Segmenting with ITK (I)
Some notes

UNIVERSIDAD DE GRANADA

Autor: Víctor Torres de la Torre

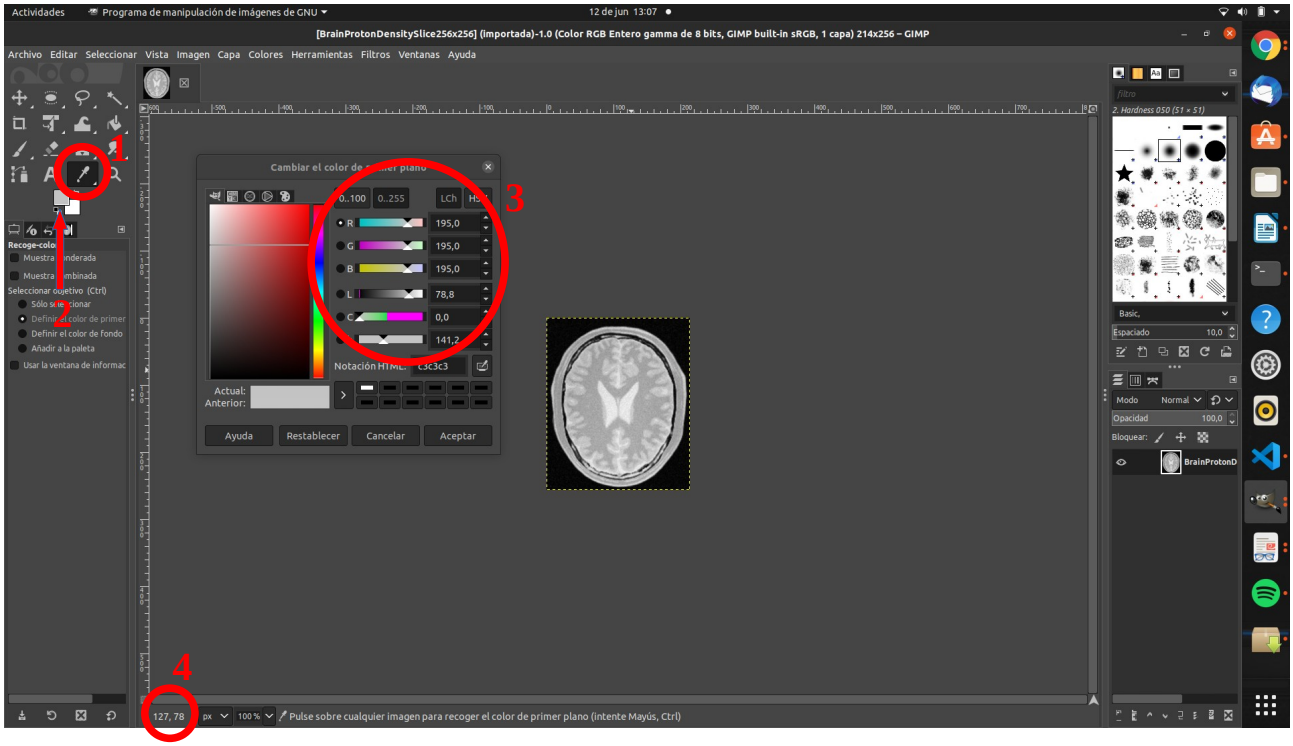
Task 1

Write a program that reads an image from a file and displays the result of applying the following filters to the input image:

ConnectedThresholdImageFilter , NeighborhoodConnectedImageFilter , and ConfidenceConnectedImageFilter .

The idea is to obtain several results by combining the free parameters of each filter, and by using or not a noise reduction filter over an input image before applying each one of these filters. The idea is to learn how the different filter parameters affect the segmentation result and observe how the application, or not, of a noise reduction filter previously in the pipeline affects the segmentation result. Try this program with the input images: BrainProtonDensitySlice256x256.png , coronaryAngiogram.png , saltAndPepperNoise1.jpg , and gaussianNoise.jpg .

Lo primero que necesitamos a la hora de experimentar con estos algoritmos es tener una idea tanto del valor de intensad como de la posición de un pixel en la imagen, de esta forma podremos proporcionar a los algoritmos un punto de partida y, además, tendremos un idea de cuales deberán de ser los valores máximos y mininos de intensidad para hacer la segmentación. Para poder obtener esta información de forma rápida se usará la herramienta Gimp:



Usando la herramienta Recoge-Color dentro de Gimp (1), podremos posicionar el puntero en un punto concreto de la imagen, al hacer click el color principal seleccionado será el valor del pixel sobre el que hemos pulsado. Haciendo click en el color (2) se nos despliega la ventana de cambiar de color, ahí podremos ver el valor RGB del pixel (3).

Por último, en la parte inferior izquierda de Gimp (4) podemos ver el valor x,y del pixel sobre el que está el puntero.

Una vez tenemos la información necesaria para empezar a segmentar, vamos a realizar diferentes experimentos con los diferentes algoritmos de segmentación propuestos en el task 1.

1. ConnectedThresholdImageFilter:

Para todos los experimentos y pruebas lanzaremos el algoritmo a la imagen orígional y a la imagen tras haber aplicado un filtro

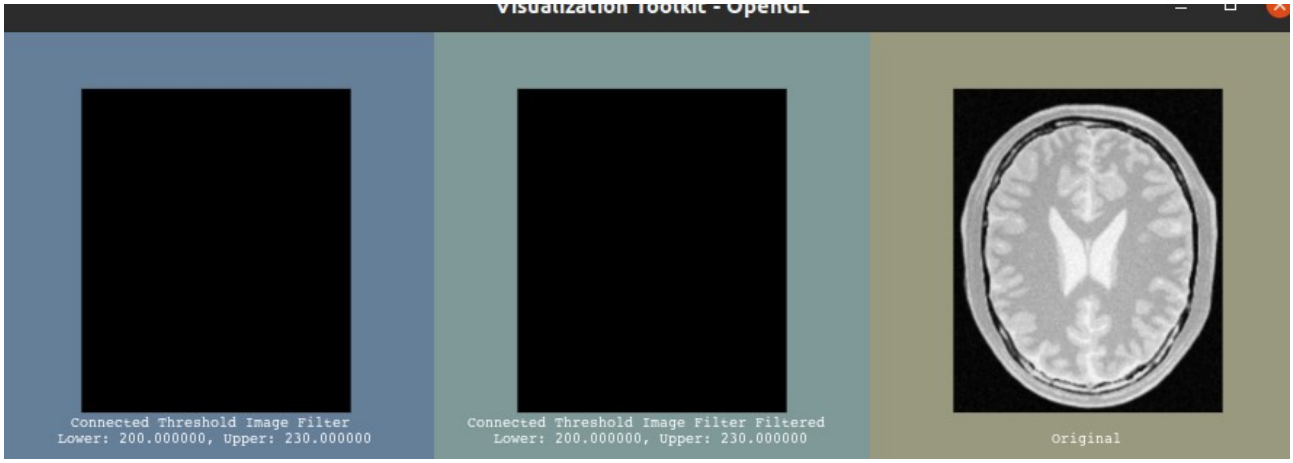
1.1. Experimento 1 (BrainProtonDensitySlice256x256.png):

Intentando segmentar la parte central de la imagen, para ellos los valores que hemos obtenido en gimp de posición y color son:

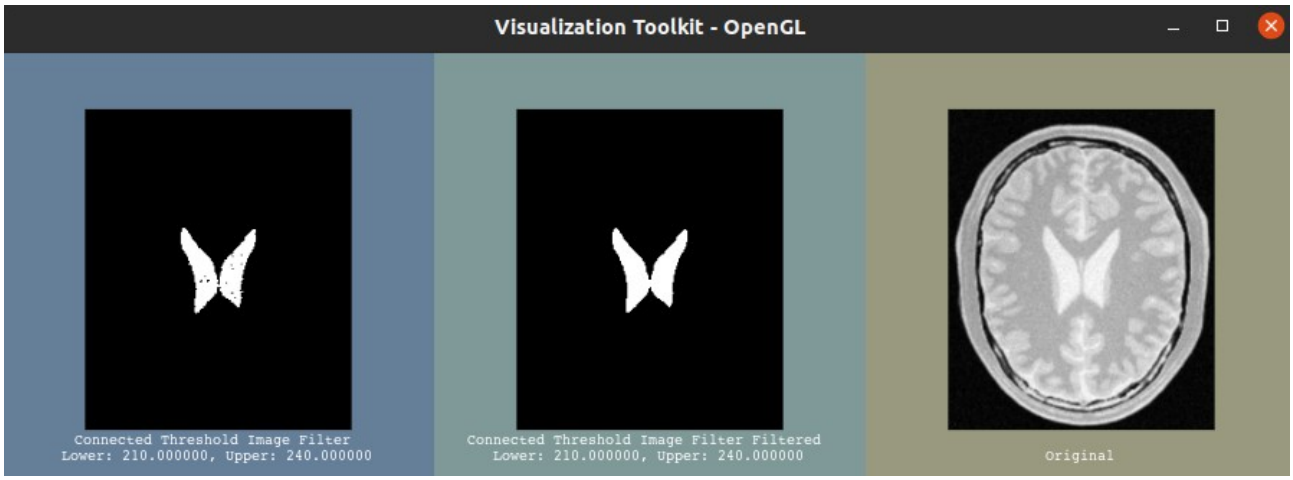
Valores Gimp
Intensidad: 222
Valor de X: 115
Valor Y: 130

Se han realizado diferentes pruebas de valores en función de lo obtenido en Gimp hasta llegar al resultado esperado. Los valores se pueden ver en la leyenda de las imágenes de cada prueba:

Prueba 1:



Prueba 2:



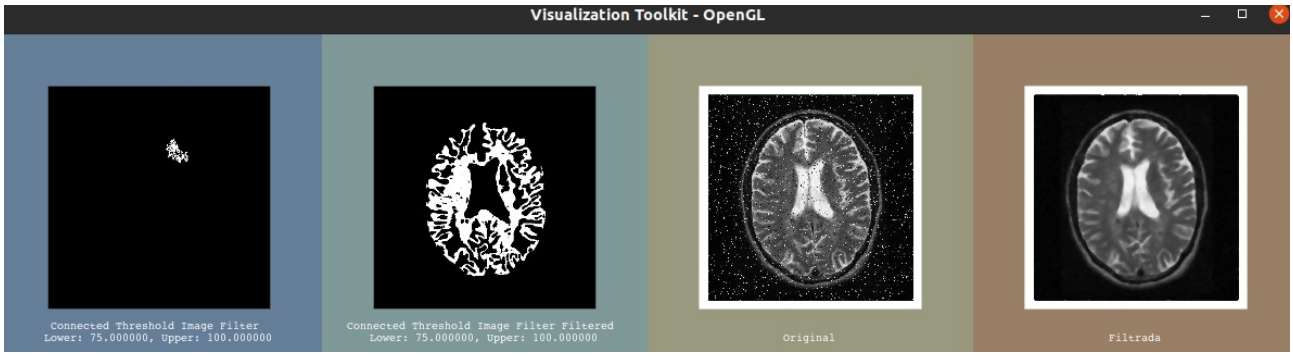
Se obtienen mejores resultados cuando previamente se la ha aplicado alguna técnica de reducción de sonido.

1.2 Experimento 2 ():

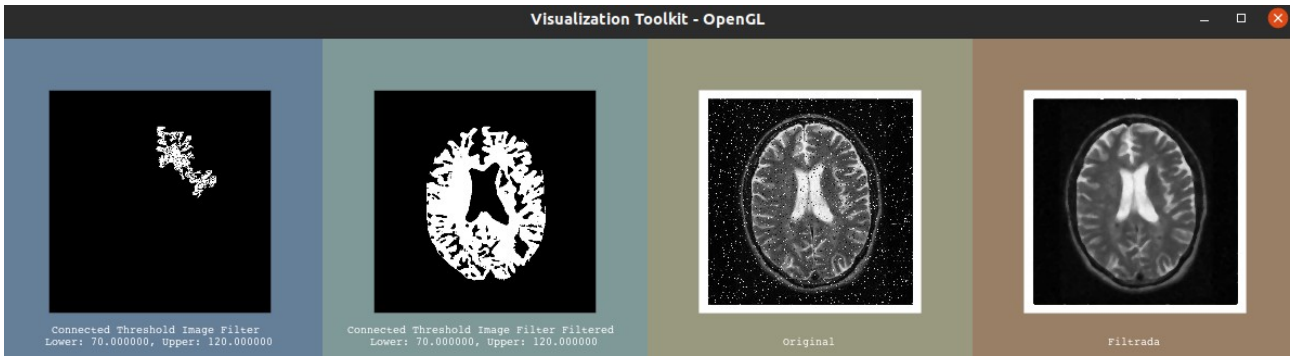
En este caso intentaremos obtener la sección intermedia de la imagen y usaremos un filtro de mediana igualmente.

Valores Gimp
Intensidad: 82
Valor de X: 305
Valor Y: 116

Prueba 1: 305 136 75 100



Prueba 2: 305 136 70 120



En este caso y de forma mas evidente que en el anterior, vemos el efecto negativo que el ruido hace a la hora de segmentar, aplicando un filtro de mediana con kernel a 4 y dando los límites correctos, hemos conseguido en la prueba 1 obtener una segmentación aceptable de la parte que buscábamos.

2. NeighborhoodConnectedImageFilter:

Para todos los experimentos y pruebas lanzaremos el algoritmo a la imagen original y a la imagen tras haber aplicado un filtro

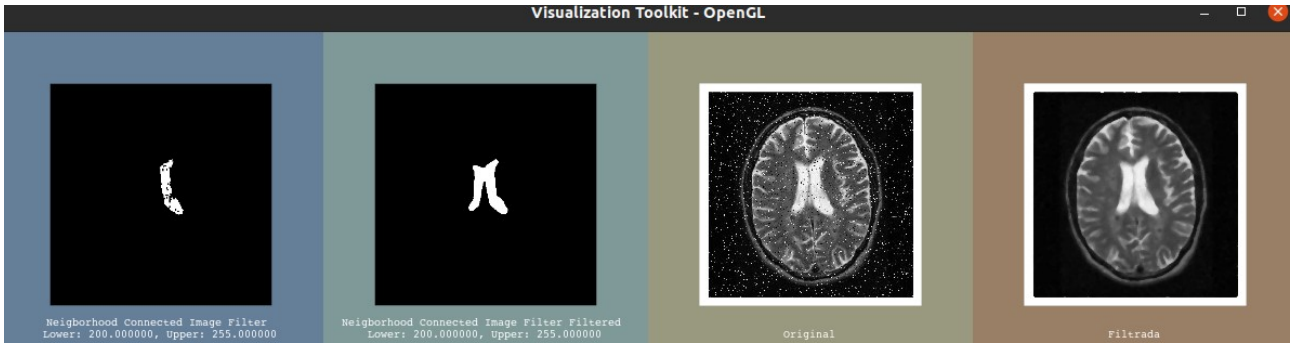
2.1 Experimento (BrainProtonDensitySlice256x256.png):

Esta vez intentaremos segmentar la parte central de la imagen, para ello los valores que hemos obtenido en gimp de posición y color son:

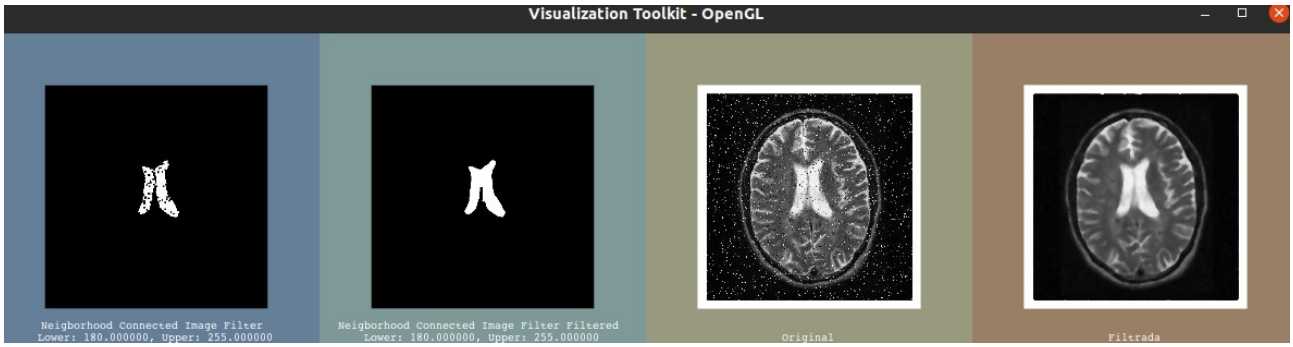
Valores Gimp
Intensidad: 254
Valor de X: 195
Valor Y: 112

Se han realizado diferentes pruebas de valores en función de lo obtenido en Gimp hasta llegar al resultado esperado. Los valores se pueden ver en la leyenda de las imágenes de cada prueba:

Prueba 1: 297 291 200 255



Prueba 1: 297 291 180 255



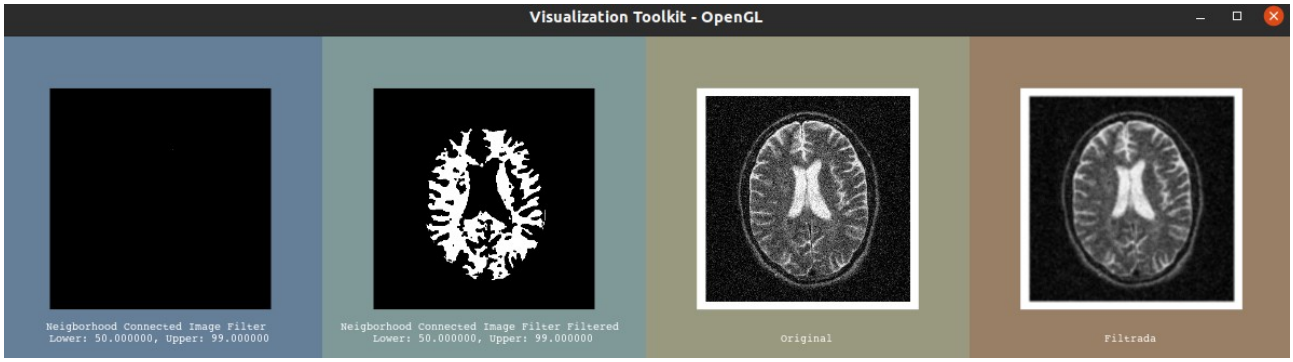
Se puede observar que bajando el umbral inferior obtenemos una segmentación más fidedigna ya que los pixeles centrales son descartados en la prueba 1 mientras se integra parte de ellos en la prueba 2.

2.2 Experimento (BrainProtonDensitySlice256x256.png):

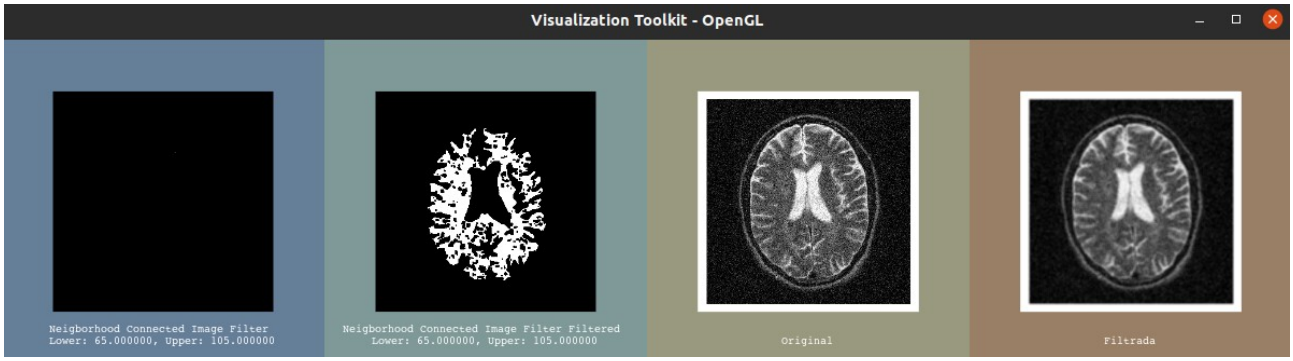
Esta vez intentaremos segmentar la parte intermedia de la imagen, teniendo en cuenta que esta vez aplicaremos un filtro gaussiano por la naturaleza gaussiana del ruido al que nos enfrentamos. Para este experimento los valores que hemos obtenido en gimp de posición y color son:

Valores Gimp
Intensidad: 70
Valor de X: 298
Valor Y: 147

Prueba 1: 298 147 50 99



Prueba 2: 298 147 65 105



En este caso llegamos a la conclusión de que no es posible hacer una segmentación adecuada si no eliminamos el ruido antes. Una vez tenemos esto claro, sabemos que los valores de umbral van a definir la cantidad de detalle que tendremos en nuestra segmentación. Mientras en la prueba 1 hemos obtenido un segmento más sólido al haber ampliado el intervalo del umbral, en la prueba 2, es menos sólido pero con más detalle, excluyendo de la segmentación zonas más oscuras del interior de la zona a segmentar.

3. ConfidenceConnectedImageFilter:

Para todos los experimentos y pruebas lanzaremos el algoritmo a la imagen original y a la imagen tras haber aplicado un filtro

3.1 Experimento (coronaryAngiogram.png):

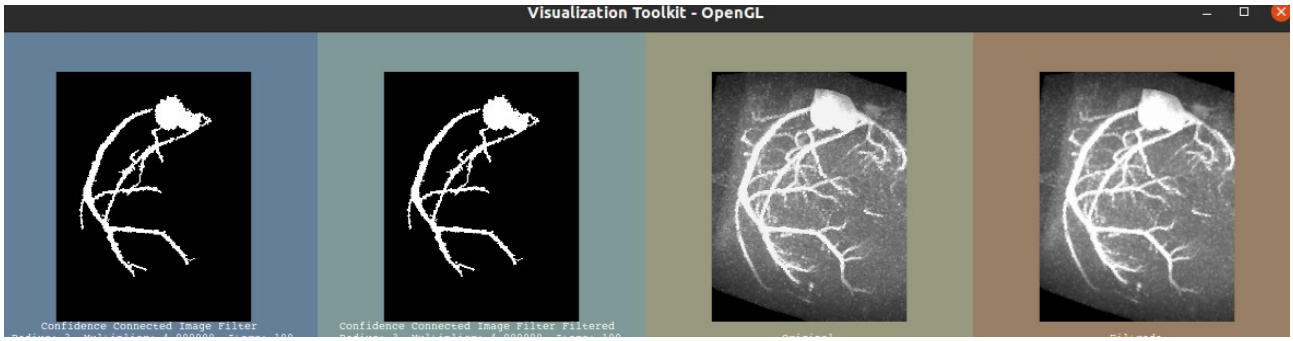
Esta vez intentaremos segmentar la ramificación de la imagen, para ello los valores que hemos obtenido en gimp de posición y color son:

Valores Gimp
Valor de X: 136
Valor Y: 172

Parámetros:

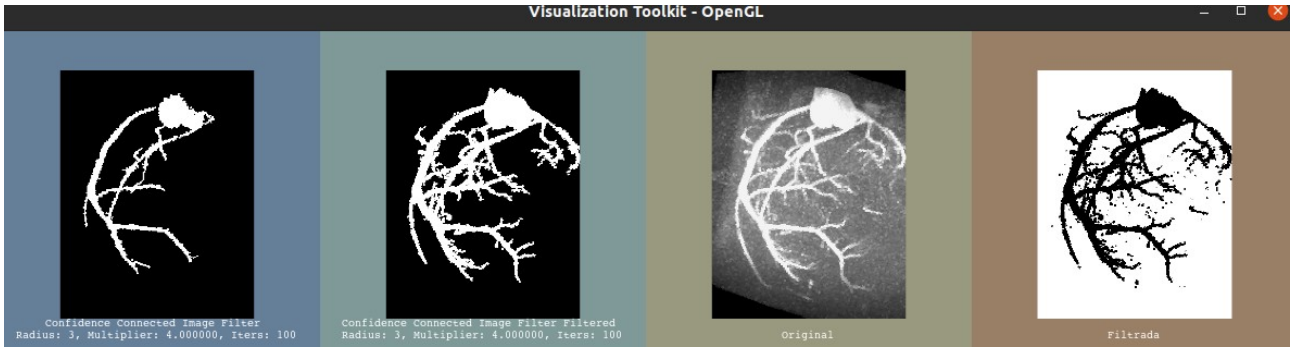
radius: 3
multiplier: 4
iters: 25

Prueba 1: 136 172 3 4 25



Prueba 2: 136 172 3 4 25

En este caso lo que hemos hecho ha sido aplicar a la imagen ya filtrada por la media un BinaryThresholdFilter:



En este experimento podemos comprobar que aplicando el preprocesamiento correcto a la imagen podemos obtener resultados mucho más buenos. Los valores de los parámetros del BinaryThresholdFilter son outside=255, inside=0, lower = 160, upper = 255.

3.2 Experimento (coronaryAngiogram.png):

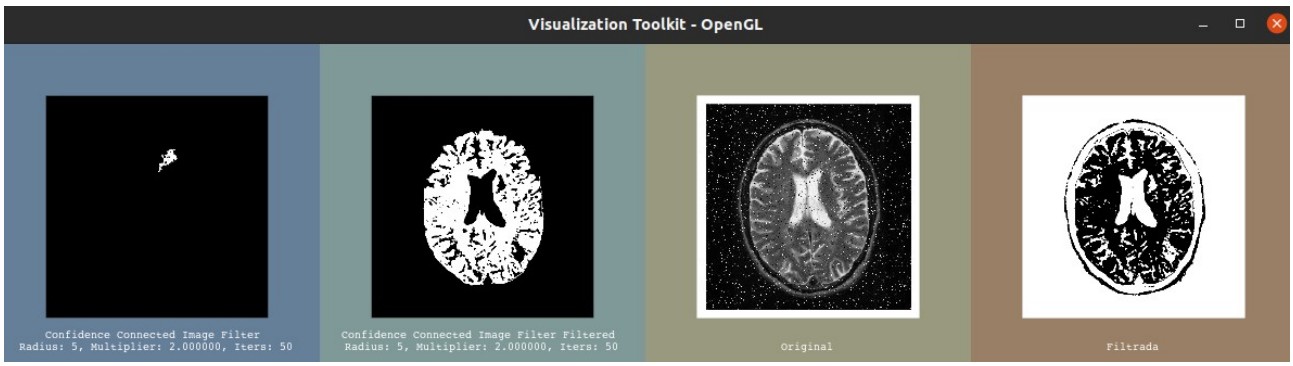
Esta vez intentaremos segmentar la parte media de la imagen, para ello el valore que hemos obtenido en gimp de posición es:

Valor de X: 306
Valor Y: 147

Parámetros:

radius: 5
multiplier: 2
iters: 50

Prueba 1: 306 147 5 2 50



Para este experimento, la primera imagen la hemos segmentado a partir de la imagen filtrada por mediana, la segunda, al igual que antes, la hemos hecho un BinaryThresholdFilter.

Concluimos diciendo que para este algoritmo de segmentación ha quedado demostrado que aplicar un BinaryThresholdFilter sobre la imagen antes de segmentar da muy buenos resultados.

Task 2

Now it’s time to put on with real stuff! The code in WatershedSegmentation1.cxx shows how to useWatershedImageFilter . I would strongly recommend to read ITK documentation of itk::WatershedImageFilter, especially the sections ‘Description of the input to this filter’ and ‘Some notes on filter parameters’.Run this

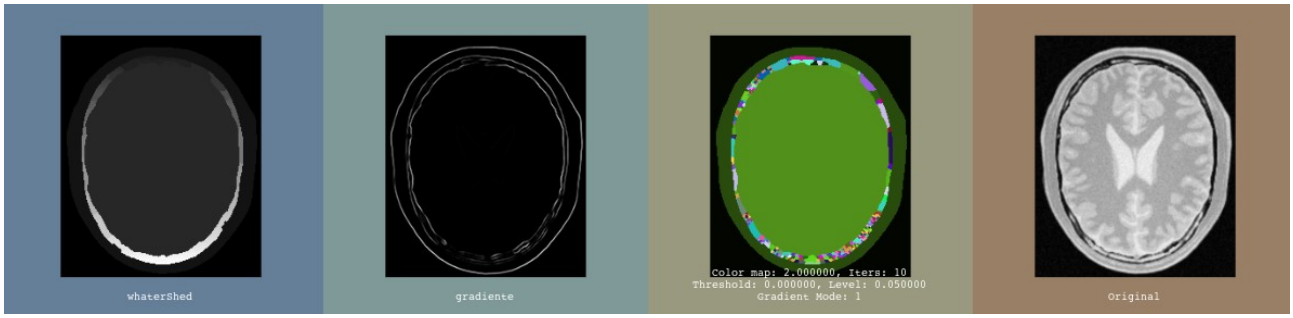
Jugaremos con los diferentes parámetros en diferentes imágenes para después discutir los resultados. Los parámetros con los que jugaremos son:

-Conductancia
-Difusión
-Umbral
-Nivel
-Gradiente (Siempre lo pondremos a 1)

2.1 BrainProtonDensitySlice256x256

Experimento 1:

-Conductancia: 2
-Difusión: 10
-Umbral: 0
-Nivel: 0.05



Como vemos, parece que necesitamos profundizar más, probemos subiendo la conductancia.

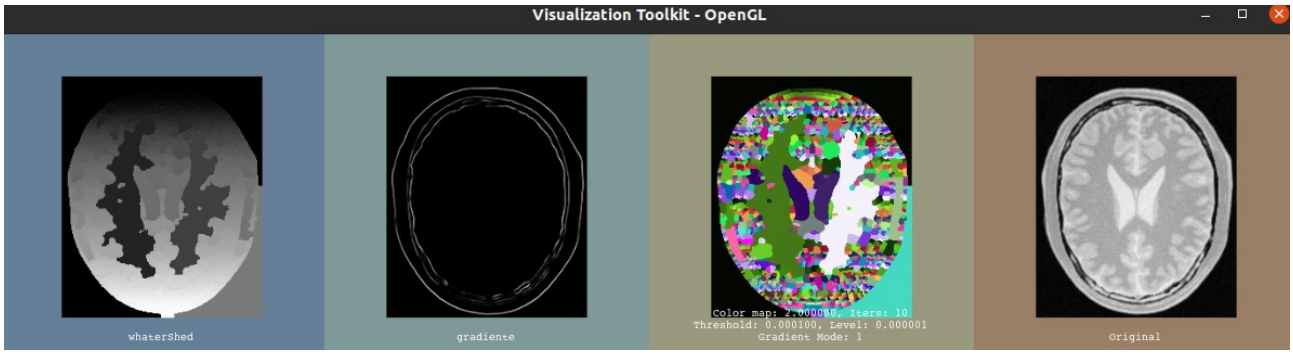
Experimento 2:

-Conductancia: 10
-Difusión: 10
-Umbral: 0
-Nivel: 0.05

Como esperábamos, el resultado a mejorado considerablemente, podemos diferenciar las partes de la imagen aunque no con toda precisión.

Experimento 3:

-Conductancia: 10
-Difusión: 2
-Umbral: 0.0001
-Nivel: 0.000001

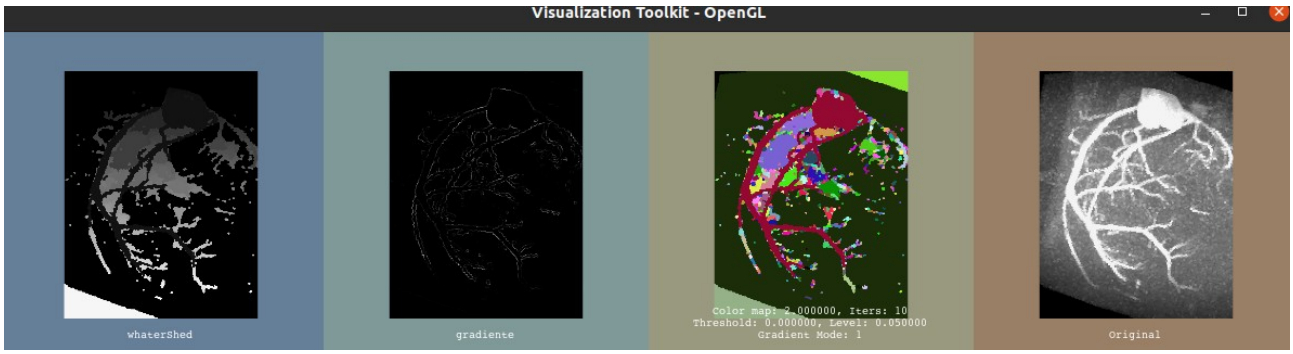


Volviendo al valor de conductancia anterior y haciendo una bajada de nivel, además de incluir un pequeño umbral que evite la sobresegmentación, hemos obtenido este resultado. Se podría decir que no hemos mejorado el resultado del experimento anterior, o al menos no de forma significativa.

2.2 coronaryAngiogram

Experimento 1:

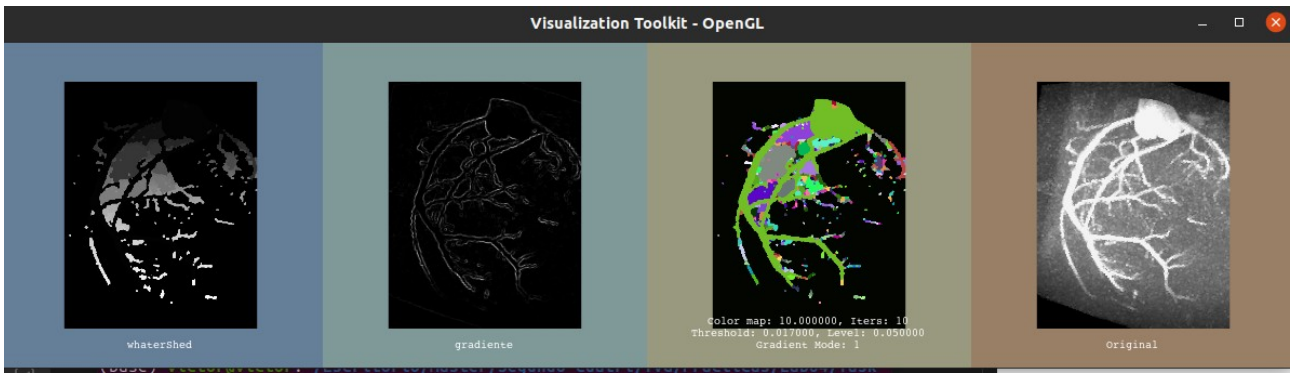
-Conductancia: 2
-Difusión: 10
-Umbral: 0
-Nivel: 0.05



Para esta imagen obtenemos un mejor resultado que en el anterior, debido posiblemente a que el contraste es más marcado y resulta más fácil al algoritmo encontrar el objetivo.

Experimento 2:

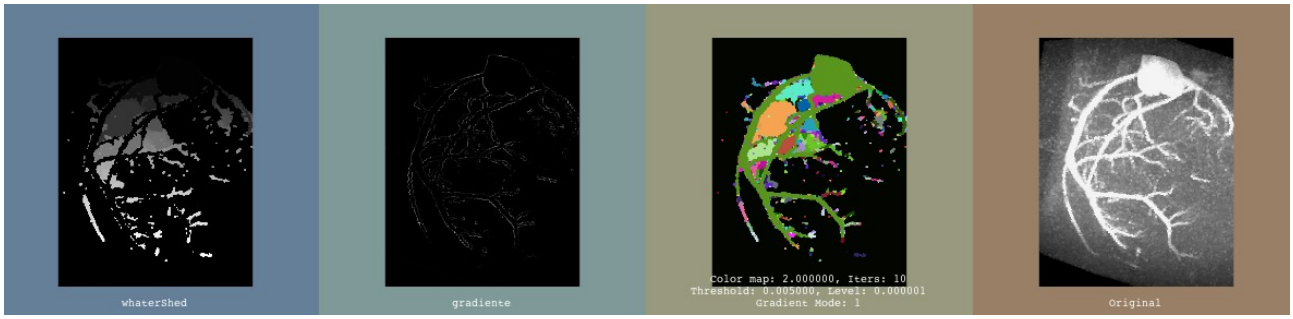
-Conductancia: 10
-Difusión: 10
-Umbral: 0.017
-Nivel: 0.05



Para este experimento, hemos necesitado colocar un umbral para evitar la sobresegmentación obteniendo unos resultados aceptablemente buenos.

Experimento 3:

-Conductancia: 10
-Difusión: 2
-Umbral: 0.005
-Nivel: 0.000001



Para este experimento hemos tenido que subir el umbral por el motivo que se comentaba en experimentos anteriores obteniendo un resultado muy similar al del experimento 2.

Concluimos diciendo que, aunque jugando con los parámetros podemos obtener resultados aceptables, esta técnica parece dar peor resultado que las utilizadas en la task 1.

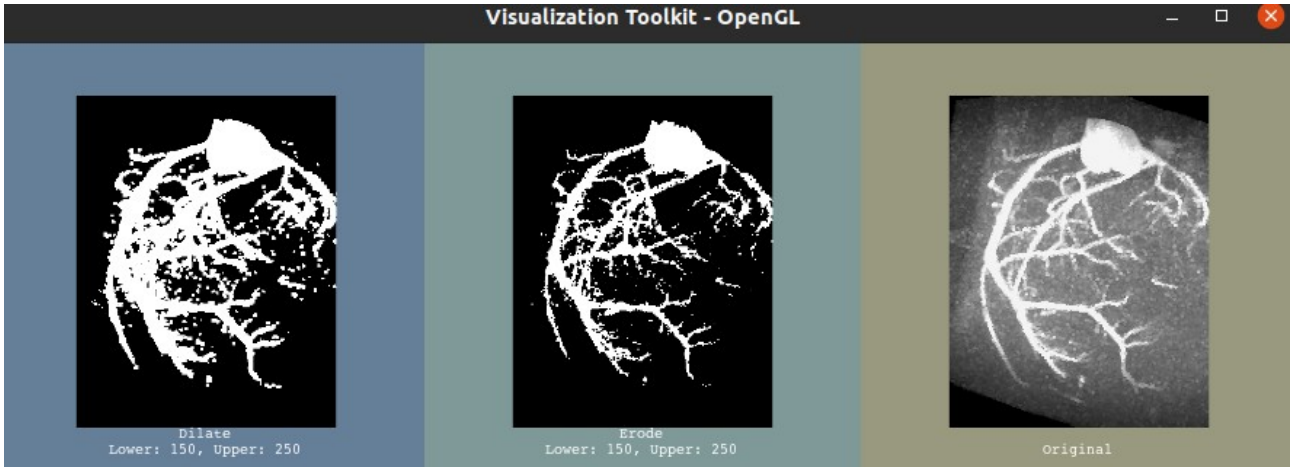
Apartado 4

Mathematical morphology filters are an important tool in image processing. ITK provides us with filters that operate on binary images and filters that operate on grayscale images. Read the section on “Mathematical Morphology” in the ITK guide2 and try the code examples.

Vamos a usar el algoritmo mathematiclMorphologyBinary (MMB).

Experimento 1

Usamos la imagen coronaryAngiogram con unos parámetros de umbral inferior y superior [150, 250]. El resultado es el siguiente:



Experimento 2

Usamos la imagen coronaryAngiogram con unos parámetros de umbral inferior y superior [180, 255]. El resultado es el siguiente:

Concluimos el apartado 4 diciendo que ajustando los umbrales en función de la imagen, de una forma sencilla, se pueden obtener buenos resultados.