Resumen sobre la implementación

Las diferentes técnicas empleadas para la segmentación de pulmón se encuentran implementadas de manera individual en el código adjunto. Fundamentalmente, se ha empleado la librería para el procesamiento de imagen médica, *The Insight Toolkit*, ITK, bien en su versión original para C++ o en su versión wrapeada para Python, SimpleITK.

ITK encapsula en clases, los diferentes filtros para el procesado de imágenes que la componen.
Para la conexión de varios objetos de tipo filtro, ITK cuenta con diversas interfaces, que resultan transparentes para los usuarios e incluso en la mayoría de ocasiones, para los desarrolladores. Esta arquitectura permite ensamblar los filtros que representan diferentes operaciones de forma inmediata, teniendo como único requisito, establecer para cada entrada de una operación de filtrado, la salida del filtro que le precede, sin interactuar con el resto del algoritmo.

Tanto para el desarrollo de la herramienta proporcionada, como para la implantación de futuras técnicas, no resulta especialmente óptimo utilizar únicamente software desarrollado en C++, si no que, puede resultar muy conveniente utilizar las funciones wrapeadas de SimpleITK, los filtros desarrollados puramente en Python, las funciones de ImageJ o Matlab, entre otros muchos ejemplos. Por esta razón, la arquitectura implementada, siguiendo la filosofía de ITK, proporciona interfaces entre distintos lenguajes de programación y entornos de procesamiento, de manera que se pueden encadenar filtros entre estos.

La versión del código facilitado, cuya arquitectura principal está implementada en Python, cuenta con tres interfaces diferentes. Dichas interfaces permiten unir filtros de ITK (C++), compilados y proporcionados en forma de ejecutables individuales, filtros de SimpleITK y los métodos desarrollados puramente en Python para el tratamiento de las imágenes.

La arquitectura se recoge principalmente en el archivo *Segmentation.py*. Dentro de este se encuentra la codificación correspondiente a la estructura de clases (ver figura) que permite ejecutar de manera encadenada la sucesión de operaciones implementadas en diferentes lenguajes.

La clase principal, *A_Filter*, se define como una clase abstracta, obligando a las clases derivadas a implementar el método *execute*, siendo en este método donde se recogen las particularidades de cada tipo de filtro empleado.
Para poder realizar filtrados encadenados, *A_Filter* cuenta con los atributos *Input_Path_image* y *Output_Path_Image* que indican las direcciones donde se almacenan las imágenes, de entrada y de salida del filtro respectivamente. Siempre que proceda, dado que no siempre se cuenta con un objeto imagen (filtros externos) o no siempre se opta por guardar la imagen resultante de un filtro (la implementación permite decidir que resultados guardar).

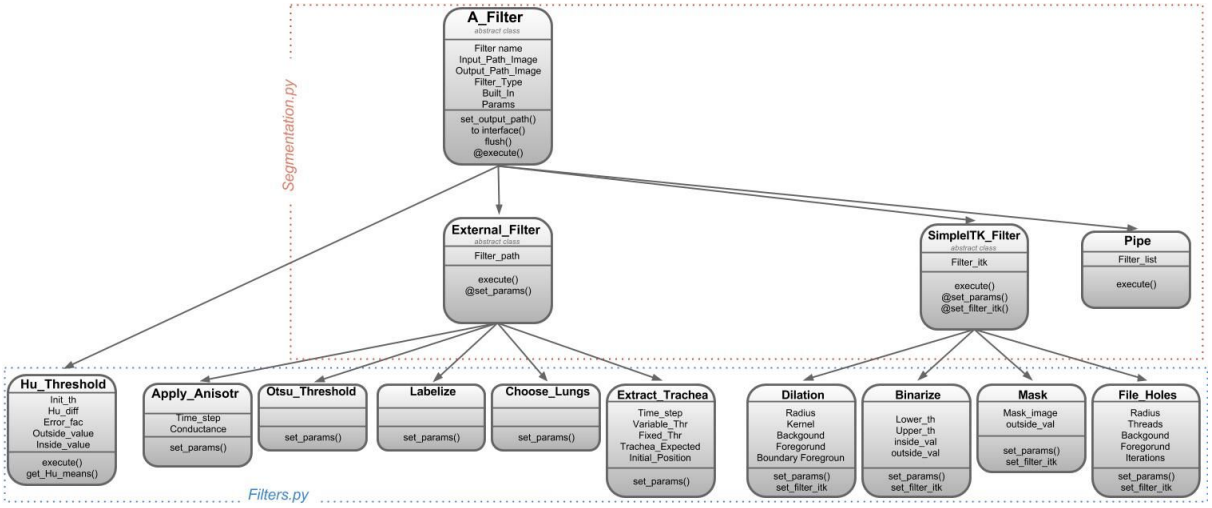
Los filtros desarrollados con Python derivan directamente de *A_Filter*.

Por otro lado, los filtros desarrollados en C++ con ITK se encadenan utilizando la clase abstracta derivada de *A_Filter*, *External_Filter*. Esta clase impone la inclusión de un método *set_params* en las clases que heredan de la misma (filtros de ITK). Este último método, necesita que se establezcan los parámetros a utilizar en las funciones (definidas en la variable *path*) que se llaman de forma externa en el método execute de *External_Filter*.
Se incluyen los siguientes filtros de tipo *External_Filter* :
• *Apply_Anisotropic*: Encargado de realizar el pre-procesado de la imagen para eliminar ruido. La implementación en C++ se adjunta en el archivo *apply_anisotropic.cxx* incluido en *utilities*.
• *Otsu_Threshold*: Utilizado en la umbralización entre tejido aireado y no - aireado. También incluido en utilities.
• *Labelize*: Filtro para establecer la conectividad entre diferentes estructuras y el filtrado morfológico de las mismas. El código fuente se encuentra en la carpeta lungs con el nombre *labelize_main.cxx*
• *Choose_Lungs*: Este filtro busca el objeto más centrado en la segmentación de los diferentes blobs. También está incluido en la carpeta lungs como *choose_lungs_main.cxx*.
• *Extract_Trachea*: Este filtro aúna todo el código necesario para el proceso complejo de segmentación de la vía aérea. El código fuente se recoge dentro de la carpeta airways en los archivos *label_air_main.cxx*, *SegmentAirwayTree_FM.cxx* y *itkTreeSegmentationImageFilter.txx*.

En el caso de la última interfaz implementada hasta el momento, los filtros wrapeados para Python de la librería de SimpleITK, se deriva de *A_Filter* la clase abstracta *SimpleITK_Filter*, que en este caso impone la definición de un objeto filtro de SimpleITK y de sus parámetros.

Dentro del fichero *Filters.py* se encuentran cuatro clases derivadas de *SimpleITK_Filter* :
Dilation y *Binarize* encargadas de realizar operaciones morfológicas y binarizaciones en varios procesos; *Mask* utilizada para extraer la vía aérea de la imagen original y obtener el resultado final de los pulmones segmentados mediante enmascarado y *File_Holes* para el postprocesado que permite incluir las lesiones.

Finalmente, para facilitar la creación de flujos de trabajo específicos (por ejemplo, diferentes modelos animales), se introduce la clase *Pipe*. Dicha clase contiene una lista de filtros que se derivan de la clase *A_Filter*, y que se encadenan tras la llamada al método execute.
El resto de archivos que componen el código fuente incluyen funciones de ayuda para realizar diferentes tareas básicas a lo largo del procesado, como las que aparecen en el archivo *UsefullFunctions.py*.



Instalación y uso

Se proporciona el script *install.sh* dentro de la carpeta *TLS*. Este script se encarga de buscar y descargar todos los paquetes necesarios para el funcionamiento de la aplicación, así como de su compilación.
El script puede ser ejecutado con un único parámetro opcional, el número de núcleos a utilizar en las operaciones de compilación de código, por lo tanto este será ejecutado como:

```
$ sudo ./install [número de núcleos]
```

El script crea la carpeta *TLS_BUILD* en el mismo directorio donde se encuentre *TLS* e incluirá los archivos binarios necesarios para utilizar la herramienta.

Por último, a modo de ejemplo paradigmático de uso, se incluye el script de python *Application.py* (en el interior de *TLS*). Dentro del cual se ha creado un *Pipe* específico para las imágenes proporcionadas[1] (dentro de la función *pamplona_lungs_segmentation*) que segmenta la imágenes proporcionadas y guarda los resultados obtenidos en la carpeta de *Results*.

[1] Notar que las imágenes han sido modificadas ligeramente para hacer coincidir la tráquea en el primer corte axial.