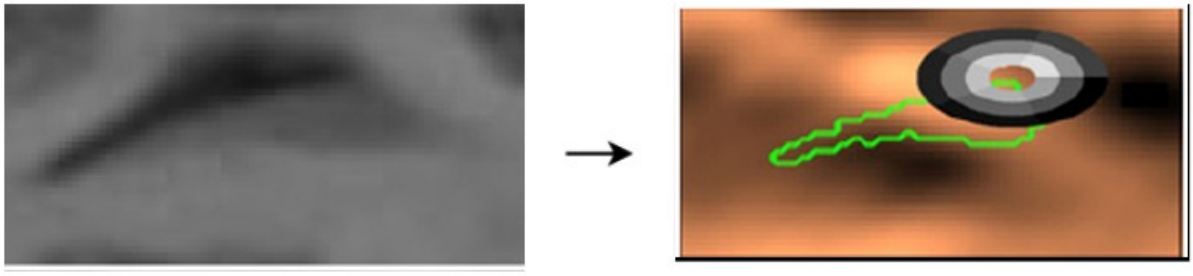


Figura 11) Izquierda: Imagen original. Derecha: Representación visual de correlograma circular sobre pixel tratado

3.1.1.1.1 Máscaras circulares

En este apartado se definirá con detalle la naturaleza de los correlogramas circulares. Existen tres variables a tener en cuenta, R , el radio de la circunferencia, S, el número de sectores triangulares, y C el número de subdivisiones circulares. En la Figura 12 (izquierda) puede verse la máscara utilizada, extraída directamente de Matlab.

Se utilizan tres sectores y dos subdivisiones circulares, obteniendo siete secciones entorno al píxel, sin tener en cuenta el fondo. Para el entrenamiento del clasificador se han utilizado dos medidas distintas de esta máscara, existiendo descriptores computados con la máscara de radio 5 y de radio 3. Para que los resultados sean correctos, el tamaño no debe variar al computar los descriptores para un nuevo sujeto de entrada (módulo de prueba).

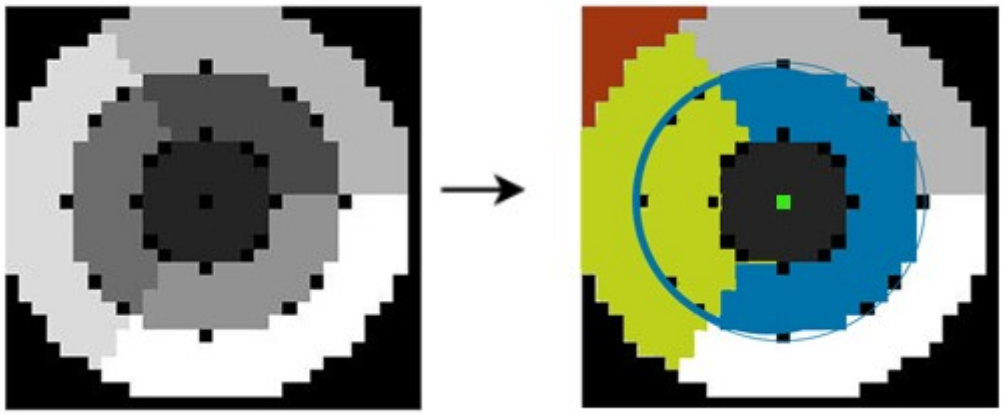


Figura 12) Máscara descriptora utilizada. Fondo (naranja), subdivisión circular (azul), sector triangular (amarillo), pixel tratado (verde). Secciones totales 7 + fondo

3.1.1.2 Cómputo de vector descriptor

Por lo tanto, para cada píxel de entrada, *p*, se computa un vector de características con las siguientes propiedades:

•Las primeras siete posiciones contienen el valor medio de gris, g(p), de cada sección.

$$(C_1 \ldots C_7)_p = \frac{\sum_{p_1}^{p_n} g(p)}{n} \in (S1 \ldots S7)$$

•Las siguientes 21, desde C_8 hasta C_{28} contienen las diferencias de nivel de gris entre cada pareja de sectores, lo que captura todas las relaciones espaciales de intensidad de gris en la vecindad de *p*.
• Dado S=7, 21 son las posibles permutaciones sin repetición de las 7 secciones.

Este proceso se realiza dos veces, para los dos radios de entrada, y se concatena el resultado posteriormente, obteniendo por tanto para cada píxel un vector descriptor de 56 características en total.

$$Vc_{(p)} = [C_1, C_2, \ldots, C_{56}]$$

3.1.1.3 Submuestreo de datos

Con el grado de desarrollo tecnológico actual resulta computacionalmente imposible utilizar todo el conjunto de entrenamiento. Por ello es menester seleccionar una muestra apropiada que ofrezca un compromiso entre tiempo de computación y calidad de predicción. El volumen de datos de entrada es tan elevado que se han incorporado distintos métodos para realizar una selección que permita completar el proceso de entrenamiento en un tiempo razonable.

A partir de la segmentación manual (ground truth) el programa separa todos los píxeles marcados como

'caudado' de aquellos marcados como 'fondo'. A continuación, como puede apreciarse en la Figura 16 se realiza una selección de *n* puntos equiespaciados. *n* puede ser definido por el usuario, modificando esta variable de forma manual en la definición de los parámetros iniciales del algoritmo. No obstante, existen ciertos valores predefinidos que facilitan un ajuste apropiado de esta variable. Hay dos estrategias alternativas de submuestreo configuradas.

- Se utilizan la totalidad de píxeles de la imagen marcados como 'caudado', y el mismo número de 'fondo' para definir la cantidad de píxeles equiespaciados que se seleccionan.

- Se utilizan la mitad de píxeles de la imagen marcados como 'caudado', y el mismo número de fondo como medida de distancia entre cada píxel que se selecciona.

Se realiza esta distinción porque el tiempo requerido para la computación es muy elevado, este tema se tratará en detalle en el apartado 3.4.3

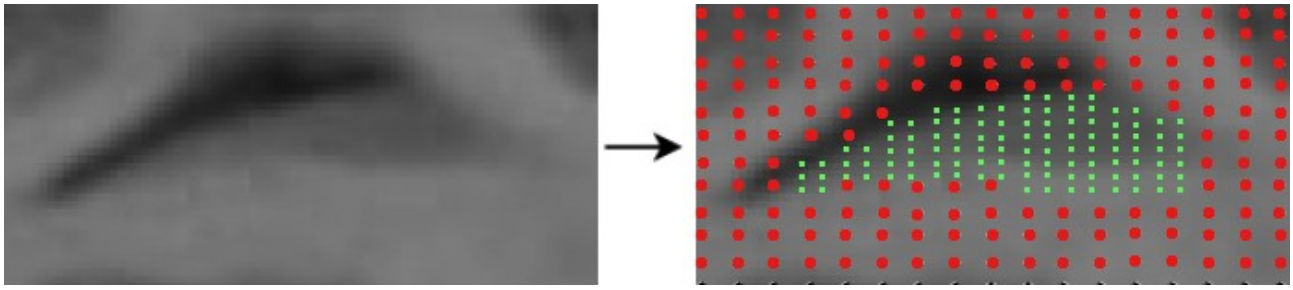


Figura 16)Representación visual aproximada del proceso de subsampling. Píxel de muestra 'fondo'(rojo), píxel de muestra 'caudado' (verde).

3.1.1.3.1 Propuesta alternativa de mejora

Con la finalidad de mejorar el resultado, esta propuesta consiste en descartar todos aquellos píxeles que a partir de la segmentación basada en atlas se determina que son semilla de fondo o de caudado. Para realizar la selección de este modo hay que introducir el valor predefinido -3, que puede ajustarse en la creación de parámetros del módulo de entrenamiento.

La motivación para implementar esta variación es que:

•Intuitivamente resulta lógico pensar que utilizar información para entrenar que posteriormente es descartada incrementa el coste computacional del entrenamiento.

•El clasificador generaliza el problema en demasía, de modo que no se ajusta exactamente a los requerimientos.

•Una reducción de la dimensionalidad del problema permite realizar un muestreo mucho más exhaustivo de la región a tratar, Figura 17.

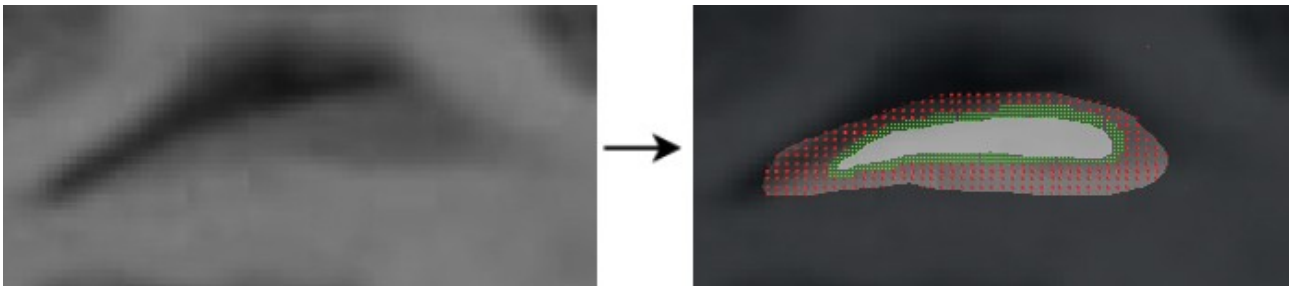


Figura 17) Representación visual aproximada de mi propuesta para el proceso de subsampling. Píxel de muestra 'fondo'(rojo), píxel de muestra 'caudado' verde, semilla de 'fondo' oscurecida, semilla de 'caudado' aclarada.

3.1.1.4 Entrenamiento de Support Vector Machine

Una vez obtenido el conjunto final de entrenamiento, la propuesta de este proyecto es utilizar una support vector machine como medio de clasificación supervisada.

Una support vector machine es un modelo de aprendizaje supervisado. Se trata de un algoritmo que mapea los datos de entrada a un espacio de alta dimensionalidad, y seguidamente busca el hiperplano que ofrece una mayor separación entre las distintas clases.
Existen correlaciones no intuitivas entre los datos de entrada, que pueden ser expuestas mediante esta técnica. Esto es debido a que pese a que el problema sea especificado en un espacio de dimensionalidad finita, por ejemplo el mundo real, es muy posible que los datos no sean separables linealmente en dicho espacio. Por lo tanto, dado un nuevo dato de entrada podemos introducirlo en el espacio y evaluar a qué lado del hiperplano se halla, siendo así clasificado. Para una visualización de estos conceptos revisar la Figura 18.
Para garantizar un tiempo razonable de ejecución los mapeos entre el espacio real y el hiperespacio de la SVM se diseñan para asegurar que el producto vectorial sea computado con facilidad, para ello se definen en términos de una función núcleo kernel *k(x,y)* que se ajusta al problema y proyecta los datos. Esta función puede ser entre otras polinomial, gaussiana, o como la que se aplica en este proyecto, lineal *u'*v.*

A) B)

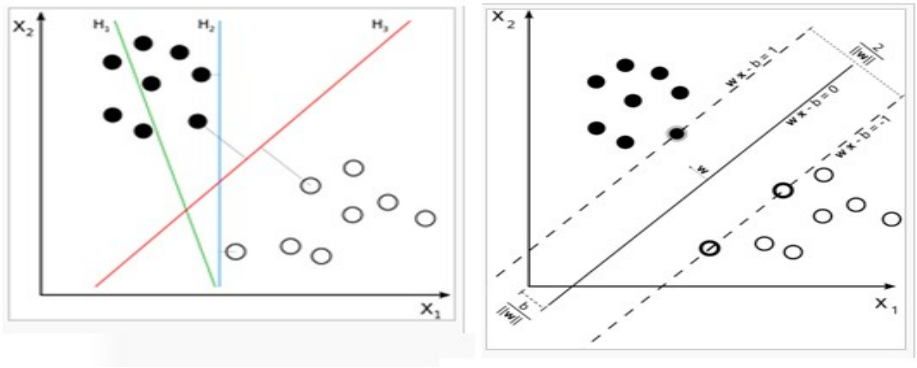


Figura 18 : A) Hiperplano objetivo (en rojo) B) los datos sobre el margen se denominan vectores soporte.

Una vez hallado este hiperplano los datos (puntos) que permanecen sobre el margen se almacenan y se utilizan para determinar a qué clase pertenece un nuevo dato expuesto al modelo. Estos puntos se conocen como vectores soporte.

3.1.2 Fase de prueba

Para la fase de prueba se ha reutilizado la estructura modular de Braincut Unsupervised. Para satisfacer las especificaciones del problema se han adaptado una serie de módulos que se explican a continuación.

3.1.2.1 Preparación para Graphcut

Con la finalidad de optimizar el resultado e investigar posibles vertientes de mejora se han introducido parámetros configurables en la preparación para GraphCut. Estas modificaciones permiten un ajuste manual de la configuración del problema, y facilitan la investigación y el desarrollo del mismo. Por defecto su valor se ha fijado en el que mejor resultado ofrece.

3.1.2.1.1 Definición de semillas

Se ha implementado un parámetro que permite distinguir si queremos que la semilla de caudado definida en la sección 2.2.3 contenga una única componente conexa, o varias, Figura 19.

Figura 19) Semilla de caudado (gris) semilla fondo (negro) zona indefinida (blanco). Izquierda configuración singleseed, derecha configuración multiseed.

En ciertos casos, resulta conveniente contar con una única semilla, descartando la componente más pequeña. En la Figura 20, pueden apreciarse las ventajas de especificar que el sistema trabaje con una sola semilla (singleseed). Se elimina el error cometido al forzar la segmentación de la componente más pequeña. Esto es debido a que Graphcut implementa un algoritmo mincut/maxflow. Para este algoritmo es imprescindible contar con dos puntos, como mínimo, que constituyen la semilla de fondo y de caudado. Esta imposición implica que aunque Graphcut sea capaz de mejorar la segmentación reduciendo el área de la segunda componente, parte de este error será arrastrado a Graphcut.

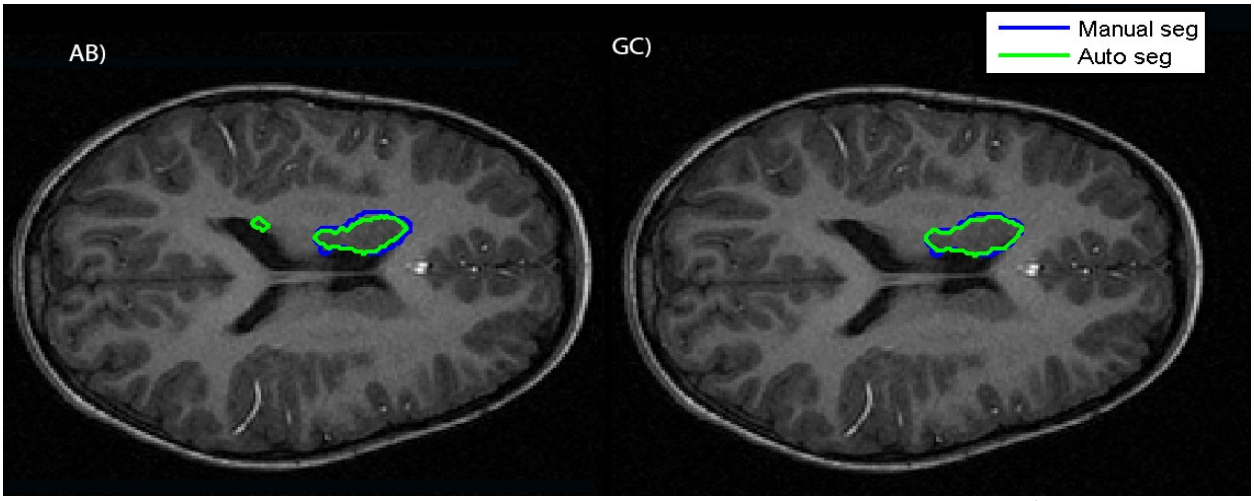


Figura 20) El uso de una única componente conexa aumenta el solapamiento con el ground truth en comparación con la segmentación basada en atlas

Por contra, también existen situaciones en las que las dos componentes conexas de semilla están muy juntas y contenidas en la segmentación manual (GT), y por lo tanto la segmentación resultante mejora el resultado en este contexto, Figura 21.

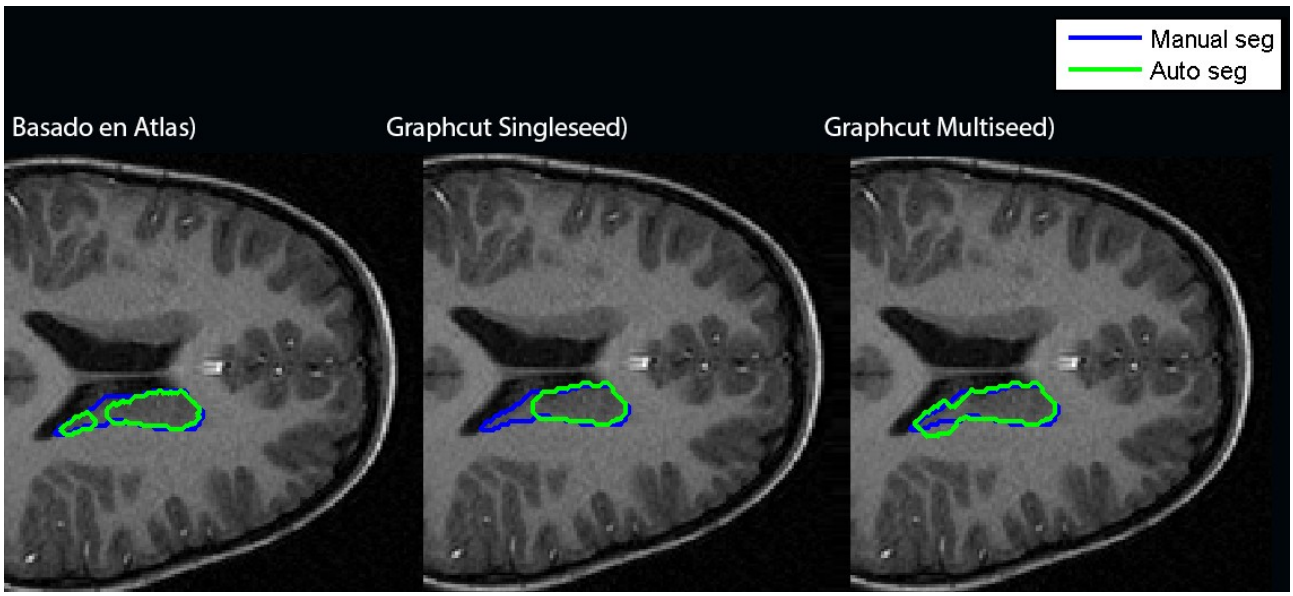


Figura 21) La configuración multiseed ofrece un resultado mejor porque delimita con más precisión el área del caudado de la segmentación manual.

Existe cierta controversia en torno a este parámetro, pues a los médicos que realizaron la segmentación manual se les solicitó una sola componente conexa. No obstante el resultado de la segmentación basada en atlas puede contener más de una, Figura 20. Este hecho implica que si nos ceñimos al ground truth es plausible que el resultado sea mejor configurando graphcut para una única componente conexa. No obstante, es posible que configurando el programa como multiseed el resultado se ajuste más a la realidad, aunque la comparación con la segmentación manual sea peor.
Empíricamente, en el apartado de Resultados se ha ajustado este parámetro al que mejor resultado cuantitativo ofrece, que se ha determinado que es configurando el sistema como singleseed, obteniendo

una ganancia respecto a multiseed de aproximadamente el 4%.

3.1.2.1.2 Definición de erosión y dilatación

Se ha implementado una variable de configuración que permite establecer si la semilla de caudado debe ser erosionada o no. Empíricamente se ha apreciado que el resultado de la segmentación basada en atlas suele ser un subconjunto de la segmentación manual, (Figura 22) es decir, la semilla está contenida en la segmentación manual.

Con la finalidad de conseguir un ajuste más apropiado se ha facilitado la manipulación de estos parámetros.
En la mayoría de casos la segmentación basada en atlas está contenida en la segmentación manual (Ground Truth), y en escasas ocasiones se produce la situación inversa, por esa razón se ha optado por no erosionar.

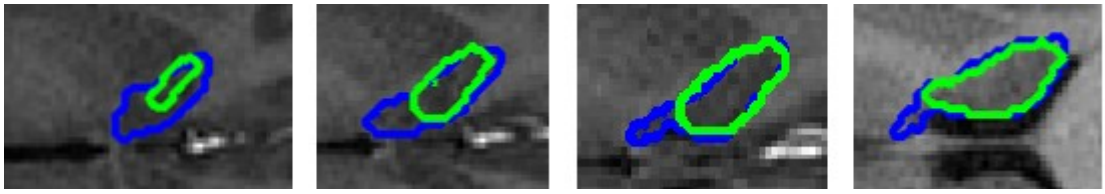


Figura 22) El resultado de la segmentación basada en atlas (verde) está contenido en la segmentación manual (azul)

3.1.2.2 GraphCut supervisado

Para introducir el término supervisado en el algoritmo se ha optado por modificar el cómputo del potencial unario (2.2.4.1) de forma tal que el nuevo potencial se define como:

PUp ("caudado"*) = PNSp (*"caudado"*) + PSp ("caudado")*
PUp ("fondo"*) = PNSp (*"fondo"*) + PSp ("fondo")*

PNSp especifica el potencial no supervisado, y *PSp* especifica el nuevo potencial obtenido a partir del resultado de clasifiación obtenido aplicando el clasificador calculado en la fase de entrenamiento.
Seguidamente, para un nuevo sujeto, se utiliza la confianza de la support vector machine en la clasificación como medida de probabilidad de que un píxel pertenezca al caudado. Por lo tanto obtenemos que:
PSp ("caudado"*) = −ln(Ps(Lp = "caudado"))*
PSp ("fondo"*) =−ln(Ps(Lp = "fondo"))*

Ps(Lp = "caudado") y Ps(Lp = "fondo") especifican la confianza del clasificador sobre el descriptor de p, *Ps(Lp = "caudado") = Clasif(p), Ps(Lp = "fondo")=-Clasif(p).*

3.2 Implementación

En esta sección se describen detalles de la implementación que se ha llevado a cabo a partir del marco teórico explicado hasta ahora, así como el diseño que se ha utilizado.

3.2.1 Plataforma y librerías de desarrollo

Se han utilizado un conjunto de librerías y programas especializados que facilitan la implementación del proyecto. En los puntos que suceden, se puede ver una explicación detallada de cada uno de éstos.

3.2.1.1 MATLAB

El desarrollo de este proyecto se ha implementado en el lenguaje de alto nivel Matlab, un lenguaje especializado en el cálculo numérico y matricial.
La unidad básica de datos que emplea MATLAB es una matriz que no requiere ser dimensionada, esto le permite resolver muchos problemas de computación técnica, en mucho menor tiempo que otros lenguajes de naturaleza similar, como C o Java.
Es importante destacar que la programación en MATLAB suele implementarse mediante scripts, es decir funciones que se invocan unas a otras transmitiéndose información.

3.2.1.2 SPM toolbox

SPM (statistical parametric mapping) es una librería especializada en el tratamiento y la manipulación de imágenes cerebrales por resonancia magnética[6]. Ha sido desarrollada en el University College of London (UCL)

3.2.1.3 OpenCV y Visual Studio

OpenCV es una librería de visión artificial desarrollada por intel, que cuenta con diversas optimizaciones para el desarrollo de software en este contexto. Está implementada en C y C++, y para facilitar la utilización de la misma una parte del algoritmo está implementada en C++, en el entorno de desarrollo (IDE) Visual Studio.

3.2.1.4 LIBSVM

LIBSVM es una librería especializada en aprendizaje automático[7]. En particular proporciona las herramientas necesarias para poder incluir una máquina de vectores soporte (SVM) en otras aplicaciones de un modo sencillo y optimizado. Soporta multitud de lenguajes, entre los que se incluye MATLAB.

Para el entrenamiento y el testeo de la support vector machine es preciso utilizar dos instrucciones:

La función 'svmpredict', para la fase de prueba, cuyos argumentos de entrada son los siguientes:

- La matriz de características para cada pixel

- El clasificador previamente computado
- El tipo de kernel que quiere utilizarse (en nuestro caso lineal)

'svmpredict' retorna los siguientes valores:

- La matriz con el resultado de la clasificación, es decir las etiquetas, fondo o caudado, en particular 1 o 0.
- La precisión de la predicción
- La confianza del clasificador en la predicción

La función svmtrain, requiere de los siguientes parámetros para el entrenamiento:

- Vector de etiquetas: 'caudado' y 'fondo' se representan como 1 y 0 respectivamente, están dispuestos del mismo modo que la matriz de entrenamiento.
- Matriz de entrenamiento: Contiene el conjunto de entrenamiento, cada fila corresponde a una muestra (píxel), cada columna a una característica
- Opciones de libsvm: Incluye multitud de argumentos posibles, para este proyecto conciernen la selección del tipo de núcleo, lineal en este caso ('-t 0'), y el coste *k**. *k*, se determinó empíricamente en [2] mediante un proceso de validación cruzada, estableciendo su optimalidad en 0,22..

Esta invocación retorna un fichero que contiene el clasificador, que puede ser utilizado para la fase de prueba.

Este parámetro k* determina el ratio entre vectores soporte y datos, su valor suele fijarse entre 0.1 y 0.8. Valores cercanos a 0.1 indican que hay pocos vectores soporte en relación al número de datos introducido, esto implica que el margen es pequeño, y la mayoría de veces nuevos datos pueden ser clasificados, correctamente o no. Esta modificación se conoce como Soft Margin y fue definida por Corinna Cortes y Vladimir N. Vapnick

3.2.2 Diseño

Para una mayor comprensión del funcionamiento de los distintos módulos, y la interacción entre los mismos se han elaborado una serie de diagramas explicativos mediante BOUML, un software diseñado para la elaboración de diagramas en notación UML.
Para no inducir a la confusión, es importante destacar que el diseño de estos diagramas de secuencia e interacción no coincide exactamente con esta notación, por ejemplo dado que este proyecto en MATLAB se ha desarrollado mediante scripts, y no objetos, lo que en UML clásico sería la instancia y el

nombre de la clase, en este caso corresponden al fichero en que se halla el script invocado, como nombre del método invocado se ha considerado apropiado indicar "run(parámetros)", queriendo indicar que el script es iniciado con "parámetros" como argumento.
Pese a sutiles diferencias como la citada anteriormente la utilización de este software y sus utilidades facilita en gran medida la comprensión del funcionamiento interno del algoritmo, por lo que se ha considerado oportuno incluirlo.

3.2.2.1 Fase de prueba

Para la fase de prueba, se han adaptado los módulos de Braincut, y se han realizado los cambios oportunos para optimizar los resultados. En el diagrama 1, puede verse la interacción entre los mismos y sus argumentos tanto de entrada como de salida.

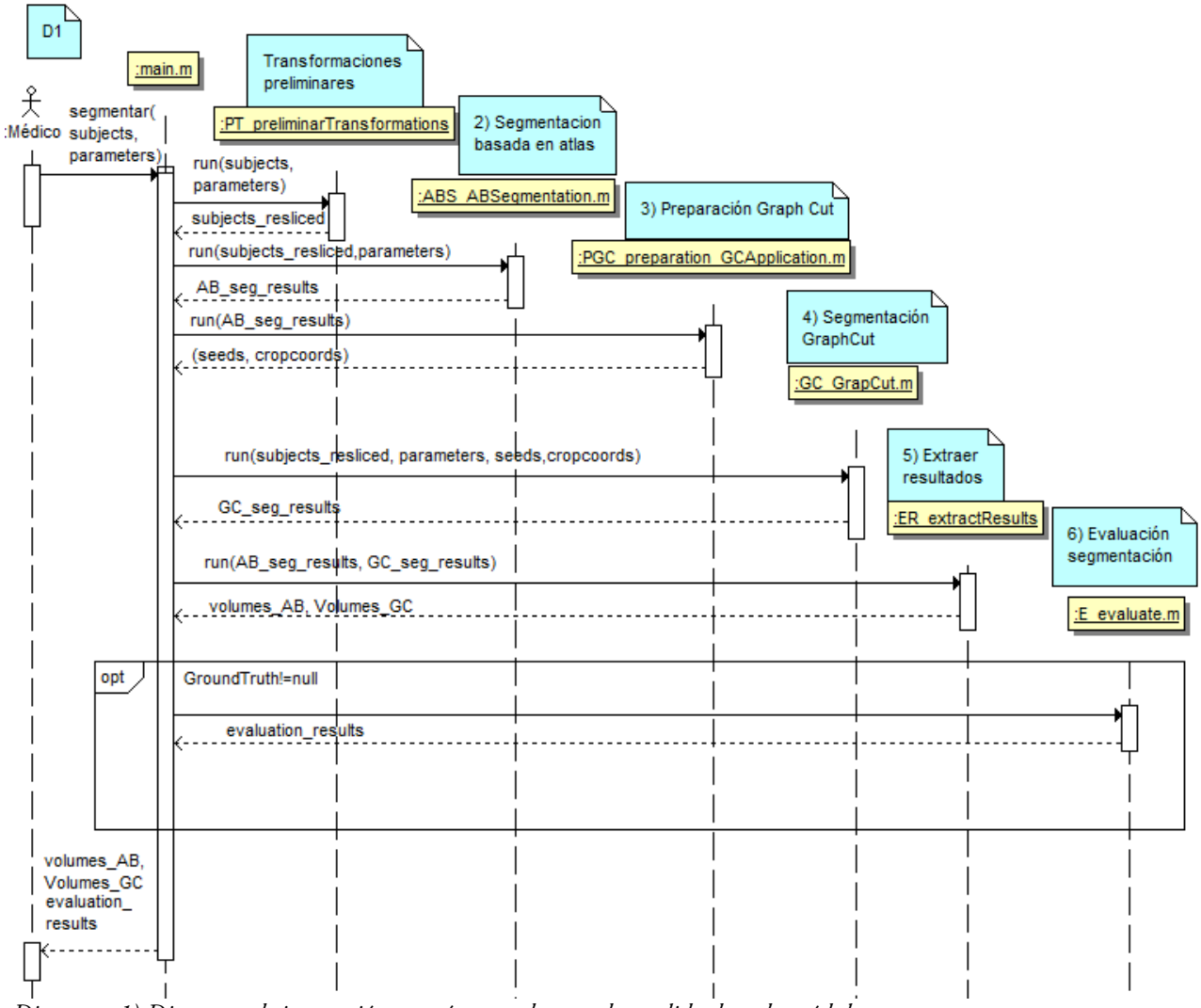


Diagrama 1) Diagrama de interacción y parámetros de entrada y salida de cada módulo.

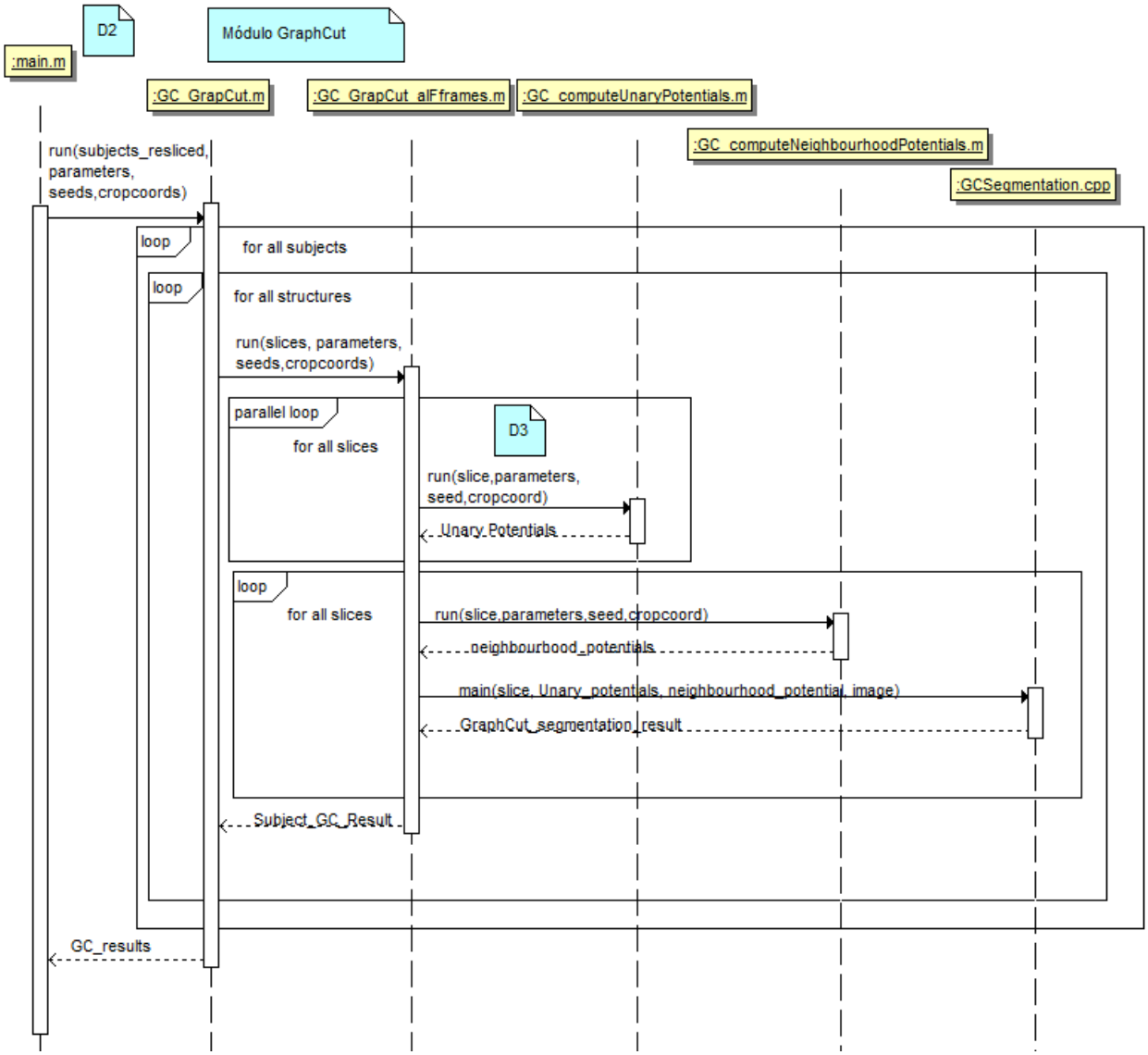


Diagrama 2) Diagrama de secuencia del funcionamiento del módulo Graphcut

En el Diagrama 2, se detalla la implementación de la etapa 4 del diagrama anterior, la segmentación Graphcut. Ha sido imprescindible modificar la lógica interna del módulo para permitir la paralelización. Las modificaciones quedan reflejadas en el código en el cómputo de los potenciales unarios, que se realizan en un bucle paralelizado independiente previo. Ha sido imprescindible introducir este cambio pues ciertas variables comunes impedían la paralelización de todo el proceso.
Además, es en el script GC_computeUnaryPotentials.m donde se ha introducido el potencial supervisado, Diagrama 3. Los descriptores deben ser computados del mismo modo que en la fase de entrenamiento, para conservar la integridad de las características.

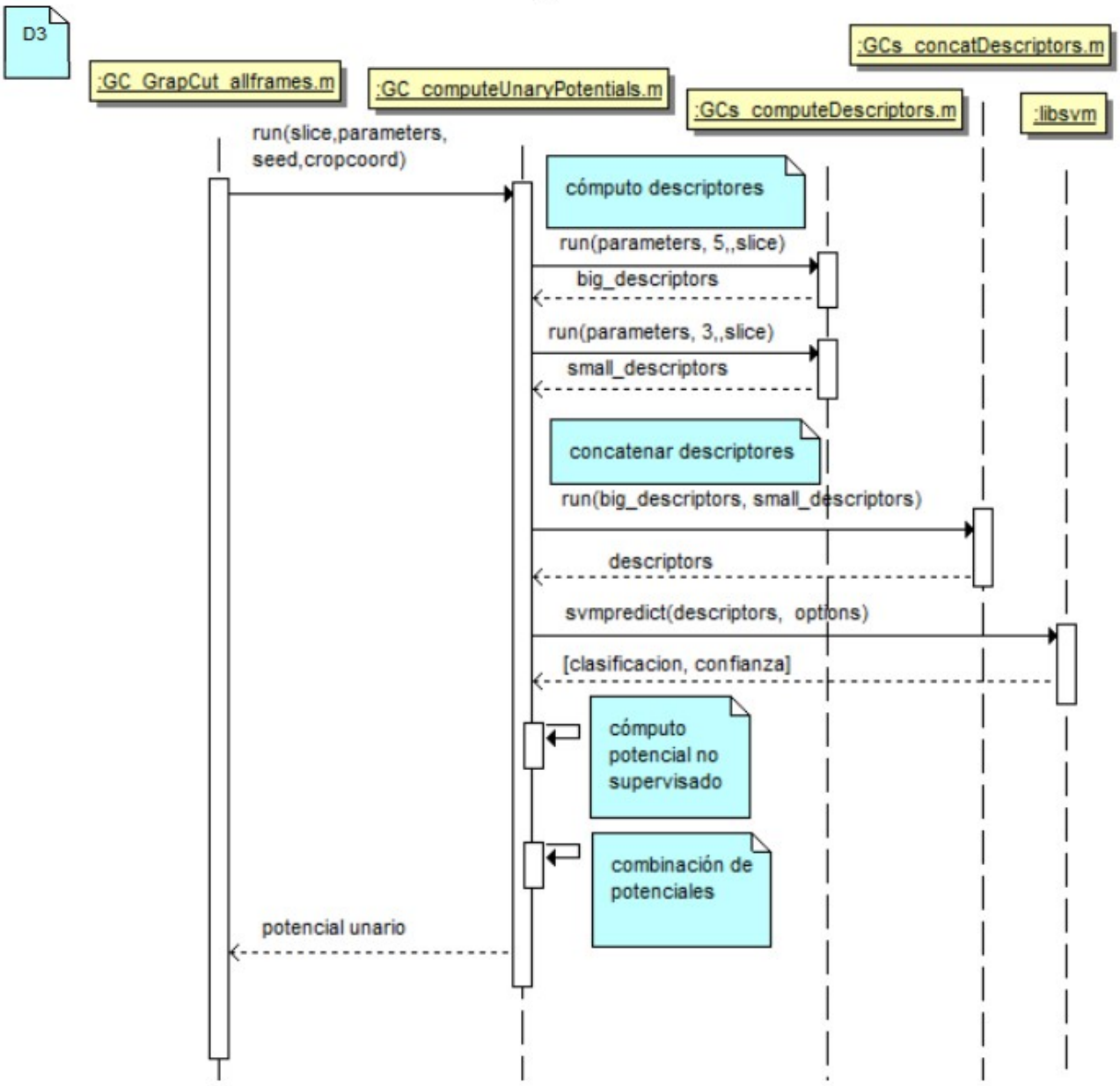


Diagrama 3 Diagrama ilustrativo del funcionamiento del cómputo de los potenciales unarios

3.2.2.2 Fase de entrenamiento

En la fase de entrenamiento, a partir de las imágenes originales el programa devuelve un clasificador, 'classifier.mat'. Este fichero se puede utilizar a posteriori para segmentar incluyéndolo en la carpeta que contiene el main, 'Code'. En el anexo 1 se detalla la organización de las carpetas y el manual de uso.

El diagrama 4 de invocaciones especifica los pasos previos requeridos para obtener los argumentos necesarios para el entrenamiento.

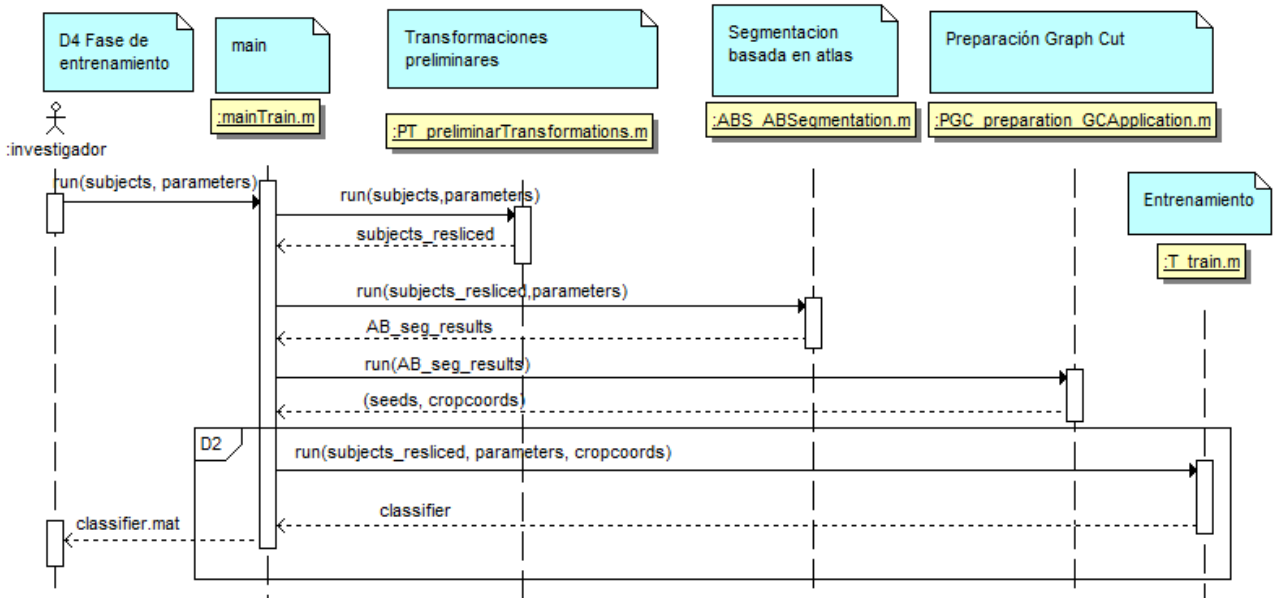


Diagrama 4 diagrama de invocación de módulos, y parámetros de entrada y salida

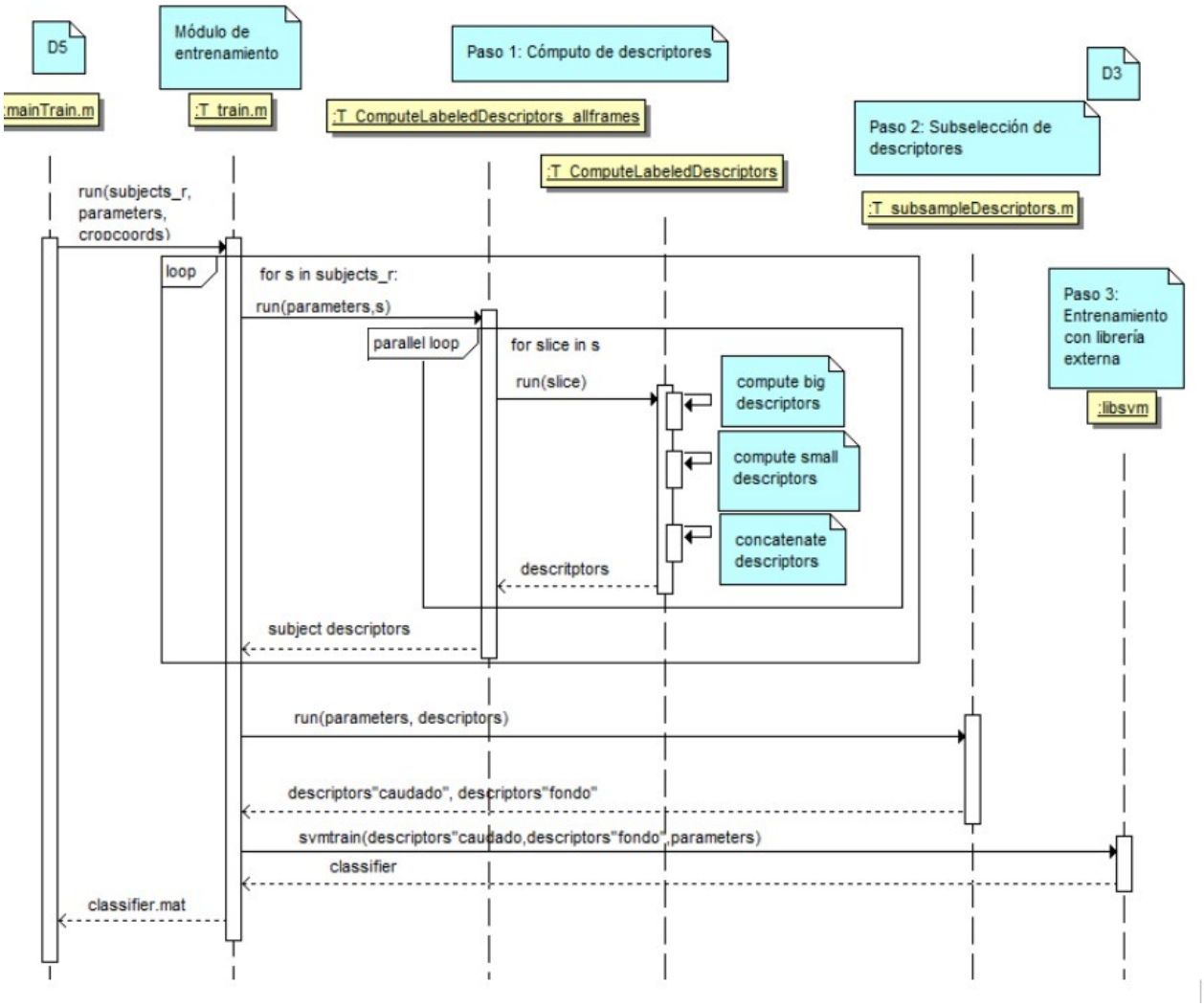


Diagrama 5) Diagrama de secuencia ilustrativo del funcionamiento del módulo de entrenamiento.

4 Experimentos y resultados

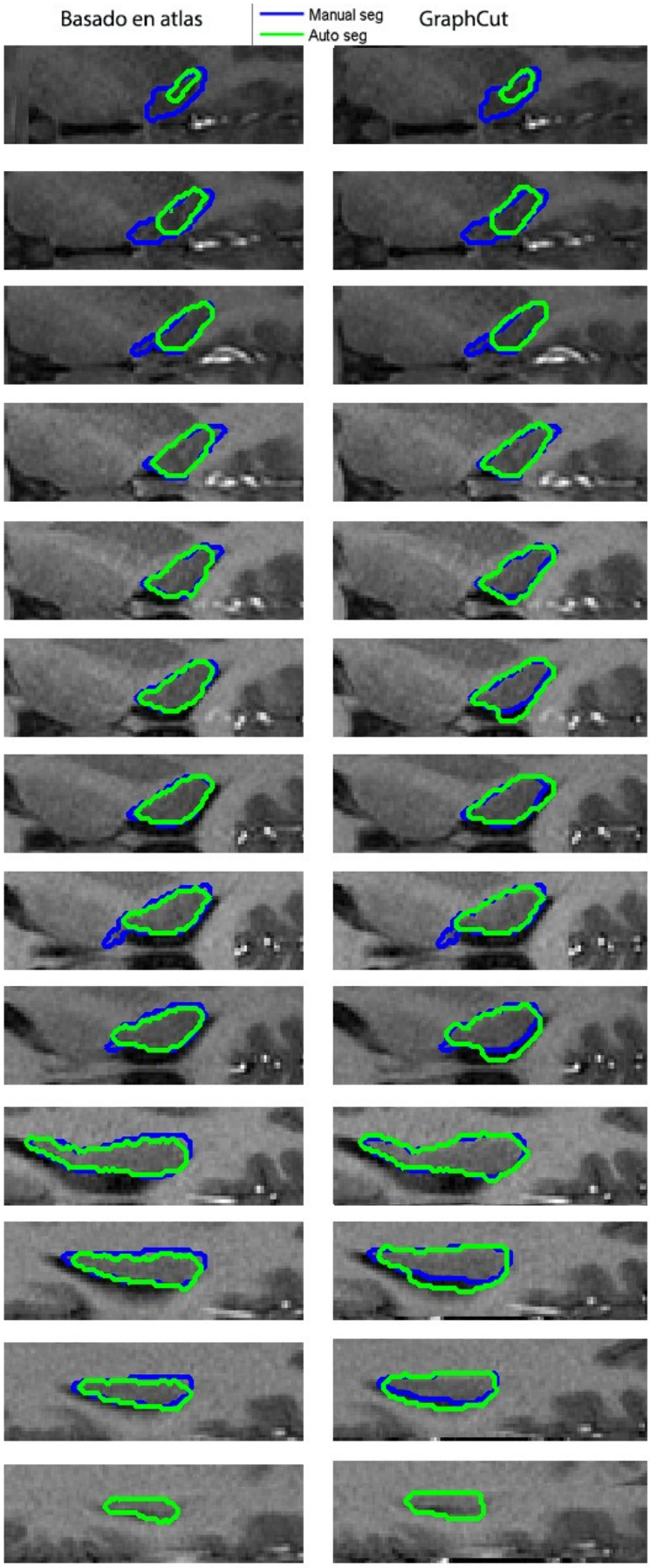
En este apartado se exponen los resultados obtenidos mediante los distintos métodos, y se hace un análisis crítico y en profundidad de los mismos.

4.1 Datos

Para las pruebas realizadas se ha utilizado un conjunto de 79 sujetos, la mitad controles y la otra mitad diagnosticados como TDAH por la Unidad de Psiquiatría Infantil en el Hospital Vall d'Hebron, de Barcelona.
Este proceso fue coordinado por la Unidad de Investigación en Neurociencia Cognitiva (URNC) de la Fundación IMIM. La edad media de los sujetos es de 11 años, y se caracterizan por ser ambidiestros, a este hecho se le asocia un desarrollo similar de los hemisferios izquierdo y derecho del cerebro. Las imágenes por resonancia magnética del cerebro fueron extraídas con una máquina cuyo campo magnético es de sistema 1,5T. La resolución de las exploraciones es de 256 x 256 x 60 píxeles con cortes de un grosor de 2 mm.

4.2 Resultados cualitativos

En las siguientes figuras se pueden ver los resultados de segmentación obtenidos a partir de cada método, se han seleccionado dos ejemplos de segmentación completa. El primero caso ilustra una segmentación en la que atlas based ligeramente superior.. En el segundo caso se muestra una segmentación que el método GraphCut realiza mejor que la segmentación basada en atlas



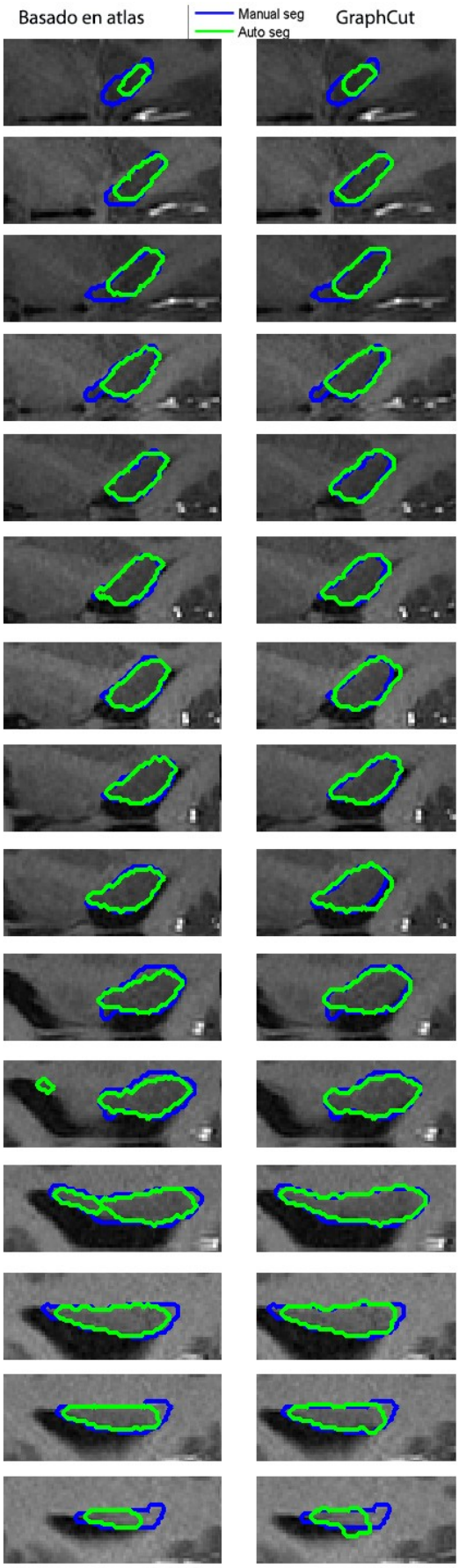
En esta figura se puede ver una comparación completa de ambos métodos de segmentación para el caudado derecho de un sujeto. Se aprecia como ambos métodos tienen problemas con los contornos estrechos del caudado de las primeras slices.

Se puede apreciar que la segmentación mediante GraphCut realiza una segmentación volumétrica más grande, propiciando una mejor segmentación en diversas slices. No obstante esta no siempre se ajusta correctamente a los contornos del Ground Truth (en verde). En ocasiones Graphcut relaja demasiado las restricciones y la segmentación excede el área del ground truth.

Un entrenamiento más exhaustivo y un reajuste de calibración podría solventar este problema. Este tema se trata con mayor profundidad en las conclusiones.

Se aprecia que el resultado de ambos algoritmos es muy similar, no obstante en este caso la segmentación basada en atlas es ligeramente mejor, numéricamente del orden del 5%.

En la última slice se comete un error significativo pues no existe ground truth, no obstante la segmentación basada en atlas sí define una semilla en la misma. La no erosión de la semilla impide en este caso particular que la segmentación GraphCut reduzca el error cometido, que es arrastrado e incluso incrementado.



A diferencia del caso anterior, en esta ocasión GraphCut segmenta ligeramente mejor que el método basado en atlas, numéricamente del orden de un 4%.

En esta Figura se aprecia como el mayor volumen de Graphcut incrementa el solapamiento.

También resulta un buen ejemplo de porqué la configuración con una única componente conexa como semilla evita arrastrar el error de la segmentación basada en atlas.

4.3 Resultados cuantitativos

4.3.1 Medidas de evaluación

Para la evaluación cuantitativa de los resultados se han utilizado una serie de medidas. Si bien existe cierta correlación entre las mismas, ofrecen una buena impresión a grandes rasgos de los resultados obtenidos.

- **VO: Volumetric overlap (en%):** Número de píxeles en la intersección de la segmentación automática y de la segmentación manual, dividido por el número de píxeles en la unión de ambas.

- **SI: Volumetric similarity index (en %):** Intersección de las segmentaciones entre el volumen medio de ambas.

- **VD: Relative absolute volume difference (en %):** Se comparan los volúmenes de la segmentación manual y automática, el porcentaje indica la diferencia entre los mismos.

- **AD: Average symmetric absolute surface distance (en mm):** Se determinan los píxeles fronterizos en ambas segmentaciones. Para cada píxel en estos conjuntos, se determina el más cercano en el otro conjunto, mediante la distancia euclídea, se realiza para todos los píxeles de frontera y se promedia, obteniendo 0 en una segmentación perfecta.

- **RMSD: Symmetric RMS (Root Mean Square) surface distance (en mm):** Esta medida es similar a la medida anterior, pero guarda las distancias al cuadrado entre los dos grupos de píxeles fronterizos. Se extrae la raíz del promedio de los valores al cuadrado y se obtiene el valor.

- **MC: Maximum symmetric absolute surface distance (en mm):** Se toma el máximo valor de las distancias calculadas en las medidas anteriores. Este valor es 0 para una segmentación perfecta.

4.3.2 Resultados obtenidos

La siguiente tabla resume los resultados, en media, de cada uno de los métodos, junto con la desviación estándar .

Caudado derecho

Medida de evaluación	Basado en atlas	GraphCut	Propuesta alternativa
SI	83'981+-2,972%	83'06+-3,053%	83,071+-3,039%
VO	72'495+-4,317%	71'14+-4,331%	71,155+-4,306%
VD	12'616+-6,435%	9'682+-9,866%	9,611+-9,915%
AD	2,374+-0,636µm	2,458+-0,555µm	2,458+-0,553µm
RMSD	6,79+-2,103µm	6,792+-1,921µm	6,82+-1,193µm
MC	7'923+-8,453mm	7'386+-8,2mm	7,452+-8,311mm
GT Volume	4797'97mm3	4797'97mm3	4797,97mm3
Auto Volume	4218'53mm3	5114'85mm3	5110,60mm3

Caudado izquierdo

Medida de evaluación	Basado en atlas	GraphCut	Propuesta alternativa
SI	82,993 +-3,37 %	82'871+-3,488%	82'863+-3,447%
VO	71,066+-4,784%	70'892+-4,788%	70'88+-4,79%
VD	15,040 +-7,110 %	10'28+-8,359%	10'237+-8,271%
AD	2,485+-0,749µm	2,482+-0,718µm	2,486+-0,718µm
RMSD	7,2+-3,3µm	7,1+-3,199µm	7,123+-3,2µm
MC	9'18+-12,3mm	8'68+-12,2 mm	8'69+-12,2 mm
GT Volume	4717'17mm3	4717'17 mm3	4717'17 mm3
Auto Volume	3997'40mm3	5040'41 mm3	5042'09mm3

Se ha estimado la mejora obtenida con respecto a Graphcut unsupervised en un 2%

Como puede apreciarse los resultados son muy similares, aunque no llega a mejorarse la segmentación basada en atlas. Aun así se ha demostrado que el margen de mejora es muy amplio. Los resultados obtenidos mediante la propuesta alternativa de submuestreo son un buen indicativo de que un entrenamiento más exhaustivo mejorará el resultado.

Graphcut establece un volumen resultante que suele ser mayor que el ground truth. Por contra la segmentación basada en atlas ofrece como resultado un volumen más pequeño que la segmentación manual.

4.3.3 Tiempo de computación

En esta sección se realiza un análisis de los tiempos de ejecución que requiere cada fase del algoritmo. Este proceso se ha ejecutado en un ordenador con un procesador AMD Phenom II X6 1100T, de 6 núcleos a 3.3 Ghz cada uno. Es importante destacar también que la implementación de 'libsvm' no soporta la paralelización de ninguna de las fases expuestas previamente, hecho que ralentiza en gran medida la fase de entrenamiento.

4.3.3.1 Fase de entrenamiento

El proceso de entrenamiento es una tarea que requiere de mucho tiempo de computación. El cómputo de los descriptores y la selección del conjunto de datos suponen una inversión temporal mínima, que puede obviarse si se compara con el tiempo que requiere 'LIBSVM' para entrenar el nuevo clasificador. 'LIBSVM' carece de herramientas de control del progreso, no obstante sí que permite la distinción de las dos fases fundamentales, la proyección de los datos en el hiperespacio, y la búsqueda del hiperplano objetivo.

4.3.3.1.1 Graphcut

- El tiempo requerido para la proyección es directamente proporcional al número de datos de entrenamiento, para un conjunto de aproximadamente cien mil ejemplos de 'fondo', y cien mil de 'caudado'. Los valores utilizados en este proyecto, el tiempo total de esta fase ha sido de aproximadamente unas 350 horas.

- 'LIBSVM' ha sido implementado de tal forma que siempre convergirá a una solución, no obstante el tiempo requerido varía en función del problema. Empíricamente el tiempo que ha requerido para convergir dadas las proyecciones anteriores ha sido de aproximadamente 250 horas.

Por lo tanto, empleando el método convencional de submuestreo, el tiempo total de entrenamiento asciende a unos 25 días.

4.3.3.1.2 Propuesta alternativa

Por diversos problemas técnicos ajenos a Graphcut no ha sido posible entrenar un clasificador con el mismo número de datos de entrada. Inestabilidades en el sistema operativo forzaban el reinicio del proceso de entrenamiento.
Aun así, se ha realizado la prueba con una tercera parte de los elementos con los que se entrenó siguiendo el método convencional y se han obtenido resultados ciertamente prometedores.

Mediante este procedimiento se ha conseguido una sutil mejora en la segmentación con respecto al método convencional, y una mejora dramática en el tiempo de computación del proceso de entrenamiento, que se ha visto reducido a un 20%, de unos 25 días a 5.

4.3.3.2 Fase de prueba

Se ha tratado de optimizar el tiempo de computación del módulo de prueba en la medida de lo posible. A tal efecto, se ha modificado el código original del módulo Grapchut para permitir la paralelización del cómputo de los descriptores. En un sistema monohilo, esta tarea conlleva un total de unos 103 minutos aproximadamente, la inclusión de la paralelización permite dividir este tiempo entre el número de núcleos disponibles para el programa. En este caso, utilizando un procesador AMD Phenom II X6 1100t a 3.3Ghz dedicado, es decir seis núcleos, el tiempo consumido por esta fase es de 16 minutos aproximadamente.
Es importante tener presente que Matlab únicamente permite la paralelización de más de 4 hilos a partir de la versión r2009a, momento en el que el máximo pasa a ser 8. Para la implementación de este proyecto se ha utilizado la versión R2008 b y la R2012 a.

La segmentación basada en atlas, implementada sin paralelización requiere de aproximadamente 12 minutos.

Si a estos dos valores añadimos el tiempo requerido para realizar las transformaciones necesarias, es decir la anonimización, el cómputo de las semillas, el cálculo del área mínima contenedora del caudado, la aplicación del algoritmo graphcut, y la extracción y evaluación de resultados, para una estructura de un sujeto el tiempo total requerido, es de unos 30 minutos.

4.4 Evaluación del proyecto

4.4.1 Distribución temporal y diagrama de Gantt

En el diagrama 6 (diagrama de Gantt) podemos apreciar cómo se han distribuido cada una de las tareas a lo largo del desarrollo de este proyecto.

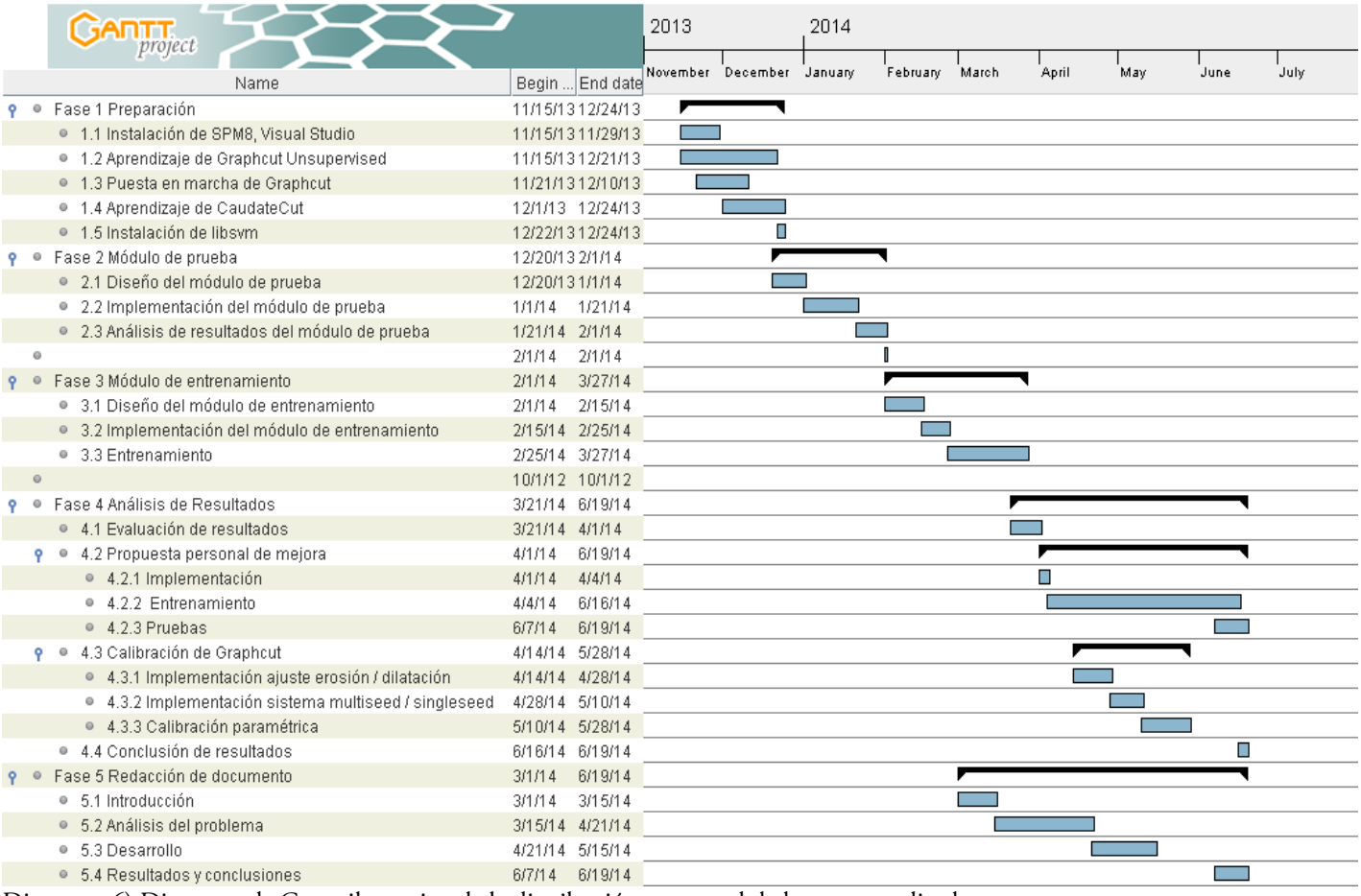
- Fase 1: Dada la dimensionalidad del problema y la cantidad de código se requirió de muchas horas para el entendimiento del problema y del código, y la familiarización con el entorno.

- Fase 2: Al no mejorar la segmentación basada en atlas se convino que el siguiente paso era reentrenar el clasificador.

- Fase 3: En esta fase se implementó un nuevo módulo que permitía reentrenar el clasificador a partir de los datos de entrada, no obstante al validar los resultados se vio que el resultado seguía sin mejorar.

- Fase 4: En este punto se consideró que el paso apropiado era intentar mejorar la segmentación basada en atlas en el contexto de Graphcut.

- Fase 5: Para la redacción de este documento se han aprovechado los largos tiempos de entrenamiento y las pruebas.



Name	Begin ...	End date
Fase 1 Preparación	11/15/13	12/24/13
1.1 Instalación de SPM8, Visual Studio	11/15/13	11/29/13
1.2 Aprendizaje de Graphcut Unsupervised	11/15/13	12/21/13
1.3 Puesta en marcha de Graphcut	11/21/13	12/10/13
1.4 Aprendizaje de CaudateCut	12/1/13	12/24/13
1.5 Instalación de libsvm	12/22/13	12/24/13
Fase 2 Módulo de prueba	12/20/13	2/1/14
2.1 Diseño del módulo de prueba	12/20/13	1/1/14
2.2 Implementación del módulo de prueba	1/1/14	1/21/14
2.3 Análisis de resultados del módulo de prueba	1/21/14	2/1/14
	2/1/14	2/1/14
Fase 3 Módulo de entrenamiento	2/1/14	3/27/14
3.1 Diseño del módulo de entrenamiento	2/1/14	2/15/14
3.2 Implementación del módulo de entrenamiento	2/15/14	2/25/14
3.3 Entrenamiento	2/25/14	3/27/14
	10/1/12	10/1/12
Fase 4 Análisis de Resultados	3/21/14	6/19/14
4.1 Evaluación de resultados	3/21/14	4/1/14
4.2 Propuesta personal de mejora	4/1/14	6/19/14
4.2.1 Implementación	4/1/14	4/4/14
4.2.2 Entrenamiento	4/4/14	6/16/14
4.2.3 Pruebas	6/7/14	6/19/14
4.3 Calibración de Graphcut	4/14/14	5/28/14
4.3.1 Implementación ajuste erosión / dilatación	4/14/14	4/28/14
4.3.2 Implementación sistema multiseed / singleseed	4/28/14	5/10/14
4.3.3 Calibración paramétrica	5/10/14	5/28/14
4.4 Conclusión de resultados	6/16/14	6/19/14
Fase 5 Redacción de documento	3/1/14	6/19/14
5.1 Introducción	3/1/14	3/15/14
5.2 Análisis del problema	3/15/14	4/21/14
5.3 Desarrollo	4/21/14	5/15/14
5.4 Resultados y conclusiones	6/7/14	6/19/14

Diagrama 6) Diagrama de Gantt ilustrativo de la distribución temporal de las tareas realizadas en este proyecto

4.4.2 Evaluación económica

Los costes de desarrollo del mismo han supuesto una inversión de un total de 450 horas, si fijamos el salario a 10€/h, el total asciende a 4.500 € como coste de producción.

Para poder estimar el coste de implementación de este proyecto, en este apartado se plantea un escenario práctico de uso del software implementado. En concepto de materiales, y licencias se requiere de lo siguiente.

Ordenador 650€:

- 16 GB de RAM: 100€
- Procesador multicore de alto rendimiento, por ejemplo el utilizado en este proyecto: 200 €
- El resto de componentes es decir caja, placa base, tarjeta gráfica, etc. Más modesto: 350€

Licencias 210€:

- Las licencias para MATLAB, y Visual Studio 2010 ascienden a 210€
- LIBSVM y spm8 no requieren de licencia.

Recursos humanos 600€:

Para que la utilización del software se realice de forma apropiada es necesario que el técnico que va a utilizar el software se familiarice con el entorno y dedique una serie de horas a un proceso de formación. Se estima que la dedicación necesaria del técnico del sistema es de unas 40 horas, a 10€/h el total asciende a 400€

Por lo tanto, el coste total de la puesta en marcha de este proyecto en un contexto práctico real suma un total de 1.260€. Este valor no incluye el salario de la persona que opera el software.

5 Conclusión

5.1 Objetivos alcanzados

Inicialmente se fijaron una serie de objetivos a completar.

- El objetivo principal era incluir un nuevo método supervisado a la librería BrainSegmentation, objetivo que ha sido completado con éxito.

- Se han completado con éxito el desarrollo de un algoritmo supervisado de segmentación automática, y se han implementado las dos fases, la de entrenamiento y la de prueba de forma correcta.

- Se ha mantenido el diseño modular que facilita el desarrollo continuo del ciclo de vida del software.

- No se ha conseguido mejorar la segmentación basada en atlas. Mediante la implementación de un método mejor de selección de ejemplos de entrenamiento se ha probado que aún existe un amplio margen de mejora. Debido a un tiempo de dedicación finito no se han podido completar los procedimientos necesarios para incluirlos en este documento en el plazo disponible.

5.2 Discusión

Inicialmente el objetivo era implementar exclusivamente el módulo de prueba, y utilizar un clasificador precomputado de CaudateCut [2]. A partir de ese momento el objetivo era seguir desarrollando convenientemente. No obstante, al no mejorar la segmentación basada en atlas se convino que el siguiente paso era reentrenar el clasificador.

Como tno se ha conseguido mejorar la segmentación basada en atlas, en este preciso momento resulta contraproducente utilizar Graphcut para segmentar en un contexto práctico de uso diario. Es más lógico realizar la segmentación basada en atlas y considerar exclusivamente su resultado. Aun así, los resultados obtenidos invitan a seguir desarrollando el algoritmo. La posibilidad de mejorar la segmentación es una realidad. Este hecho implica que Graphcut supervised tiene mucho potencial para poder ser aplicado en un escenario real en poco tiempo.

Debido a la diferencia en los resultados obtenidos con respecto a CaudateCut[2], se requirió de una alta inversión temporal para investigar posibles mejoras en el nuevo contexto, es decir GraphCut supervised. Resultados prometedores motivan más que nunca que se prosiga el desarrollo de este algoritmo.

5.3 Trabajo futuro

Los resultados obtenidos abren una puerta a continuar con el desarrollo del proyecto, para el que pueden plantearse multitud de nuevos objetivos.

El primer objetivo que debe fijarse es reentrenar utilizando la propuesta alternativa de muestreo, de tal modo que se utilice el conjunto de entrenamiento computable más grande posible. Esta modificación probablemente haga que se mejoren los resultados de la segmentación basada en atlas

Implementar un proceso de validación cruzada (cross-validation) que ajuste con precisión los nuevos parámetros configurables probablemente resulte en una mejor segmentación.

La inclusión del nuevo método de submuestreo del training set invita a plantearse la posibilidad de revisar el proyecto original de investigación CaudateCut[2], para incluir esta mejora y presentar los nuevos resultados obtenidos.

Paralelizar la segmentación basada en atlas y Graphcut supervised por completo optimizaría en gran medida el tiempo de computación, y por lo tanto es un objetivo a tener presente.

La inclusión de una interfaz gráfica en la librería repercutiría muy positivamente en la usabilidad del sistema, por lo que también es un objetivo a plantearse.

6 Bibliografía

1.
Christopher M. Bishop, Pattern Recognition and Machine Learning. Springer.2006.

2.
Laura Igual, Joan Carles Soliva, Antonio Hernandez-Vela, Sergio Escalera, Xavier Jimenez, Oscar Vilarroyaand Petia Radeva. A Fully-Automatic Caudate Nucleus Segmentation of Brain MRI: Application in Volumetric Analysis of Pediatric Attention-Deficit/Hyperactivity Disorder, BioMedical EngineeringOnLine 2011, 10:105 doi:10.1186/1475-925X-10-105, ISSN: 1475-925X.

3.
Laura Igual; Joan Carles Soliva; Sergio Escalera; Roger Gimeno; Oscar Vilarroya; Petia Radeva. Automatic Brain Caudate Nuclei Segmentation and Classification in Diagnostic of Attention-Deficit/Hyperactivity Disorder. Computerized Medical Imaging and Graphics, 36(8), pp.591-600, 2012. doi:10.1016/j.compmedimag.2012.08.002.

4.
40. Escalera S, Fornés A, Pujol O, Lladós J, Radeva P: Circular Blurred Shape Model for Multiclass Symbol Recognition.
IEEE Transactions on Systems, Man, and Cybernetics 2010.

5.
Isabel Lobato González. ENTORNO PARA SEGMENTACIÓN DE IMÁGENES POR RESONANCIA MAGNÉTICA CEREBRALES. Trabajo final de Grado de Ingeniería Informática, Universidad de Barcelona. 2013.

6.
SPM http://www.fil.ion.ucl.ac.uk/spm/

7.
LIBSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/

8.
http://science.howstuffworks.com/mri1.htm

9.
http://en.wikipedia.org/wiki/*

7 Anexo: Manual de uso

7.1 Información general para el uso de la aplicación

Para poder ejecutar la aplicación hay que realizar unos pasos preliminares.

1) Hay que añadir la librería OpenCV a las variables de entorno, para ello desde Windows hay que ir a: Mi PC > botón derecho> Propiedades > Conguración avanzada del sistema > Variables de entorno > Buscar variable 'Path' > Editar. Y añadir al final de la línea (sin espacios):

;<Path OpenCV>\Release;<Path OpenCV>\Debug

Donde <Path OpenCV> es la ruta donde está la carpeta OpenCV 2.1 bin, de la carpeta Code.
Por ejemplo si el código está en: C: se debería añadir al final de la línea:

;C:\BrainCut\Code\OpenCV 2.1 bin\Release;C:\BrainCut\Code\OpenCV 2.1 bin\Debug

2) Añadir todas las imagines .nii o .hdr que se quieran segmentar dentro de la carpeta Brain-Cut\Data\Input

3) Añadir (si es preciso) todas las imágenes segmentadas manualmente dentro de la carpeta BrainCut\Data\GT

7.1.1 Parámetros configurables

Se han añadido una serie de parámetros configurables en la función 'create_parameters', su modificación se recomienda exclusivamente para fines de investigación y desarrollo.

- parameters.structures: Estructuras a segmentar, pueden ser 'Caudate_L' o 'Caudate_R, o ambos.
- parameters.singleseed: Variable boleana, 1 para una unica componente conexa como semilla, 0 para multiseed.
- parameters.erode: Variable boleana, 1 para erosionar, 0 para no erosionar
- parameters.computeDescriptors: Variable boleana que determina si los descriptores ya han sido computados, y por lo tanto si deben recalcularse o no. 1 para recalcular, 0 para cargar.
- parameters.subsampleDescriptors: Variable boleana, especifica si ya se ha realizado el submuestreo de datos de entrenamiento, 1 para computar, 0 para cargar. Los ficheros para cargar deben estar en: ..src\BrainCut\Data\Temporal_Output\ y ser nombrados 'bgdsubsampled.mat' y

'fgdsubsampled.mat respectivamente'

- parameters.samples: Variable numérica que determina el ratio de submuestreo, se recomienda usar las opciones preoconfiguradas -1,-2,-3.

- parameters.visualize: Variable boleana, 1 para visualizar y almacenar resultados obtenidos, 0 para estrictamente segmentar (significativamente más rápido)

7.2 Distribución de carpetas

Para una buena organización se requiere una buena distribución de carpetas, de este modo el usuario solo deberá acceder a algunas de ellas, haciendo todo mucho mas intuitivo y usable.

Data

Esta carpeta contiene todos los archivos y datos, que se requieren, se computan y se almacenan a lo largo de la ejecución del algoritmo. Están estructurados del siguiente modo:

- Carpeta GT: Carpeta contenedora de los Ground Truth. Estas son las IRMs segmentadas manualmente por médicos, y con las que se hará la comparación para ver cuán efectivo es el método.
- Carpeta Input:Esta es la carpeta donde el usuario colocará las IRMs a segmentar, ya sea en formato '.nii' o '.hdr'.
- Carpeta Output: Esta carpeta contiene la salida del algoritmo, es decir los resultados del proceso.

- Carpeta Temporal_Output: Contiene diferentes datos que se van obteniendo a lo largo del proceso. Contiene los resultados de cada módulo, hecho que permite ahorrar mucho tiempo si ya se ha computado un módulo previamente. Los datos de Input no deben variar para que se mantenga la integridad de estos elementos, es muy importante que de modificar los datos de entrada se elimine por completo esta carpeta, para evitar errores.

Code

Contiene todos los scripts y funciones del código, con un prefijo que indica a que fase pertenece, siendo estas:

- train: Training
- ABS: AB Segmentation
- PGC: Preparation for GC Application
- GC: Graph Cut
- ER: Extract results
- E: Evaluate

Además, contiene el clasificador 'classifier.mat'. Si se reentrena es preciso que el nuevo clasificador sustituya éste.