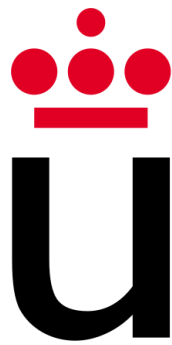


Práctica 2:

Segmentación de Imágenes Médicas

Análisis de Imagen Médica

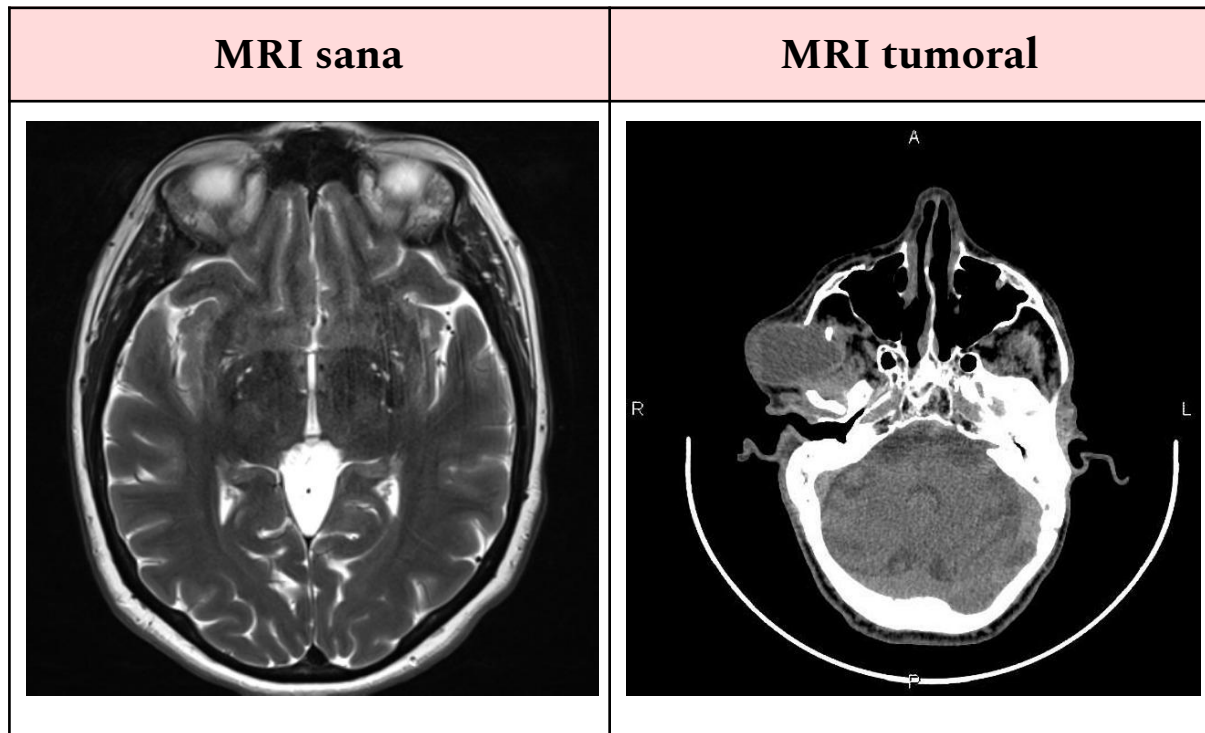


**Universidad
Rey Juan Carlos**

Sofía Gutiérrez Díaz
Teresa Vargas Rodríguez
Daniel Mancheño Castaño

0. Importar imágenes y librerías. [1], [2]:

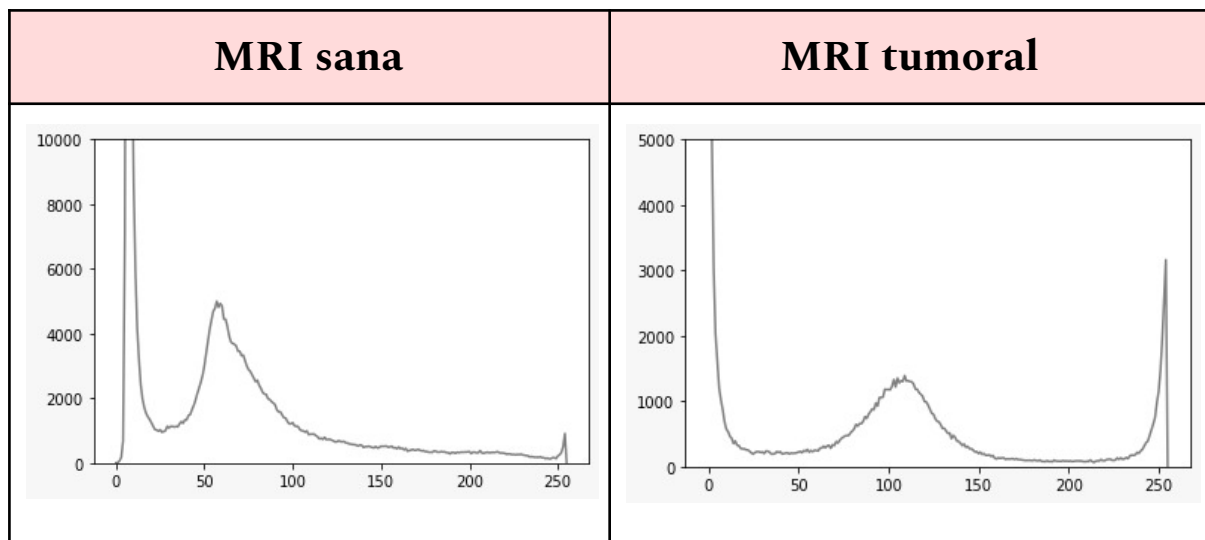
Comenzamos importando las librerías que se van a utilizar para construir el código en *python3*, cómo serían *numpy*, *scikit-image* y *opencv*. Las imágenes a filtrar, una imagen de resonancia magnética sana y una de resonancia magnética tumoral, se han importado con la librería de *scikit-image*.



1. Umbralización, crecimiento de regiones y Watershed:

Comenzamos visualizando los histogramas de cada una de las imágenes, para ello empleamos la función de la librería *opencv*.

Los histogramas de ambas imágenes son trimodales: la imagen de MRI sana contiene dos picos en la zona de grises oscuros y otro en el extremo derecho; mientras que en la imagen de MRI tumoral aparecen dos picos en los extremos y uno en la zona de grises intermedios.

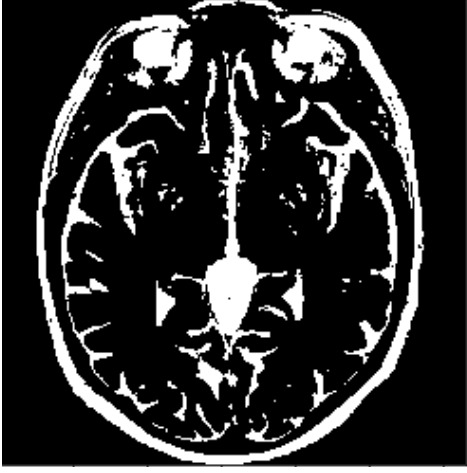




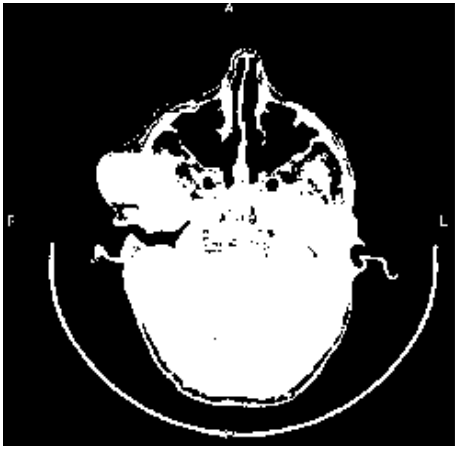
1.1 Umbralización [3], [4], [5]:

En la imagen de MRI sana, la estructura seleccionada para segmentar se corresponde con este pico en las intensidades de grises claros. Por el contrario, en la imagen de MRI tumoral buscamos aislar la región que se corresponde al pico en intensidades de gris medias.

Debido a la desproporción en píxeles oscuros y claros en ambas imágenes -los píxeles del fondo toman valor nulo, desequilibrando el histograma-, el método de Otsu estará contraindicado: el umbral no separará adecuadamente las estructuras. Aplicando esta técnica, obtenemos unos umbrales recomendados de **103** y **85**, para la MRI sana y la MRI tumoral respectivamente.

Manualmente, elegimos un umbral de **200** para la MRI sana. La MRI tumoral, por el contrario, no resulta ser compatible con esta técnica, puesto que la estructura no es diferenciable analizando únicamente los niveles de grises.

MRI sana Otsu	MRI sana Manual
	

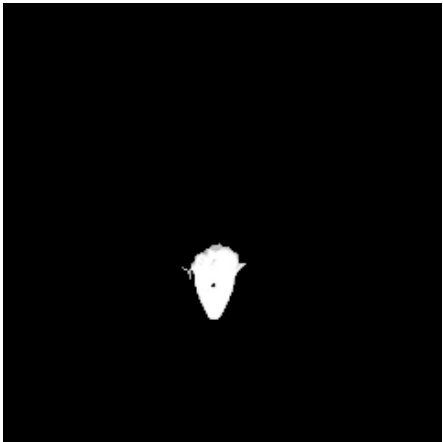
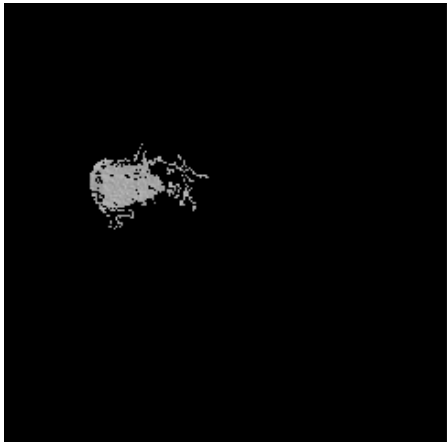
MRI tumoral Otsu	MRI tumoral Manual
	

1.2. Crecimiento de regiones [8]:

Se ha utilizado el método de crecimiento por regiones clásico en ambas imágenes por igual, cuya implementación en código depende de tres parámetros: *semilla*, *tolerancia* y el *nuevo valor*.

- La *semilla* se define como el pixel de inicio del algoritmo. Es imprescindible elegir una semilla dentro de la región a segmentar, en caso contrario, se corre el riesgo de no segmentar la estructura deseada.

- La técnica de segmentación por regiones se basa en la identificación de píxeles con intensidades similares. El parámetro de *tolerancia* permite modelar la diferencia de intensidad entre dos píxeles necesaria para considerarlos como parte de estructuras diferentes. Por ejemplo, si se establece una tolerancia de 10, todos los píxeles que varíen un máximo de 10 unidades de gris de la semilla se considerarán parte de la región de interés. En caso de sobreestimar este parámetro, la región de interés podría abarcar píxeles que no pertenecen al objeto deseado. Por el contrario, si se eligiera un valor demasiado bajo, podríamos no estar segmentando la zona al completo.
- El *nuevo valor* constituye la intensidad de los píxeles de la región de interés. Indicar que el valor de este parámetro no afecta a la segmentación en sí. No obstante, considerando que la salida generada es la imagen original con la zona segmentada superpuesta, es recomendable elegir un valor que permita diferenciar el objeto del resto de estructuras. En nuestro caso, para visualizar mejor el resultado de la segmentación, hemos decidido realizar la sustracción de la imagen segmentada y la imagen original.

MRI sana	MRI tumoral
	

1.3. Algoritmo de Watershed [9]:

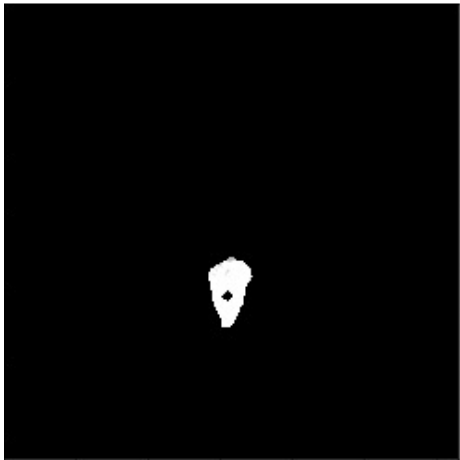
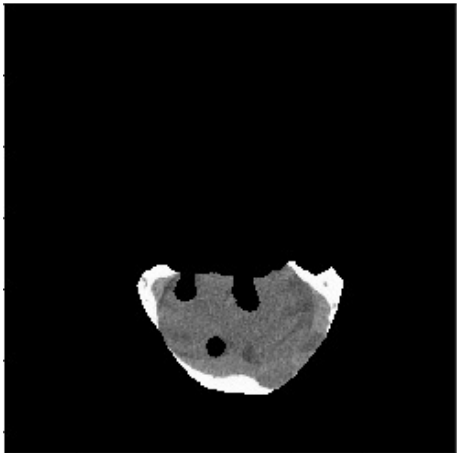
En este apartado se procesarán las imágenes utilizando el algoritmo de Watershed. El funcionamiento de este algoritmo es bastante sencillo, mediante la función *watershed* del módulo *segmentation* de *skimage*, podemos segmentar una imagen basándose en unos marcadores. Esta función toma dos parámetros:

- *image*: Será nuestra imagen a segmentar
- *markers*: Son los marcadores que se utilizan para definir las regiones que se sabe que pertenecen a objeto o al fondo. Puede ser tanto una matriz binaria que representan las zonas de objeto y fondo, como unas coordenadas.

La dificultad de este algoritmo reside en hallar los marcadores adecuados: Primero se debe binarizar la imagen (más adelante entenderemos el por qué), para después calcular un mapa de distancias euclidianas y con estas hallar los píxeles de la región conocida y los del fondo. Para calcular las distancias, vease como de lejos está un píxel blanco al píxel negro más cercano, usamos la función *distanceTransform* de la librería *opencv*. Dicha función recibe tres parámetros principales:

- *src*: Este parámetro es la imagen de la que vamos a calcular las distancias. La imagen sólo puede ser binaria.
- *distanceType*: el tipo de distancia a calcular, permite calcular distancias como la de Manhattan o la de Chebyshev, pero nosotros calcularemos la euclidiana.
- *maskSize*: el tamaño de la máscara utilizado para calcular la distancia.

Después de calcular las distancias, mediante umbralización, separamos la región que consideramos conocida (mayor distancia) de la que no (menor distancia). Después, mediante la función *connectedComponents*, etiquetamos los píxeles que están conectados y asignamos los marcadores. A continuación se muestran los resultados obtenidos:

MRI sana	MRI tumoral
	

1.4 Discusión de métodos de segmentación sin procesamiento de imágenes:

Método de segmentación	MRI sana	MRI tumoral
Umbralización Manual	Rendimiento limitado debido a que el umbral elegido no aísla la zona de interés que queremos segmentar.	Resultados pobres debido al bajo contraste y alta similitud de niveles de gris entre el tumor y las regiones adyacentes.
Método de Otsu	Resultados deficientes al tener una descompensación en la frecuencia de niveles de grises. El umbral escogido no segmenta bien nuestra región de interés. Se obtienen peores resultados que de la segmentación manual.	Mala segmentación pues el umbral seleccionado no permite diferenciar el tumor debido a que tiene un nivel de gris medio, parecido a la imagen en general.

Método de segmentación	MRI sana	MRI tumoral
Crecimiento de Regiones	Más efectivo que la umbralización manual ya que no deja residuos de otras zonas anatómicas del cerebro y segmenta la zona de interés correctamente debido a la uniformidad en tono de la misma y la gran diferencia en color con las regiones adyacentes.	No tan efectivo como en la MRI sana, pero mejor que la umbralización. No funciona del todo bien debido a la similitud en niveles de grises que se encuentran alrededor del tumor que queremos segmentar y al ruido que tiene la imagen.
Watershed	<p>Mal rendimiento porque identifica una zona oscura dentro de la región de interés como borde y no como estructura, por lo que se excluye dicho grupo de píxeles del objeto.</p> <p>En comparación con la segmentación por umbralización, mientras que este método segmenta mejor la región de interés se observan numerosos residuos. Mediante el método de <i>Watershed</i> el objeto se segmenta peor pero no aparecen estos artefactos.</p>	<p>Rendimiento extremadamente deficiente debido a dificultades para identificar bordes debido al bajo contraste y a la complejidad de la imagen.</p> <p>En cuanto a la umbralización, el rendimiento es igual de malo que el método <i>Watershed</i>.</p>

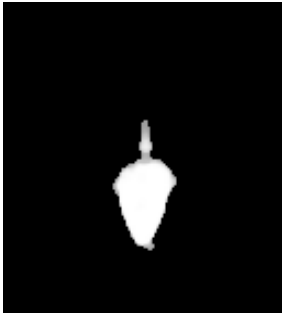

1.5 Métodos de segmentación de imágenes pre y postprocesadas:

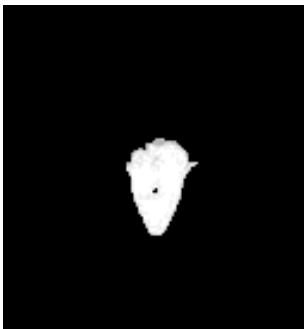
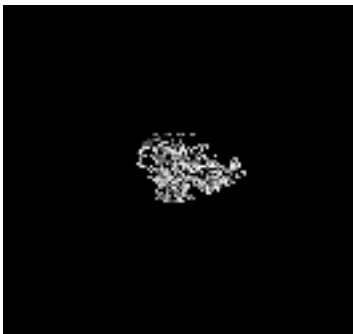
1.5.1 Crecimiento de Regiones:

En este caso hemos probado dos métodos diferentes de preprocesado.

Primero hemos aplicado un filtro gaussiano [6] a las imágenes del cerebro y del tumor, pero el resultado ha sido peor que en el método sin preprocesar. El problema de haber suavizado la imagen reside en la fusión de regiones que antes no estaban juntas, dando lugar a una segmentación incorrecta.

En cuanto al segundo método, hemos decidido enventanar [3] las imágenes para aumentar el contraste. Esto ha dado un buen resultado en caso de la imagen del cerebro sana, pero un mal resultado en el caso de la imagen tumoral. Esto se debe a la cantidad de zonas con un mismo nivel de gris que el tumor que se encuentra en la imagen.

MRI sana con filtro	MRI tumoral con filtro
	

MRI sana enventanada	MRI tumoral enventanada
	

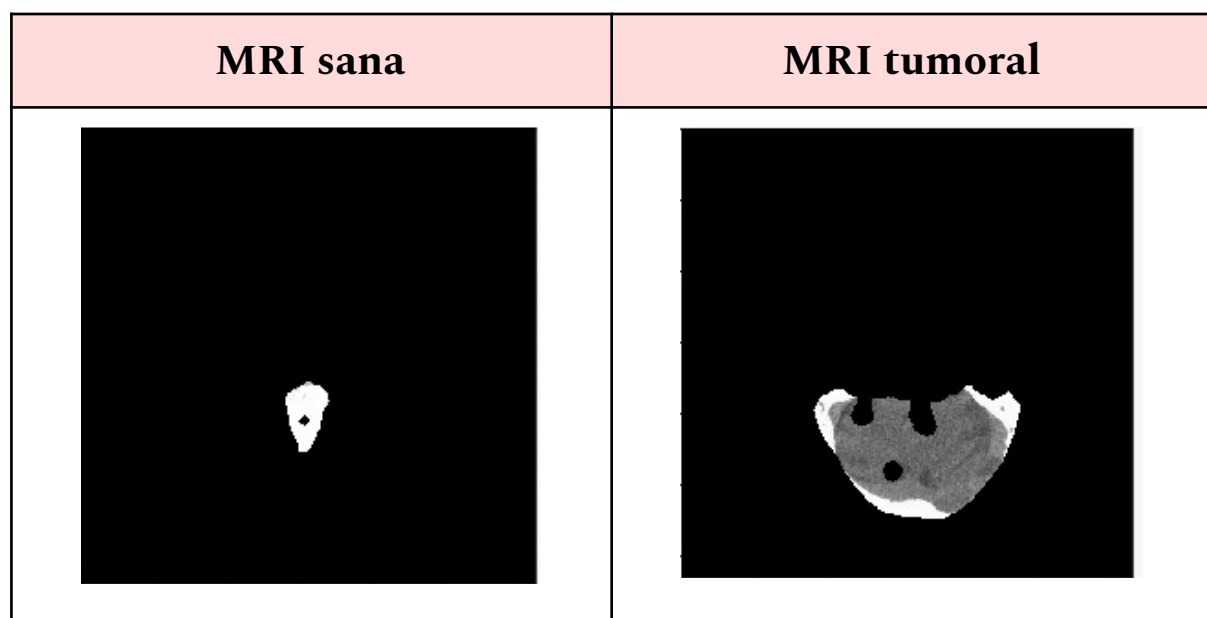
1.5.2 Watershed:


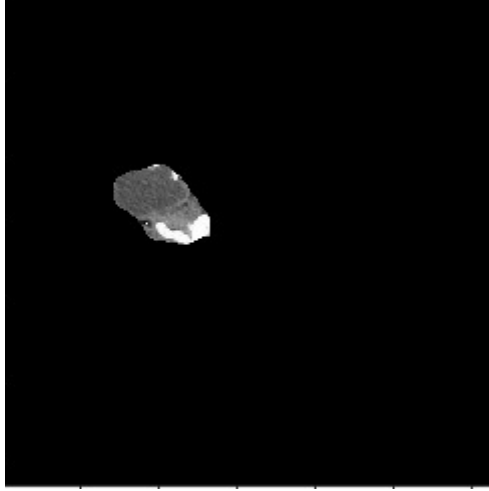
Para este caso hemos visto que el pre y postprocesado toman mucha importancia, por lo que nos hemos enfocado principalmente en él para mejorar los resultados.

En cuanto al preprocesado, hemos decidido suavizar los contornos y eliminar el ruido de la imagen binaria mediante un filtro gaussiano [6], de esta forma, las distancias se distribuyen de mejor manera, dándole más importancia a masas grandes de cerebro. En el caso de la imagen con tumor, en caso de no realizar el preprocesado la región de interés ni siquiera aparecía en el resultado final.

Para la imagen del tumor, además del filtro, hemos aplicado una erosión [7] para separar las dos regiones principales que eran detectadas, ya que si no lo hacíamos, el algoritmo detectaba como única región todo el cerebro.

Aún después del preprocesado, veíamos que en el caso del tumor, la región segmentada no coincidía exactamente con la región que deseábamos segmentar. Es por este motivo que decidimos realizar una dilatación de la región segmentada, dando como resultado más exacto.



MRI sana procesada	MRI tumoral procesada
	

2. Contornos Activos:

2.1 Funcionamiento [10] [11]:

El modelo de contornos activos es un método de segmentación basado en la detección de bordes en imágenes donde se conoce previamente la forma aproximada de la región a segmentar. La principal ventaja de esta técnica reside en su resistencia al ruido.

Se trata de algoritmo iterativo que se inicializa con la generación de una curva elástica, llamada *snake*, cerca del objeto. La *snake* se sentirá atraída por los altos gradientes asociados a los bordes de la región de interés, y se deformará en cada iteración, de manera similar a una goma, para adaptarse al objeto hasta conseguir delinear la estructura de interés.

La deformación de estas curvas elásticas están modeladas mediante ecuaciones de energía:

$$\begin{aligned}
E_{\text{snake}}^* &= \int_0^1 E_{\text{snake}}(\mathbf{v}(s)) ds \\
&= \int_0^1 E_{\text{int}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s))
\end{aligned}$$

Donde $\mathbf{v}(s)$ es la función de la serpiente y s son los diferentes puntos que conforman la curva $\mathbf{v}(s)$.

La energía interna (E_{int}) establece las características de deformación del contorno elástico mediante la restricción de la velocidad de cambio de la curva. Para ello se ponderan la primera y segunda derivada de la *snake*, donde los coeficientes determinan el grado en el cual esta se puede estirar o curvar. Es gracias a estas condiciones de flexibilidad que el modelo es resistente al ruido, puesto que coordina el movimiento de todos los puntos conjuntamente y evita que se se vean atraídos por gradientes generados por ruido.

$$E_{\text{int}} = \alpha(s) \left| \frac{\delta \mathbf{v}}{\delta s} \right|^2 + \beta(s) \left| \frac{\delta^2 \mathbf{v}}{\delta s^2} \right|^2$$

Por otro lado, la energía de la imagen (E_{image}) se corresponde a la fuerza que atrae la serpiente hacia la región de interés. Siendo el objetivo aislar un objeto concreto, aprovecharemos la presencia de los bordes de manera que dicha función de energía será proporcional al gradiente de la imagen. Se aplicará entonces un suavizado Gaussiano para potenciar dichos gradientes.

$$|\nabla I * G_\sigma|$$

Donde ∇I es el gradiente de las intensidades de la imagen y G_σ un filtro Gaussiano.

Para lograr la deformación de la serpiente se deberá maximizar el gradiente en los puntos s de la serpiente $\mathbf{v}(s)$, o equivalentemente, minimizar el negativo del gradiente.

$$E_{\text{image}} = -|\nabla I * G_{\sigma}|$$

Combinado ambas energías (E_{image} y E_{int}) se obtiene la expresión de E_{Snake}^* , la cual buscaremos minimizar. Los puntos s de la curva se moverán en cada iteración del algoritmo en dirección de máximo gradiente (o mínimo gradiente negativo) hasta rodear finalmente el objeto al completo.

2.2 Parámetros [12]:

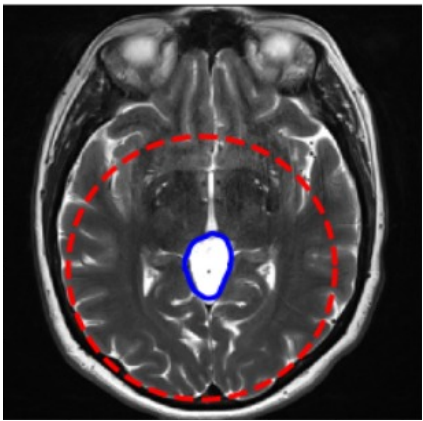
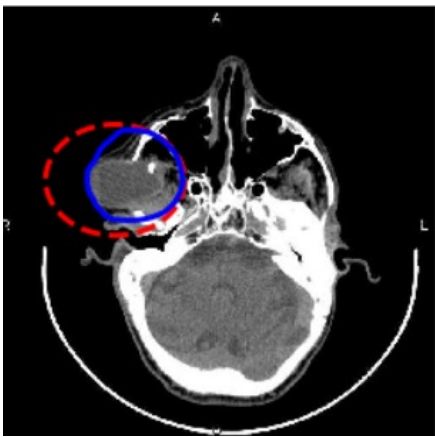
La función de `skimage.active_contour` que permite implementar esta metodología en *python* recibe los siguientes parámetros:

- El parámetros *snake*, que establece las coordenadas iniciales de la curva, se corresponde a una matriz (N,2) donde N es el número de puntos que forman la curva (puntos s). La *snake* deberá contener suficientes puntos iniciales para rodear el objeto de interés. En nuestro caso, se han inicializado curvas periódicas con formas redondas.
- Los parámetros *alpha* y *beta* forman parte de la ecuación de energía interna. Como se menciona previamente, modelan la velocidad de contracción y la suavidad de la curva. A mayor valor de estos parámetros, más rápido se deformará y más suave será la *snake*. Valores más bajos de *alpha* y *beta* aumentan, respectivamente, la flexibilidad de la curva y permiten un mayor trazado de líneas finas.
- Los parámetros *w_linefloat* y *w_edgfloat* controlan la atracción de la curva a zonas brillantes y a los bordes respectivamente. Aumentando el valor de estos coeficientes se aumenta dicha fuerza de atracción. Se corre el riesgo de que la *snake* se vea atraída por zonas ruidosas si este valor es demasiado elevado. Así mismo, mediante el uso de valores negativos podemos repeler el *snake* de valores altos de gris y de los contornos.
- Mediante los parámetros *gamma* y *convergence* se ajusta la velocidad y el umbral de convergencia de la *snake*. Aumentando el valor de *gamma*

conseguimos alcanzar el óptimo más rápido, corriendo el riesgo de que la *snake* se vuelva inestable. Por el contrario, valores más bajos ralentizarán la convergencia, de manera que se requerirán más iteraciones, aunque se mejorará la estabilidad del modelo.

- Los parámetros *max_px_move* y *max_num_iter* limitan tanto la distancia máxima, medida en píxeles, que puede recorrer la *snake* en cada iteración como el número total de iteraciones hasta alcanzar la curva óptima. Modificando el número de píxeles que tiene permitido moverse la *snake* definirá la velocidad a la que convergerá el modelo. De manera similar, si disminuimos el número de iteraciones se fuerza al modelo a converger más rápido, corriendo el riesgo de una segmentación incompleta. Por el contrario, aumentando las iteraciones, el proceso se alargará pero mejorará la segmentación.
- El parámetro de *boundary_condition* se corresponde a un *string* que establece las condiciones de contorno para la *snake*.

2.3 Segmentación mediante Contornos Activos [12]:

MRI sana	MRI tumoral
	

La segmentación por contornos activos devuelve unos resultados excelentes en el caso de la MRI sana, comparables con la segmentación por *Watershed* de la imagen

preprocesada, mientras que en la MRI tumoral no se logra aislar correctamente la zona de interés.

En la MRI sana este método funciona adecuadamente debido al alto contraste del objeto frente al fondo, al poco ruido y a la presencia de bordes claramente identificables.

En cambio, para la MRI tumoral no se logra ajustar bien la *snake* a la región de interés debido a la presencia de estructuras vecinas con intensidades elevadas. Estos focos distorsionan el contorno pues actúan como fuentes de atracción. Se ha probado a modificar el valor de los coeficientes de *alpha* y *beta* para reducir la flexibilidad de la curva; y de los parámetros de *w_linefloat* y *w_edgfloat*, para disminuir la fuerza de atracción hacia estas zonas blancas, sin mucho éxito. Por lo tanto, concluimos que la segmentación por contornos activos funciona ligeramente peor que el método de *Watershed* de la imagen de MRI sana preprocesada aunque demuestra ser bastante más efectiva para el caso de la imagen preprocesada de la MRI tumoral.

A pesar de los problemas que surgen a la hora de aislar el tumor, consideramos que el método de contornos activos supera a la segmentación por umbralización y por *Watershed* de imágenes sin preprocesar en ambos casos.

3. Otros métodos de segmentación: Random-Walker

3.1 Funcionamiento [13], [14]:

El método de segmentación de imágenes de *Random-Walker* es una técnica local basada en grafos. Se utiliza especialmente en imágenes ruidosas o de bajo contraste, o en caso de que los bordes de zonas anatómicas de interés aparezcan difusos. Se basa en la asignación de etiquetas, objeto y fondo, dependientes de la intensidad de píxeles vecinos a los píxeles de la imagen.

El algoritmo se inicializa creando un grafo a partir de la imagen, donde cada pixel representa un nodo y las aristas se corresponden a las conexiones entre nodos a

vecindad 8, vease todos los píxeles en contacto con un determinado píxel central. Estas aristas se pueden ponderar para ajustar el proceso de segmentación según sea necesario. A continuación se asignan unas etiquetas iniciales a unos píxeles semilla, clasificándolos como objeto o fondo según su intensidad. Además, se asignan probabilidades de transición a todas las aristas según la diferencia entre los niveles de gris de los píxeles que cada arista conecte.

Una vez finalizado este proceso da comienzo la segmentación: Se colocan unas partículas llamadas *caminantes* o *random-walkers* sobre píxeles no etiquetados y se mueven hasta alcanzar un píxel etiquetado. Para decidir a qué píxel dirigirse, se calculan las probabilidades de transición mediante el algoritmo de dijkstra, también conocido como el algoritmo de caminos mínimos. Una vez calculado el camino mínimo en términos de niveles de gris, véase qué píxel es más probable que tenga una intensidad similar al nodo que se desea etiquetar (sobre el que se encuentra el *caminante*), el *caminante* se desplaza en esta dirección, donde la probabilidad de transición es más alta, hasta alcanzar una semilla. Una vez el *caminante* llega a su destino, al píxel sobre el que comenzó se le asigna la etiqueta de dicho píxel semilla.

Este proceso se repite para todos los píxeles de la imagen hasta que todos los nodos hayan sido clasificados como objeto o fondo.

Tal y como se menciona anteriormente, esta técnica es muy robusta respecto al ruido. Esto se debe a que es un modelo de probabilidades que tiene en cuenta la intensidad de los píxeles y sus vecinos, de forma que cada decisión de movimiento de un caminante se basa en información local. Como el ruido es aleatorio, los caminantes siguen patrones locales más fuertes e ignoran los píxeles ruidosos.

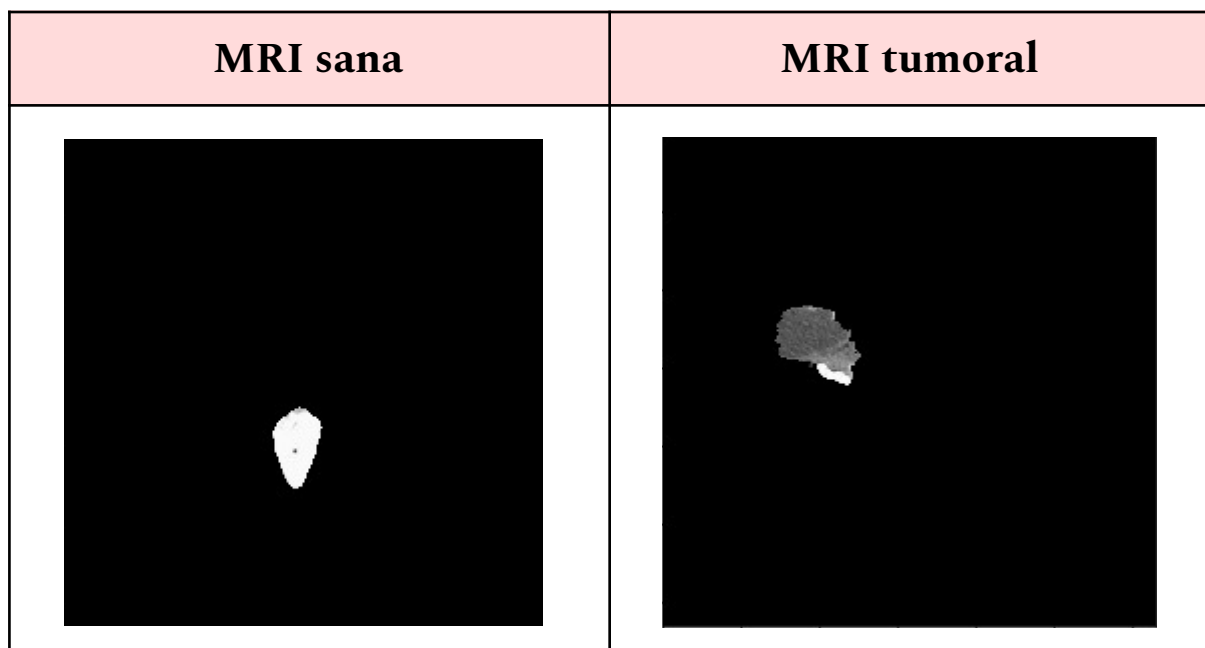
3.2 Parámetros [15]:

La función de *skimage.random_walker* que permite implementar esta metodología en *python* recibe los siguientes parámetros:

- El parámetros *data* es la imagen que queremos segmentar, siendo esta un array 2D de Numpy.

- El parámetro *labels* es una matriz NumPy que especifica las etiquetas de semillas para la segmentación. En nuestro caso estamos usando los mismos marcadores que hemos usado en el método *Watershed*.
- Mediante el parámetro *beta* hemos controlado la influencia de la suavidad en la segmentación. Este parámetro regula la penalización por cambios bruscos en la intensidad de píxeles entre regiones vecinas. Al aumentar su valor hemos favorecido que nuestras segmentaciones tengan límites más suaves entre regiones.
- El parámetro *mode* controla el tipo de segmentación. En nuestro caso hemos usado el modo *bf* (flujo bidireccional). El flujo bidireccional toma en cuenta tanto las etiquetas de las semillas como la información de intensidad de la imagen para determinar la probabilidad de que un píxel pertenezca a una etiqueta específica. Ayuda a que el algoritmo sea más robusto en el caso de que la región a segmentar tenga bordes irregulares o regiones no completamente conectadas.

3.3 Segmentación mediante Método Random-Walker [15]:



El método de segmentación Random-Walker logra segmentar adecuadamente tanto la MRI sana como la tumoral. Esta técnica es tan potente como difícil de

implementar, puesto que dependiendo de los marcadores seleccionados el resultado de la segmentación puede variar.

En el caso de la MRI sana, consideramos que es este método es el que mejor aísla la estructura de interés: no sólo preserva los bordes, sino que evita erosionar o dilatar en exceso el objeto, tal y como ocurre en la segmentación mediante el método de *Watershed* y el de contornos activos.

En cuanto a la MRI tumoral, al ser una imagen ruidosa y con bajo contraste, se espera obtener mejores resultados en comparación con el resto de técnicas. Consideramos que para segmentar el tumor en esta imagen los métodos más efectivos serían tanto el método de *Watershed* como el método de *Random-Walker*: En caso de buscar unos bordes más suaves, se recomienda emplear *Watershed* sobre una imagen suavizada, mientras que en caso de desear una segmentación más exacta se sugiere utilizar la segmentación *Random-Walker*, puesto que logra excluir mejor el hueso del resultado final.

4. Bibliografía

Librerías:

- [1] Numpy: (No date) *NumPy*. Available at: <https://numpy.org/> (Accessed: 16 October 2023).
- [2] Scikit-Image: *News* (no date) *scikit*. Available at: <https://scikit-image.org/> (Accessed: 16 October 2023).

Umbralización:

- [3] programación, T. de. (2017, August 17). Histogramas OpenCV Python. <https://acodigo.blogspot.com/2017/08/histogramas-opencv-python.html>

[4] *Skimage.filters#*. *skimage.filters* - *skimage* 0.22.0 documentation. (n.d.). https://scikit-image.org/docs/stable/api/skimage.filters.html#skimage.filters.threshold_otsu

[5] Rosebrock, A. (2023, June 8). *OpenCV thresholding (cv2.threshold)*. PyImageSearch. <https://pyimagesearch.com/2021/04/28/opencv-thresholding-cv2-threshold/>

Crecimiento de regiones:

[6] Atul, K. &, & Atul, K. &. (2019, May 6). *Cv2.medianBlur()*. TheAILearner. <https://theailearner.com/tag/cv2-medianblur/>

[7] GeeksforGeeks. (2023, January 4). *Python opencv: Cv2.erode() method*. GeeksforGeeks. <https://www.geeksforgeeks.org/python-opencv-cv2-erode-method/>

[8] *Skimage.segmentation#*. *skimage.segmentation* - *skimage* 0.22.0 documentation. (n.d.-a). https://scikit-image.org/docs/stable/api/skimage.segmentation.html#skimage.segmentation.flood_fill

Watershed:

[9] *Skimage.segmentation#*. *skimage.segmentation* - *skimage* 0.22.0 documentation. (n.d.-a). <https://scikit-image.org/docs/stable/api/skimage.segmentation.html#skimage.segmentation.watershed>

Contornos activos:

- [10] Rieiro, B. R., & José, R. M. de D. (2011). Capítulo 3/Contornos Activos. In *Segmentación Estéreo Mediante Contornos activos Proyecto Fin de Máster*. essay, s.n. <https://biblus.us.es/bibing/proyectos/abreproy/70265/fichero/Capítulo+3.pdf>
- [11] Kass, M., Witkin, A. & Terzopoulos, D. Snakes: Active contour models. *Int J Comput Vision* 1, 321–331 (1988). <https://doi.org/10.1007/BF00133570>
- [12] *Skimage.segmentation#*. skimage.segmentation - skimage 0.22.0 documentation. (n.d.).https://scikit-image.org/docs/stable/api/skimimage.segmentation.html#skimage.segmentation.active_contour

Random Walker:

- [13] Pennsylvania State University. (n.d.). <http://vision.cse.psu.edu/people/chenpingY/paper/grady2006random.pdf>
- [14] YouTube. (2017, February 24). *Random walks for segmentation*. YouTube. https://www.youtube.com/watch?v=nrTEeiK6bcE&ab_channel=MedicalImageAnalysis
- [15] *Random Walker segmentation#*. Random walker segmentation - skimage 0.22.0 documentation. (n.d.). https://scikit-image.org/docs/stable/auto_examples/segmentation/plot_random_walker_segmentation.html#sphx-glr-auto-examples-segmentation-plot-random-walker-segmentation-py