Trabajo Fin de Máster

Máster Universitario en Ingeniería de Telecomunicación

Segmentación de Imágenes Médicas aplicando ICA y Kernelized Fuzzy c-Means

Autora: Ana Isabel García Noguer

Tutora: Susana Hornillo Mellado

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Máster
Máster Universitario en Ingeniería de Telecomunicación

Segmentación de Imágenes Médicas aplicando ICA y Kernelized Fuzzy c-Means

Autora:
Ana Isabel García Noguer

Tutora:
Susana Hornillo Mellado
Profesora Contratada Doctora

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo Fin de Máster: Segmentación de Imágenes Médicas aplicando ICA y Kernelized Fuzzy c-Means

Autora: Ana Isabel García Noguer

Tutora: Susana Hornillo Mellado

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente/a:

Vocales:

Secretario/a:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El/La Secretario/a del Tribunal

AGRADECIMIENTOS

A Mamá, Papá y Rafalito.

Ana Isabel García Noguer

Sevilla, 2019

RESUMEN

La segmentación de tejidos cerebrales es un reto importante, a la vez que tedioso, debido a que las imágenes de resonancia magnética tomadas de tejidos cerebrales presentan niveles de escala de grises muy similares, lo que hace que se puedan confundir unos tejidos con otros con cierta facilidad. En este trabajo se desarrollan diversas técnicas como *K-Means*, *Fuzzy C-Means* y clúster kernelizado con fuzzy c-means (KFCM) combinado con análisis de componentes independientes (ICA) que obtienen una segmentación de los distintos tejidos o regiones de interés en las imágenes cerebrales de resonancia magnética (MRI).

El método propuesto parte imágenes cebrebrales multimodales denominadas T1-Weighted, T2-Weigthed y PD-Weighted. En primera instancia, se aplica a estas imágenes un pre-procesado en el que se elimina el cráneo de las imágenes cerebrales. A través del análisis ICA se extraen tres componentes independientes. Como las imágenes multimodales son consideradas como una combinación lineal de señales, aplicar ICA hace que se produzca una mejora en el contraste de las imágenes cerebrales.

El resultado de extraer las tres componentes independientes será la entrada de los distintos algoritmos de clasificación para extraer los tejidos cerebrales. Haciendo un análisis de los resultados del experimento, el método propuesto es capaz de extraer con precisión formas complicadas de los tejidos cerebrales.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

NOTACIÓN

ICA	*Independent Component Analysis*
MRF	*Markov Random Fields*
BSS	*Blind Source Separation*
MRI	*Magnetic Resonance Imaging*
TAC	*Tomografía Axial Computerizada*
CT	*Computerized Tomography*
SPECT	*Single Photon Emission Computed Tomography*
PET	*Positron Emission Tomography*
US	*Ultrasound*
KFCM	*Kernelized Fuzzy c-Means*
PCA	*Principal Component Analysis*
ICAMM	*Independent Component Analysis Mixture Model*
EICAMM	*Enhanced ICA Mixture Model*
KICA	*Kernelized Independent Component Analysis*
FCM	*Fuzzy C-Means*
MEPFC	*Maximum Entropy Principle-based Fuzzy Clustering*
PNFCM	*Possibilistic Neurofuzzy C-Means Algorithm*
PCM	*Possibilistic C-Means Algorithms*
GKFCM	*Gaussian Kernel Fuzzy C-Means*
RSF	R*egion Scalable Fitting*
KWFLICM	*Kernel Weigthed Fuzzy Factor Local Information C-Means*

SKFCM	*Spatial constraint Kernelized Fuzzy C-Means*
GRBF	*Gaussian Radial Base Function*
Vóxel	*Volumetric Píxel*
WM	*White Matter*
GM	*Gray Matter*
CSF	*CerebroSpinal Fluid*
VP	*Verdaderos Positivos*
FP	*Falsos Positivos*
FN	*Falsos Negativos*
VN	*Verdaderos Negativos*
TPR	*True Positive Rate*
ACC	*Accuracy*
TNR	*True Negative Rate*
PPV	*Positive Predictive Value*
NPV	*Negative Predictive Value*

1 ESTADO DEL ARTE

Nadie puede construir un mundo mejor sin mejorar a las personas. Cada uno debe trabajar para su propia mejora.

- Marie Curie -

A medida que han ido avanzando las tecnologías del tratamiento de imagen médica, se ha producido en paralelo un aumento de la demanda de medios visuales que sean más sofisticados y puedan servir de ayuda a la inspección anatómica de estructuras, identificación de enfermedades, análisis de características en una región de interés de una imagen médica y registro y reconstrucción de imágenes.

De entre las distintas tecnologías desarrolladas para satisfacer esta demanda, la segmentación de imágenes juega un papel indispensable. Las imágenes médicas están principalmente limitadas por bajo contraste, presencia de ruidos, intensidad no homogénea y otras ambigüedades que complican enormemente esta tarea de segmentación. Además, si comparamos las imágenes médicas con el resto de las imágenes, las primeras tienden a presentar mayor dificultad a la hora de recuperar las distintas formas a segmentar, haciendo necesario el uso de técnicas más precisas que ayuden a realizar esta tarea.

La segmentación de imagen hace referencia a la técnica que divide una imagen en diferentes regiones y se usa típicamente para identificar regiones de interés. En la literatura se pueden encontrar diversos algoritmos que se han venido empleando para segmentar imágenes, como el Análisis en Componentes Independientes (ICA, *Independent Component Analysis*) [1] [2] , *fuzzy clustering* [3], métodos basados en *level set* [4] [5], campo aleatorio de Markov (MRF, *Markov Random Fields*) [6] y redes neuronales [7], entre otras, cada una con sus ventajas y desventajas.

La mayoría de las imágenes médicas se toman mediante técnicas de adquisición especiales tales como la resonancia magnética (MRI, *Magnetic Resonance Imaging*), en la que se prestará especial interés en este trabajo, la tomografía axial computerizada (TAC o CT, *Computerized Axial Tomography*), tomografía por emisión de un único fotón (SPECT, *Single Photon Emission Computed Tomography*), la tomografía por emisión de positrones (PET, *Positron Emission Tomography*) y ultrasonidos (US, *Ultrasound*). Debido a sus ventajas sobre otras técnicas de

diagnóstico por imagen [8] , la resonancia magnética ha sido ampliamente utilizada para la segmentación. Es por ello, como bien se ha mencionado anteriormente, que las imágenes con las que se va a trabajar en este proyecto serán las MRI. En el Capítulo 3 se detallará cómo se adquieren estas imágenes.

Para reducir la amplia incertidumbre presente en las imágenes médicas, este estudio se esfuerza por integrar el análisis de componentes independientes (ICA) junto con la técnica de separación de conjuntos llamada *kernelized fuzzy c-means* (KFCM) y el preprocesado de imágenes como algoritmo computacional para usar en la segmentación de imágenes médicas.

Como técnica estadística y computacional para revelar factores ocultos que subyacen a conjuntos de variables aleatorias, mediciones o señales, nace ICA, que define un modelo de generación para combinaciones de datos provenientes de múltiples variables, asumiendo que son mezclas de algunas variables latentes desconocidas. Fue desarrollado originalmente como medio para resolver el problema de la separación ciega de fuentes o BSS (*Blind Source Separation*) [9].

El problema de la separación ciega de fuentes consiste en estimar un conjunto de señales fuente de las que no tenemos información a priori, es decir, a partir de mezclas de señales sin tener conocimiento alguno de los mecanismos de mezclas utilizados para la combinación de estas. Para desarrollar una clasificación de estas señales sin supervisión, los datos se modelan como una mezcla de clases descrita por combinaciones lineales de señales independientes no gausianas. ICA se ha utilizado de forma exitosa para separar señales multivariadas, pero se deben cumplir para ello dos limitaciones estadísticas.

El Análisis de Componentes Independientes (ICA, *Independent Component Analysis*) es un método para la separación ciega de fuentes, basado en la independencia estadística de dichas fuentes. A diferencia del Análisis de Componentes Principales (PCA, *Principal Component Analysis*) en ICA, no se supone que las componentes del método sean ortogonales para su separación y únicamente se asume la independencia estadística de las componentes.

Lee et al. [10] proponen el modelo de mezclas ICA (ICAMM, *ICA Mixture Model*) para superar una de las limitaciones de ICA, que es suponer que las señales fuente son independientes. El algoritmo ICAMM encuentra los componentes independientes y la matriz de mezclas para cada clase y calcula la probabilidad de pertenencia a la clase para cada patrón en el conjunto de datos. Las reglas de aprendizaje para el ICAMM se obtuvieron utilizando el método de ascenso de gradiente para maximizar la función de datos de probabilidad de registro.

Para intentar mejorar el rendimiento de ICAMM, *Oliveira y Romero* [1] propusieron una mejora del modelo de mezclas ICA que denotaron como EICAMM (*Enhanced ICA Mixture Model*). Este algoritmo implementa algunas modificaciones en las reglas de aprendizaje del algoritmo ICAMM original. EICAMM es similar al algoritmo ICAMM original, ya que ambos están enfocados en la maximización de la información mutua propuesta por *Bell y Sejnowski* [11].

ICA y su extensión ICAMM han sido ampliamente aplicados en diversos campos de investigación [12]. *Jenssen y Eltoft* [2] utilizaron ICA para generar un banco de filtros de datos estadísticamente dependientes para la segmentación de texturas. A la salida del banco de filtros se tienen imágenes ICA (estadísticamente independientes) fruto del aprendizaje a su paso por el mismo a partir de imágenes de texturas. De esta manera el método es capaz de segmentar la imagen en las distintas texturas.

Tateyama et al. [13] propusieron un método basado en el *kernel* del modelo de análisis de componentes independientes (KICA, *Kernelized Independent Component Analysis*) para clasificación de imágenes MRI. KICA, basado en una aproximación no lineal para extraer componentes independientes en imágenes, parecía producir una mejora significativa en los conjuntos de datos de resonancia magnética cerebral clasificando correctamente las distintas texturas de las imágenes MRI cerebrales.

Continuando con la investigación acerca de otros métodos utilizados para la segmentación de imágenes de resonancia magnética, el *fuzzy clustering* (o agrupación difusa si se traduce al español) resulta considerablemente beneficiosa gracias a su capacidad de retener mucha más información de la imagen original que los métodos para la segmentación de imágenes mencionados anteriormente.

En el caso particular de las imágenes médicas, los algoritmos de *clustering* tratan de asociar cada píxel de cada imagen a una clase de tejido aplicando la noción de similitud a una clase. Por lo tanto, se puede adoptar esta misma postura para la segmentación de imágenes en función de sus tejidos. A diferencia del método de clasificación *K-means* [14], el algoritmo *Fuzzy C-Means* (FCM) permite a los píxeles pertenecer a múltiples clases variando el grado de pertenencia a cada una de las clases. Por lo tanto, los algoritmos basados en *Fuzzy C-Means* son particularmente adecuados para manejar un problema de incertidumbre relacionado con la segmentación de imágenes médicas multimodales.

Masulli and Schenone [15], después de realizar algunas pruebas con algoritmos de *clustering* que contenían el algoritmo FCM tales como *Maximum Entropy Principle-based Fuzzy Clustering* (MEPFC) [16] y dos versiones de *Possibilistic C-Means Algorithms* (PCMs) [17]- [18], introdujeron su algoritmo de clustering llamado *Possibilistic Neurofuzzy C-Means Algorithm* (PNFCM).

El algoritmo KFCM (*Kernelized Fuzzy C-Means*) [19]- [20] se desarrolló sustituyendo la distancia Euclídea original usada en el algoritmo FCM (*Fuzzy C-Means*) por una distancia inducida mediante un kernel (o *kernel-induced*) y aplicando una penalización espacial en las funciones de pertenencia a un grupo. *Kannan et al.* [21] proponen un algoritmo FCM con kernel basado en la tangente hiperbólica para segmentar imágenes de resonancia magnética de mama. El algoritmo fue desarrollado integrando el kernel, es decir, la función tangente hiperbólica, y los métodos basados en Lagrange con el algoritmo FCM actuando como función objetivo obteniendo así una segmentación efectiva.

Gupta et al. [3] presentaron un método de segmentación híbrida que combinaba un *clustering* basado en FCM con Kernel Gaussiano (GKFCM, *Gaussian Kernel Fuzzy C-Means*) [22] con un modelo de contornos activos impulsado por una función de energía basada en R*egion Scalable Fitting* (RSF) [23] para segmentar imágenes médicas de ultrasonido. En este método, el resultado obtenido del método GKFCM es utilizado para inicializar el contorno que, mediante el modelo de contornos activos, se extiende para identificar las regiones estimadas.

GKFCM también ayuda a estimar los parámetros de control usados en los procesos de curva de evolución y consigue mejores resultados sustituyendo la distancia Euclídea por la distancia *kernel-induced.* La función RSF es la responsable de atraer el contorno hacia los límites del objeto y no requiere de un proceso de reinicialización cuando la curva está evolucionando.

Gong et al. [24] propusieron una mejora para FCM llamada KWFLICM (*Kernel Weigthed Fuzzy Factor Local Information C-Means*), introduciendo un factor difuso ponderado de compensación y un kernel (*Kernel Weigthed Fuzzy Factor*). El factor de compensación, dependiendo de la distancia espacial de un píxel con todos los píxeles vecinos y de su diferencia de nivel de gris, es introducido en la función objetivo para garantizar que no haya presencia de ruido y conservar la precisión de la imagen.

Para mejorar la robustez frente al ruido y evitar los valores atípicos en la segmentación, el algoritmo KWFLICM incorpora además una medida de distancia del kernel en su función objetivo. De esta forma, el algoritmo consigue buenos resultados aplicados a la segmentación de imágenes y es robusto frente a diversos tipos de ruido.

A pesar de las ventajas que presentan los métodos mencionados anteriormente, capaces de segmentar de forma más homogénea, reduciendo los falsos positivos y eliminando rudio, la mayoría de ellos todavía necesitan de conocimientos previos acerca del número de clústers en los que se dividen las imágenes a segmentar. Este dato puede no ser un dato conocido en futuras imágenes a segmentar.

Existen numerosos criterios de validación [25] diferentes para estimar el número de clústeres que ayudan a alcanzar mayor precisión en la segmentación de una imagen dada cuando se procesan mediante los métodos KFCM o SKFCM (*Spatial constraint Kernelized Fuzzy C-Means*) [26]- [27].

2 ALGORITMOS DE CLASIFICACIÓN

Me gusta la gente que lucha contra las adversidades. Me gusta la gente que busca soluciones. Me gusta la gente que valora a sus semejantes no por un estereotipo social ni por cómo lucen.

- Mario Benedetti -

El aprendizaje de máquina (*Machine Learning*) estudia el aprendizaje automático a partir de datos (*data-driven*, gobernado por los datos) para conseguir hacer predicciones precisas a partir de observaciones con datos previos. La clasificación automática de objetos o datos es uno de los objetivos del aprendizaje de máquina [28].

En este contexto, se pueden considerar tres tipos de algoritmos.

- **Clasificación supervisada:** disponemos de un conjunto de datos (por ejemplo, imágenes de letras escritas a mano) que vamos a llamar datos de entrenamiento y cada dato está asociado a una etiqueta (a qué letra corresponde cada imagen). Construimos un modelo en la fase de entrenamiento (*training*) utilizando dichas etiquetas, que nos dicen si una imagen está clasificada correcta o incorrectamente por el modelo. Una vez construido el modelo podemos utilizarlo para clasificar nuevos datos que, en esta fase, ya no necesitan etiqueta para su clasificación, aunque sí la necesitan para evaluar el porcentaje de objetos bien clasificados.
- **Clasificación semisupervisada:** algunos datos de entrenamiento tienen etiquetas, pero no todos. Este último caso es muy típico en clasificación de imágenes, donde es habitual disponer de muchas imágenes en su mayoría no etiquetadas. Estos se pueden considerar algoritmos supervisados que no necesitan todas las etiquetas de los datos de entrenamiento.
- **Clasificación no supervisada:** los datos no tienen etiquetas (o no queremos utilizarlas) y estos se clasifican a partir de su estructura interna (propiedades, características).

Los métodos que se van a usar en esta investigación se sitúan en los algoritmos de clasificación no supervisada. En este capítulo se van a desarrollar, de entre todos, los distintos métodos de *clustering* que se han empleado en este trabajo para realizar la segmentación de imágenes MRI cerebrales en sus distintos tejidos.

2.1. El Método *K-Means*

K-means [28]- [29] es un algoritmo de clasificación no supervisada (*clustering*) que agrupa objetos en k grupos basándose en sus características. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o clúster. Se suele usar la distancia cuadrática.

Para ejecutar el algoritmo se debe pasar como entrada el conjunto de datos y un valor k. El conjunto de datos serán las características (*features*) para cada punto. El método se puede describir en tres pasos:

1. **Inicialización:** Las posiciones iniciales de los k centroides serán asignadas de manera aleatoria de cualquier punto del conjunto de datos de entrada. A partir de aquí se realiza un proceso de iteración en 2 pasos.

2. **Asignación objetos a los centroides:** cada objeto de los datos es asignado a su centroide más cercano.

3. **Actualización de los centroides:** se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.

El algoritmo itera entre estos dos últimos pasos hasta cumplir un criterio de detención:

- si no hay cambios en los puntos asignados a los grupos,
- si la suma de las distancias se minimiza,
- o si se alcanza un número máximo de iteraciones.

Se resuelve un problema de optimización, siendo la función para optimizar (minimizar) la suma de las distancias euclídeas cuadráticas de cada objeto al centroide de su clúster.

Los objetos se representan con vectores reales de d dimensiones $(x_1, x_2, \dots, x_n)$ y el algoritmo *k-means* construye k grupos donde se minimiza la suma de distancias de los objetos, dentro de cada grupo $S = \{S_1, S_2, \dots, S_k\}$ a su centroide. El problema se puede formular de la siguiente forma:

$$\min_{S} E(\mu_i) = \min_{S} \sum_{i=1}^{k} \sum_{X_j \in S_i} \left\| x_j - \mu_i \right\|^2 \quad (1)$$

donde S es el conjunto de datos cuyos elementos son los objetos x_j representados por los vectores, donde cada uno de sus elementos representa una característica o atributo. Se tienen k grupos o clústeres con sus correspondientes centroides μ_i.

En cada actualización de los centroides, desde el punto de vista matemático, se impone la condición necesaria de extremo a la función $E(\mu_i)$ que, para la función cuadrática (1) es:

$$\frac{\delta E}{\delta \mu_i} = 0 \Longrightarrow \mu_i^{t+1} = \frac{1}{\left| S_i^{(t)} \right|} \cdot \sum_{X_j \in S_i^{(t)}} x_j \quad (2)$$

y se toma el promedio de los elementos de cada grupo como nuevo centroide, donde t indica el número de iteración.

Las principales ventajas del método *k-means* son que es un método sencillo y rápido. Pero es necesario decidir el valor de *k* y el resultado final depende de la inicialización de los centroides. En principio no converge al mínimo global sino a un mínimo local.

Para esclarecer estos conceptos sería interesante ver un caso práctico [30]. Este ejemplo ha sido implementado en *Matlab R2017a*.

2.1.1 Clasificación de la Flor Iris en sus tres variantes mediante K-Means.

El set de datos de la flor del Iris (o Fisher Iris *dataset*) es un conjunto de datos multivariante introducido por Ronald Fisher en su artículo de 1936 [31] como un ejemplo de análisis discriminante lineal. El set de datos contiene 50 muestras de cada una de tres especies de Iris: Iris setosa, Iris virginica e Iris versicolor. A continuación, se representan las tres especies.

Figura 1. Ejemplo K-means- Especies de Iris [32]

Para cada una de estas especies se midieron cuatro rasgos de cada muestra:

- Longitud del sépalo
- Ancho del sépalo
- Longitud del pétalo
- Ancho del pétalo

Los sépalos son los que envuelven a las otras piezas florales en las primeras fases de desarrollo, cuando la flor es solo un capullo, por su parte los pétalos son la parte inferior del perianto y comprende las partes estériles de una flor. En la siguiente imagen se puede observar la diferencia entre pétalo y sépalo.

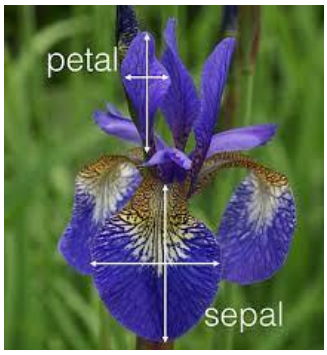


Figura 2. Ejemplo K-means – Diferencia entre sépalo y pétalo [32]

El primer paso es cargar los datos en Matlab y almacenarlos en la variable |X|. Como es un set de datos incorporado ya en Matlab solo habría que cargarlos mediante la función *load*. Se tomarán como predictores los valores de longitud y ancho del pétalo. En la Figura 3 se representa el set de datos |X|.

```
load fisheriris
X = meas(:,3:4);
```

El grupo más grande parece estar dividido en una región de varianza más baja y una región de varianza más alta. Esto podría indicar que el clúster más grande está formado por dos conjuntos superpuestos. Se establece, por tanto, un valor de *k* igual a 3 y se aplica *k-means* al conjunto de datos, como sigue:

```
% Cluster the data.  Specify _k_ = 3 clusters.
rng(1); % For reproducibility
[id

x,C] = kmeans(X,3);
```

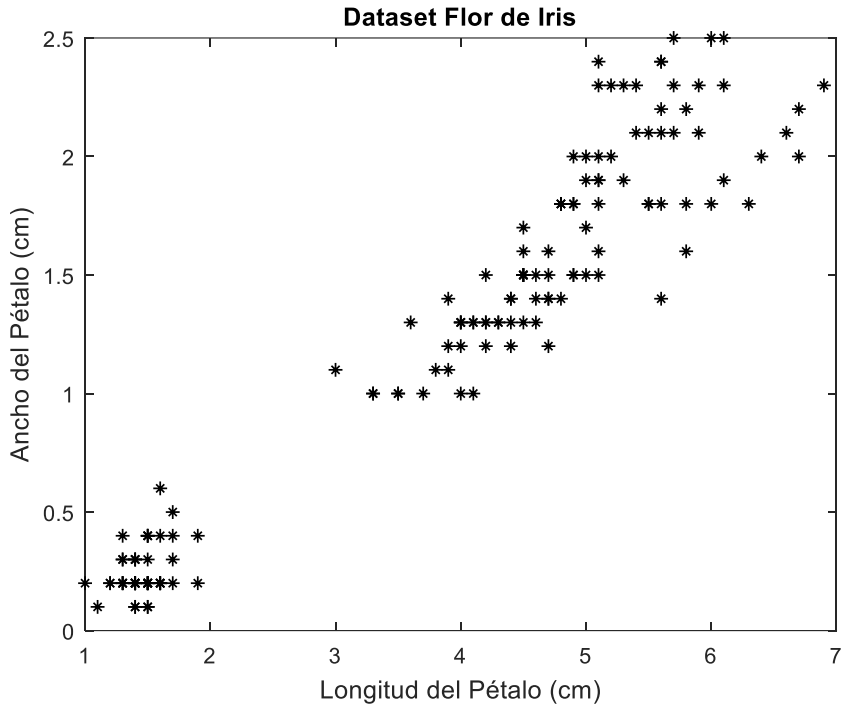


Figura 3. Ejemplo K-means - Representación gráfica del set de datos Flor de Iris

La función *kmeans* devuelve:

- **idx**, que es un vector de índices de agrupación predichos que corresponden a las observaciones en |X|.
- **C**, que es una matriz de 3 × 2 que contiene las ubicaciones finales de los centroides.

Para visualizar mejor los resultados de aplicar K-means, se aplica de nuevo la función *kmeans* para calcular la distancia desde cada centroide a las observaciones en una cuadrícula o grid.

```
x1 = min(X(:,1)):0.01:max(X(:,1));
x2 = min(X(:,2)):0.01:max(X(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)]; % Defines a fine grid on the plot

% Assigns each node in the grid to the closest centroid
idx2Region = kmeans(XGrid,3,'MaxIter',1,'Start',C);
```

De esta manera, ya se pueden visualizar las 3 regiones en las que la función *kmeans* ha dividido el set de datos.

2.2. El método *Fuzzy C-Means* (FCM)

El agrupamiento difuso (del inglés, *fuzzy clustering*) es una clase de algoritmos de agrupamiento donde cada elemento tiene un grado de pertenencia difuso a los grupos.

Este tipo de algoritmos surge de la necesidad de resolver una deficiencia del agrupamiento exclusivo, como el método K-means, que considera que cada elemento se puede agrupar inequívocamente con los elementos de su clúster y que, por lo tanto, no se asemeja al resto de los elementos.

Esto se logra representando la similitud entre un elemento y un grupo por una función, llamada función de pertenencia, que toma valores entre cero y uno. Los valores cercanos a uno indican una mayor similitud, mientras que los cercanos a cero indican una menor similitud. Por lo tanto, el problema del agrupamiento difuso se reduce a encontrar una caracterización de este tipo que sea óptima.

En particular, el algoritmo fuzzy c-means [33] - [34] se basa en la minimizacion de una función objetivo o de costo denominada c-means, que es la mostrada a continuación.

$$J_m(X;U,V) = \sum_{i=1}^{c}\sum_{k=1}^{N} \mu_{ik}^{m} \cdot \|x_k - v_i\|_A^2 \tag{3}$$

sujeto a la restricción

$$\sum_{i=1}^{c} \mu_{ik} = 1, \forall\, k = 1,2,\dots,N \tag{4}$$

donde $v_i \in V$ es el centro del *i*-ésimo clúster a determinar, $x_k \in X$ es el *k*-ésimo elemento de los datos de entrenamiento, $\mu_{ik} \in U$ es el grado de pertenencia de x_k al grupo i, cuyo valor está en el intervalo cerrado [0,1], m es el exponente de peso mayor que 1 que determina el grado de difusividad (fuzziness) de los grupos resultantes, c es el número de grupos y N el número de datos de entrenamiento. La distancia $\|x_k - v_i\|_A^2$ se define como sigue:

$$D_{ikA}^2 = \|x_k - v_i\|_A^2 = (x_k - v_i)^T \cdot A \cdot (x_k - v_i) \tag{5}$$

donde A es una medida de distancia. El algoritmo FCM usa A = I, la matriz identidad, en cuyo caso la distancia es la distancia Euclídea. Los puntos estacionarios de la función objetivo (3) se hallan al expandirla con las restricciones por medio de los multiplicadores de Lagrange:

$$\bar{J}_m(X;U,V,\lambda) = \sum_{i=1}^{c}\sum_{k=1}^{N} \mu_{ik}^{m} \cdot D_{ikA}^2 + \sum_{k=1}^{N} \lambda_k \cdot \left(\sum_{i=1}^{c} \mu_{ik} - 1\right) \tag{6}$$

Hallando el gradiente de (6) $\bar{J}_m$ con respecto a U, V y λ e igualandolo a cero, y si además $D_{ikA}^2 > 0, \forall i, k$, entonces (U, V) minimizan la función objetivo si y solo si:

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{D_{ikA}^2}{D_{jkA}^2} \right)^{\frac{2}{m-1}}}, 1 \leq i \leq c \; y \; si \; 1 \leq k \leq N \tag{7}$$

$$v_i = \frac{\sum_{k=1}^{N} \mu_{ik}^m \cdot x_k}{\sum_{k=1}^{N} \mu_{ik}^m}, 1 \leq i \leq c \tag{8}$$

Debido a que la suma $\sum_{i=1}^{c} \mu_{ik} = 1, \forall\, k = 1,2, \dots, N$, se cumple que $0 < \sum_{i=1}^{c} \mu_{ik} < N\,, \forall i = 1,2, \dots, c$. La ecuación (8) muestra que el centro v_i es la media ponderada de los datos que pertenecen a un grupo, donde los pesos de ponderación son los grados de pertenencia.

La partición difusa de los datos se lleva a cabo a través de un proceso de optimización iterativa de la función objetivo (3) actualizando los valores de μ_{ik} y de v_i. El algoritmo FCM se detiene cuando se cumple:

$$\max \left\| \mu_{ik}^{(j+1)} - \mu_{ik}^{(j)} \right\| < \varepsilon \tag{9}$$

donde ε es un criterio de terminación entre 0 y 1, y j es paso de iteración. El algoritmo que se lleva a cabo es el siguiente:

- **Paso 1:** Considere un conjunto de datos **X** de N ejemplos, $X = \{x_1, \dots, x_N\}$.
- **Paso 2:** Fije el número de grupos o clases c ($2 \leq c \leq N$) y el criterio de terminación ε.
- **Paso 3:** Fije un nivel apropiado de difusividad (*fuzziness*) de grupo m.
- **Paso 4:** Inicialice la matriz $U_{N \times c}$ con valores aleatorios si $\mu_{ik} \in [0,1]$ y $\sum_{i=1}^{c} \mu_{ik} = 1$.
- **Paso 5:** Calcule los centros de los grupos v_i usando la ecuación (8).
- **Paso 6:** Actualice la matriz de pertenencia $U = \left[\mu_{ik}\right]_{i=1,\dots,c;k=1,\dots,N}$ de acuerdo con (7).
- **Paso 7:** Repita desde el Paso 5 si $\left\| \mu_{ik}^{(j+1)} - \mu_{ik}^{(j)} \right\| > \varepsilon$.

De la misma forma que para el algoritmo K-means, se muestra a continuación una implementación del mismo en Matlab para ver su funcionamiento.

2.2.1. Clasificación de un conjunto de datos mediante el algoritmo Fuzzy c-Means

Se aplicará el algoritmo FCM al mismo set de datos que en el ejemplo del apartado anterior. Por tanto, de igual forma, el primer paso es cargar los datos en Matlab y almacenarlos una variable *X*. Como es un set de datos incorporado ya en Matlab solo habría que cargarlos mediante la función *load*.

```
load fisheriris
X = meas(:,3:4);
```

A continuación, se aplica el algoritmo FCM, se hace un post-procesado para visualizar mejor el resultado y se representa gráficamente. Como resultado de aplicar el algoritmo FCM se obtienen tanto los centroides, como la matriz *idx*, que contiene el grado de pertenencia de cada punto en cada grupo. Los valores de la matriz *idx*, entre 0 y 1, indican desde la no pertenencia al grupo hasta la pertenencia completa.

```
% Cluster the data.  Specify _k_ = 3 clusters.
rng(1); % For reproducibility
C,idx]=fcm(X,3);

x1 = min(X(:,1)):0.01:max(X(:,1));
x2 = min(X(:,2)):0.01:max(X(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)]; % Defines a fine grid on the plot

idx2Region = kmeans(XGrid,3,'MaxIter',1,'Start',C);
    % Assigns each node in the grid to the closest centroid
```

En la siguiente figura se muestran ambos algoritmos implementados sobre el mismo set de datos a modo de comparativa para ver las diferencias entre el algoritmo K-Means y el algoritmo FCM. Se puede apreciar a simple vista que las regiones no son exactamente iguales y que los centroides varían ligeramente.

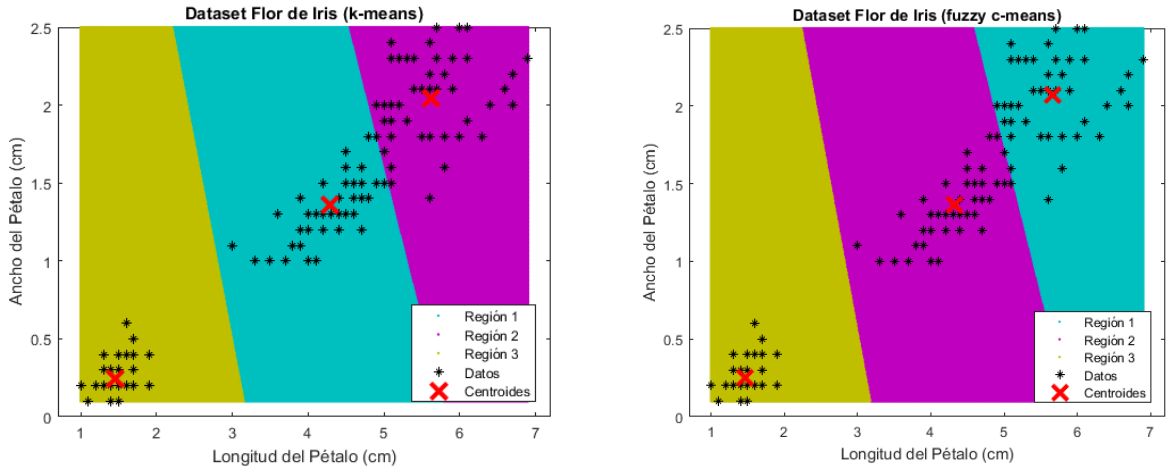


Figura 4. Comparativa de Resultados aplicando K-Means y FCM al set de datos Flor de Iris

¿Es mejor *k-means* o *fuzzy c-means*? La elección de un método u otro para la separación de regiones depende en gran parte de los datos que se tengan de partida. Para el ejemplo anterior, mediante un estudio visual de los resultados, resulta prácticamente irrelevante la elección de uno u otro, aunque habría que hacer un análisis numérico que calculara cuál de los dos tiene un

porcentaje mayor de acierto en su predicción. En el Capítulo 4 de este trabajo se estudiarán a fondo los resultados obtenidos para ambos métodos.

2.3. El método Kernelized Fuzzy C-Means (KFCM)

El algoritmo KFCM (*Kernelized Fuzzy C-Means*) [19]- [20] se desarrolló sustituyendo la distancia Euclídea original usada en el algoritmo FCM (*Fuzzy C-Means*) por una distancia inducida mediante un kernel (o *kernel-induced*) y aplicando una penalización espacial en las funciones de pertenencia a un grupo.

El algoritmo Kernelized Fuzzy C-Means [20]- [25] en el que se ha basado esta investigación se construye mediante la siguiente función objetivo

$$J_m = \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^{m} \cdot \left\|\Phi(x_i) - \Phi(c_j)\right\|^2, 1 \le m \le \infty \qquad (10)$$

donde m es un número real mayor que 1 que será el exponente de u_{ij}, Φ es un mapeo implícito no lineal, x_i y c_j son, respectivamente, el i-ésimo dato medido y el centro del j-ésimo grupo en el espacio original.

El mapeo implícito no lineal se define como sigue: Se realiza un mapeo desde el espacio de datos p hasta el espacio de características mapeado d, Φ: X $\rightarrow F$ ($x \in R^p \rightarrow \Phi(\mathrm{x}) \in R^d, d > p$), es decir, el conjunto de datos de entrada $\{x_1, \ldots, x_n\} \subseteq X$, que pertenecen a un espacio vectorial de baja dimensión, se mapea a un espacio de características d-dimensional potencialmente más alto.

El propósito final de este mapeo es convertir el problema no lineal en el espacio de entrada original en uno potencialmente lineal pero en un espacio de características dimensionalmente mayor. El *kernel* $K(x, y)$ en el espacio de características se puede representar como sigue.

$$K(x, y) = \langle \Phi(\mathrm{x}), \Phi(y) \rangle \qquad (11)$$

donde $\langle \Phi(\mathrm{x}), \Phi(y) \rangle = \Phi^{\mathrm{T}}(\mathrm{x})\Phi(\mathrm{y})$ denota la operación del producto interno. En el estudio, se adopta la función de *kernel* de base radial gaussiana (GRBF, *Gaussian Radial Base Function*) y se describe como

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{\sigma^2}\right) \qquad (12)$$

donde σ es un parámetro del *kernel* gaussiano. Sustituyendo la función $K(x, y)$ en la ecuación (10) se tiene que

$$\left\|\Phi(x_i) - \Phi\left(c_j\right)\right\|^2 = K(x_i, x_i) - 2K\left(x_i, c_j\right) + K(c_j, c_j) \tag{13}$$

Considerando que $K(x, c) = \exp(-\|x - c\|^2/\sigma^2)$, que la función entre un punto y sí mismo $K(x_i, x_i)$ es igual a uno, que la función ente un centroide y sí mismo $K(c_j, c_j)$ es igual a uno y que $\Phi^{\mathrm{T}}(\mathrm{x_i})\Phi\left(\mathrm{c_j}\right)$ es igual a $\Phi(\mathrm{x_i})\Phi^{\mathrm{T}}\left(\mathrm{c_j}\right)$, se puede reescribir la expresión (10) como

$$J_m = 2\sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^m \cdot (1 - K(x_i, c_j)), 1 \leq m \leq \infty \tag{14}$$

Como se ha desarrollado en el apartado anterior para el algoritmo FCM (*Fuzzy C-Means*), la función objetivo (14) se puede minimizar sujeta a la rescricción de U (p. e., $\sum_{i=1}^{c} \mu_{ij} = 1, \forall\, j = 1,2, \ldots, N$). La matriz de membresía U se puede obtener de la expresión siguiente.

$$u_{ij} = \frac{\sum_{k=1}^{c}\left(1 - K(x_i, c_k)\right)^{1/(m-1)}}{\left(1 - K\left(x_i, c_j\right)\right)^{1/(m-1)}} \tag{15}$$

Donde c es el número de clústeres; c_j y c_k son los centroides de los clústers difusos j y k, respectivamente y el parámetro m (conocido como *fuzzifier*) es un factor de peso exponencial para cada uno de los elementos u_{ij}. El centroide c_j se obtiene a partir de la expresión

$$c_j = \sum_{i=1}^{n} \frac{u_{ij}^m \cdot K\left(x_i, c_j\right) \cdot x_i}{u_{ij}^m \cdot K\left(x_i, c_j\right)} \tag{16}$$

2.3.1. Aplicación KFCM. Segmentación de regiones en imágenes.

Siguiendo la dinámica de los apartados anteriores, se presenta un ejemplo de aplicación del algoritmo KFCM implementado en Matlab R2017a. Con motivo de facilitar la comprensión de este algoritmo, se ha elegido un ejemplo aplicado a la segmentación de imágenes.

En este caso se va a utilizar el algoritmo para segmentar regiones en una imagen de resonancia magnética de un corte cerebral que se pasa como parámetro de entrada. Los píxeles de la imagen son, para este ejemplo, el conjunto de datos a clasificar. Se representa, a continuación, la imagen o *dataset* de entrada.

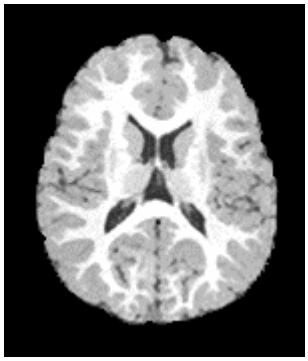


Figura 5. Ejemplo KFCM - Imagen MRI de entrada.

Los parámetros escogidos para la función objetivo son:

- **Número de Clústeres:** 4
- **Valor del exponente *m*:** 2
- **Número máximo de Iteraciones:** 500
- **Epsilon ε:** 10^{-5}
- **Delta:** 10

El código empleado para ejecutar este ejemplo es el siguiente:

```
%Cargamos la imagen de entrada y la representamos
Img = imread('1.bmp');
figure (1),imshow(Img,[]);

% Datos de entrada
cluster_n=4;        % número de clústeres
m=2;                % valor del exponente m
iter_max=500;       % número máximo de iteraciones
e=1e-5;             % epsilon
delta=10;           % valor de delta (enre 5 y 15)

%Preparamos los datos para introducirlos en el algoritmo kfcm
Vinit=KFCM_init(Img,cluster_n);

%Aplicamos el algoritmo KFCM
Img_label=KFCM_Img(Img,Vinit,cluster_n,m,iter_max,e,delta);

%Representamos el clúster completo en que ha dividido el algoritmo
figure(2);
imshow(Img_label,[]);

%Representamos cada uno de los clústeres (cada uno es un tejido)
figure(3);imshow(Img_label==1);
figure(4);imshow(Img_label==2);
figure(5);imshow(Img_label==3);
figure(6);imshow(Img_label==4);
```

La funciones empleadas en este ejemplo se incluyen en los Anexos de este trabajo. Como resultado de aplicar el algoritmo kfcm se obtienen las siguientes imágenes. En primer lugar, Img_label representa todas las clases o clústeres superpuestos.

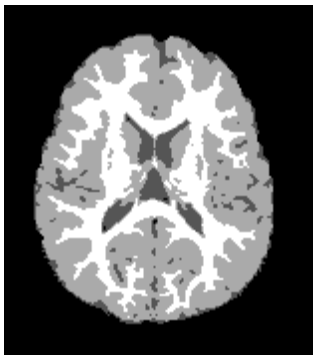
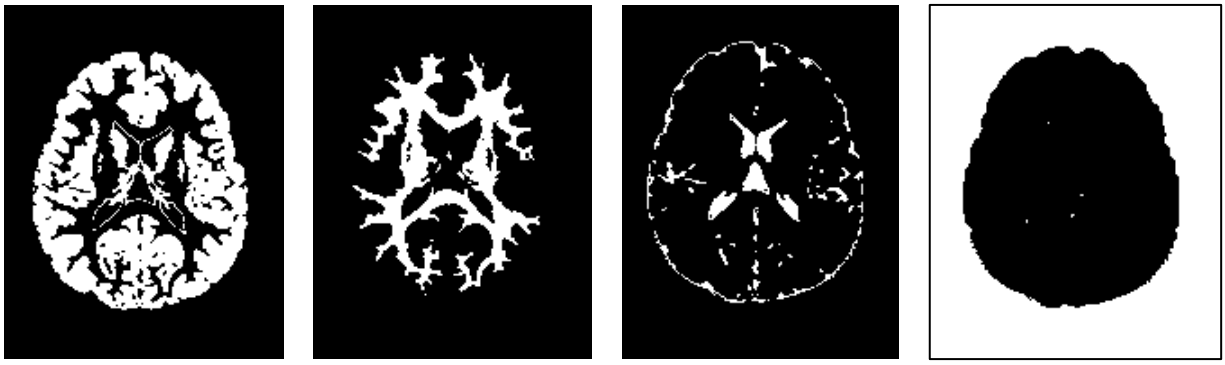


Figura 6. Ejemplo KFCM – Clases superpuestas resultados de algoritmo KFCM

Si separamos estos clústeres o clases por colores (o etiquetas) en distintas imágenes se pueden observar los distintos tejidos cerebrales segmentados. Se puede observar este resultado en la siguiente figura.



(a) (b) (c) (d)

Figura 7. Ejemplo KFCM Resultados

De (a) a (c) Distintos tejidos cerebrales segmentados. (d) Fondo segmentado

2.4. El Análisis de Componentes Independientes (ICA)

El Análisis de Componentes Independientes (ICA, *Independent Component Analysis*) es un método para la separación ciega de fuentes, basado en la independencia estadística de dichas fuentes. A diferencia del Análisis de Componentes Principales (PCA, *Principal Component Analysis*) en ICA, no se supone que las componentes del método sean ortogonales para su separación y únicamente se asume la independencia estadística de las componentes. A continuación, se detalla una motivación para el método, así como la teoría necesaria para su construcción y aplicación [35].

2.4.1. Mezcla de Fuentes, el efecto *Cocktail Party*

Se supone una reunión donde se encuentran varias personas hablando al mismo tiempo (de aquí el nombre *Cocktail Party*). Suponer que sólo se tiene interés en escuchar a una de las personas que está hablando, pero lo que se percibe realmente es una mezcla de todas las voces y sonidos presentes en la reunión. Sin embargo, el cerebro humano tiene la valiosa capacidad de aislar o separar los diferentes sonidos que está escuchando y concentrarse únicamente en la voz que se encuentra interesado.

Se considera ahora la siguiente situación, hay dos locutores (1 y 2) y cada uno de ellos está hablando, por lo tanto, se generan dos señales acústicas $\mathbf{s1}$ y $\mathbf{s2}$. Junto con los dos locutores, se encuentran dos micrófonos que graban el sonido que se genera cerca a ellos. Los micrófonos no reciben independientemente cada una de las señales s1 y s2 que se están generando. En lugar de eso reciben una mezcla de las dos señales. El método ICA permite recuperar las dos señales originales.

Las señales pueden ser reprentadas como $\boldsymbol{s1} = \{\boldsymbol{s_{11}}, \boldsymbol{s_{12}}, \dots, \boldsymbol{s_{1N}}\}$ y $\boldsymbol{s2} = \{\boldsymbol{s_{21}}, \boldsymbol{s_{22}}, \dots, \boldsymbol{s_{2N}}\}$ donde N es el número de muestras en el tiempo y $\boldsymbol{s_{ij}}$ representa la amplitude de la señal s_i en cada instante de tiempo j-ésimo. Ambas señales pueden verse de forma matricial:

$$S = \begin{pmatrix} s1 \\ s2 \end{pmatrix} = \begin{pmatrix} (\boldsymbol{s_{11}}, \boldsymbol{s_{12}}, \dots, \boldsymbol{s_{1N}}) \\ (\boldsymbol{s_{21}}, \boldsymbol{s_{22}}, \dots, \boldsymbol{s_{2N}}) \end{pmatrix} \tag{17}$$

donde $S \in R^{pxN}$ representa el espacio que está definido por las señales fuentes y p indica el número de señales fuente. Las señales fuente $s1$ y $s2$, se pueden mezclar formando combinaciones lineales de las primeras, por ejemplo, de la siguiente forma

$$\begin{aligned} x_1 &= a \times s_1 + b \times s_2 \\ x_2 &= c \times s_1 + d \times s_2 \end{aligned} \tag{18}$$

Con *a, b, c* y *d* son coeficientes de mezcla, cada uno de ellos diferentes, teniendo en cuenta que provienen de distintos sensores localizados en diferentes puntos, por lo que cada sensor ha capturado una señal diferente, mezcla de las originales. La ecuación (18) se puede reescribir de la siguiente manera.

$$X = \begin{pmatrix} x1 \\ x2 \end{pmatrix} = \begin{pmatrix} as_1 + bs_2 \\ c_{s1} + ds_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = As \tag{19}$$

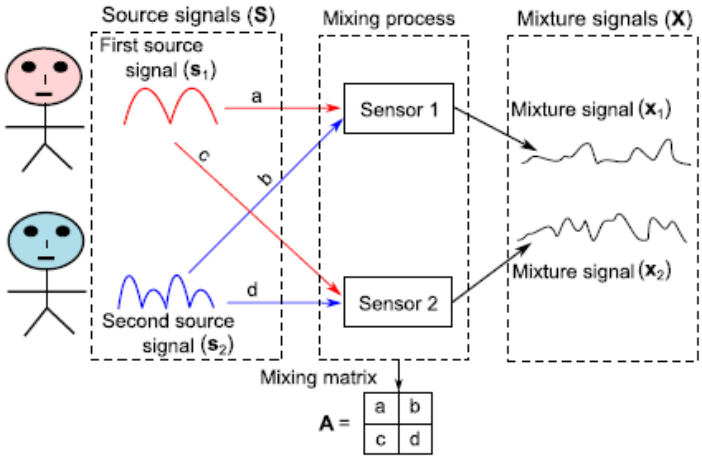


Figura 8. Representación gráfica del proceso de mezcla de las dos señales s1 y s2. [36]

En la práctica, sólo conocemos el vector de mezclas X que se genera al muestrear algún evento. El método ICA consiste en aplicar un algoritmo que nos permita encontrar una matriz de separación W de tal modo que $y = Wx$ sea una buena aproximación del vector s, es decir $y \simeq s$.

2.4.2. Condiciones para el buen funcionamiento del método ICA

Para construir el algoritmo para el método, debemos asumir las siguientes condiciones [35] que son necesarias para que el método genere buenos resultados:

- **Independencia Estadística**

  La independencia estadística es la idea primordial para toda la construcción del método, ya que queremos encontrar dentro de nuestras señales mezcladas, aquellas que son estadísticamente más independientes respecto a las demás. Podemos definir la independencia estadística como sigue:

  Sean $x_1, x_2, \dots, x_m$ un conjunto de variables aleatorias con función de densidad de probabilidad $f(x_1, x_2, \dots, x_m)$, entonces estas variables son mutuamente independientes si

  $$f(x_1, x_2, \dots, x_m) = f_1(x_1) f_2(x_2) \dots f_m(x_m) \qquad (20)$$

  donde f_i es la función de desidad marginal de x_i. Para la construcción del método se asumirá que cada una de las señales s_i sea estadísticamente independiente respecto a las demás.

- **Función de densidad de probabilidad no Gaussiana**

  Las señales fuente independientes deben tener función de densidad de probabilidad no Gaussiana, de esta forma se garantiza que las componentes independientes efectivamente se pueden separar. Esto se puede ver en las siguientes figuras: La Figura 9 muestra una combinación lineal de dos señales aleatorias con función de densidad de porbabilidad no Gaussiana. En ésta se observan claramente que las aristas determinan la combinación lineal de los datos, po lo que es posible separar las señales independientes

originales. Si la condición no se cumple, es decir si las señales tienen función de densidad Gaussiana, los datos estaran agrupados cerca a la media estadística de cada señal y no sera posible separar las señales independientes. La Figura 10 ilustra esto.

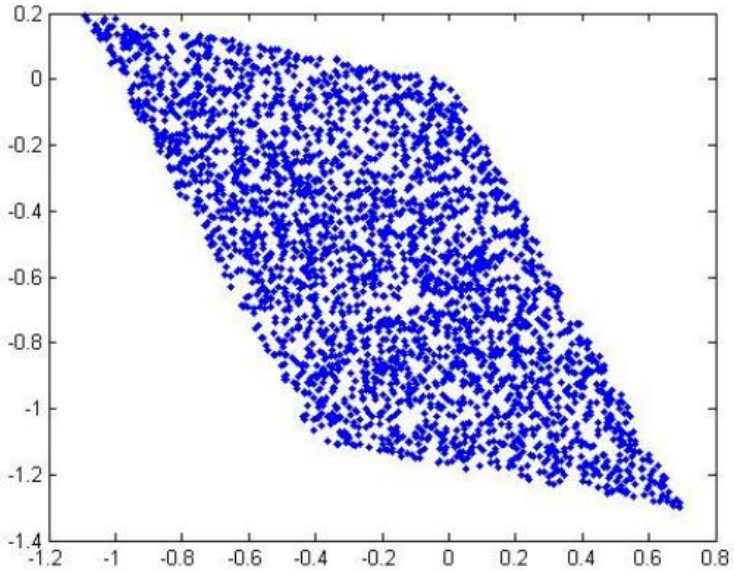


Figura 9. Combinación de señales con densidad de probabilidad no gaussiana [35]

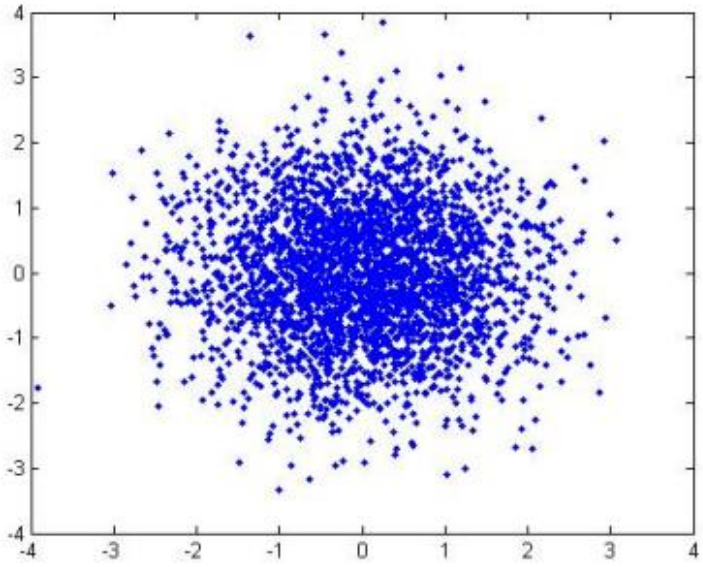


Figura 10. Combinación de señales con densidad de probabilidad gaussiana [35]

2.4.3. Construcción del método ICA

2.4.3.1. Centrado y Blanqueo de los datos

El pre-procesado de blanqueo de datos antes del algoritmo del método ICA ayuda a garantizar la convergencia de éste y también facilita algunas condiciones para la construcción del algoritmo.

El primer paso para el blanqueo consiste en centrar los datos restando la media de todas las señales. Dada una mezcla de señales (X), la media es μ y el proceso de centrado se puede calcular de la siguiente forma:

$$D = X - \mu = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} x_1 - \mu \\ x_2 - \mu \\ \vdots \\ x_n - \mu \end{pmatrix} \quad (21)$$

donde D es la señal fruto de la mezcla de señales después del proceso de centrado. Se puede volver a sumar la media tras el finalizer todo el proceso de ICA.

Una vez centrados, el siguiente paso es blanquear los datos, lo que significa transformar las señales en señales no correlacionadas y luego reescalarlas para que tengan varianza unidad. Este paso incluye dos pasos principales, que son:

- **Incorrelación:** El objetivo de este paso es incorrelar los datos, es decir, hacer que una señal sea no correlacionada con otra y viceversa. Dos variables son incorreladas si su covarianza es cero. Para hacer este proceso de incorrelación ICA usa la técnica de PCA. Se calculan los autovectores que van a formar el nuevo espacio PCA. El primer paso es calcular la martriz de covarianza. La matriz de covarianza de dos variables (x_i, x_j) cualesquiera se expresa como $\Sigma_{ij} = E\{x_i, x_j\} - E\{x_i\}E\{x_j\} = E[(x_i - \mu_i)(x_j - \mu_j)]$. Cuando se tienen muchas variables, la matriz de covarianza se calcula como $\Sigma = \mathrm{E}[DD^T]$, donde D son los datos centrados.

  La matriz se obtiene caculando los autovalores (λ) y los autovectores (V), sabiendo que $V\Sigma = \lambda V$. Los autovectores representan los componentes principales que son las direcciones del espacio de componentes principales y los autovalores son los escalares que indican la magnitud de los autovectores. El autovector que tiene el autovalor mas alto es el primer componente principal (PC_1) y tiene máxima varianza. Para incorrelar los datos, se proyectan en el nuevo espacio calculado como $U = V \cdot D$.

- **Escalado:** A continuación se reescalan cada señal no correlacionada para conseguir que tengan varianza unidad. Por tanto, cada vector U se reescala mediante la expresión $Z = \lambda^{\frac{1}{2}} \cdot U = \lambda^{\frac{1}{2}} \cdot V \cdot D$, siendo Z los datos blanqueados.

En la figura Figura 11 se muestra el procedimiento de pre-procesado de forma esquemática.

2.4.3.2. Teorema del Límite Central

Como se ha mencionado anteriormente, la idea principal para el método ICA es la búsqueda de independencia estadística entre las variables. Pues bien, el siguiente teorema nos da una relación entre la medida de gaussianidad para variables aleatorias y la independencia estadística.

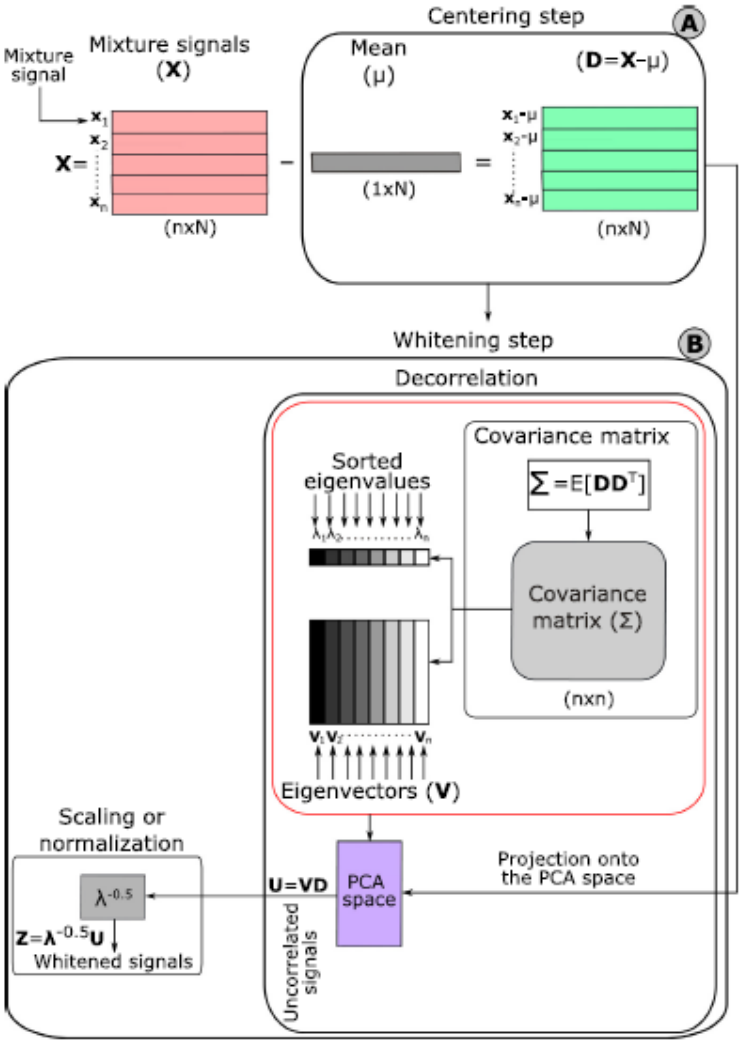


Figura 11. Representación esquemática de la fase de pre-procesado en ICA [36]

Teorema del Limite Central: Sea $x_1, x_2, \ldots, x_n$ una muestra aleatoria de tamaño *n,* es decir, una secuencia de variables aleatorias independientes e idénticamente distribuidas con valores esperados dados por μ y varianzas finitas dadas por σ^2. Suponiendo que interesa el promedio de la muestra S_n de estas variables aleatorias.

$$S_n = \frac{(x_1 + \cdots + x_n)}{n} \tag{22}$$

Por la ley de los grandes números, los promedios de la muestra convergen en probabilidad y casi seguramente al valor esperado μ cuando n tiende a infinito. El teorema del límite central afirma que a medida que n se hace más grande, la distribución de la diferencia entre la media de la muestra S_n y su límite μ, cuando n es muy grande, se aproxima a la distribución normal con media 0 y varianza σ^2.

Por el teorema del limite central, las señales mezcladas x siempre van a ser más gaussianas que las señales independientes originales s. El trabajo del método consistirá entonces en tratar de reducir la gaussianidad que presentan las señales mezcladas para encontrar las señales

independientes, geométricamente, es casi como "girar" los datos hacia los ejes que determinan las componentes independientes y de esta forma minimizar la gaussianidad entre las señales.

2.4.3.3. Negentropia y medida de No-gaussianidad

¿Cómo se logra entonces reducir la gaussianidad? La medida de gaussianidad está relacionada con la independencia estadística entre las variables. Cuanto mayor sea la gaussianidad más información comparten las variables y menos independientes son entre si. Se define primero el concepto de Entropía *H* para un valor aleatorio $y = [y_1, \dots, y_n]$:

$$H(y) = -\int f(y) \cdot log f(y) dy \tag{23}$$

La entropía se puede considerar como una medida de cantidad de información mutua que ontienen los elementos del vector. En base a la entropía *H* podemos definir la función de negentropía *J* como:

$$J(y) = H(y_{gauss}) - H(y) \tag{24}$$

donde y_{gauss} es un vector aleatorio gaussiano con la misma matriz de covarianza de y. Si la entropía mide la cantidad de información mutua en el vector, la negentropía determina la cantidad de información no mutua entre los elementos del vector. Es decir, a mayor negentropía, menor es la gaussianidad de los elementos del vector. La negentropía la se puede considerar como una medida de **no-gaussianidad**.

Dado que la esperanza de una variable aleatoria se define como: $E[X] = \int xf(x)dx$, se puede utilizar una aproximación de la negentropía a través de la esperanza del vector aleatorio de la siguiente forma [35]:

$$J(y_i) \approx c[E\{G(y_i)\} - E\{G(V)\}] \tag{25}$$

donde G es una función llamada función de constraste, c es una constante irrelevante y v es una variable gaussiana con media cero y varianza uno. Esta aproximación de la función de negentropía ayuda a definir el objetivo del método ICA, si se hace $y_i = w^T x$, donde *x* es una de las componentes mezcladas y w^T es el vector desmezclado para este x. La meta será maximizar la función *J* dada por:

$$J(w) \approx [E\{G(w^T x)\} - E\{G(v)\}] \tag{26}$$

donde w es el vector m−dimensional de mezcla de forma que $E\{G(w^T x)^2\} = 1$. Esta última condición es simplemente de escala, para que los vectores no crezcan arbitrariamente.

Ahora se puede generalizar la función anterior para toda la matriz de desmezclado *W*, teniendo en cuenta que se maximizan cada una de las negentropías cuando la suma de todas ellas es máxima. El problema final de maximización será el siguiente:

$$\max \sum_{i=1}^{m} J(w_i)$$
$$\text{s.a. } E\{G(w_k^T x)\} - E\{G(w_j^T x)\} = \delta_{jk} \tag{27}$$

La restricción del problema de maximización garantiza que cada una de las componentes de desmezclado w_j no esté correlacionada con las demás. Dos vectores aleatorios y_i, y_j están no correlacionados si $E[(y_i)(y_j)] - E(y_i)E(y_j) = 0$. Dado que los datos han sido blanqueados, la media de cada vector es cero. Por consiguiente, $E(y_i) = 0$ y la condición de no correlación se reduce a

$$\mathrm{E}[(y_i)(y_j)] = \delta_{jk} \tag{28}$$

donde

$$\delta_{jk} = \begin{cases} 1 \; si \; j = k \\ 0 \; si \; j \neq k \end{cases} \tag{29}$$

2.4.3.4. Algoritmo ICA en una dimensión

Se observa que el óptimo de $J(w)$ se obtiene en un óptimo de $E\{G(w^T x)\}$. Aplicando el método de multiplicadores de Langrange, el óptimo de $E\{G(w^T x)\}$ bajo las condiciones $E\{G(w^T x)^2\} = 1, \|w\|^2 = 1$ se obtiene cuando

$$\nabla(E\{G(w^T x)\}) = \beta \nabla(\|w\|^2) \tag{30}$$

$$E\{x g(w^T x)\} - \beta w = 0 \tag{31}$$

Donde $g = G'$, β será entonces:

$$\beta = w_0^T E\{x g(w_0^T x)\} \tag{32}$$

$$\beta = E\{w_0^T x g(w_0^T x)\} \tag{33}$$

con w_0 el valor óptimo de w.

Para solucionar la ecuación $E\{x g(w^T x)\} - \beta w = 0$ y encontrar el óptimo W_0, aplicaremos el método de Newton. Se denota el lado izquiero de la ecuación anterior como *F* y se obtiene el Jacobiano JF como

$$JF(w) = E\{x x^T g'(w^T x)\} - \beta I \tag{34}$$

Para simplificar la expresión, se aproxima el término $E\{x x^T g'(w^T x)\}$, como $E\{x x^T\} \cdot E\{g'(w^T x)\} \simeq E\{g'(w^T x)\} I$. Esto es así debido a que se garantiza la independencia entre los datos. Ahora la matriz Jacobiana es diagonal y puede representarse como:

$$JF(w) = E\{g'(w^T x)\}I - \beta I \ = (E\{g'(w^T x)\} - \beta)I \tag{35}$$

La expresión anterior permite evitar el cálculo de la inversa de la matriz Jacobiana al aplicar el método de Newton, ya que la inversa de la matriz de reduce a:

$$JF(w) = \frac{1}{E\{g'(w^T x)\} - \beta} I \tag{36}$$

Aplicando el método de Newton se obtiene:

$$w^+ = w \ - \frac{[E\{xg(w^T x)\} - \beta w]}{[E\{g'(w^T x)\} - \beta]} \tag{37}$$

$$w^* = w^+ / \|w^+\| \tag{38}$$

donde w^* denota el nuevo valor de w. Se puede simplificar el algoritmo multiplicando la ecuación anterior por $\beta - E\{g'(w^T x)\}$, quedando:

$$w^+ = [E\{xg(w^T x)\} - E\{g'(w^T x)\} \cdot w] \tag{39}$$

2.4.3.5. El algoritmo FastICA

El algoritmo FastICA utiliza la función de contraste G en la forma $g(x) \ = \ tanh(x)$ y se define como sigue:

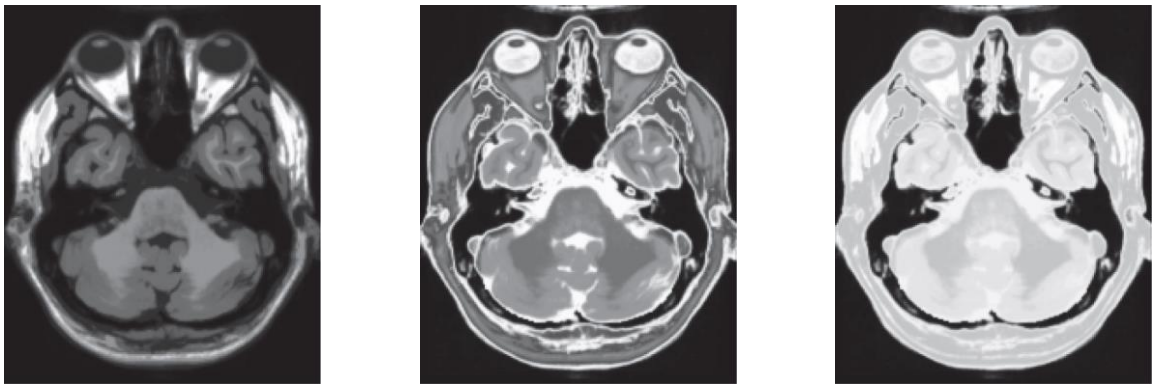
1. Elegir un vector de peso aleatorio *w* como vector inicial.
2. Actualizar la matriz de desmezclado W mediante la expression $w_{n+1} = [E\{xg(w_n^T x)\} - E\{g'(w_n^T x)\} \cdot w_n]$ donde $g(x) \ = \ tanh(x)$.
3. Normalizar los pesos de la matriz mediante $w_{n+1} = w_{n+1} / \|w_{n+1}\|$
4. Actualizar $n \leftarrow n + 1$ y volver al paso 2 si $\|w_{n+1} - w_n\| \geq \varepsilon$, donde $\varepsilon \geq 0$ es una constante de valor muy pequeño.

3 EL MÉTODO

"Einstein se equivocaba cuando decía que 'Dios no juega a los dados con el universo'. Considerando las hipótesis de los agujeros negros, Dios no solo juega a los dados con el universo: a veces los arroja donde no podemos verlos".

- Stephen Hawking-

Como bien se mencionaba en el Estado del Arte de este documento, el objetivo de este trabajo es desarrollar un algoritmo de segmentación de tejidos cerebrales a partir de imágenes de resonancia magnética o MRI (*Magnetic Resonance Imaging*). Para realizar dicho estudio, se han tomado como imágenes las recogidas en la base de datos web *BrainWeb* [37]. Estas imágenes no son imágenes de cerebros reales, sino imágenes simuladas de volúmenes cerebrales, ampliamente utilizadas para hacer este tipo de estudios. En la siguiente figura se puede observar un corte cerebral de imagen de resonancia magnética en sus tres variantes T1-Weigthed, T2-Weighted y PD-Weighted.



(a) (b) (c)

Figura 12. Corte de imágenes MRI cerebrales T1-weighted, T2-weighted y PD-weighted.

Para hacer el estudio sobre las distintas técnicas de segmentación se han tomado tres volúmenes de cerebros simulados procedentes de la base de datos *BrainWeb*. De estos volúmenes (compuestos por 181 cortes o secciones cerebrales) se ha extraído el corte 94, por ser un corte central y en el que se observan mejor los tejidos objeto de la segmentación. Este corte, de tamaño

181x217 tendría un grosor de los tejidos de 1 mm, 0% de ruido y 0% de *intensity nonuniformity ("RF").*

3.1. La Imagen por Resonancia Magnética Nuclear

Para conocer en profundidad como se capturan estas imágenes de forma real, a continuación, se introduce el procedimiento de generación de las imágenes por resonancia magnética.

La Imagen por Resonancia Magnética (MRI, *Magnetic Resonance Imaging*) es una técnica no ionizante con capacidad tridimensional, excelente contraste para los tejidos blancos y alta resolución espacial (~ 1mm). Se basa en la medición de las propiedades magnéticas de los tejidos, en especial de los protones (núcleos de hidrógeno *H*) presentes en el agua y, en menor medida, en los lípidos.

El paciente es colocado dentro de un potente imán que produce un campo magnético estático muy intenso (1-3 Teslas) y se aplica una señal de radiofrecuencia (RF) a través del cuerpo de forma que los protones absorben parte de la energía de la RF al entrar en resonancia magnética. Cuando cesa la señal de RF, la energía absorbida es reemitida en forma de señal de RF. Esta emisión de RF es medida mediante una bobina sintonizada. La técnica de MRI mide el contenido en hidrógeno de los vóxeles (del inglés, *volumetric pixel*) individuales de cada corte transversal del paciente. Los átomos de hidrógeno *H* producen una señal debido a su resonancia magnética. La información espacial se obtiene al utilizar gradientes de campo magnético en las tres dimensiones, es decir, las frecuencias de resonancia de cada vóxel varían linealmente con su localización.

Los núcleos con un número impar de protones presentan la propiedad de la resonancia magnética:

- Cada protón es una carga positiva que gira alrededor de un eje produciendo un pequeño bucle de corriente que genera un dipolo magnético.
- Este dipolo tiene un momento magnético *m*, representado por un vector apuntando de N a S (ver Figura 13).

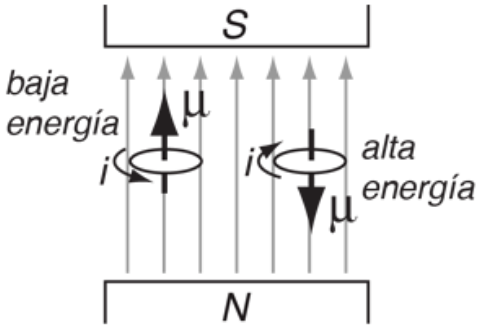


Figura 13. Momento Dipolar Magnético. [38]

En ausencia de un campo magnético, los núcleos apuntan en direcciones aleatorias, es decir, efecto magnético neto es cero. En presencia de un campo eléctrico *B*, los dipolos magnéticos se alinean con un cierto ángulo respecto a la dirección paralela o antiparalela al campo magnético.

Debido al menor coste energético que tiene alinearse en paralelo al campo, un número algo superior de dipolos se orientan en esta dirección (aprox. 3 ppm (*partes por millón*)). Esta pequeña diferencia de orientaciones provoca un momento magnético neto M_z en la dirección de *B.* La MRI sólo puede detectar el exceso de dipolos en esta orientación, no el número total de protones.

Los dipolos sometidos a un campo magnético estático sufren un movimiento de precesión orientado hacia la dirección del campo. comienzan a girar alrededor del eje que define la dirección del campo con una frecuencia fija (frecuencia de Larmor [39]).

El vector magnético *m* de cada dipolo se descompone en:

- Una componente longitudinal $m_z$que apunta a la dirección Z.
- Una componente transversal m_{xy} que gira en el plano XY.

Considerando todos los protones detectables de un vóxel, vectores m_z se suman formando una magnetización longitudonal neta M_z, mientras que los vectores m_{xy} se cancelan entre sí.

Resonancia Magnética: Al someter a un dipolo a una señal de RF a la frecuencia de Larmor, se le aplica justo la energía requerida para cambiar su orientación de paralelo a antiparalelo.

Mientras se mantiene la señal de RF, más dipolos cambian de orientación, girando el momento magnético neto *M*.

- **Pulso de 180º**: Se aplica un pulso de RF con una duración y energía tal que produce un giro completo del momento magnético *M*, invirtiendo su componente M_z.

- **Pulso de 90º:** Se aplica un pulso de RF con una duración y energía tal que produce un giro de 90º del momento magnético *M*, cancelando su componente longitudinal M_z y creando un momento magnético transversal M_{xy} perpendicular a *B* y que gira en el plano XY a la frecuencia de Lamor.

Cuando el pulso de 90º cesa, el vector magnético M_{xy} continúa girando en el plano transversal XY durante un tiempo. Este giro induce una señal de RF en una bobina a la frecuencia de resonancia. Mediante técnicas de codificación espacial y procesado de la señal se pueden identificar las señales procedentes de cada vóxel de la matriz de adquisición. Sólo el componente M_{xy} produce la señal medible, pero su valor es proporcional al valor de M_z previo al pulso.

Inmediatamente después del pulso de 90º la señal de MR es máxima. A partir de ese instante los

dipolos comienza a volver a su orientación original. Como consecuencia: M_z empieza a crecer hasta su valor original, M_{xy} decae, reduciéndose proporcionalmente la amplitud de la señal de RF inducida. Sin embargo, la frecuencia permanece igual. El vector del momento magnético *M* es la suma de sus componentes longitudinal y transversal. Mientras M_z aumenta y M_{xy} decrece, el vector *M* se mueve en espiral del plano transversal al longitudinal.

Esto desencadena en el conocido fenómeno de relajación, o *free inducction delay,* se debe a dos procesos independientes:

- Relajación ***spin–lattice*** (recuperación T1): Los protones excitados van cediendo su exceso de energía a la estructura molecular (lattice) y se vuelven a alinear en paralelo al eje Z. La componente M_z reaparece lentamente de manera exponencial con una constante de tiempo T1.

- Relajación ***spin–spin*** (decaimiento T2): Los núcleos, que estaban girando en el plano XY de manera coherente (en fase), empiezan a cederse energía entre ellos, perdiendo la coherencia de fase. La componente M_{xy} decae rápidamente de manera exponencial con una constante de tiempo T2.

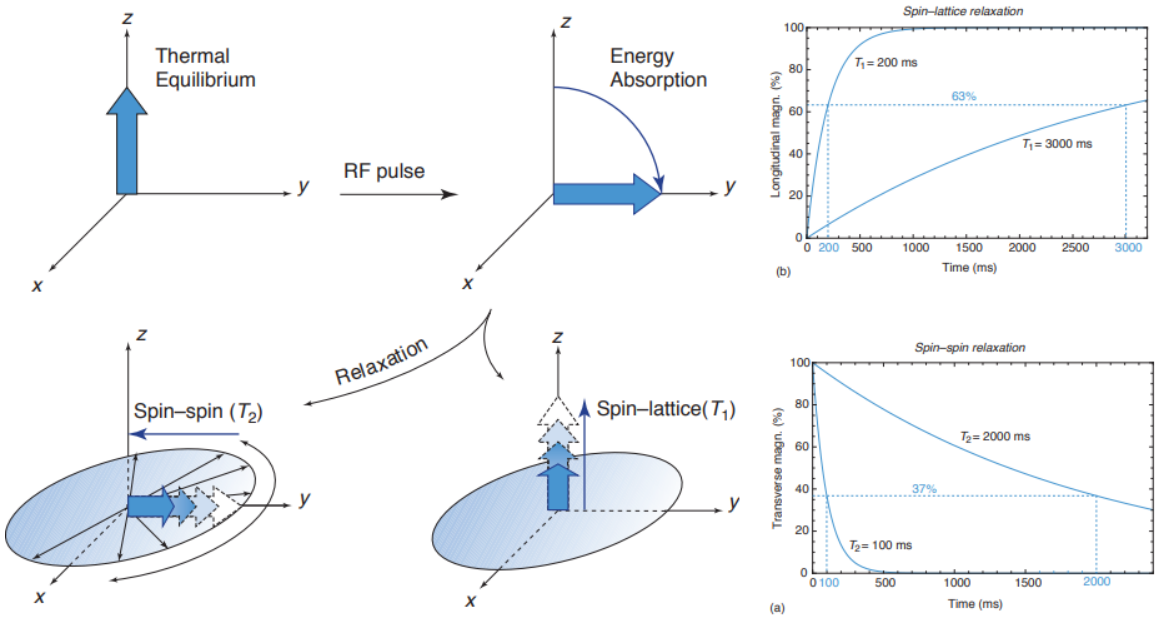


Figura 14. Fenómenos de relajación *spin-lattice* y *spin-spin* [40].

El valor de pico de la señal inducida es proporcional a:

- La densidad de protones (*PD*, el número de protones por mm^3) del vóxel.
- La ratio giromagnética del núcleo.
- La intensidad del campo magnético estático *B*

Sólo los protones móviles generan señales:

- Los que forman parte de moléculas complejas o están inmovilizados en el hueso no contribuyen a la señal.
- La mayor parte de la señal procede del agua corporal.
- El aire atrapado no produce ninguna señal.
- La grasa tiene un PD algo superior al resto de tejidos blandos.
- La materia gris tiene un PD algo superior a la materia blanca.
- Los tejidos blandos en general tienen una PD similar.

3.2. El método

Retomando de nuevo el método, se podría hacer una división de este en varias fases, para poder ver de forma más clara dicho proceso de segmentación. Se subdivide, por tanto, el capítulo en los distintos subapartados de los que se compondrá dicho procedimiento. A continuación, se enumera cada una de las etapas del método:

- Fase de pre-procesado
- Fase de segmentación
- Fase de análisis de los resultados

Figura 15. Fases del algoritmo de segmentación implementado

3.2.1. Pre-procesado

En primer lugar, tomando las imágenes de la base de datos de tipo T1, T2 y PD, se aplica un pre-procesado a las imágenes para su posterior segmentación en los distintos tejidos de interés para el estudio.

Se dividirá en dos partes claramente diferenciales: eliminación del cráneo y centrado y blanqueado de los datos (se entiende por datos los pixeles de cada imagen). En la Figura 16 se muestra un diagrama de esta primera fase de pre-procesamiento.

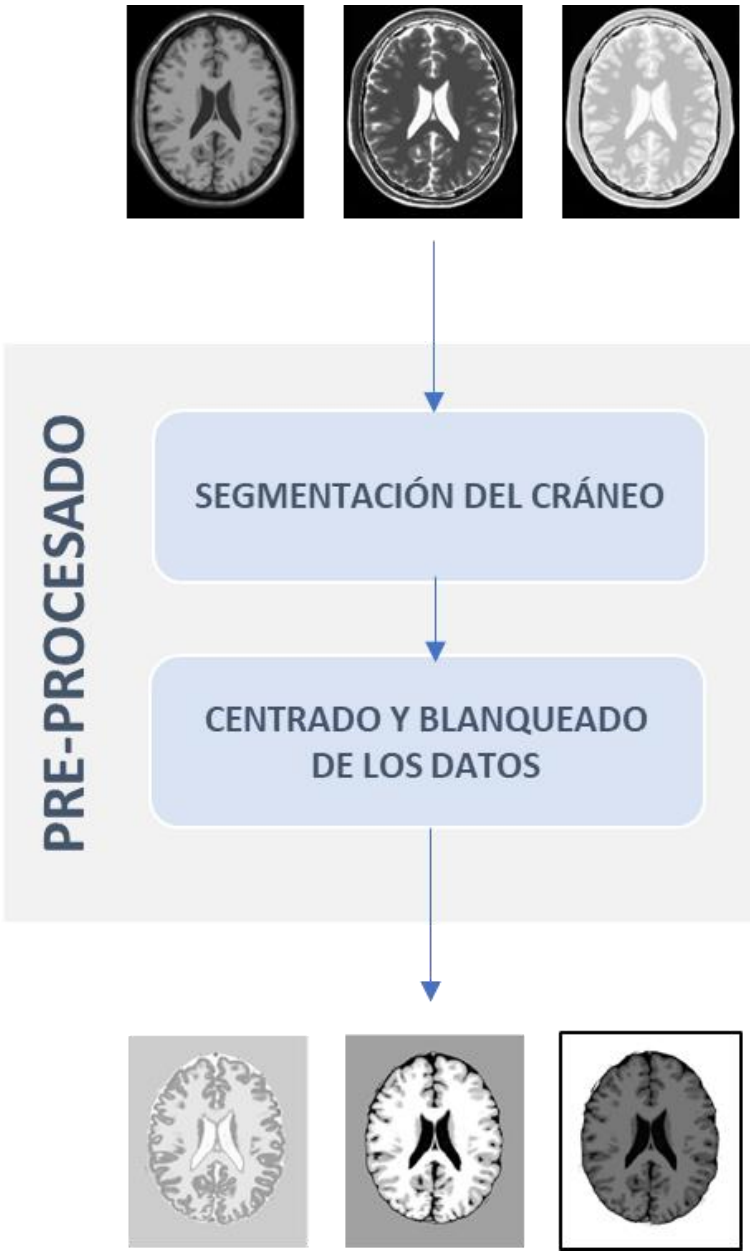


Figura 16. Esquema de la fase de pre-procesado

La segmentación del cráneo es un paso esencial en la mayoría de los análisis cuativativos de la estructura cerebral. Esta segmentación consiste en eliminar tanto la estructura ósea del cráneo como las meninges, que son las membranas de tejido conectivo que cubren todo el sistema nervioso central, añadiéndole una protección blanda que complementa a la dura de las estructuras óseas.

Para realizar esta tarea, se calcula un umbral que separará los tejidos cerebrales de los tejidos no cerebrales acorde con el histograma de la imagen en escala de grises. Con este umbral se transforma la imagen en una imagen binaria, es decir, se binariza la imagen. Para eliminar objetos falsos aislados y huecos que queden como resultado de esta binarización, se aplican

operaciones morfológicas obteniendo una nueva imagen binarizada.

Para eliminar los píxeles blancos del fondo que no son de interés se realiza una operación morfológica de erosión. El último paso es construir una máscara que servirá para eliminar el cráneo de la imagen original. Por otro lado, se puede obtener el cráneo segmentado aplicando la máscara inversa. En resumen, los pasos a seguir serían los siguientes:

- **Paso 1.** Tomar el volumen de entrada y seleccionar la *slice* 94.
- **Paso 2.** Rotar la imagen 180 grados
- **Paso 3.** Reescalar la imagen para que sus valores estén entre 0 y 255.
- **Paso 4.** Pasar la imagen a valores *uint8*.
- **Paso 5.** Seleccionar un valor umbral y pasar la imagen a binaria.
- **Paso 6.** Realizar operaciones morfolóficas para rellenar los huecos. Con esto ya se tiene la máscara para segmentar el tejido cerebral.
- **Paso 7.** Calcular la máscara complementaria. Se obtiene entonces la máscara para segmentar el cráneo.
- **Paso 8.** Aplicar ambas máscaras a la imagen original y segmentar el cerebro y el cráneo.

En los Anexos se adjunta el código correspondiente a este procedimiento. A continuación, se muestra un diagrama de los pasos descritos.

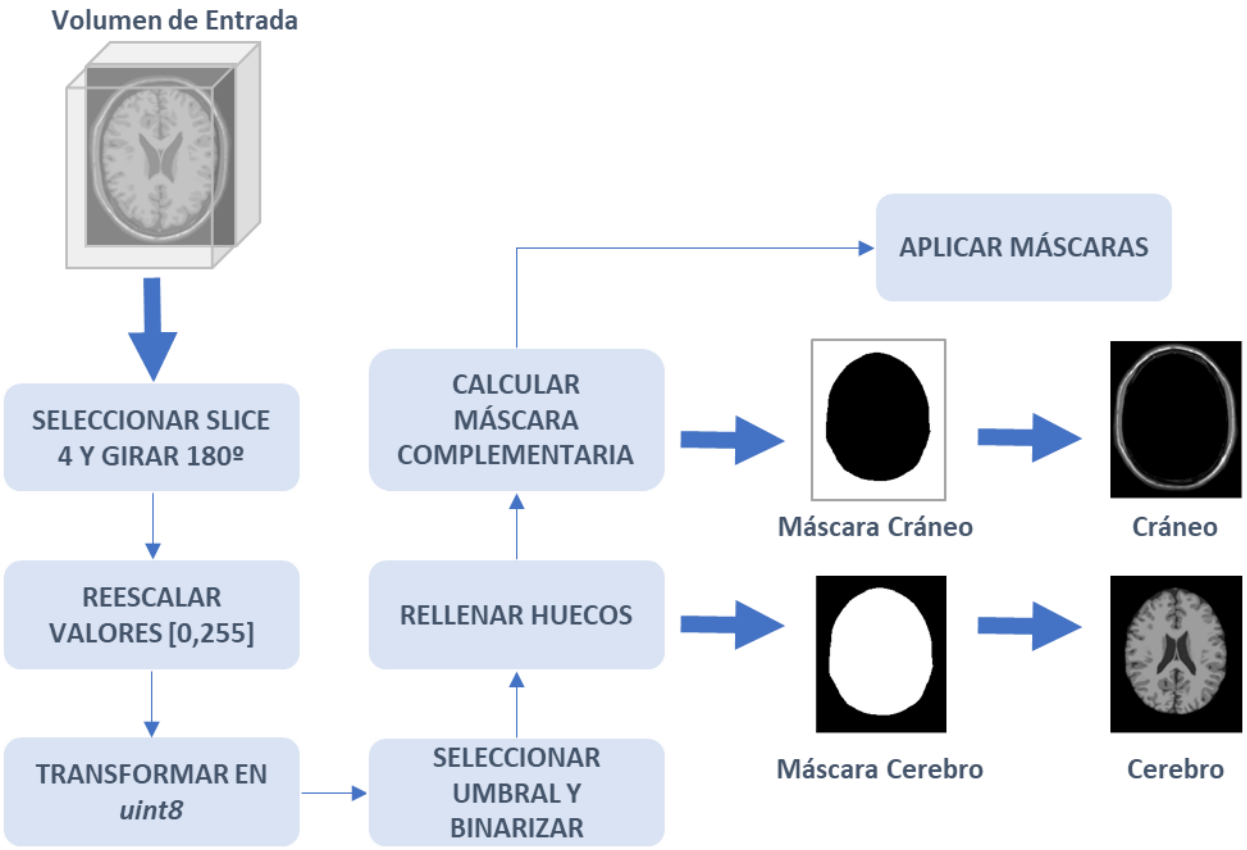
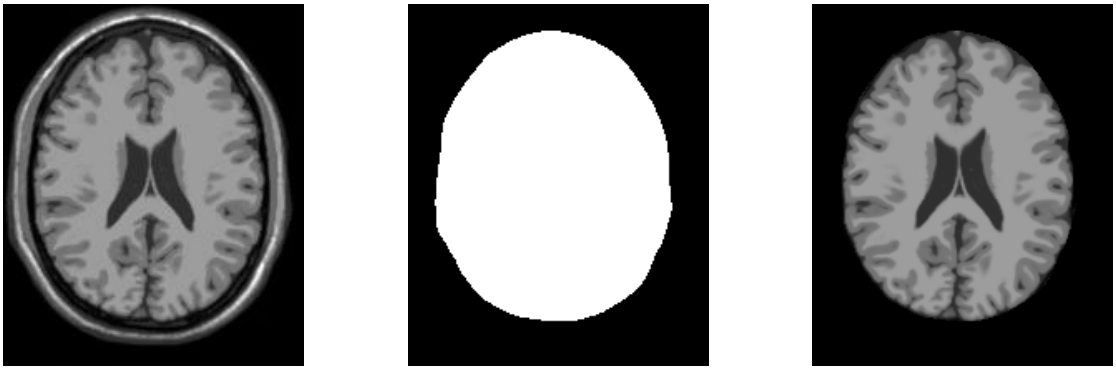


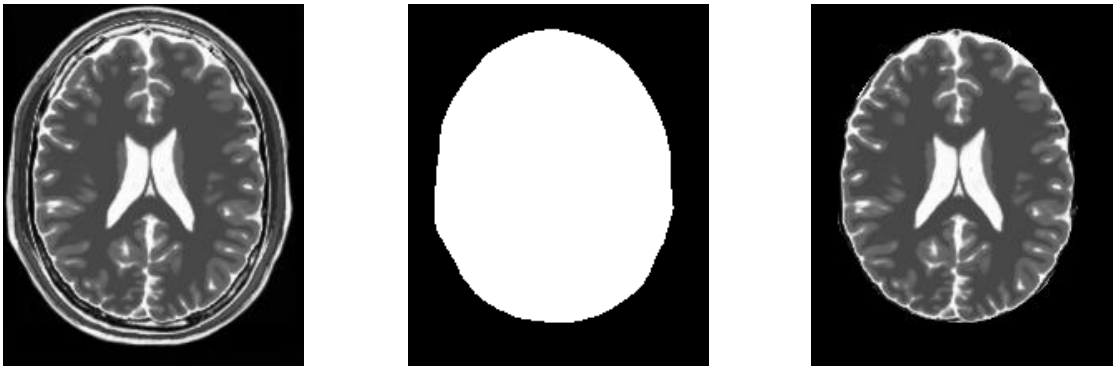
Figura 17. Pre-procesado. Procedimiento de Eliminación del Cráneo

En las siguientes figuras se pueden observar los resultados de aplicar el proceso de la eliminación del cráneo a los tres tipos de imágenes multimodales (T1-weighted, T2-weighted y PD-weighted, respectivamente).



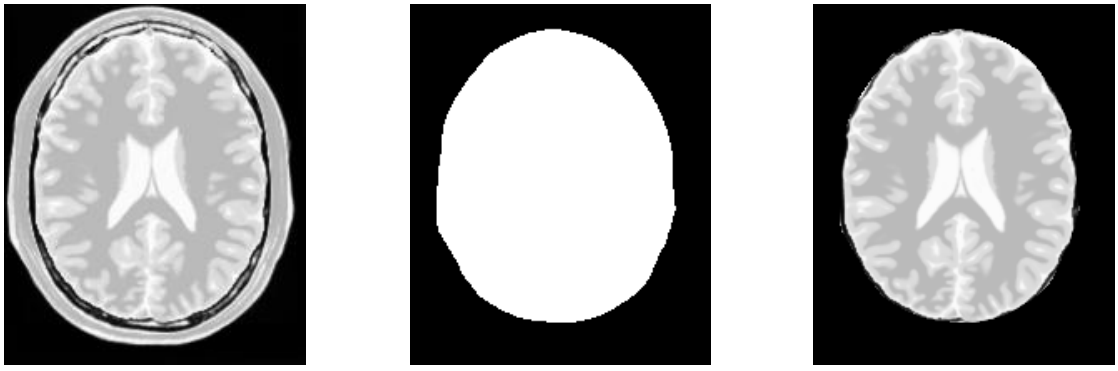
(a) (b) (c)

Figura 18. Eliminación del cráneo en Imagen T1-weigthed: (a) Imagen Original, (b) Máscara, (c) Imagen segmentada.



(a) (b) (c)

Figura 19. Eliminación del cráneo en Imagen T2-weigthed: (a) Imagen Original, (b) Máscara, (c) Imagen segmentada.



(a) (b) (c)

Figura 20. Eliminación del cráneo en Imagen PD-weigthed: (a) Imagen Original, (b) Máscara, (c) Imagen segmentada.

Una vez se tiene el cráneo segmentado en las imágenes T1-weighted, T2-weighted y PD-weighted hay que pasar a aplicar el centrado y el blanqueado de los datos. Para ello, como es costoso trabajar con las imágenes en forma matricial y por separado, lo primero es almacenar en una variable *X* estos datos para su posterior procesamiento. Por tanto, se define la variable *X* como el set de imágenes o señales observadas representadas por x_n con n=1, 2 y 3, respectivamente.

$$X = [x_1, x_2, x_3]^T \tag{40}$$

El proceso de centrado de los datos es tan sencillo como restar a la variable *X* el valor de su media $E[X]$.

$$X = [x_1, x_2, x_3]^T \rightarrow \oplus \rightarrow X_{centrado} = X - X_{media}$$
$$\uparrow -$$
$$X_{media} = E[X]$$

Figura 21. Centrado de los datos X

Tras realizar el centrado, hay que completar el proceso de preprocesado con el blanqueado de los datos. Para ello se emplea sobre los datos centrados la función *fastica,* en Matlab, en su modalidad de blanqueado de datos, esto es:

```
%Blanqueado de los datos
[whitesig, wm, dwm] = fastica(X, 'only', 'white');

%Reajuste de los datos para mostrar en las 3 imágenes multimodales
ws1=reshape(whitesig(1,:),M,N);
ws2=reshape(whitesig(2,:),M,N);
ws3=reshape(whitesig(3,:),M,N);
```

El proceso de blanqueado de datos se detalla en el Capítulo 2, en la sección 2.4.3. titulada *Construcción del método ICA.* Los resultados de aplicar este blanqueado a las tres imágenes multimodales se muestran a continuación, en la siguiente figura.

(a) (b) (c)

Figura 22. Imágenes Blanqueadas: (a)T1-Weigthed, (b) T2-Weigthed y (c) PD-Weighted

3.2.2. Análisis de Componentes Independientes

Las señales a la salida del proceso serán almacenadas en la variable S, descrita por S_p con p=1, 2 y 3, respectivamente. Cada una de ellas representará un tejido cerebral: WM (*White Matter* o materia blanca), GM (*Gray Matter* o materia gris) y CSF (*CerebroSpinal* Fluid o líquido cefalorraquídeo).

$$S = [s_1, s_2, s_3]^T \tag{41}$$

La relación entre las imágenes X y los tejidos cerebrales S se pueden representar mediante la siguiente expresión:

$$X = AS \tag{42}$$

Donde A es la matriz de mezclado y tiene dimensiones 3x3. Si se tiene la capacidad de obtener la matriz A, entonces los tejidos cerebrales pueden extraerse de la siguiente forma:

$$S = A^{-1}X \tag{43}$$

Sin embargo, la matriz A es todavía desconocida. Se puede conseguir una aproximación de la matriz A mediante el análisis de la independencia estadística y mediante operaciones matemáticas. Con la matriz inversa $W = A^{-1}$ (conocida como matriz de separación) se puede predecir una aproximación de S, denotada como $\hat{S}$, que representaría la aproximación de los tejidos cerebrales. Se puede expresar con la siguiente expresión matemática:

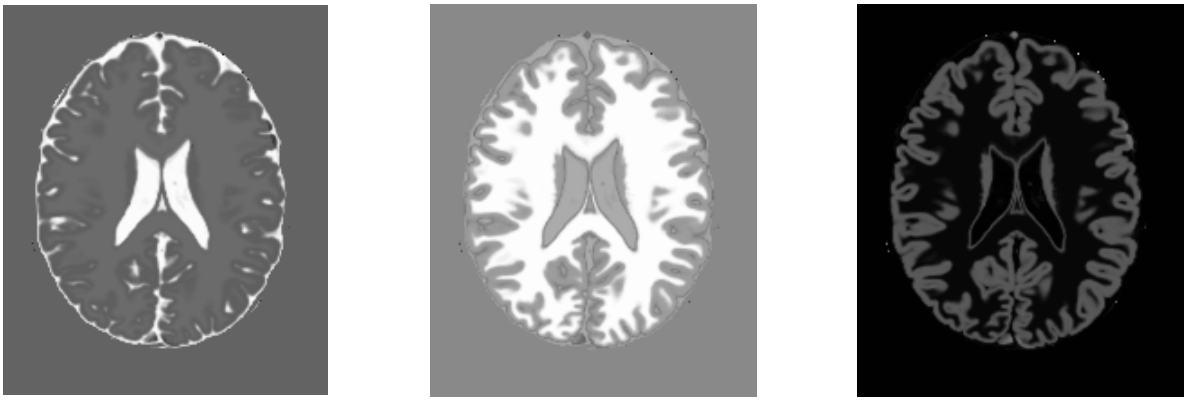
$$S \simeq \hat{S} = WX \tag{44}$$

En este trabajo se emplea el algoritmo FastICA para calcular la matriz W y poder analizar más a fondo cada píxel en la imagen multiespectral para una correcta clasificación del tejido cerebral.

Ya se menciona en el Capítulo anterior que FastICA [12]- [18] utiliza un método de descenso de gradiente para maximizar la medida de No-gaussianidad, acelerando así la estimación de los componentes independientes de las señales fuente.

Este resultado será utilizado en la siguiente fase del método para poder estudiar los casos de clasificación mediante las técnicas K-Means, FCM y KFCM con y sin ICA, pudiendo analizar si los resultados son mejores o peores en función de si se aplica previamente ICA o no. En la siguiente

figura se puede observar el resultado de aplicar ICA a los datos, donde se aprecian en forma de imagen, las tres componentes independientes obtenidas a partir de las tres imágenes multimodales T1-weighted, T2-weighted y PD-weighted.



(a) (b) (c)

Figura 23. Análisis de componentes independientes. (a), (b) y (c) son las tres componentes independientes resultado de aplicar FastICA al conjunto de datos blanqueados.

Las tres componentes independientes se corresponden con los tejidos WM (*White Matter* o materia blanca), GM (*Gray Matter* o materia gris) y CSF (*CerebroSpinal Fluid* o líquido cefalorraquídeo). Estas imágenes no dan mucha información a simple vista, pero son de alta importancia para los siguientes pasos del método, necesarios para poder aplicar ciertas técnicas de segmentación.

En los Anexos se recoge el código correspondiente a este apartado de análisis de componentes independientes.

3.2.3. Segmentación de tejidos cerebrales

Teniendo en cuenta todo lo explicado anteriormente, sobre las imágenes pre-procesadas (con o sin ICA) se aplican los siguientes métodos de clasificación:

- *K-Means*
- FCM, *Fuzzy C-Means*
- KFCM, *Kernelized Fuzzy C-Means*

Primero se aplican todos los métodos a las imágenes pre-procesadas sin aplicar ICA y posteriormente se aplican los mismos métodos a las imágenes después de haberlas pasado por la función ICA. Por tanto, tenemos 6 segmentaciones diferentes que posteriormente habrá que estudiar para valorar cuál de ellas es la que tiene un mejor resultado. Si se quiere estudiar la implementación de estos algoritmos en Matlab, en los Anexos se recoge el código correspondiente a este apartado del método.

4 RESULTADOS

Una persona ingeligente resuelve un problema.
Una persona sabia lo evita.

- Albert Einstein -

En este Capítulo se mostrarán los resultados obtenidos después de aplicar tres métodos de segmentación: K-Means, FCM y KFCM aplicado directamente a los datos en crudo o aplicado a los datos después de aplicarles el algoritmo FastICA, de manera que en el primer caso las imágenes son variables dependientes unas de otras y en el segundo caso se trabaja con componentes independientes.

También se realizará un análisis de estos resultados, donde se determinará qué método es el más eficiente para segmentar esta clase de imágenes médicas. Para ello se calcularán el grado de similitud entre la imagen resultado de la segmentación y su verdad de referencia tomada de la base de datos *BrainWeb*.

El primer resultado que se va a mostrar es el que se obtiene de aplicar al corte central (numero 94) el pre-procesado y los distintos algoritmos con y sin FastICA a las imágenes de entrada. Posteriormente, como estudio adicional, se han hecho pruebas para distintos cortes, que no sean el corte central del volumen cerebral. De esta manera podemos hacer un estudio más completo de los resultados de la clasificación.

En la Figura 24, de (a) hasta (c) se muestran las imágenes originales T1-weighted, T2-weighted y PD-weighted. De (e) hasta (f) se pueden observar los resultados de las imágenes sin la región del cráneo, tras pasar por el proceso de segmentación de la fase de pre-procesado. De (g) hasta (i) los resultados de las imágenes tras centrar y blanquear los datos y aplicar la extracción de componentes independientes mediante el algoritmo FastICA.

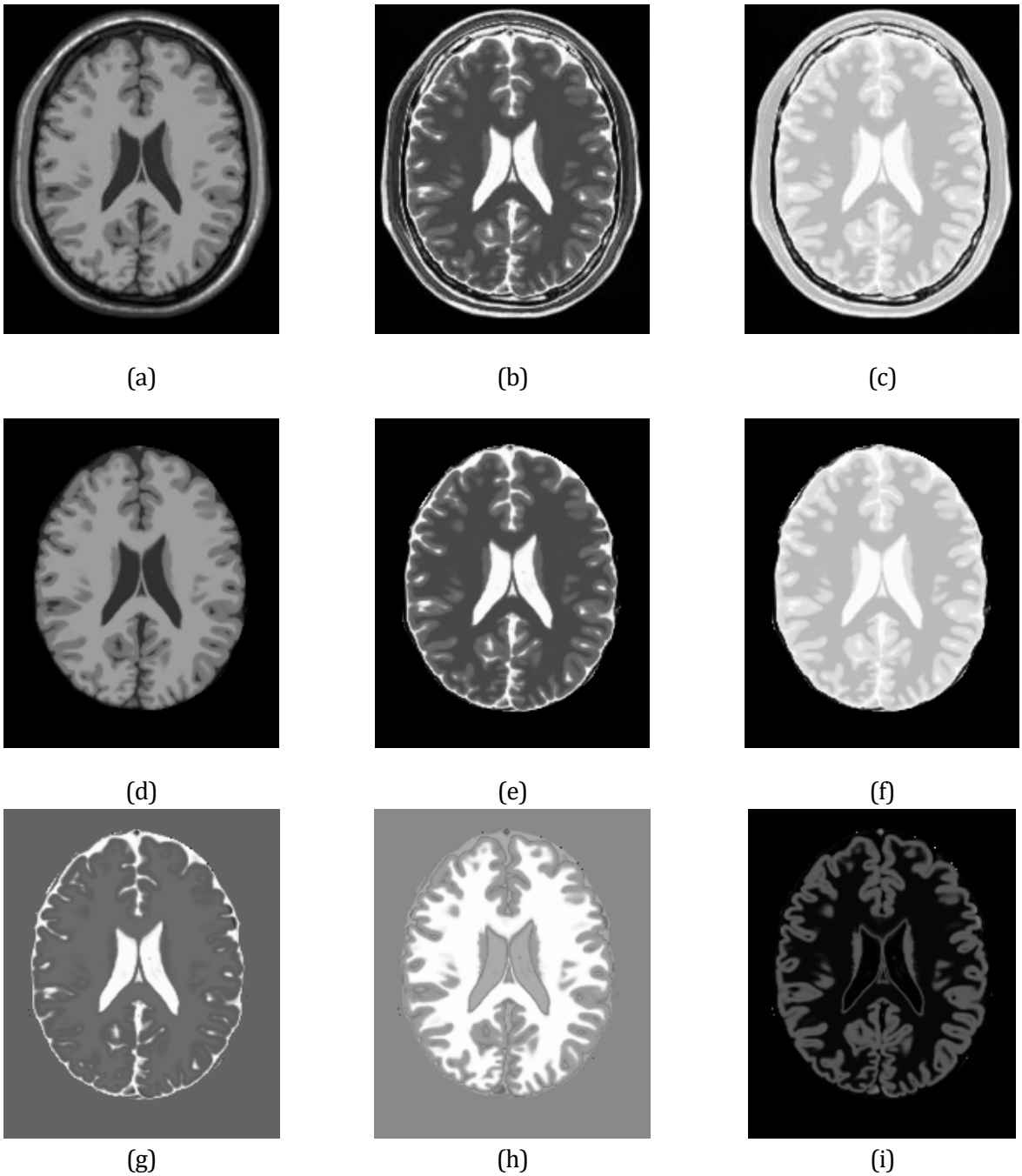
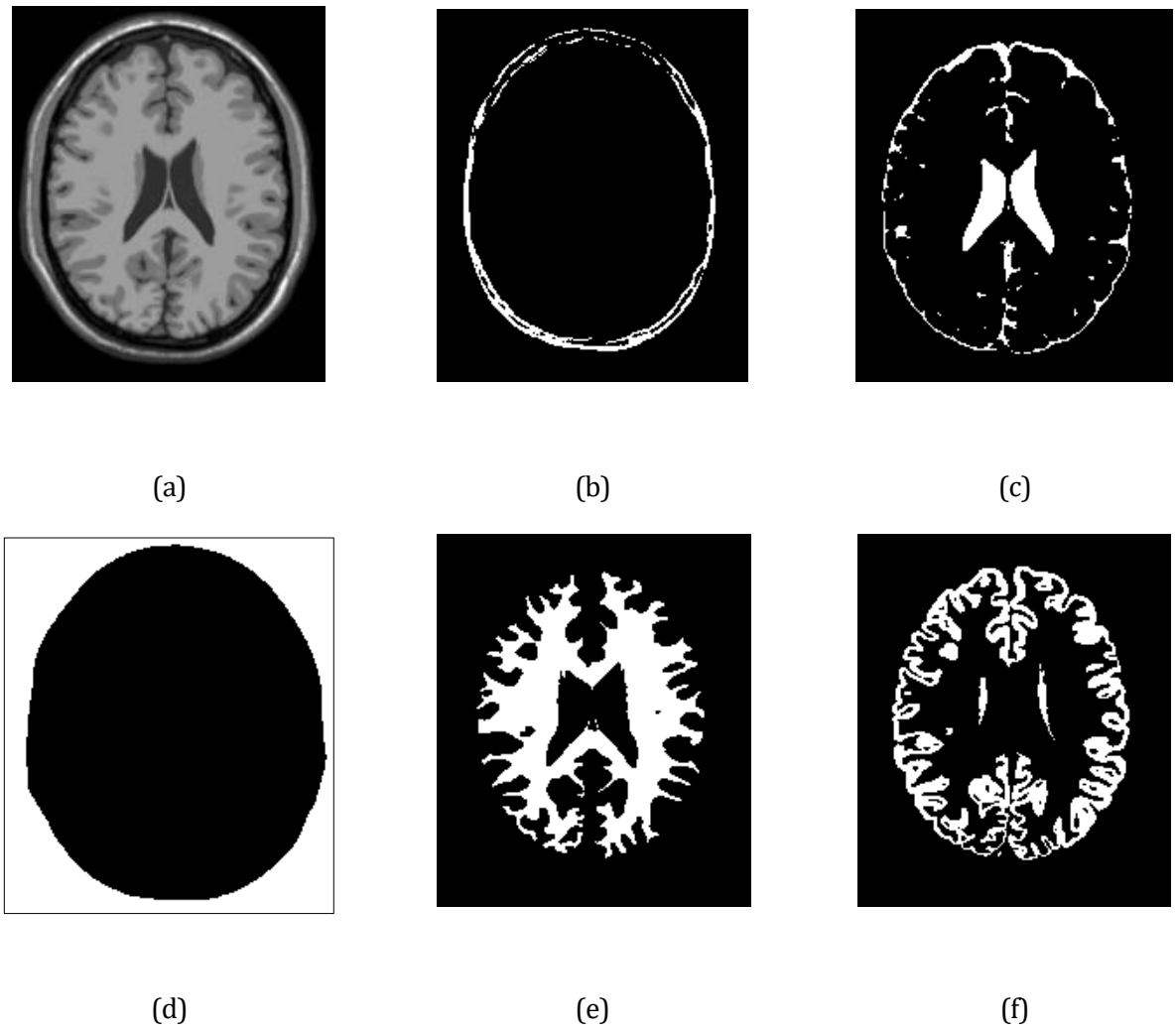


Figura 24. Componentes Independientes extraídas mediante algoritmo FastICA.

Antes de mostrar los resultados de la segmentación sería interesante conocer cuáles son las imágenes o verdades de referencia en las que se va a basar el estudio de viabilidad de esta segmentación. Las verdades de referencia son las que delimitan de forma fiable (lo que un médico podría indicar) los distintos tejidos. En la siguiente figura se observan las verdades de referencia de los distintos tejidos.



(a) (b) (c)

(d) (e) (f)

Figura 25. Verdades de Referencia

En (a) se observa la imagen T1-weigthed original. (b) corresponde con el cráneo, (c) es el líquido cefalorraquídeo, (d) es el fondo, (e) la materia blanca y (f) la materia gris. Con estos datos ya podemos analizar tanto a simple vista como cuantitativamente la calidad de los resultados de las segmentaciones que se mostrarán a continuación.

En las siguientes figuras se van a mostrar los resultados de segmentar, con los distintos métodos indicados anteriormente, las imágenes MRI cerebrales mencionadas anteriormente, no aplicando el método FastICA, en primera instancia, y, posteriormente, aplicando el método FastICA para estudiar así las diferencias.

4.1. Resultados obtenidos para los distintos métodos sin aplicar ICA

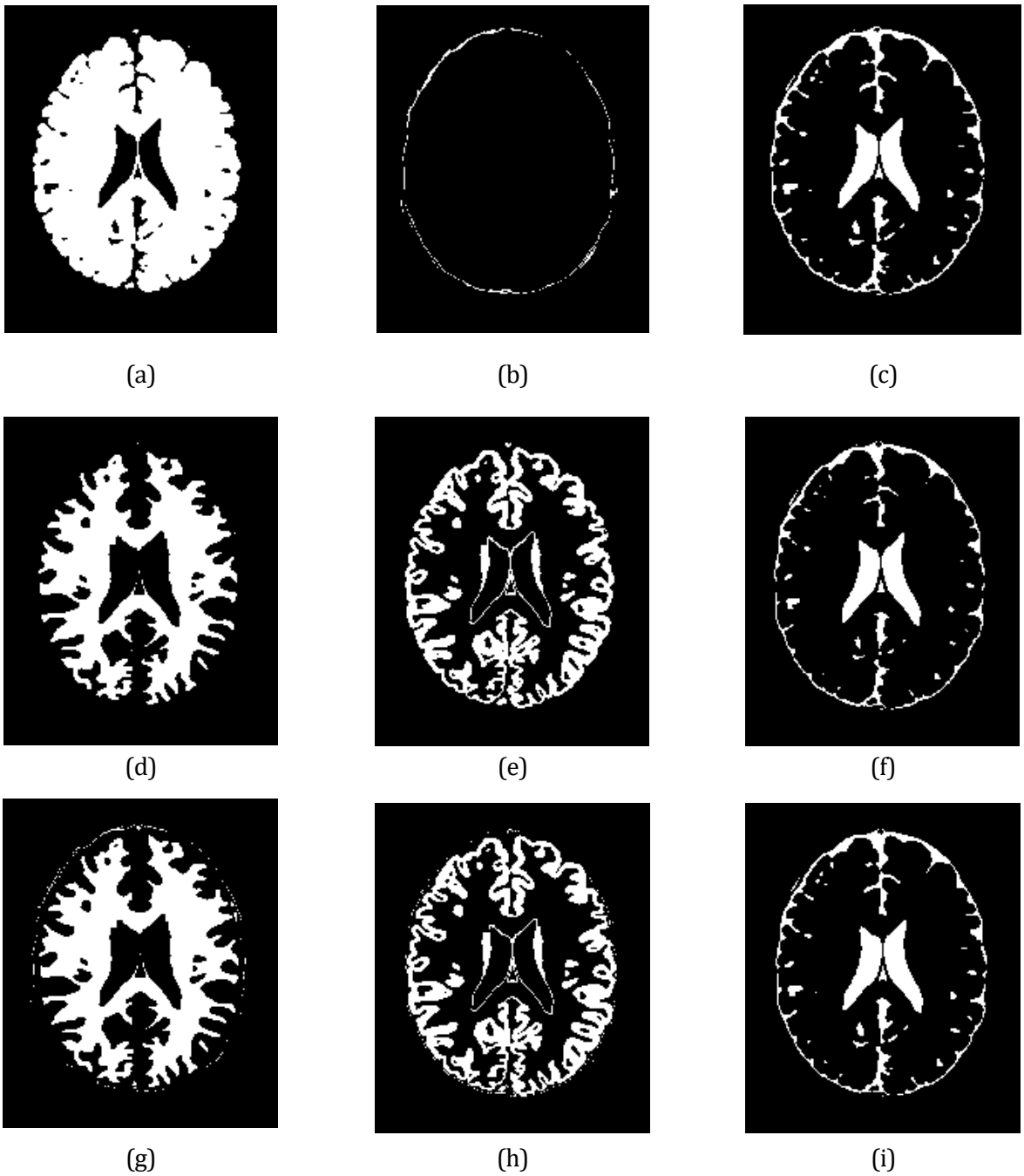
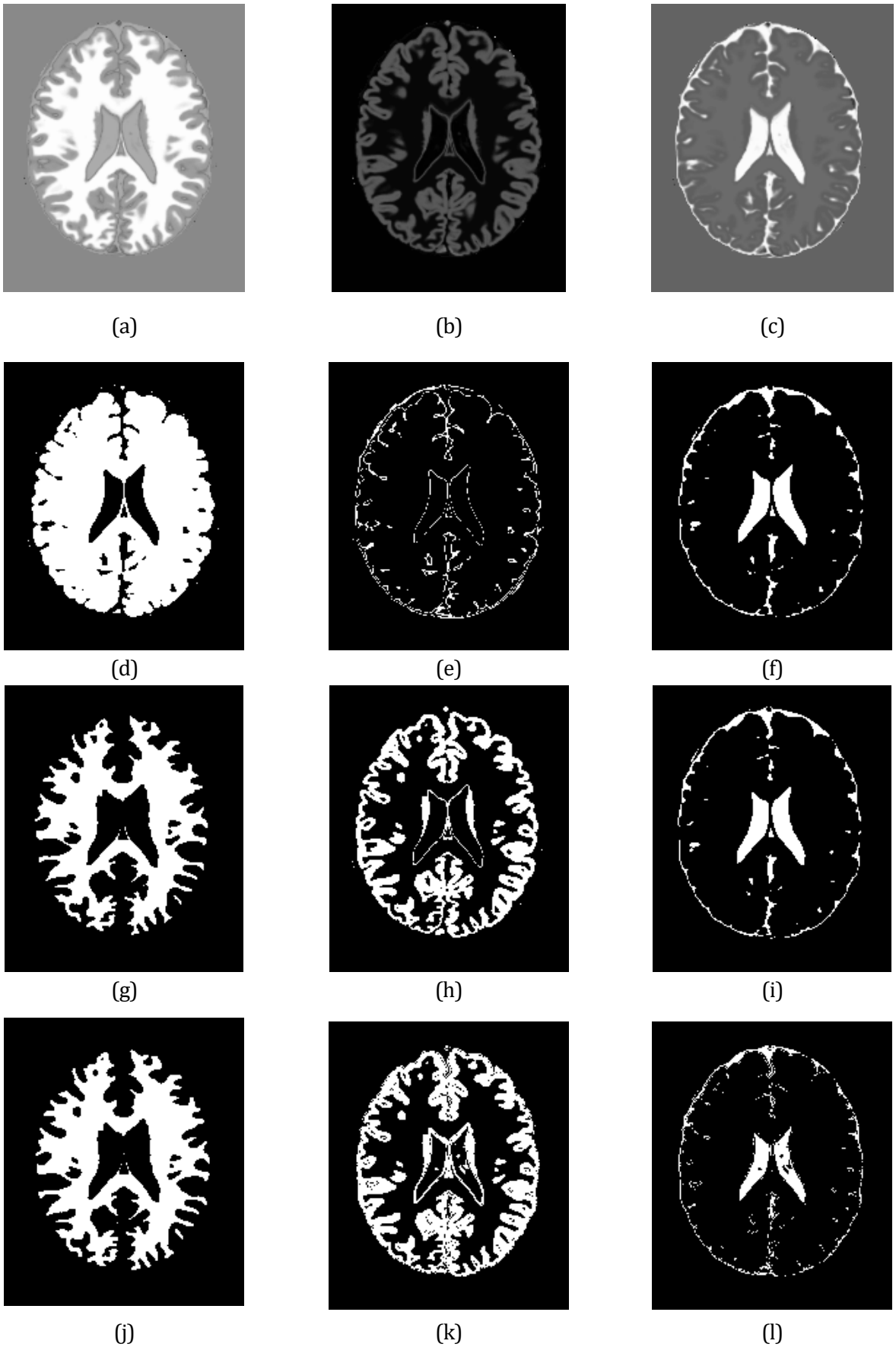


Figura 26. Resultados de la segmentación con los distintos métodos y sin aplicar ICA.

De la (a) a la (c) los resultados de la segmentación aplicando *K-Means*. Se puede observar que no hace una buena segmentación de los tejidos que se están buscando. De la (d) a la (f) los resultados de la segmentación aplicando el algoritmo FCM (*Fuzzy C-Means*). Ahora sí se aprecian, respectivamente, la materia blanca, la materia gris y el líquido cefalorraquídeo o CSF. De la (g) a la (i) se muestran los resultados de la segmentación tras aplicar el algoritmo KFCM (*Kernelized Fuzzy C-Means*). En este caso se vuelve a observar una buena clasificación.

4.2. Resultados obtenidos para los distintos métodos aplicando ICA



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figura 27. Resultados de la segmentación con los distintos métodos y aplicando ICA.

De la (a) a la (c) las tres componentes independientes resultantes de aplicar FastICA a las imágenes pre-procesadas. De la (d) a la (f) los resultados de la segmentación aplicando el algoritmo K-Means con ICA. De la (g) a la (i) los resultados de aplicar FCM (*Fuzzy C-Means*) a estas componentes. Se aprecian, respectivamente, la materia blanca, la materia gris y el líquido cefalorraquídeo o CSF. De la (j) a la (l) se muestran los resultados de la segmentación tras aplicar el algoritmo KFCM (*Kernelized Fuzzy C-Means*). En este caso se vuelve a observar una buena clasificación. A continuación se presentará un análisis de calidad de los resultados obtenidos.

4.3. Estudio de los Resultados

Una vez se ha hecho un análisis preliminar de forma visual, es momento de pasar al análisis cuantitativo de los resultados. Para la valoración cuantitativa de los resultados se hará uso de tablas de confusión. Las tablas de confusión recogen información acerca de las predicciones de la siguiente forma [41]:

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Tabla 1. Modelo Tabla de Confusión

La observación sería lo correspondiente a las imágenes de las verdades de referencia, es decir, los distintos tejidos cerebrales. La prección serían las segmentaciones realizadas de cada uno de estos tejidos. Se van a analizar los tejidos por separado y se recogerán los resultados en una tabla resumen. Los términos recogidos en la tabla de confusión son los siguientes [41]:

- **VP (Verdaderos Positivos):** este término se conoce en inglés como TP (*True Positive*). La observación es de un tejido en concreto y la predicción lo cataloga como tal.

- **FP (Falsos Positivos):** en inglés es conocido como FP (*False Positive*). La observación no pertenece al tejido objeto de estudio y la predicción lo cataloga como tejido.

- **FN (Falsos Negativos):** en inglés se conoce con el mismo término FN (*False Negative*). La observación pertenece al tejido y la predicción no lo cataloga como tejido.

- **VN (Verdaderos Negativos):** del inglés TN (*True Negative*). No es tejido y la predicción tampoco lo identifica como tal.

Partiendo de estos valores de las tablas de confusión se pueden calcular una serie de parámetros de calidad muy utilizados para medir la calidad de los algoritmos de clasificación de imágenes médicas. Estos parámetros de calidad se definen a continuación [41]:

- **TPR (Sensibilidad, *Sensitivity* o *True Positive Rate***): es la relación entre el número de píxeles segmentados por el algoritmo que, también son segmentados por el experto y la cantidad total de píxeles segmentados por el experto.

$$Sensitivity = \frac{VP}{VP + FN} \tag{45}$$

- **ACC (Precisión o *Accuracy*):** Representa el número de píxeles (%) correctamente identificados.

$$Accuracy = \frac{VP + VN}{VP + FP + VN + FN} \tag{46}$$

- **TNR (Especificidad, *Specificity* o *True Negative Rate*):** Porcentaje de negativos correctamente identificados, respecto a los negativos de la imagen segmentada por un experto.

$$Specificity = \frac{VN}{VN + FP} \tag{47}$$

- **PPV (Valor Predictivo Positivo o *Positive Predictive Value*):** mide la sobre-segmentación calculando la relación entre los píxeles segmentados por el algoritmo que también son segmentados por los expertos y la cantidad de píxeles segmentados.

$$Positive\ Predictive\ Value = \frac{VP}{VP + FP} \tag{48}$$

- **NPV (Valor Predictivo Negativo o *Negative Predictive Value*):** Es la proporción entre los valores negativos indicados por un experto frente a la suma de falsos negativos detectados por el algoritmo y los valores negativos que indica el experto.

$$Negative\ Predictive\ Value = \frac{VN}{VN + FN} \tag{49}$$

- **J (Similitud o *Similarity*)** [42]**:** Se calcula mediante el Coeficiente de Jaccard, que mide el grado de similitud entre dos conjuntos, sea cual sea el tipo de elementos. Dados dos conjuntos, A y B, cada uno de ellos con n valores binarios, el coeficiente de Jaccard es una medida útil de la superposición de los valores que A y B comparten. Los valores de los conjuntos A y B pueden ser 0 o 1. Se definen, por tanto:

    - M_{11} representa el número total de veces que el valor en el conjunto A y el mismo valor en el conjunto B es 1. Se corresponde con el valor VP.

    - M_{01}representa el numero total de veces que el valor en el conjunto A es 0 y el mismo valor en el conjunto B es 1. Se corresponde con el valor FP.

    - M_{10} representa el número total de veces que el valor del conjunto A es 1 y el mismo valor en el conjunto B es 0. Se corresponde con el valor FN.

    - M_{00} represents el número total de veces que el valor del conjunto A y el mismo valor en el conjunto B son ambos igual a 0. Se corresponde con el valor VN.

    La suma de M_{AB} tiene que ser igual a n. Con estas definiciones, se uede calcular el coeficiente de Jaccard como:

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} = \frac{VP}{FP + FN + VP} \tag{50}$$

Ahora bien, una vez definidos todos estos términos, el siguiente paso es mostrar los resultados en función de estos, del tejido segmentado y del método utilizado. En primer lugar, se muestran los resultados tras aplicar el algoritmo K-Means a las imágenes sin aplicar ICA.

4.3.1. Análisis de calidad del método K-Means sin aplicar ICA

	Sensitivity	*Accuracy*	*Specificity*	*PPV*	*NPV*	*Similarity*
White Matter	99,69 %	82,12 %	77,02 %	55,73 %	99,88 %	55,64 %
Gray Matter	0,00 %	83,93 %	98,62 %	0,00 %	84,92 %	0,00 %
CSF	96,61 %	97,13 %	97,17 %	69,68 %	99,76 %	68,01 %
Average	**65,43 %**	**87,72 %**	**90,93 %**	**41,80 %**	**94,85 %**	**41,21 %**

Tabla 2. Análisis de calidad del método K-Means sin aplicar ICA

4.3.2. Análisis de calidad del método FCM sin aplicar ICA

	Sensitivity	*Accuracy*	*Specificity*	*PPV*	*NPV*	*Similarity*
White Matter	95,79 %	96,01 %	96,07 %	87,61 %	98,74 %	84,37 %
Gray Matter	86,07 %	94,39 %	95,85 %	78,42 %	97,51 %	69,59 %
CSF	91,84 %	97,86 %	98,26 %	78,11 %	99,44 %	73,04 %
Average	**91,23 %**	**96,08 %**	**96,72 %**	**81,38 %**	**98,56 %**	**75,66 %**

Tabla 3. Análisis de calidad del método FCM sin aplicar ICA

En los tres métodos se obtiene un buen valor de precisión o accuracy, ya que este valor representa el número de píxeles (%) correctamente identificados e indica que los distintos algoritmos están efectuando una buena clasificación (en los tres casos por encima del 87%), pero no se puede centrar la vista sólo en este resultado.

4.3.3. Análisis de calidad del método KFCM sin aplicar ICA

	Sensitivity	*Accuracy*	*Specificity*	*PPV*	*NPV*	*Similarity*
White Matter	95,31 %	95,83 %	95,98 %	87,32 %	98,60 %	83,72 %
Gray Matter	86,56 %	94,36 %	95,72 %	77,99 %	97,60 %	69,57 %
CSF	95,92 %	97,54 %	97,65 %	73,34 %	99,71 %	71,12 %
Average	**92,59 %**	**95,91 %**	**96,45 %**	**79,55 %**	**98,63 %**	**74,80 %**

Tabla 4. Análisis de calidad del método KFCM sin aplicar ICA

Los porcentajes de similitud o *similarity* también medidas muy importantes. Este valor representa, valga la redundancia, el grado de similitud entre dos conjuntos, sea cual sea el tipo de elementos. Para el caso de K-Means no se obtiene un buen resultado de clasificación del tejido. Ya se podía apreciar visualmente en el apartado 4.1. que no se estaba distinguiendo entre la materia blanca y la materia gris correctamente. Por tanto, se podría decir que el K-Means no será un clasificador definitivo para este tipo de estudio.

Para el caso de FCM y KFCM se tienen mejores resultados. Se puede decir que una similitud por encima del 70% es un buen resultado de clasificación. Se procede, a continuación, al estudio de calidad de los mismos algoritmos tras aplicar el método FastICA a las imágenes resultado del pre-procesado.

4.3.4. Análisis de calidad del método K-Means con ICA

	Sensitivity	*Accuracy*	*Specificity*	*PPV*	*NPV*	*Similarity*
White Matter	99,66 %	82.66 %	77.73 %	56.49 %	99.87 %	56.38 %
Gray Matter	5,89 %	83,01 %	96,51 %	22,83 %	85,41 %	4,91 %
CSF	84,85 %	98,03 %	98,92 %	84,14 %	98,98 %	73,15 %
Average	**63,46 %**	**87,90 %**	**91,05 %**	**54,48%**	**94,75 %**	**44,81 %**

Tabla 5. Análisis de calidad del método K-Means con ICA

4.3.5. Análisis de calidad del método FCM con ICA (ICFCM)

	Sensitivity	*Accuracy*	*Specificity*	*PPV*	*NPV*	*Similarity*
White Matter	94,04 %	96,58 %	97,32 %	91,07 %	98,25 %	86,10 %
Gray Matter	86,55 %	94,68 %	96,11 %	79,57 %	97,60 %	70,81 %
CSF	87,76 %	97,93 %	98,62 %	81,08 %	99,17 %	72,84 %
Average	**89,45 %**	**96,39 %**	**97,35 %**	**83,90 %**	**98,34 %**	**76,58 %**

Tabla 6. Análisis de calidad del método ICFCM

Se aprecian resultados similares al caso anterior, pero mejorados estos dos primeros casos al aplicar ICA. Buenos resultados de precisión o *accuracy* y mejora considerable en los resultados de similitud o *similarity.* De igual forma, no K-Means no sería un buen algoritmo de clasificación y FCM sí.

4.3.6. Análisis de calidad del método KFCM con ICA (ICKFCM)

	Sensitivity	*Accuracy*	*Specificity*	*PPV*	*NPV*	*Similarity*
White Matter	95,51 %	96,11 %	96,28 %	88,17 %	98,66 %	84,67 %
Gray Matter	86,67 %	92,00 %	92,93 %	68,23 %	97,55 %	61,75 %
CSF	62,31 %	96,54 %	98,84 %	78,36 %	97,49 %	53,17 %
Average	**81,49 %**	**94,88 %**	**96,01 %**	**78,25 %**	**97,90 %**	**66,53 %**

Tabla 7. Análisis de calidad del método ICKFCM

Para el caso de KFCM con ICA (ICKFCM) se obtiene una disminución de la similitud del 4% que da a entender que aplicar FastICA no ayudaría a mejorar el resultado de la clasificación de los tejidos. Sin embargo, el porcentaje de accuracy está por encima del 90%, lo que indica que el porcentaje de píxeles bien clasificados es muy elevado. Habrá que hacer una valoración final para ver qué está ocurriendo con este caso.

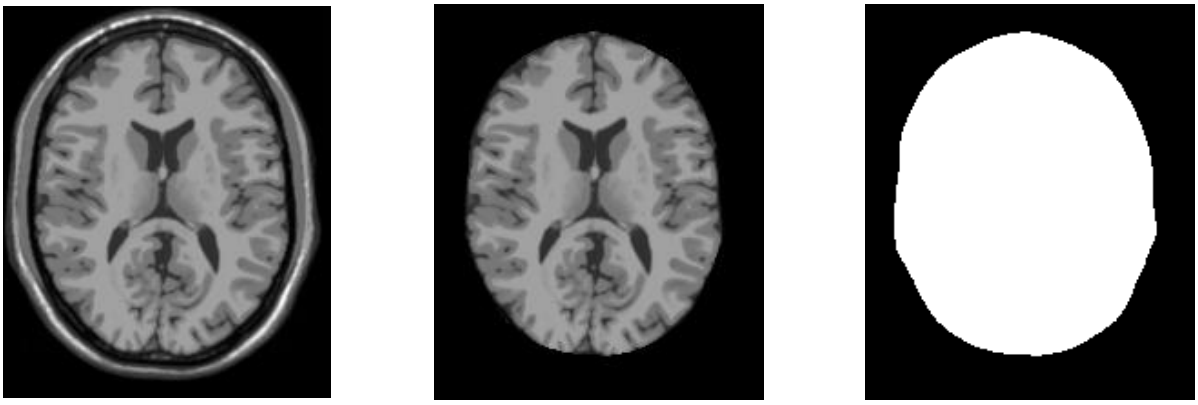
4.4. Pruebas adicionales

Para completar aún más el estudio se han realizado una serie de pruebas adicionales a los métodos de clasificación implementados que sirvan para comprobar la veracidad de los mismos.

Se ha probado, por tanto, el algoritmo con otros cortes (slices) distintos al corte 94, que es en el que se ha centrado el estudio, por ser un corte central y donde se visualizan de forma más clara los distintos tejidos. Se mostrarán los resultados para los cortes 85 y 75.

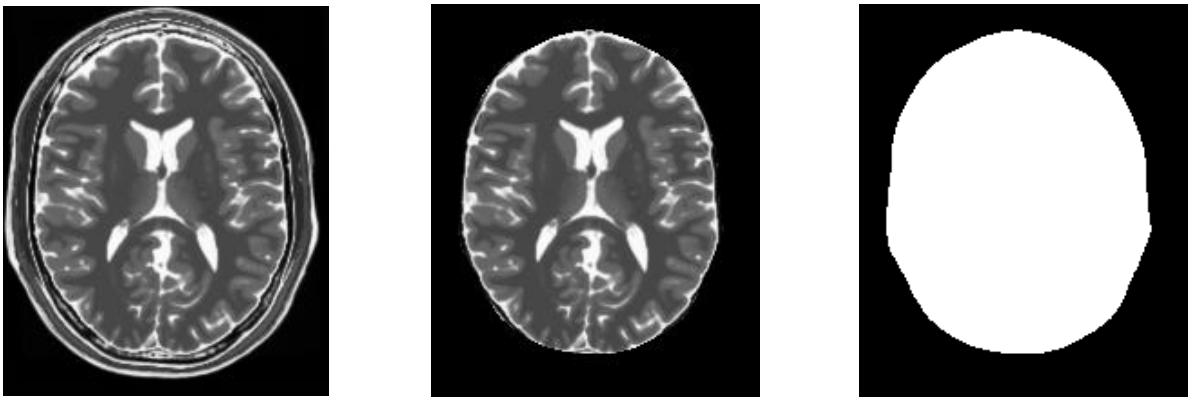
En primer lugar se muestran los resultados del pre-procesado. En la siguiente figura se puede observar como se comporta el algoritmo de pre-procesado con estos nuevos cortes.

4.4.1. Resultados del pre-procesado para el corte número 85.



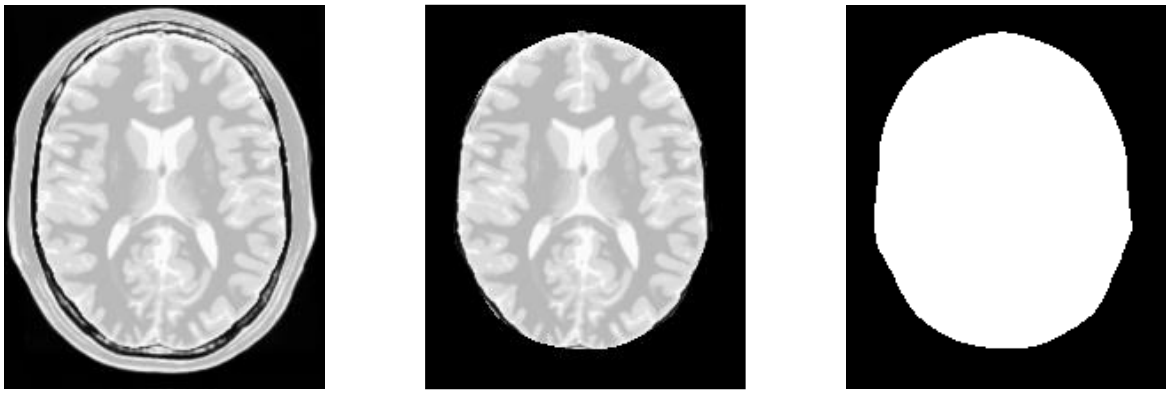
(a) (b) (c)

Figura 28. Eliminación del cráneo en Imagen T1-weigthed: (a) Imagen Original, (b) Máscara, (c) Imagen segmentada.



(a) (b) (c)

Figura 29. Eliminación del cráneo en Imagen T2-weigthed: (a) Imagen Original, (b) Máscara, (c) Imagen segmentada.

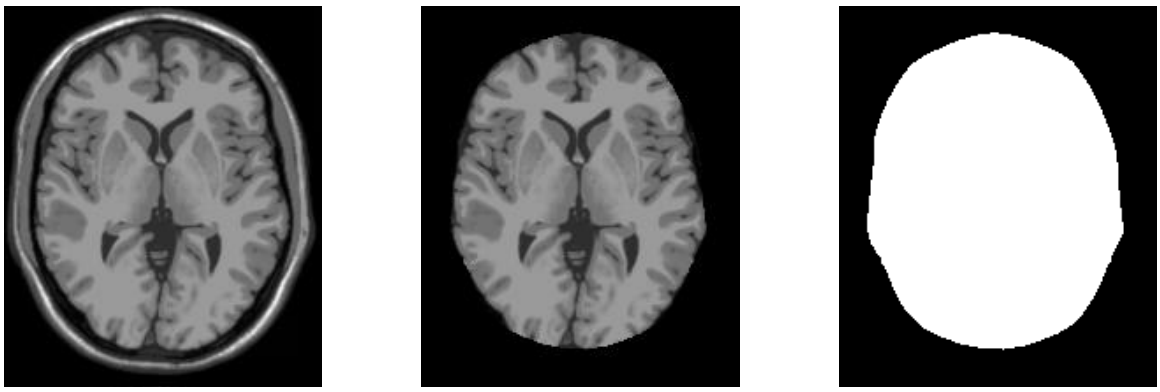


(a) (b) (c)

Figura 30. Eliminación del cráneo en Imagen PD-weigthed: (a) Imagen Original, (b) Máscara, (c) Imagen segmentada.

4.4.2. Resultados del pre-procesado para el corte número 75

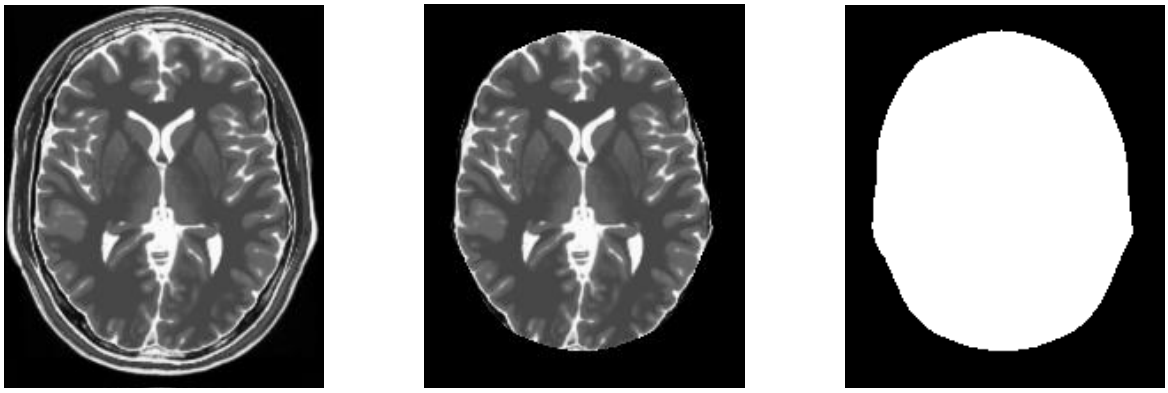
Se aplica el mismo procedimiento para el corte número 75 y se muestran en la siguientes figuras los resultados obtenidos.



(a) (b) (c)

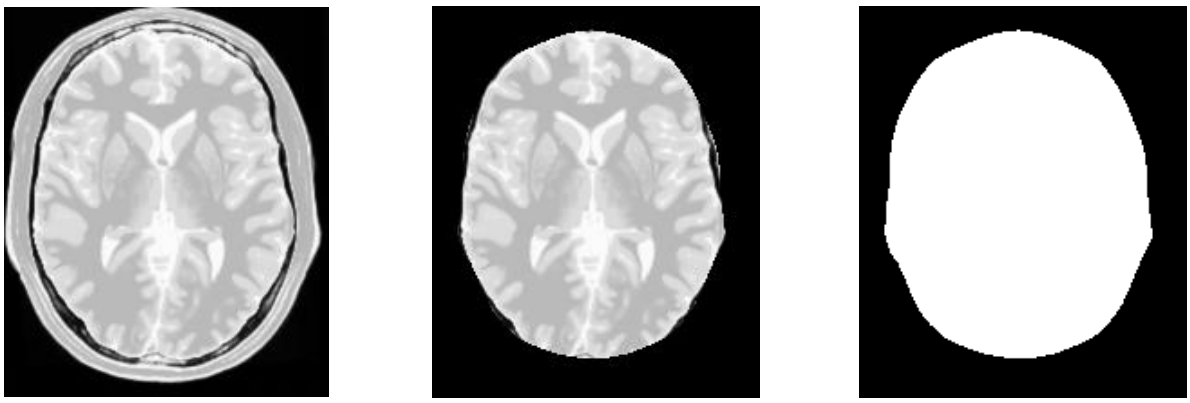
Figura 31. Eliminación del cráneo en Imagen T1-weigthed: (a) Imagen Original, (b) Imagen segmentada, (c) Máscara

Se aprecia una buena segmentación del cráneo y del tejido cerebral tanto para el corte número 75 como para el corte número 85. Esto sirve para justificar el buen funciamiento de este primer proceso de segmentación, no importando la forma que tenga el cráneo para hacer una buena segmentación de este.



(a) (b) (c)

Figura 32. Eliminación del cráneo en Imagen T2-weigthed: (a) Imagen Original, (b) Imagen segmentada, (c) Máscara



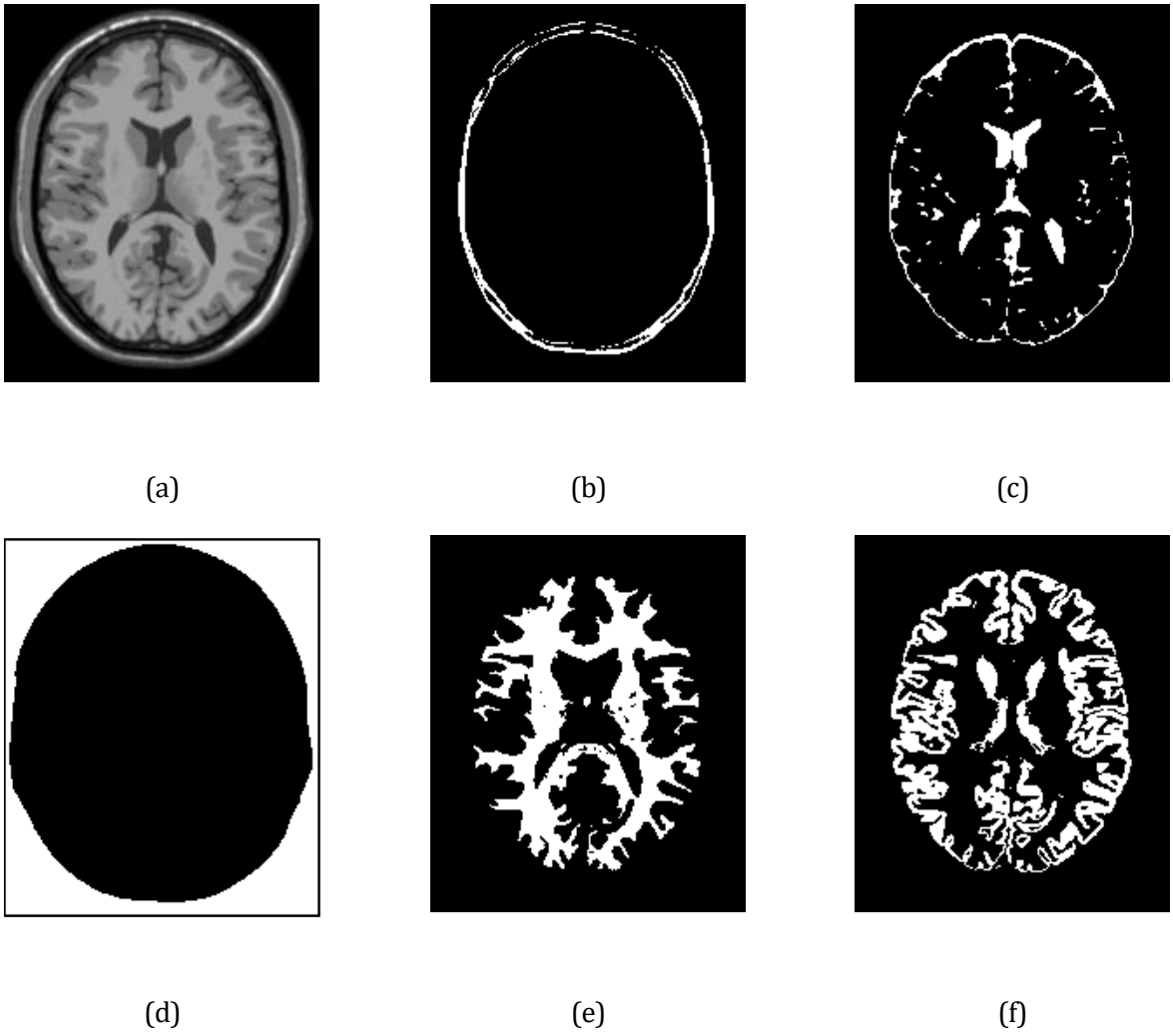
(a) (b) (c)

Figura 33. Eliminación del cráneo en Imagen PD-weigthed: (a) Imagen Original, (b) Imagen segmentada, (c) Máscara

A continuación, se muestran los distintos resultados tras aplicar todos los métodos en los dos cortes seleccionados para el estudio. Previamente se muestran las verdades de referencia para poder visualizar la calidad del resultado.

4.4.3. Resultados para los distintos métodos con el corte número 85.

El procedimiento seguido para hacer el análisis de resultados es el mismo que se ha realizado para el corte central. Se partirá de las verdades de referencia de los distintos tejidos para el corte número 85 y se compararán con los tejidos resultantes mediante los parámetros de calidad descritos al principio de este capítulo. Por tanto, a continuación, se representan las verdades de referencia para el corte objeto de estudio.



(a) (b) (c)

(d) (e) (f)

Figura 34. Verdades de Referencia para el corte número 85

En (a) se observa la imagen T1-weigthed original. (b) corresponde con el cráneo, (c) es el líquido cefalorraquídeo, (d) es el fondo, (e) la materia blanca y (f) la materia gris. Con estos datos ya podemos analizar tanto a simple vista como cuantitativamente la calidad de los resultados de las segmentaciones que se mostrarán a continuación.

Algoritmo K-Means

(a) (b) (c)

Figura 35. Resultados de la segmentación del corte 85 para el algoritmo K-Means

De la (a) a la (c) los resultados de la segmentación aplicando *K-Means*. Se puede observar que no hace una buena segmentación de los tejidos que se están buscando, al igual que ocurría con el corte central inicial.

Análisis de Resultados de K-Means

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	99,92 %	77,17%	71,60 %	46,30 %	99,97 %	46,28 %
Gray Matter	0,05 %	80,86 %	99,06 %	1,31 %	81,48 %	0,05 %
CSF	90,96 %	96,40 %	96,74 %	63,98 %	99,41 %	60,16 %
Average	**63,64 %**	**84,81 %**	**89,13 %**	**37,19 %**	**93,62 %**	**35,49 %**

Tabla 8. Análisis de calidad del método K-means para el corte 85

Algoritmo FCM

(a) (b) (c)

Figura 36. Resultados de la segmentación del corte 85 para el algoritmo FCM

De la (a) a la (c) los resultados de la segmentación aplicando FCM (*Fuzzy C-Means*). Ahora sí se aprecian, respectivamente, la materia blanca, la materia gris y el líquido cefalorraquídeo o CSF.

Análisis de Resultados de FCM

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	93,44 %	95,65 %	96,19 %	85,74 %	98,35 %	80,87 %
Gray Matter	89,37 %	92,84 %	93,62 %	75,92 %	97,50 %	69,64 %
CSF	85,42 %	97,13 %	97,87 %	71,87 %	99,06 %	64,02 %
Average	**89,41 %**	**95,20 %**	**95,89 %**	**77,84 %**	**98,30 %**	**71,51 %**

Tabla 9. Análisis de calidad del método FCM para el corte 85

Algoritmo KFCM

(a) (b) (c)

Figura 37. Resultados de la segmentación del corte 85 para el algoritmo KFCM.

De la (a) a la (c) los resultados de la segmentación aplicando KFCM (*Kernelized Fuzzy C-Means*). Se aprecian bien los tres tejidos, materia blanca, materia gris y fluído cefalorraquídeo (CSF), respectivamente.

Análisis de Resultados de KFCM

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	92,27 %	95,64 %	96,46 %	86,48 %	98,07 %	80,64 %
Gray Matter	90,06 %	92,50 %	93,05 %	74,50 %	97,65 %	68,84 %
CSF	89,68 %	96,73 %	97,18 %	66,89 %	99,33 %	62,11 %
Average	**90,67 %**	**94,95 %**	**95,56 %**	**75,95 %**	**98,35 %**	**70,53 %**

Tabla 10. Análisis de calidad del método KFCM para el corte 85

Algoritmo K-Means con ICA

(a) (b) (c)

Figura 38. Resultados de la segmentación del corte 85 para el algoritmo K-Means con ICA

Análisis de Resultados de K-means con ICA

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	92,25 %	95,98 %	96.89 %	87,93 %	98,07 %	81,88 %
Gray Matter	91,89 %	92,27 %	92,36 %	73,03 %	98,06 %	68,61 %
CSF	85,89 %	96,78 %	97,48 %	68,42 %	99,08 %	61,50 %
Average	**90,01 %**	**95,01 %**	**95,57 %**	**76,46 %**	**98,40 %**	**70,66 %**

Tabla 11. Análisis de calidad del método K-means con ICA para el corte 85

Algoritmo FCM con ICA (ICFCM)

(a) (b) (c)

Figura 39. Resultados de la segmentación del corte 85 para el algoritmo FCM con ICA

Análisis de Resultados para ICFCM

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	90,44 %	96,12 %	97,52 %	89,94 %	97,65 %	82,13 %
Gray Matter	89,27 %	92,91 %	93,73 %	76,25 %	97,48 %	69,85 %
CSF	78,47 %	97,03 %	98,21 %	73,64 %	98,62 %	61,26 %
Average	**86,06 %**	**95,35 %**	**96,48 %**	**79,94 %**	**97,91 %**	**71,08 %**

Tabla 12. Análisis de calidad del método ICFCM para el corte 85

Algoritmo KFCM con ICA (ICKFCM)

(a) (b) (c)

Figura 40. Resultados de la segmentación del corte 85 para el algoritmo KFCM con ICA

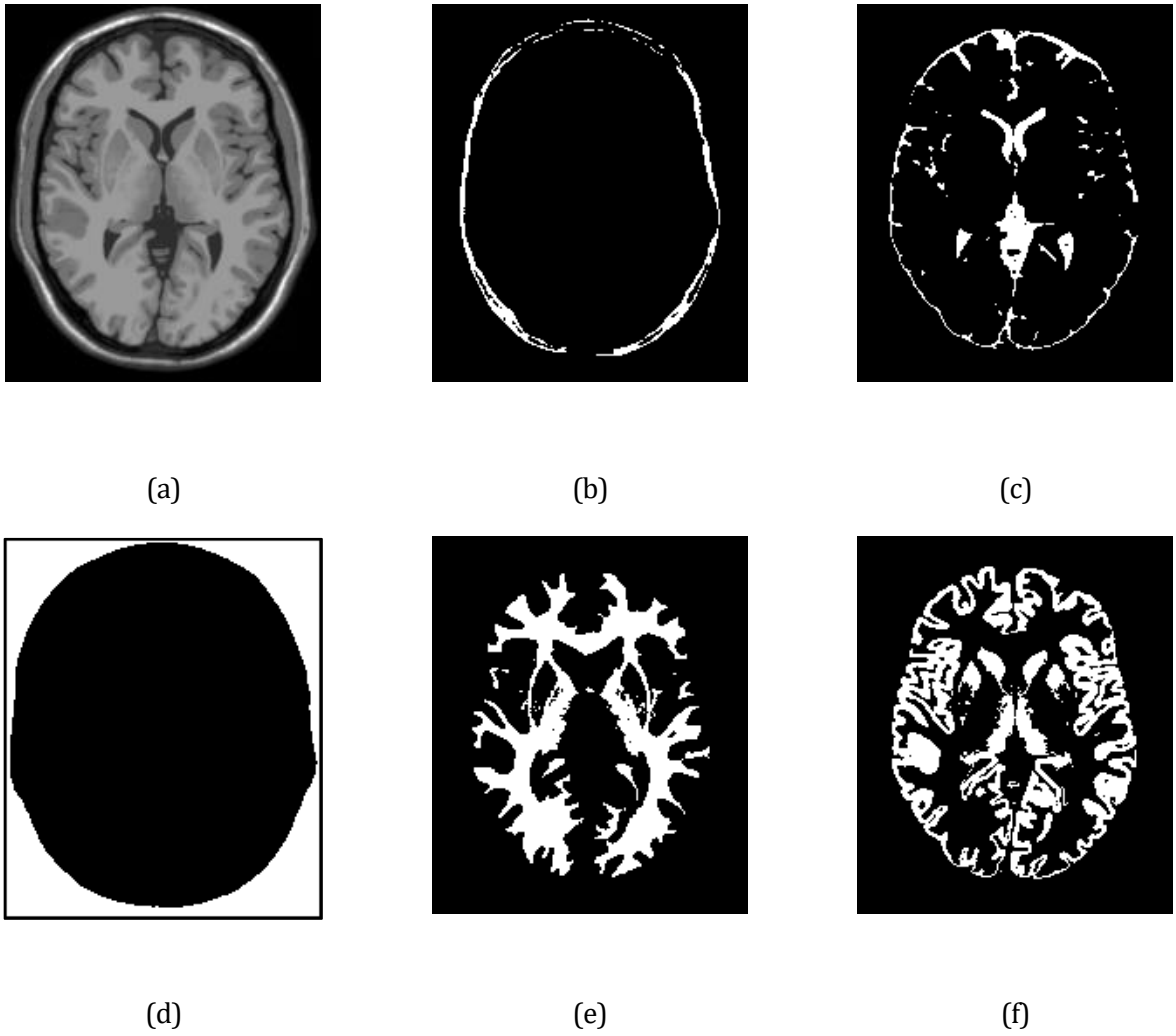
Análisis de Resultados para ICKFCM

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	96,14 %	91,67 %	90,57 %	71,43 %	98,96 %	69,44 %
Gray Matter	89,67 %	91,65 %	92,10 %	71,88 %	97,53 %	66,38 %
CSF	50,55 %	96,85 %	99,79 %	94,12 %	96,94 %	49,00%
Average	**78,78 %**	**93,39 %**	**94,15 %**	**79,14 %**	**97,81 %**	**61,60 %**

Tabla 13. Análisis de calidad del método KFCM con ICA (ICKFCM) para el corte 85

4.4.4. Resultados para los distintos métodos con el corte número 75

El procedimiento seguido para hacer el análisis de resultados es el mismo que se ha realizado para el corte central y para el corte 85. Se partirá de las verdades de referencia de los distintos tejidos para el corte número 75 y se compararán con los tejidos resultantes mediante los parámetros de calidad descritos al principio de este capítulo. Por tanto, a continuación, se representan las verdades de referencia para el corte objeto de estudio.



(a) (b) (c)

(d) (e) (f)

Figura 41. Verdades de Referencia para el corte número 75

En (a) se observa la imagen T1-weigthed original. (b) corresponde con el cráneo, (c) es el líquido cefalorraquídeo, (d) es el fondo, (e) la materia blanca y (f) la materia gris. Con estos datos ya podemos analizar tanto a simple vista como cuantitativamente la calidad de los resultados de las segmentaciones que se mostrarán a continuación.

Algoritmo K-Means

(a) (b) (c)

Figura 42. Resultados de la segmentación del corte 75 para el algoritmo K-Means

Análisis de Resultados para K-Means

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	100 %	74,39 %	68,37 %	42,62 %	100 %	42,62 %
Gray Matter	0,00 %	79,40 %	98,99 %	0,00 %	80,04 %	0,00 %
CSF	88,53 %	96,70 %	97,22 %	67,13 %	99,25 %	61,76 %
Average	**62,84 %**	**83,49 %**	**88,19 %**	**36,58 %**	**93,09%**	**34,79 %**

Tabla 14. Análisis de calidad del método K-means para el corte 75.

Algoritmo FCM

(a) (b) (c)

Figura 43. Resultados de la segmentación del corte 75 para el algoritmo FCM

Análisis de Resultados para FCM

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	95,95 %	94,15 %	93,73 %	78,24 %	98,99 %	75,75 %
Gray Matter	87,05 %	92,98 %	94,44 %	79,46 %	96,72 %	71,07 %
CSF	83,24 %	97,33 %	98,23 %	75,10 %	98,91 %	65,24 %
Average	**88,74 %**	**94,82 %**	**95,46 %**	**77,60 %**	**98,21 %**	**70,68 %**

Tabla 15. Análisis de calidad del método FCM para el corte 75.

Algoritmo KFCM

(a) (b) (c)

Figura 44. Resultados de la segmentación del corte 75 para el algoritmo KFCM

Análisis de Resultados para KFCM

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	94,78 %	94,79 %	94,79 %	81,06 %	98,72 %	77,60 %
Gray Matter	87,41 %	88,16 %	88,34 %	64,92 %	96,60 %	59,37 %
CSF	9,18 %	93,35 %	98,74 %	31,86 %	94,43 %	7,67 %
Average	**63,79 %**	**92,1 %**	**93,95 %**	**59,28 %**	**96,58 %**	**48,21 %**

Tabla 16. Análisis de calidad del método KFCM para el corte 75

Algoritmo K-Means con ICA

(a) (b) (c)

Figura 45. Resultados de la segmentación del corte 75 para el algoritmo K-Means con ICA

Análisis de Resultados para K-Means con ICA

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	95,04 %	94,84 %	94,79 %	81,09 %	98,78 %	77,80 %
Gray Matter	89,28 %	92,23 %	92,96 %	75,79 %	97,23 %	69,47 %
CSF	83,70 %	96,98 %	97,83 %	71,22 %	98,94 %	62,55 %
Average	**89,34 %**	**94,68 %**	**95,19 %**	**76,03 %**	**98,31 %**	**69,94 %**

Tabla 17. Análisis de calidad del método K-Means con ICA para el corte 75

Algoritmo FCM con ICA (ICFCM)

(a) (b) (c)

Figura 46. Resultados de la segmentación del corte 75 para el algoritmo ICFCM

Análisis de Resultados para ICFCM

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	93,72 %	95,43 %	95,83 %	84,08 %	98,48 %	79,60 %
Gray Matter	86,54 %	92,83 %	94,39 %	79,19 %	96,60 %	70,51 %
CSF	77,61 %	97,19 %	98,44 %	76,19 %	98,56 %	62,46 %
Average	**85,95 %**	**95,15 %**	**96,22 %**	**79,82 %**	**97,88 %**	**70,85 %**

Tabla 18. Análisis de calidad del método ICFCM para el corte 75

Algoritmo KFCM con ICA (ICKFCM)

(a) (b) (c)

Figura 47. Resultados de la segmentación del corte 75 para el algoritmo ICKFCM

Análisis de Resultados para ICKFCM

	Sensitivity	***Accuracy***	***Specificity***	***PPV***	***NPV***	***Similarity***
White Matter	97,49 %	91,84 %	90,51 %	70,70 %	99,35 %	69,44 %
Gray Matter	85,95 %	92,05 %	93,56 %	76,71 %	96,42 %	68,16 %
CSF	73,08 %	97,27 %	98,82 %	79,91 %	98,28 %	61,74 %
Average	**85,50 %**	**93,72 %**	**94,29 %**	**75,77 %**	**98,01 %**	**66,44 %**

Tabla 19. Análisis de calidad del método ICKFCM para el corte 75

4.5. Síntesis final de los resultados

Se hace una comparativa, en media, de los porcentajes de calidad obtenidos como Similitud o *Similarity* (coeficiente de Jaccard) para los distintos métodos aplicados al corte central (94) y a los dos adicionales (75 y 85), Con esto, se puede hacer una síntesis final que sirva como conclusión del estudio realizado.

4.5.1. Síntesis final para el corte central (corte 94)

	K-Means	***FCM***	***KFCM***	**ICA K-Means**	***ICFCM***	***ICKFCM***
White Matter	55,64 %	84,37 %	83,72 %	56.38 %	86,10 %	69,44 %
Gray Matter	0,00 %	69,59 %	69,57 %	4,91 %	70,81 %	68,16 %
CSF	68,01 %	73,04 %	71,12 %	73,15 %	72,84 %	61,74 %
Average	**41,21 %**	**75,66 %**	**74,80 %**	**44,81 %**	**76,58 %**	**66,44 %**

Tabla 20. Comparativa final de resultados para corte central 94

Para el corte 94, en los métodos de clasificación K-Means y FCM se observa una clara mejoría de los resultados si sobre las imágenes aplicamos previamente el algoritmo FastICA. De esta manera se consigue un mejor resultado en cuando a la medida de similitud. Sin embargo, se aprecia un empeoramiento de la similitud en los resultados del algoritmo KFCM en función de si se aplica o no FastICA.

4.5.2. Síntesis final para el corte número 85

	K-Means	***FCM***	***KFCM***	**ICA K-Means**	***ICFCM***	***ICKFCM***
White Matter	46,28 %	80,87 %	80,64 %	81,88 %	82,13 %	84,67 %
Gray Matter	0,05 %	69,64 %	68,84 %	68,61 %	69,85 %	61,75 %
CSF	60,16 %	64,02 %	62,11 %	61,50 %	61,26 %	53,17 %
Average	**35,49 %**	**71,51 %**	**70,53 %**	**70,66 %**	**71,08 %**	**66,53 %**

Tabla 21. Comparativa final de resultados para corte 85

De nuevo se presenta la misma casuística: en los métodos de clasificación K-Means y FCM se observa una clara mejoría de los resultados si sobre las imágenes aplicamos previamente el algoritmo FastICA, pero se aprecia un leve empeoramiento de la similitud en los resultados del algoritmo KFCM en función de si se aplica o no FastICA.

4.5.3. Síntesis final para el corte número 75

Para el caso del corte 75 se observa exactamente el mismo comportamiento que en los casos anteriores. A continuación, los resultados de calidad en valores de similitud.

	K-Means	***FCM***	***KFCM***	**ICA K-Means**	***ICFCM***	***ICKFCM***
White Matter	42,62 %	75,75 %	77,60 %	77,80 %	79,60 %	69,44 %
Gray Matter	0,00 %	71,07 %	59,37 %	69,47 %	70,51 %	68,16 %
CSF	61,76 %	65,24 %	7,67 %	62,55 %	62,46 %	61,74 %
Average	**34,79 %**	**70,68 %**	**48,21 %**	**69,94 %**	**70,85 %**	**66,44 %**

Tabla 22. Comparativa final de resultados para corte 75

5 CONCLUSIONES Y LÍNEAS FUTURAS

Hay que ser feliz, aunque sea sólo por molestar.

- Joaquín Sabina -

Se puede considerar, tras el estudio de investigación realizado, que el trabajo desarrollado tiene unos resultados, en general, positivos. El algoritmo para la eliminación previa del cráneo se comporta correctamente, a pesar de las variaciones en la forma del cráneo al recorrer el volumen y seleccionar distintos cortes o *slices*.

Los algoritmos FCM (*Fuzzy C-Means*) y KFCM (*Kernelized Fuzzy C-Means*) presentan buenas clasificaciones de los distintos tejidos. Sin embargo, se ha podido comprobar que el algoritmo K-Means no es un buen clasificador para este tipo de imágenes cerebrales debido a los resultados obtenidos tras aplicar el mismo tanto con ICA como sin ICA. En algunos casos hace una buena clasificación, pero no es suficiente para ser elegido como clasificador definitivo.

La aplicación del análisis de componentes independientes (FastICA) mejora la calidad del resultado para los casos de K-Means y FCM pero se aprecia una disminución de la calidad del 4% para el caso de KFCM. Sabiendo la dificultad que ha presentado aplicar el algoritmo KFCM para segmentar los tejidos de la materia blanca, la materia gris y el líquido cefalorraquídeo, se puede proponer como línea futura mejorar el post-procesado de este para que la calidad del resultado sea acorde a los métodos de clasificación anteriormente implementados. Además, sería también una buena línea futura implementar el algoritmo aplicando ruido a las imágenes de entrada para comprobar si los resultados siguen conservando buena calidad, teniendo en cuenta que esta bajará en alguna medida al tratarse de imágenes que han sufrido una corrupción por ruido.

Otra posible vía de investigación para mejorar este estudio sería aplicar los algoritmos de clasificación a imágenes reales ya que estas son imágenes cerebrales simuladas. También podría aplicarse a otro tipo de imágenes médicas. Estudios recientes [43] determinan que la segmentación del ventrículo izquierdo (LV o *Letf Ventricle*) a partir de conjuntos de datos de imágenes de resonancia magnética cardíaca (MRI) es un paso esencial para el cálculo de índices clínicos como el volumen ventricular y el porcentaje de eyección (bombeo de la sangre por los ventrículos), por lo que sería interesante estudiar la segmentación del ventrículo izquierdo en imágenes de resonancia magnética cardiaca con los algoritmos estudiados en este trabajo.

REFERENCIAS

[1] P. Oliveira y R. A. F. Romero, «Improvements on ICA mixture models for image pre-processing and segmentation,» *Neurocomputing,* vol. 71, nº 10-12, pp. 2180-2193, 2008.

[2] R. Jenssen y T. Eltoft, «Independent component analysis for texture segmentation,» *Pattern Recognition,* vol. 36, nº 10, pp. 2301-2315, 2003.

[3] D. Gupta, R. S. Anand y B. Tyagi, «A hybrid segmentation method based on Gaussian kernel fuzzy clustering and region based active contour model for ultrasound medical images,» *Biomedical Signal Processing and Control,* vol. 16, pp. 98-112, 2015.

[4] Y.-T. Chen, «A level set method based on the Bayesian risk for medical image segmentation Pattern Recognition,» vol. 43, nº 11, p. 3699–3711, 2010.

[5] Q. Z. a. E. Q. Dong, «Narrow band active contour model for local segmentation of medical and texture images,» *Acta Automatica Sinica,* vol. 39, nº 1, pp. 21-30, 2013.

[6] M. B. J. A. N. a. Y. Z. G. Xiao, «Segmentation of ultrasound B-mode images with intensity inhomogeneity correction,» *IEEE Transactions on Medical Imaging,* vol. 21, nº 1, pp. 48-57, 2002.

[7] H.-M. W. a. H. H.-S. Lu, «Iterative sliced inverse regression for segmentation of ultrasound and MR images,» *Pattern Recognition,* vol. 40, nº 12, pp. 3492-3502, 2007.

[8] W. E. L. G. R. K. a. F. A. J. W. M. Wells III, «Adaptive segmentation of MRI data,» *IEEE Transactions on Medical Imaging,* vol. 15, nº 4, p. 429–442, 1996.

[9] E. Sorouchyari, «Blind separation of sources, part III: stability analysis,» *Signal Processing,* vol. 24, nº 1, pp. 21-29, 1991.

[10] M. S. L. a. T. J. S. T.-W. Lee, «ICA mixture models for unsupervised classification of non-Gaussian classes and automatic context switching in blind signal separation,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, nº 10, p. 1078–1089, 2000.

[11] A. Bell y T. J. Sejnowski, «An information-maximization approach to blind separation and blind deconvolution,» *Neural Computation,* vol. 7, nº 6, p. 1129–1159, 1995.

[12] A. Hyvärinen, J. Karhunen y E. Oja, «Independent Component Analysis,» de *A Wiley-Interscience Publication, John Wiley & Sons Inc*, New York, NY, USA, 2001.

[13] T. Tateyama, Z. Nakao y Y.-W. Chen, «Classification of brain matters in MRI by Kernel

independent component analysis,» de *Proceedings of the 4th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP '08)*, Harbin, China, Agosto, 2008.

[14] S. Wu, A. C. Liew, H. Yan y M. Yang, «Cluster analysis of gene expression data based on self-splitting and merging competitive learning,» *IEEE Transactions on Information Technology in Biomedicine,* vol. 8, nº 1, p. 5–15, 2004.

[15] F. Masulli y A. Schenone, «A fuzzy clustering based segmentation system as support to diagnosis in medical imaging,» *Artificial Intelligence inMedicine,* vol. 16, nº 2, p. 129–147, 1999.

[16] G. Beni y X. Liu, «A least biased fuzzy clustering method,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 16, nº 9, p. 954–960, 1994.

[17] R. Krishnapuram y J. M. Keller, «The possibilistic C-means algorithm: insights and recommendations,» *IEEE Transactions on Fuzzy Systems,* vol. 4, nº 3, pp. 385-393, 1996.

[18] R. Krishnapuram y J. M. Keller, «Possibilistic approach to clustering,» *IEEE Transactions on Fuzzy Systems,* vol. 1, nº 2, pp. 98-110, 1993.

[19] D. Q. Zhang y S. C. Chen, «A novel kernelized fuzzy C-means algorithm with application in medical image segmentation,» *Artificial Intelligence in Medicine,* vol. 32, nº 1, pp. 37-50, 2004.

[20] E. A. Zanaty, S. Aljahdali y N. Debnath, «A kernelized fuzzy C-means algorithm for automatic magnetic resonance image segmentation,» *Journal of Computational Methods in Sciences and Engineering,* vol. 9, nº 1, pp. 123-136, 2009.

[21] S. R. Kannan, S. Ramathilagam, R. Devi y A. Sath, «Robust kernel FCM in segmentation of breast medical images,» *Expert Systems with Applications,* vol. 38, nº 4, pp. 4382-4389, 2011.

[22] M. S. Yang y H. .. Tsai, «A Gaussian kernel-based fuzzy c-means algorithm with a spatial bias correction,» *Pattern Recognition Letters ,* vol. 29, nº 12, p. 1713–1725, 2008.

[23] C. Li, C.-Y. Kao, J. C. Gore y Z. Ding, «Implicit active contours driven by local binary fitting energy,» de *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07), pp. 1–7*, IEEE,Minneapolis,Minn, USA, 2007.

[24] M. Gong, Y. Liang, J. Shi, W. Ma y J. Ma, «Fuzzy C-means clustering with local information and kernel metric for image segmentation,» *IEEE Transactions on Image Processing,* vol. 22, nº 2, pp. 573-584, 2013.

[25] E. A. Zanaty, «Determining the number of clusters for kernelized fuzzy C-means algorithms for automatic medical image segmentation,» *Egyptian Informatics Journal,* vol. 13, nº 1, pp. 39-58, 2012.

[26] J. Kang, L. Min, Q. Luan, X. Li y J. Liu, «Novelmodified fuzzy C-means algorithm with applications,» *Digital Signal Processing* , vol. 19, nº 2, pp. 309-319, 2009.

[27] D.-W. Kim, K. Lee, D. Lee y K. H. Lee, «A kernel-based subtractive clustering method,» *Pattern Recognition Letters,* vol. 26, nº 7, pp. 879-891, 2005.

[28] Universidad de Oviedo, «El algoritmo k-means aplicado a clasificación y procesamiento de imágenes,» [En línea]. Available: https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html.

[29] «Aprende Machine Learning. K-Means en Python paso a paso.,» [En línea]. Available: https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/. [Último acceso: 2019].

[30] «Ejemplo K-Means Matlab,» [En línea]. Available: https://es.mathworks.com/help/stats/kmeans.html. [Último acceso: 20 11 2019].

[31] R. A. Fisher, «The use of multiple measurements in taxonomic problems,» *Annals of Eugenics,* vol. 7, nº 2, pp. 179-188, 1936.

[32] L. González, «Blog LIGDI GONZÁLEZ - Aprende todo sobre Inteligencia Artificial,» 19 11 2019. [En línea]. Available: http://ligdigonzalez.com/machine-learning-clasificador-flor-iris-python/.

[33] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, 1981.

[34] S. Villazana, F. Arteaga, C. Seijas y O. Rodriguez, «Estudio Comparativo entre Algoritmos de Agrupamiento Basado en SVM y C-medios Difuso Aplicados a Senales Electrocardiográficas Arrítmicas,» *Revista Ingeniería UC,* vol. 19, nº 1, pp. 16-24, 2012.

[35] D. J. Sandoval Salazar, Análisis de componentes aplicado al estudio de la actividad cerebral, Bogotá, Colombia: Facultad de Ciencias, Departamento de Matem´aticas, 2014.

[36] A. Tharwat, «Independent component analysis: An introduction,» de *Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences*, 60318 Frankfurt am Main, Germany, 2018.

[37] «BrainWeb. Simulated Brain Database,» [En línea]. Available: http://brainweb.bic.mni.mcgill.ca/.

[38] C. R. Nave, 2010, «HyperPhysics,» [En línea]. Available: http://hyperphysics.phy-astr.gsu.edu/hbasees/hframe.html. [Último acceso: 2019].

[39] N. Tubridy, «Neuroradiological history: Sir Joseph Larmor and the basis of MRI physics.,» *Neuroradiology,* vol. 42, nº 11, pp. 852-855, 2000.

[40] J. V. Francés, «Imágenes por Resonancia Magnética - Presentación Power Point,» [En línea]. [Último acceso: 2019].

[41] A. I. García Noguer y I. Fondón , Segmentación automática de la excavación en retinografías basada en gradientes de color y clasificador Complex Tree, Sevilla: Universidad de Sevilla, 2015.

[42] R. Moulton y Y. Jiang, «Maximally Consistent Sampling and the Jaccard Index of Probability Distributions,» de *ICDM Workshop on High Dimensional Data Mining*, 2018.

[43] M. Avendi, A. Kheradvar y H. Jafarkhani, «A combined deep-learning and deformable-model approach to fully automatic segmentation of the left ventricle in cardiac MRI,» *Medical Image Analysis,* vol. 30, pp. 108-119, 2016.

[44] A. Hyvärinen y E. Oja, «A fast fixed-point algorithm for independent component analysis,» *Neural Computation,* vol. 9, nº 7, p. 1997, 1483-1492.

[45] E. A. Zanaty y A. Afifi, «Support vector machines (SVMs) with universal kernels,» *Applied Artificial Intelligence,* vol. 25, nº 7, pp. 575-589, 2011.

[46] N. H. Sweilam, A. A. Tharwat y N. K. Abdel Moni, «Support vector machine for diagnosis cancer disease; a comparative study,» *Egyptian Informatics Journal,* vol. 11, nº 2, pp. 81-92, 2010.

[47] «National Library of Medicine,» 2016. [En línea]. Available: http://www.nlm.nih.gov/.

[48] «Department of Medical Imaging and Radiological Technology, Yuanpei University of Medical Technology,» 2016. [En línea]. Available: http://rad.ypu.edu.tw/.

[49] R. Cárdenes, R. Luis de García y M. Bach-Cuadra, «A multidimensional segmentation evaluation for medical image data,» *Computer Methods and Programs in Biomedicine ,* vol. 96, nº 2, pp. 108-124, 2009.

[50] P. Comon, «Independent component analysis, a new concept?,» *Sig Process,* vol. 36, nº 3, p. 287–314, 1994.

[51] A. Hyvarinen, «Fast and robust fixed-point algorithms for independent component analysis,» *IEEE Trans. Neuronal Networks,* vol. 10, nº 3, pp. 626-634, 1999.

[52] A. Hyvärinen y E. Oja, «Independent component analysis: algorithms and applications,» *Neural Networks,* vol. 13, nº 4, pp. 411-430, 2000.

Anexos

Función de Eliminación del Cráneo: *skull_stripping.m*

```
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %             TRABAJO DE FIN DE MASTER            %
                %     Autora: Ana Isabel García Noguer            %
                %     Fichero: skull_stripping.m                  %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DESCRIPCION: Se trata de tomar la imagen de entrada y eli-
% minar la zona que corresponde al cráneo

function [original, salida, background] = skull_stripping(entrada)

% 1. Rotamos la imagen de entrada
entrada = entrada(:,:,94);
entrada=imrotate(entrada,180);

% Reescalamos la imagen de entrada (solo para calcular la máscara)
 maximo=max(max(entrada));
 im_rescaled=(255*entrada)./maximo;
 im_rescaled=uint8(im_rescaled);

% Calculamos la imagen binaria en base a un umbral
umbral=0.1;
im_bw = im2bw(im_rescaled,umbral);

% Realizamos operaciones morfológicas para rellenar los huecos
struct = strel('square',6);
im_dilated=imopen(im_bw,struct);

struct = strel('disk',7);
mask = imclose(im_dilated,struct);

struct = strel('disk',15);
mask = imerode(mask,struct);
mask_skull = imcomplement(mask);

%Aplicamos la máscara para obtener el cerebro
brain = zeros(size(mask)); %para que tengan igual dimensiones
brain = mask.*entrada;

%Aplicamos la máscara complementaria para obtener el cráneo
skull = zeros(size(mask)); %para que tengan igual dimensiones
skull = mask_skull.*entrada;


% Pasamos los resultados a las variables de salida
salida=brain;
original=entrada;
background=mask_skull;
end
```

Función Centrado, Blanqueado y FastICA: *independent_component_analysis.m*

```
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %             TRABAJO DE FIN DE MASTER                  %
            %     Autora: Ana Isabel García Noguer                  %
            %   Fichero: independent_component_analysis.m           %
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DESCRIPCION: Se aplica el algoritmo FastICA para
% separar las imagenes T1, T2 y PD en componentes
% independientes, que son IC1,IC2,IC3

function
[ic1,ic2,ic3]=independent_component_analysis(t1_prima,t2_prima,pd_prima)
 [M,N]=size(t1_prima);

%1. Se pasan las imagenes a vector
x1=t1_prima(:);
x2=t2_prima(:);
x3=pd_prima(:);

% 2. Señal observada X
X=[x1,x2,x3]';

% Paso 2.1. Centrar los datos (Hacer media de X cero)
Xmedia = mean2(X);
X=X-Xmedia;

% Paso 2.2. Blanqueado de los datos usando FastICA
rng(1); % For reproducibility
[whitesig, wm, dwm] = fastica(X, 'only', 'white');

ws1=reshape(whitesig(1,:),M,N);
ws2=reshape(whitesig(2,:),M,N);
ws3=reshape(whitesig(3,:),M,N);

%Paso 2.3. Aplicamos FasICA
[S,A,B]=fastica(X,'stabilization','on','approach','symm','g','skew');

% Reshape de los resultados
ic1=reshape(S(1,:),M,N);
ic2=reshape(S(2,:),M,N);
ic3=reshape(S(3,:),M,N);

% Reescalamos las imagenes
 maximo=max(max(ic1));
 ic1=(255*ic1)./maximo;
 ic1=uint8(ic1);

 maximo=max(max(ic2));
 ic2=(255*ic2)./maximo;
 ic2=uint8(ic2);

 maximo=max(max(ic3));
 ic3=(255*ic3)./maximo;
 ic3=uint8(ic3);

end
```

Función para Análisis de Calidad de los Resultados: *analysis.m*

```
              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              %             TRABAJO DE FIN DE MASTER             %
              %      Autora: Ana Isabel García Noguer            %
              %             Fichero: analysis.m                  %
              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%DESCRIPCIÓN: COMPARAMOS EL RESULTADO CON LA VERDAD DE REFERENCIA
clear all; close all; clc;

%ANALISIS
result1 = double(imread('./resultados/ickfcm_wht.tiff'));
true    = imread('./database/GT/wht.tiff');
[M,N]=size(true);

%Reescalamos los píxeles
result=zeros(M,N);
posiciones=find(result1>0);
result(posiciones)=255;

verdad=zeros(M,N);
posiciones=find(true>0);
verdad(posiciones)=255;

[m,n]=size(result);

TP=0; TN=0; FP=0; FN=0;

for i=1:m
    for j=1:n
       valor_result = result(i,j)
       valor_true = verdad(i,j)

       %si verdad == 1 y result == 1
       if (valor_true == 255 && valor_result == 255)
            TP=TP+1; %verdadero positivo
       elseif (valor_true == 255 && valor_result == 0)
            FN=FN+1; %falso negativo
       elseif (valor_true == 0 && valor_result == 0)
            TN=TN+1; %verdadero negativo
       else
            FP=FP+1; %falso positivo
       end
    end
end

%Parametros de calidad
Sensitivity = TP/(TP+FN)*100;
Accuracy = (TP+TN)/(TP+FN+TN+FP)*100;
Specificity = TN/(TN+FP)*100;
Positive_Predictive_Value = TP/(TP+FP)*100;
Negative_Predictive_Value = TN/(TN+FN)*100;
Jaccard = TP/(TP+FN+FP)*100;
```

Función *ground_truth.m*

```
              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              %             TRABAJO DE FIN DE MASTER           %
              %     Autora: Ana Isabel García Noguer           %
              %     Fichero: ground_truth.m                    %
              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DESCRIPCION:

function [salida] = ground_truth(entrada)

%Seleccionamos la slice y rotamos
entrada=entrada(:,:,1);
im=imrotate(entrada,180);
im_gray=im;

%Binarizamos la entrada
umbral=0.7;
im_bw = im2bw(im_gray,umbral);


%Construimos la verdad de referencia
 fila=ones(1,181);
 columna=ones(217,1);

 if fila == im_bw(1,:)
     fila=zeros(1,181);
     columna=zeros(217,1);
 elseif fila ~= im_bw(1,:)
     fila=ones(1,181);
     columna=ones(217,1);
 end

 im_bw(1,:)=fila;
 im_bw(end,:)=fila;
 im_bw(:,1)=columna;
 im_bw(:,end)=columna;

salida=im_bw;

end
```

Función *KFCM_init.m*

```
function [Vinit]=KFCM_init(Img,cluster_n)

%Transforma la imagen en un vector y aplica k-means al set de imágenes
Img=double(Img);
[row,col]=size(Img);
Imgtmp=reshape(Img,row*col,1);
size(Imgtmp);
[index,Vinit]=kmeans(Imgtmp,cluster_n,'emptyaction','singleton');

 end
```

Función *kfcm.m*

```
function [center, U, obj_fcn] = kfcm(data, Vinit, options)

% Change the following to set default options
default_options = [2;   % exponent for the partition matrix U
        100;    % max. number of iteration
        1e-5;   % min. amount of improvement
        1;      % info display during iteration
        10];    %
if nargin == 2
    options = default_options;
else
    % If "options" is not fully specified, pad it with default values.
    if length(options) < 5
        tmp = default_options;
        tmp(1:length(options)) = options;
        options = tmp;
    end
    % If some entries of "options" are nan's, replace them with defaults.
    nan_index = find(isnan(options)==1);
    options(nan_index) = default_options(nan_index);
    if options(1) <= 1
        error('The exponent should be greater than 1!');
    end
end

expo = options(1);          % Exponent for U
max_iter = options(2);      % Max. iteration
min_impro = options(3);     % Min. improvement
display = options(4);       % Display info or not
delta= options(5);          % delta

obj_fcn = zeros(max_iter, 1);   % Array for objective function

%Inicialización del bucle
center=Vinit;
cluster_n=length(Vinit);

%Función Kernel
k_dis=exp(-((abs(repmat(data',cluster_n,1)-
repmat(center,1,length(data)))).^2)/(delta.^2));

%Valores Uij, en forma matricial
U=((1-k_dis).^(-1/(expo-1)))./repmat( sum((1-k_dis).^(-1/(expo-
1)),1),cluster_n,1 );
size(U);

% Proceso Iterativo
for i = 1:max_iter
    tmp0=sum(U.^expo.*k_dis.*repmat(data',cluster_n,1),2);
    tmp1=sum(U.^expo.*k_dis,2);
    center=tmp0./tmp1;

    k_dis=exp(-((abs(repmat(data',cluster_n,1)-
repmat(center,1,length(data)))).^2)/(delta.^2));
    U=((1-k_dis).^(-1/(expo-1)))   ./   repmat( sum((1-k_dis).^(-1/(expo-
1)),1),cluster_n,1 );
    [U1,U2]=size(U);
    tmp=U.^2.*k_dis;
    obj_fcn(i)=sum(tmp(:));
```

```
    if display
        fprintf('Iteration count = %d, obj. fcn = %f, size(U)=%d %d\n', i,
obj_fcn(i),U1,U2);
    end
    % check termination condition
    if i > 1
        if abs(obj_fcn(i) - obj_fcn(i-1)) < min_impro, break; end
    end
end

iter_n = i; % Actual number of iterations
obj_fcn(iter_n+1:max_iter) = [];

end
```

Función *KFCM_Img.m*

```
function
[Img_label,One_index,aux]=KFCM_Img(Img,Vinit,cluster_n,m,iter_max,e,delta)
%     Img --> imagen de entrada
%     cluster_n --> necesario
%     m --> default=2
%     iter_max -> default=1000
%     e (epsilin) --> default=1e-5
%     Img_label --> Salida con tantos clústeres como se indique en
%     cluster_n

if nargin==3
    m=2;iter_max=1000;e=1e-5;delta=10;
end
if nargin==4
    iter_max=1000;e=1e-5;delta=10;
end
if nargin==5
    e=1e-5;delta=10;
end
if nargin==6
    delta=10;
end

Img=double(Img);
[nrow,ncol]=size(Img);

%Reshape de la imagen
Img_reshape=reshape(Img,nrow*ncol,1);
options = [m;iter_max;e;1;delta];

% Aplicar kfcm
[center,U,obj_fcn] = kfcm(Img_reshape,Vinit,options);



% Reajustar la salida
maxU = max(U);
maxUn=repmat(maxU,cluster_n,1);
```

```
U=U';
maxUn=maxUn';
One_index=(maxUn==U);

% Img_label=zeros(size(Img_reshape));
for i=1:cluster_n
    index=floor(find(One_index(:,i)==1));
    Img_label(index)=i;
end

aux=Img_label;

%Img_label=reshape(Img_label,nrow,ncol);

% Preparamos salida
tmp=zeros(1,cluster_n);
for i=1:cluster_n
    tmp(i)=mean(Img(Img_label==i));
end
[data,index]=sort(tmp);

Img_label0=zeros(size(Img_label));
for i=1:cluster_n
    Img_label0(Img_label==index(i))=i;
end

Img_label=Img_label0;

end
```

Script para algoritmo K-Means: prueba_kmeans.m

```
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             %             TRABAJO DE FIN DE MASTER             %
             %      Autora: Ana Isabel García Noguer            %
             %      Fichero: prueba_kmeans.m                    %
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all ; clear all ; clc

%Base de datos: https://brainweb.bic.mni.mcgill.ca
%% 1. Lectura de la imagen

[t1,info1]=loadminc('./database/T1/t1_icbm_normal_1mm_pn0_rf0.mnc');
[t2,info2]=loadminc('./database/T2/t2_icbm_normal_1mm_pn0_rf0.mnc');
[pd,info3]=loadminc('./database/PD/pd_icbm_normal_1mm_pn0_rf0.mnc');

%% 2. Aplicamos Skull Stripping
[t1_original, t1_prima, background] = skull_stripping(t1);
[t2_original, t2_prima, background] = skull_stripping(t2);
[pd_original, pd_prima, background] = skull_stripping(pd);

%Tamaño de las imagenes
[M,N]=size(t1_original);


%% Montamos el dataset
dataset1 = t1_prima(:);
```

```
dataset2 = t2_prima(:);
dataset3 = pd_prima(:);
dataset = [dataset1,dataset2,dataset3];

% 1.3. Aplicar K-means
classes=4;
rng(1); % For reproducibility
[idx,C] = kmeans(dataset,classes);

% 1.4. Reshape de los resultados
salida_kmeans=reshape(idx,M,N);

km1=zeros(M,N);
posiciones=find(salida_kmeans==1);
km1(posiciones)=1;

km2=zeros(M,N);
posiciones=find(salida_kmeans==2);
km2(posiciones)=1;

km3=zeros(M,N);
posiciones=find(salida_kmeans==3);
km3(posiciones)=1;

km4=zeros(M,N);
posiciones=find(salida_kmeans==4);
km4(posiciones)=1;
```

Script para algoritmo ICA K-Means: *prueba_ica_kmeans.m*

```
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %            TRABAJO DE FIN DE MASTER                    %
            %     Autora: Ana Isabel García Noguer                   %
            %     Fichero: prueba_ica_kmeans.m                       %
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 close all ; clear all ; clc

%Base de datos: https://brainweb.bic.mni.mcgill.ca

%% 1. Lectura de la imagen
[t1,info1]=loadminc('./database/T1/t1_icbm_normal_1mm_pn0_rf0.mnc');
[t2,info2]=loadminc('./database/T2/t2_icbm_normal_1mm_pn0_rf0.mnc');
[pd,info3]=loadminc('./database/PD/pd_icbm_normal_1mm_pn0_rf0.mnc');

%% 2. Aplicamos Skull Stripping
[t1_original, t1_prima, background] = skull_stripping(t1);
[t2_original, t2_prima, background] = skull_stripping(t2);
[pd_original, pd_prima, background] = skull_stripping(pd);

%Tamaño de las imagenes
[M,N]=size(t1_original);

%% FastICA
[ic1,ic2,ic3]=independent_component_analysis(t1_prima,t2_prima,pd_prima);
```

```
%% Montamos el dataset
dataset1 = ic1(:);
dataset2 = ic2(:);
dataset3 = ic3(:);
dataset = [dataset1,dataset2,dataset3];

% 1.3. Aplicar K-means
classes=4;
rng(1); % For reproducibility
[idx,C] = kmeans(double(dataset),classes);

% 1.4. Reshape de los resultados
[M,N]=size(t1_prima);
salida_kmeans=reshape(idx,M,N);

km1=zeros(M,N);
posiciones=find(salida_kmeans==1);
km1(posiciones)=1;

km2=zeros(M,N);
posiciones=find(salida_kmeans==2);
km2(posiciones)=1;

km3=zeros(M,N);
posiciones=find(salida_kmeans==3);
km3(posiciones)=1;

km4=zeros(M,N);
posiciones=find(salida_kmeans==4);
km4(posiciones)=1;
```

Script para algoritmo FCM: *prueba_fcm.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            TRABAJO DE FIN DE MASTER                                    %
%     Autora: Ana Isabel García Noguer                                   %
%     Fichero:prueba_fcm.m                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all
clear all
clc

%Base de datos: https://brainweb.bic.mni.mcgill.ca

%% 1. Lectura de la imagen
[t1,info1]=loadminc('./database/T1/t1_icbm_normal_1mm_pn0_rf0.mnc');
[t2,info2]=loadminc('./database/T2/t2_icbm_normal_1mm_pn0_rf0.mnc');
[pd,info3]=loadminc('./database/PD/pd_icbm_normal_1mm_pn0_rf0.mnc');

%% 2. Aplicamos Skull Stripping
[t1_original, t1_prima, background] = skull_stripping(t1);
[t2_original, t2_prima, background] = skull_stripping(t2);
[pd_original, pd_prima, background] = skull_stripping(pd);

%Tamaño de las imagenes
[M,N]=size(t1_original);
```

```
%% Montamos el dataset
dataset1 = t1_prima(:);
dataset2 = t2_prima(:);
dataset3 = pd_prima(:);
dataset = [dataset1,dataset2,dataset3];

%% Aplicar FCM
classes=4;
rng(1); % For reproducibility
[centers,U] = fcm(dataset,classes);
%

% % 2. Reshape de los resultados
umbral=0.5;
salida_fcm=U';

salida_fcm1=reshape(salida_fcm(:,1),M,N);
salida_fcm1 = im2bw(salida_fcm1,umbral);

salida_fcm2=reshape(salida_fcm(:,2),M,N);
salida_fcm2 = im2bw(salida_fcm2,umbral);

salida_fcm3=reshape(salida_fcm(:,3),M,N);
salida_fcm3 = im2bw(salida_fcm3,umbral);

salida_fcm4=reshape(salida_fcm(:,4),M,N);
salida_fcm4 = im2bw(salida_fcm4,umbral);
```

Script para algoritmo ICFCM: *prueba_ica_fcm.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             TRABAJO DE FIN DE MASTER                          %
%     Autora: Ana Isabel García Noguer                          %
%     Fichero:prueba_ica_fcm.m                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all
clear all
clc

%Base de datos: https://brainweb.bic.mni.mcgill.ca

%% 1. Lectura de la imagen
[t1,info1]=loadminc('./database/T1/t1_icbm_normal_1mm_pn0_rf0.mnc');
[t2,info2]=loadminc('./database/T2/t2_icbm_normal_1mm_pn0_rf0.mnc');
[pd,info3]=loadminc('./database/PD/pd_icbm_normal_1mm_pn0_rf0.mnc');

%% 2. Aplicamos Skull Stripping
[t1_original, t1_prima, background] = skull_stripping(t1);
[t2_original, t2_prima, background] = skull_stripping(t2);
[pd_original, pd_prima, background] = skull_stripping(pd);

%Tamaño de las imagenes
[M,N]=size(t1_original);
```

```
%% FastICA
[ic1,ic2,ic3]=independent_component_analysis(t1_prima,t2_prima,pd_prima);

%% Montamos el dataset
dataset1 = ic1(:);
dataset2 = ic2(:);
dataset3 = ic3(:);
dataset = [dataset1,dataset2,dataset3];

% 1. Aplicar FCM sobre el mismo dataset anterior
classes=4;
[centers,U] = fcm(dataset,classes);

U=U';
[M,N]=size(t1_prima);
umbral=0.5;

salida_icfcm1=reshape(U(:,1),M,N);
salida_icfcm1 = im2bw(salida_icfcm1,umbral);

salida_icfcm2=reshape(U(:,2),M,N);
salida_icfcm2 = im2bw(salida_icfcm2,umbral);

salida_icfcm3=reshape(U(:,3),M,N);
salida_icfcm3 = im2bw(salida_icfcm3,umbral);

salida_icfcm4=reshape(U(:,4),M,N);
salida_icfcm4 = im2bw(salida_icfcm4,umbral);
```

Script para algoritmo KFCM: *prueba_kfcm.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             TRABAJO DE FIN DE MASTER                                  %
%     Autora: Ana Isabel García Noguer                                  %
%     Fichero: prueba_kfcm.m                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all
clear all
clc

%Base de datos: https://brainweb.bic.mni.mcgill.ca

%% 1. Lectura de la imagen
[t1,info1]=loadminc('./database/T1/t1_icbm_normal_1mm_pn0_rf0.mnc');
[t2,info2]=loadminc('./database/T2/t2_icbm_normal_1mm_pn0_rf0.mnc');
[pd,info3]=loadminc('./database/PD/pd_icbm_normal_1mm_pn0_rf0.mnc');

%% 2. Aplicamos Skull Stripping
[t1_original, t1_prima, background] = skull_stripping(t1);
[t2_original, t2_prima, background] = skull_stripping(t2);
[pd_original, pd_prima, background] = skull_stripping(pd);

%Tamaño de las imagenes
[M,N]=size(t1_original);
```

```
%% Montamos el dataset
dataset1 = t1_prima(:); dataset2 = t2_prima(:); dataset3 = pd_prima(:);
dataset = [dataset1,dataset2,dataset3];

%% KFCM
% Datos para el Kernel tomado del paper
cluster_n=4;
m=2;
iter_max=100;
e=1e-5;
delta=150;
rng(1); % For reproducibility

%kmeans a la imagen original
Vinit=KFCM_init(dataset,cluster_n);

%Aplicamos KFCM
[Img_label,one_index,aux]=KFCM_Img(dataset,Vinit,cluster_n,m,iter_max,e,delt
a);

% Preparamos salida
aux=aux';
salida_kfcm=reshape(aux,M*N,3);

tmp=zeros(1,cluster_n);
for i=1:cluster_n
    tmp(i)=mean(dataset(salida_kfcm==i));
end
[data,index]=sort(tmp);

salida_kfcm0=zeros(size(salida_kfcm));
for i=1:cluster_n
    salida_kfcm0(salida_kfcm==index(i))=i;
end

salida_kfcm=salida_kfcm0;
salida_kfcm1=reshape(salida_kfcm(:,1),M,N);
salida_kfcm2=reshape(salida_kfcm(:,2),M,N);
salida_kfcm3=reshape(salida_kfcm(:,3),M,N);
kfcm=(salida_kfcm3+salida_kfcm1+salida_kfcm2);
kfcm=kfcm.*imcomplement(double(background));
```

Script para algoritmo ICKFCM: *prueba_ica_kfcm.m*

```
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             %            TRABAJO DE FIN DE MASTER            %
             %     Autora: Ana Isabel García Noguer           %
             %     Fichero: prueba_ica_kfcm.m                 %
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all ;clear all ; clc

%Base de datos: https://brainweb.bic.mni.mcgill.ca
```

```
%% 1. Lectura de la imagen
[t1,info1]=loadminc('./database/T1/t1_icbm_normal_1mm_pn0_rf0.mnc');
[t2,info2]=loadminc('./database/T2/t2_icbm_normal_1mm_pn0_rf0.mnc');
[pd,info3]=loadminc('./database/PD/pd_icbm_normal_1mm_pn0_rf0.mnc');

%% 2. Aplicamos Skull Stripping
[t1_original, t1_prima, background] = skull_stripping(t1);
[t2_original, t2_prima, background] = skull_stripping(t2);
[pd_original, pd_prima, background] = skull_stripping(pd);

%Tamaño de las imagenes
[M,N]=size(t1_original);

%% FastICA
[ic1,ic2,ic3]=independent_component_analysis(t1_prima,t2_prima,pd_prima);

%% Montamos el dataset
dataset1 = ic1(:); dataset2 = ic2(:); dataset3 = ic3(:);
dataset = [dataset1,dataset2,dataset3];

%% KFCM

 % Datos para el Kernel tomado del paper
cluster_n=4;
m=2;
iter_max=80;
e=1e-5;
delta=150;

rng(1); % For reproducibility

%kmeans a la imagen original
Vinit=KFCM_init(dataset,cluster_n);

%Aplicamos KFCM
[Img_label_ica,one_index,aux]=KFCM_Img(dataset,Vinit,cluster_n,m,iter_max,e,
delta);

% Preparamos salida
aux=aux';
salida_kfcm=reshape(aux,M*N,3);

tmp=zeros(1,cluster_n);
for i=1:cluster_n
    tmp(i)=mean(dataset(salida_kfcm==i));
end
[data,index]=sort(tmp);

salida_kfcm0=zeros(size(salida_kfcm));
for i=1:cluster_n
    salida_kfcm0(salida_kfcm==index(i))=i;
end

salida_kfcm=salida_kfcm0;
salida_kfcm1=reshape(salida_kfcm(:,1),M,N);
salida_kfcm2=reshape(salida_kfcm(:,2),M,N);
salida_kfcm3=reshape(salida_kfcm(:,3),M,N);
kfcm=(salida_kfcm3+salida_kfcm1+salida_kfcm2);
kfcm=kfcm.*imcomplement(double(background));
```