UNIVERSIDAD DE GRANADA

Laboratorio 5: Lectura y escritura de imágenes con ITK

TVG curso:2024/2025

Adriano García - Giralda Milena
DNI:77944452-M adrianoggm@correo.ugr.es Grupo 1

21 de junio de 2025

Índice

1. Introducción

En esta práctica se ha realizado la experimentación y realización de diversas técnicas en un total de 8 task y 2 ejercicios. He de destacar que a partir de la task3 he tenido que cambiar a python el lenguaje de programación debido que al usar la wsl la vista de visualización de quickview no funciona.

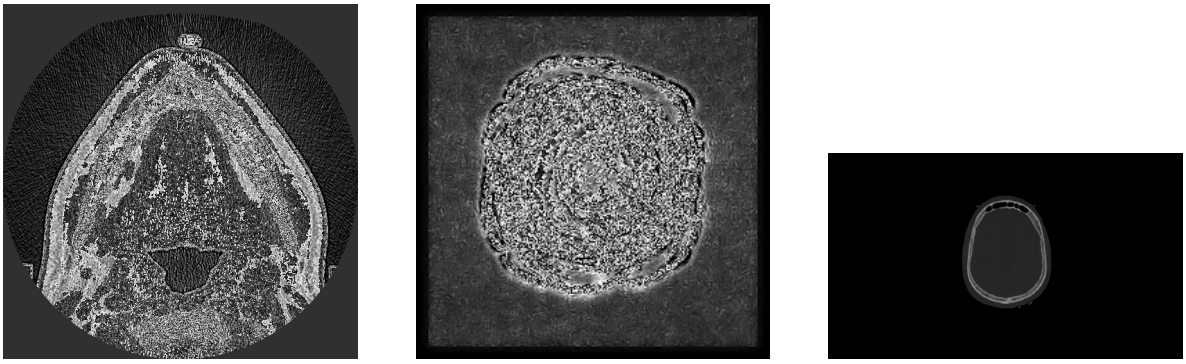
2. Tarea 1

En esta tarea, el objetivo es desarrollar un programa que implemente un conversor de formatos de imagen, aprovechando las capacidades que proporciona la librería ITK. Para ello, se debe compilar el ejemplo `ImageReadWrite.cxx` ubicado en el directorio `ITK_DIR/Examples/IO`.

Es recomendable encapsular cualquier llamada a `Update()` dentro de un bloque `try/catch`, para capturar posibles excepciones durante la ejecución del *pipeline*, tal y como se muestra en el Listado 1.

Mediante la adaptación del código proporcionado junto con fragmentos de prácticas anteriores, se obtiene un programa capaz de transformar y actualizar el formato de la imagen pasada como argumento.

Si ejecutamos el programa con las imágenes `DicomImage1.dcm`, `t50.bmp` e `IMG113`, se obtienen los siguientes resultados:



(a) IMG113. **(b)** DicomImage1 . **(c)** t50.

Figura 1: Imágenes tras ser preprocesadas y convertidas a png.

3. Tarea 2

En esta tarea se ejecuta un ejemplo que ilustra la instanciación explícita de una clase *ImageIO* (en este caso, para archivos VTK), configurando sus parámetros y conectándola a la clase `ImageFileWriter`. El código fuente se encuentra en `ImageReadExportVTK.cxx`.

Se recomienda consultar la sección correspondiente en la guía de ITK y probar el programa con la imagen `BrainProtonDensity3slices.mha`, disponible en `ITK_DIR/Examples/Data`.

Aunque el ejemplo ilustra únicamente cómo usar una clase `ImageIO` explícita con `ImageFileWriter`, también puede hacerse lo mismo con `ImageFileReader`. Este enfoque es habitual al leer archivos `.raw`

utilizando el objeto `RawImageIO`. El inconveniente de esta técnica es que los parámetros de la imagen deben especificarse manualmente en el código.

El uso directo de archivos `.raw` está fuertemente desaconsejado en imagen médica. Es preferible utilizar formatos que incluyan una cabecera, como `MetaImageIO`, `GiplImageIO` o `VTKImageIO`.

En esta tarea se realiza la lectura de imágenes con formatos distintos a `.png`, concretamente `.raw` y `.mha`. Como los archivos `.raw` carecen de metainformación, es necesario proporcionar una cabecera adecuada. En nuestro caso, se ha utilizado una cabecera `.mha` que referencia a la imagen `.raw` correspondiente.

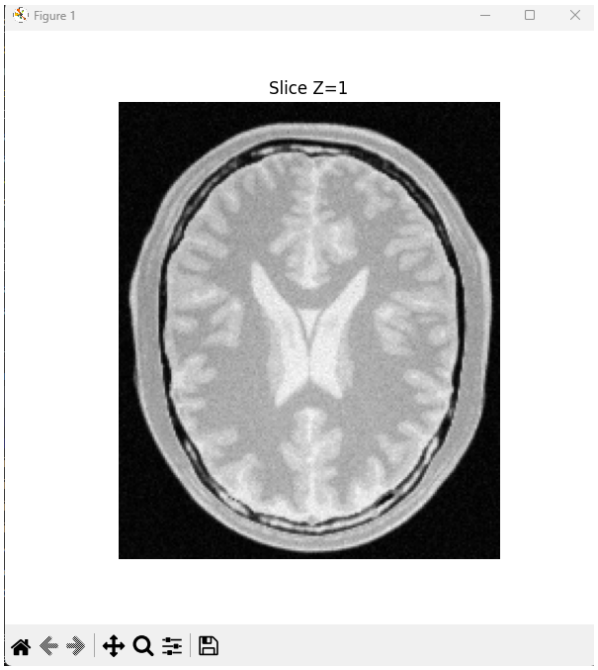


Figura 2: Código que formatea el archivo `.raw`.

Tras ejecutar el código, se obtienen los siguientes archivos de salida:

- `BrainProtonDensity3slices.mha`
- `BrainProtonDensity3slices.raw`
- `BrainProtonDensity3slicesmha.vtk`
- `BrainProtonDensity3slicesraw.vtk`

Todos representan la misma imagen, aunque visualmente pueden diferir debido a los parámetros de la cabecera, como se explicará en la Tarea 6.

4. Tarea 3

Esta tarea muestra cómo leer y escribir imágenes en color (RGB) desde y hacia un archivo. Para ello, se debe compilar el ejemplo `RGBImageReadWrite.cxx` y ejecutarlo con una imagen a color, como `VisibleWomanHeadSlice.png`, disponible en `ITK_DIR/Examples/Data`.

Dado que surgieron problemas al visualizar imágenes con el visor nativo, se optó por representarlas utilizando la biblioteca `matplotlib` en Python, como se muestra en el script `task3.py`. La imagen utilizada fue:

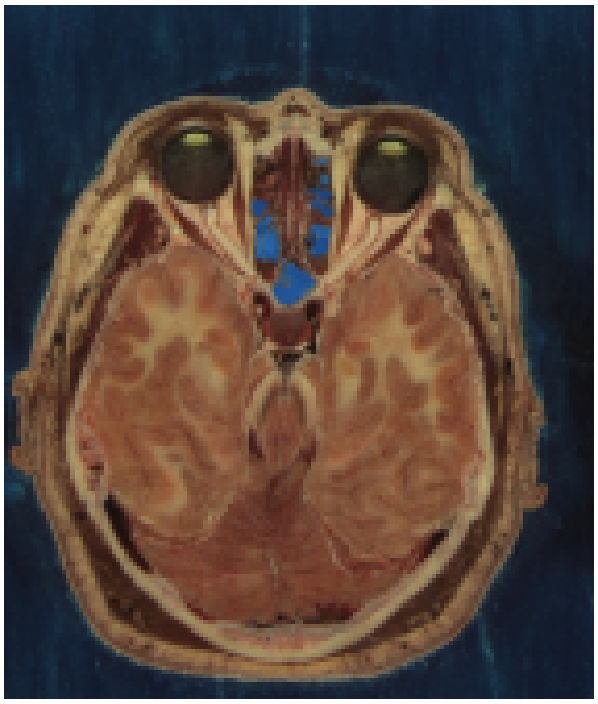


Figura 3: Visualización de la imagen `VisibleWomanHeadSlice.png` en el visor.

La salida obtenida en el visualizador coincide con la esperada, mostrando correctamente la imagen tras su carga y renderizado:

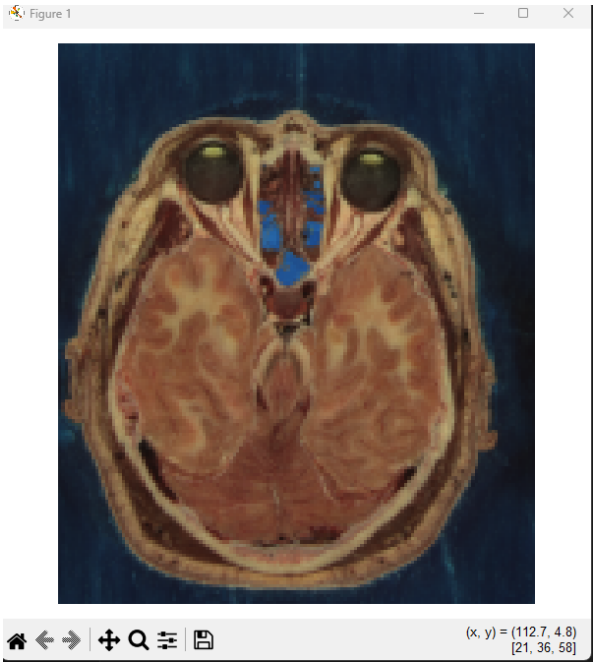


Figura 4: Visualización generada por el visor: resultado equivalente al archivo original.

5. Tarea 4

Esta tarea consiste en compilar y ejecutar el ejemplo `ImageReadCastWrite.cxx`, que muestra cómo realizar correctamente un *casting* de tipo de imagen dentro de un *pipeline*. A continuación, se desarrolla un programa que lee una imagen, aplica un filtro de rescalado de intensidad y muestra la imagen original junto a la filtrada.

Se utilizó la imagen `BrainMidSagittalSlice.png` como entrada. A continuación se muestra la imagen original junto con su versión tras la conversión de tipo de píxel (cast):

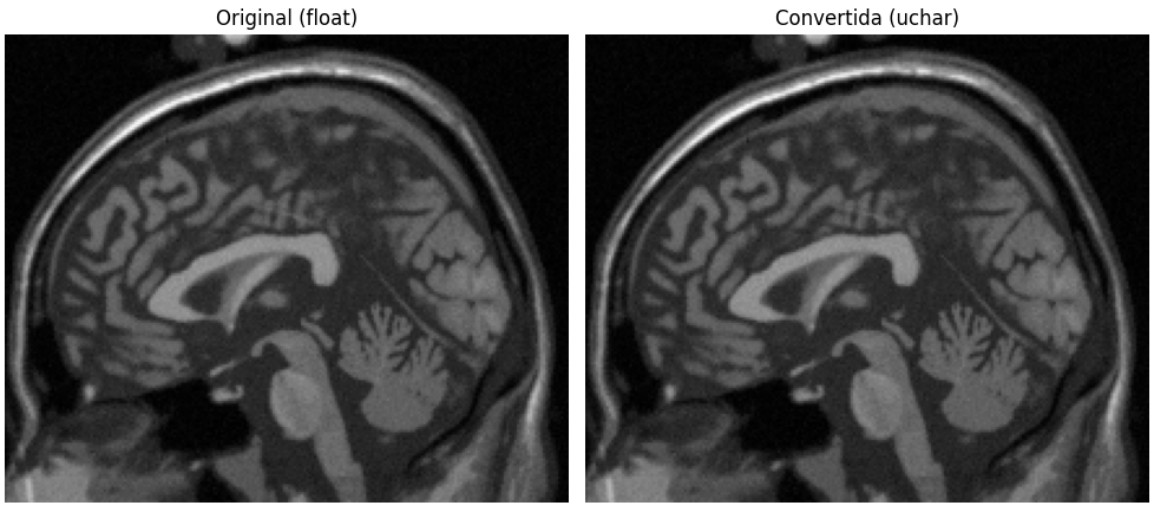


Figura 5: Comparativa entre la imagen original (`BrainMidSagittalSlice.png`) y la imagen tras aplicar `CastImageFilter`.

Observamos cómo no existen diferencias apreciables al ojo humano, pese a haber cambiado el tipo de la imagen.

6. Tarea 5

Esta tarea ilustra cómo leer una imagen y definir una región de interés (ROI) a partir de parámetros proporcionados, para posteriormente escribir dicha región a un nuevo archivo. El ejemplo está implementado en `ImageReadRegionOfInterestWrite.cxx` y debe ejecutarse con una imagen del directorio `ITK_DIR/Examples/Data`.

El programa recibe como argumento la imagen y la localización de la ROI, junto con su tamaño (por ejemplo: 50x50 píxeles centrados en una posición específica). Al ejecutarlo, se recortan los valores fuera de la región deseada, conservando únicamente la zona de interés.

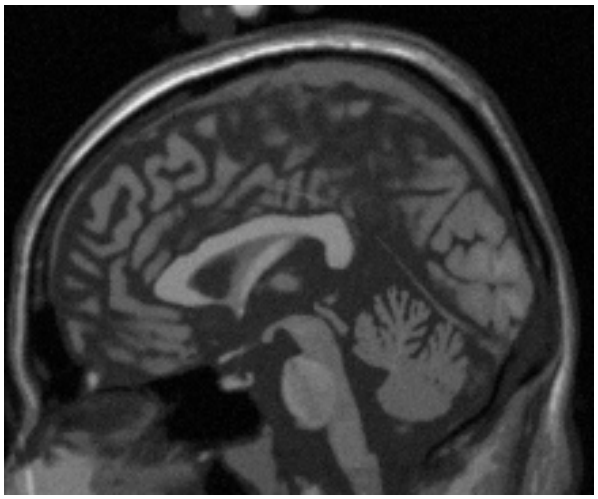


Figura 6: Imagen original: `BrainMidSagittalSlice.png`.

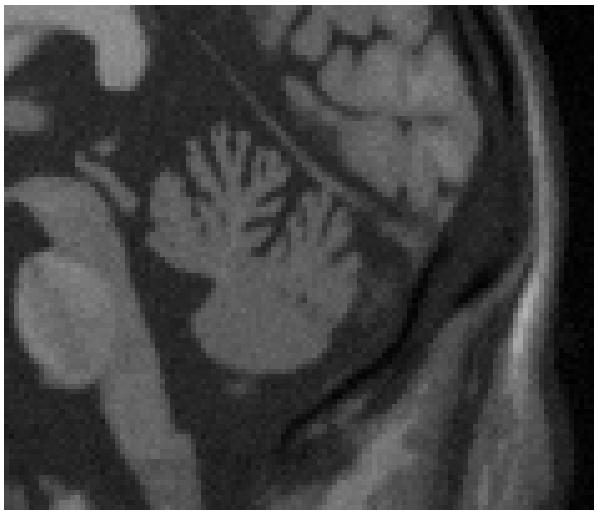


Figura 7: Imagen tras seleccionar la región de interés.

7. Tarea 6

En esta tarea se explora el proceso de extracción de una *slice* 2D a partir de un volumen 3D, utilizando el filtro `ExtractImageFilter` de ITK. La imagen de entrada puede estar almacenada en un archivo `.mha` o en un conjunto `.raw` con su correspondiente cabecera. La funcionalidad implementada permite obtener una rebanada del volumen a lo largo del eje Z (índice 2), lo cual se logra colapsando la dimensión correspondiente en la definición de la región de interés.

Conceptos técnicos

Para entender el funcionamiento de esta tarea, conviene recordar que las imágenes 3D tienen dimensiones asociadas a los ejes X, Y y Z, con índices 0, 1 y 2 respectivamente. El filtro de extracción toma como región de entrada la región completa de la imagen original, y se le indica que la nueva dimensión colapsada (con tamaño cero) será la del eje Z, es decir, se extrae una imagen bidimensional del plano XY correspondiente a una coordenada k determinada.

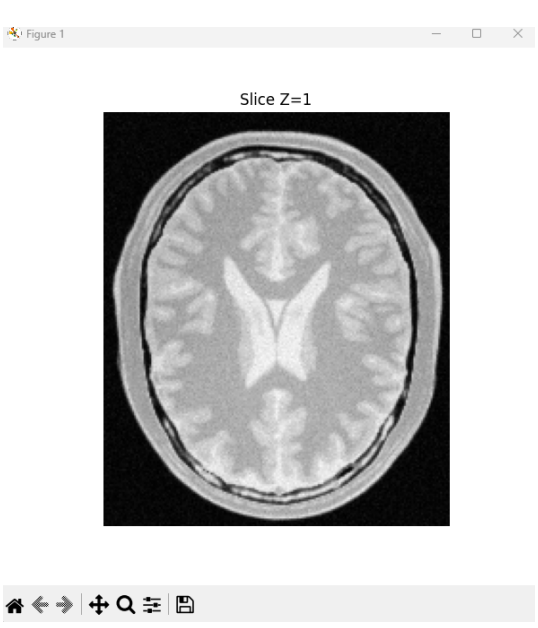
Además, se detecta si el archivo de entrada es un volumen `.raw`. En ese caso, se construye dinámicamente una cabecera `.mhd` temporal que incluye los metadatos necesarios (dimensiones, tipo de dato, espaciado, orden de los bytes, etc.) para que ITK pueda interpretar correctamente el contenido binario del archivo. Esta solución permite una lectura segura y reproducible de archivos crudos que, por sí solos, carecen de metainformación.

Visualización y ejecución

El script desarrollado permite pasar como argumentos el volumen 3D, la ruta de salida para la slice extraída y el índice k de la rebanada deseada. Si no se especifica el índice, se selecciona automáticamente la slice central del volumen.

A continuación, se presentan los resultados obtenidos al aplicar este procesamiento a dos volúmenes equivalentes, visualizando la **slice 1** de cada uno:

- `BrainProtonDensity3slices.mha`
- `BrainProtonDensity3slices.raw` (con su cabecera correspondiente)



(a) Slice extraída del volumen `.mha`.

(b) Slice extraída del volumen `.raw` con cabecera asociada.

Figura 8: Comparativa de la slice 1 extraída de ambos volúmenes procesados.

Como se puede observar, las imágenes resultantes son prácticamente idénticas, lo cual confirma que ambos archivos contienen los mismos datos volumétricos y que la cabecera generada para el archivo `.raw` es correcta.

8. Tarea 7 y 7.2

Descripción general

Esta tarea se centra en la creación de un volumen 3D a partir de una serie de imágenes 2D (`t50.bmp` hasta `t60.bmp`), y su posterior visualización como slices individuales. El objetivo es convertir dichas imágenes en un conjunto volumétrico en formato `.mhd/.raw` y comprobar que el volumen generado es correcto visualmente.

La versión adaptada del código de `Task 7.2` mejora la compatibilidad con los formatos BMP y permite visualizar de forma cómoda todas las rebanadas del volumen. Además, el código se ha modificado para aceptar parámetros de entrada como el directorio donde se encuentran las imágenes y la ruta de salida del archivo `.mhd` generado.

Proceso realizado

1. Se recorren todos los archivos `t50.bmp`, `t51.bmp`, ..., `t60.bmp`, asegurándose de que existan.
2. Se leen las imágenes utilizando la clase `ImageSeriesReader` de ITK, lo que permite cargar múltiples slices como un único volumen tridimensional.
3. Una vez creado el volumen, se guarda como un archivo en formato `MetaImage` (`.mhd + .raw`).
4. Finalmente, se visualiza el volumen generado mostrando todas sus slices en una cuadrícula.

Este flujo automatiza el proceso de reconstrucción volumétrica a partir de imágenes secuenciales bidimensionales, una técnica común en la creación de datasets en imagen médica.

Visualización del volumen generado

En la siguiente figura se muestran las slices obtenidas a partir de los archivos BMP procesados. Se puede observar una progresión suave entre las rebanadas, lo cual indica que el volumen ha sido correctamente ensamblado:

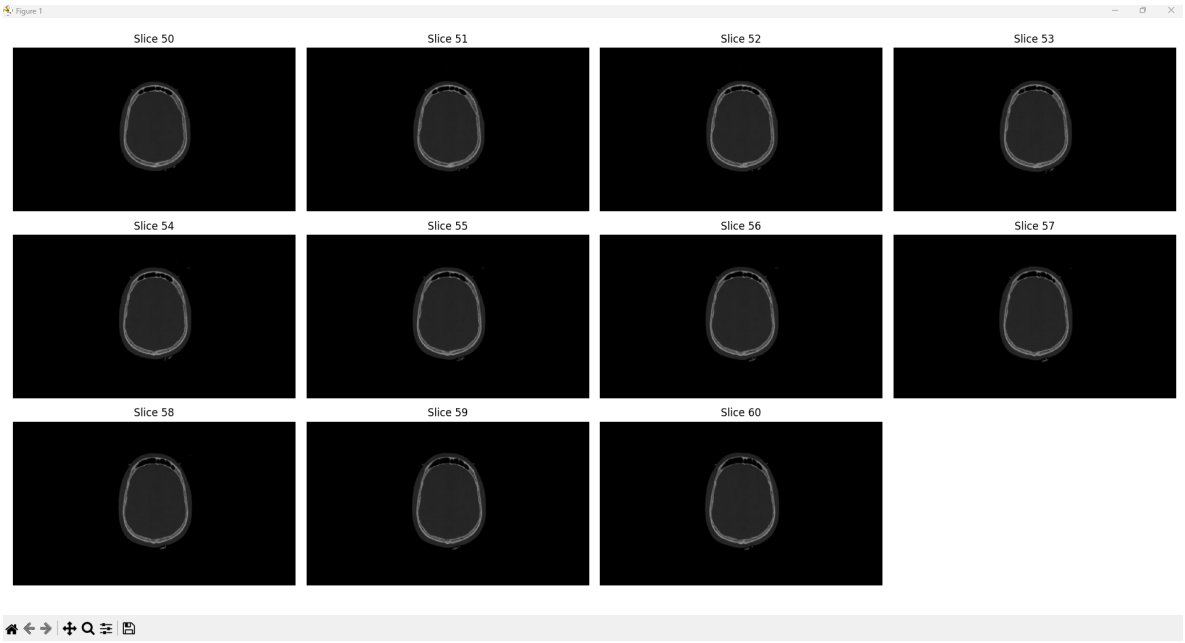


Figura 9: Visualización en cuadrícula de las slices extraídas del volumen construido a partir de los archivos `t50.bmp` hasta `t60.bmp`.

Este ejercicio muestra cómo convertir una serie de imágenes 2D en un volumen 3D compatible con ITK, lo cual resulta útil en tareas de preprocesamiento para segmentación, visualización o entrenamiento de modelos de aprendizaje automático en imágenes volumétricas. También demuestra el uso práctico de filtros como `ExtractImageFilter` para manipular y visualizar secciones del volumen resultante.

9. Tarea 8

Lectura y visualización de imágenes DICOM

DICOM (Digital Imaging and Communications in Medicine) es el estándar internacional para el almacenamiento, transmisión y visualización de imágenes médicas digitales. Está mantenido por la asociación NEMA y se utiliza ampliamente en hospitales y centros clínicos, integrando modalidades como TAC, RM, ecografía, etc. Una de sus principales ventajas es que un solo archivo DICOM puede contener tanto la imagen como metadatos clínicos esenciales (nombre del paciente, fecha, parámetros del equipo, etc.).

En ITK, el soporte para DICOM se implementa a través de la biblioteca GDCM (*Grassroots DICOM*), desarrollada por el equipo CREATIS. Esta biblioteca permite tanto la lectura de imágenes DICOM individuales como el manejo de series completas (p. ej. cortes axiales de una tomografía).

Objetivo de la tarea

El propósito de esta tarea es desarrollar un script capaz de leer y visualizar imágenes DICOM individuales (archivos `.dcm`) o directorios que contengan una serie DICOM. Para ello, se utiliza la

clase `ImageSeriesReader` de `SimpleITK`, que detecta automáticamente las distintas series disponibles en un directorio y permite cargarlas correctamente.

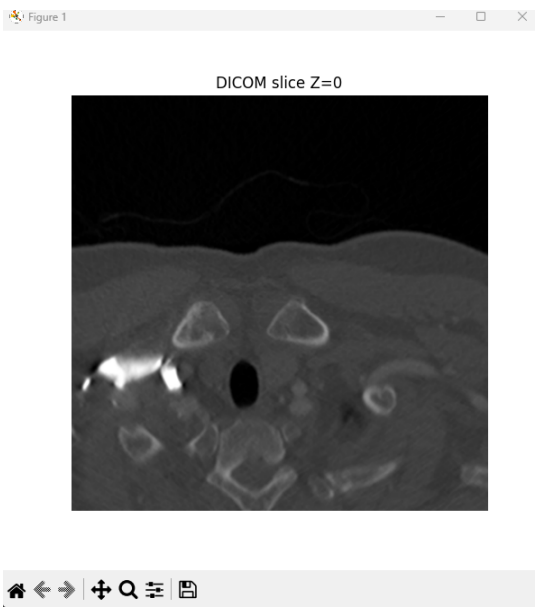
El programa acepta como argumento un archivo DICOM o un directorio que contenga una serie, y permite mostrar una rebanada concreta del volumen (índice Z) si se trata de una imagen tridimensional. Si se proporciona una imagen 2D, simplemente se visualiza sin necesidad de extracción.

Funcionamiento general

1. Si se proporciona un directorio, el programa busca series DICOM válidas y carga la primera disponible.
2. Si se proporciona un único archivo `.dcm`, se carga directamente.
3. Si la imagen es tridimensional ($Z > 1$), se extrae el frame o slice indicado por el usuario mediante el filtro `ExtractImageFilter`.
4. Finalmente, la imagen se visualiza usando `matplotlib`.

Visualización del resultado

En la siguiente figura se muestra una slice extraída de una serie DICOM leída desde un directorio. El índice Z fue proporcionado como argumento. Para el caso $Z = 0$, se obtienen los siguientes resultados:



(a) Slice extraída con índice $Z = 0$ desde el directorio DICOM2.

(b) Visualización alternativa o procesada de la misma slice ($Z = 0$) para DICOM1.

Figura 10: Extracción de la slice correspondiente a $Z = 0$ a partir de una serie DICOM.

El manejo correcto de imágenes DICOM es fundamental para cualquier flujo de trabajo clínico o investigación biomédica. Esta tarea sienta las bases para realizar análisis más avanzados como segmentación, reconstrucción tridimensional o fusión multimodal. Además, permite validar que los volúmenes DICOM se estén interpretando correctamente en términos de orden, espaciado y orientación, aspectos críticos para garantizar resultados clínicamente relevantes.

Ejercicio 1: Conversión de DICOM a PNG

Objetivo

El propósito de este ejercicio es desarrollar un programa que permita convertir una imagen en formato DICOM (`.dcm`) a una imagen en formato PNG, lo cual resulta útil para visualizar imágenes médicas en herramientas estándar o para incluirlas en informes sin necesidad de software especializado.

El programa acepta como argumentos:

- La ruta del archivo de entrada en formato DICOM.
- La ruta de salida con terminación `.png`.

Funcionamiento

El programa realiza los siguientes pasos:

1. Lee la imagen DICOM con `SimpleITK`. Si la imagen es tridimensional (por ejemplo, una serie de slices en un solo archivo), se extrae únicamente la primera rebanada ($Z = 0$).
2. Si la imagen no está en un formato de 8 bits (por ejemplo, si es de 16 bits o tipo `float`), se reescala su rango dinámico al intervalo $[0, 255]$ y se convierte a tipo `UInt8`, adecuado para el formato PNG.
3. Se guarda la imagen resultante en formato PNG en la ruta proporcionada.

Este flujo asegura que cualquier imagen DICOM válida se convierta correctamente a un formato estándar, sin pérdida significativa de información para propósitos de visualización.

Aplicación del ejercicio

Se aplicó este conversor a las dos imágenes DICOM proporcionadas en el directorio de datos:

- `DicomImage1.dcm` → `DicomImage1.png`
- `DicomImage2.dcm` → `DicomImage2.png`

Estas imágenes PNG generadas pueden abrirse con cualquier visor estándar. A continuación se muestran las visualizaciones obtenidas:

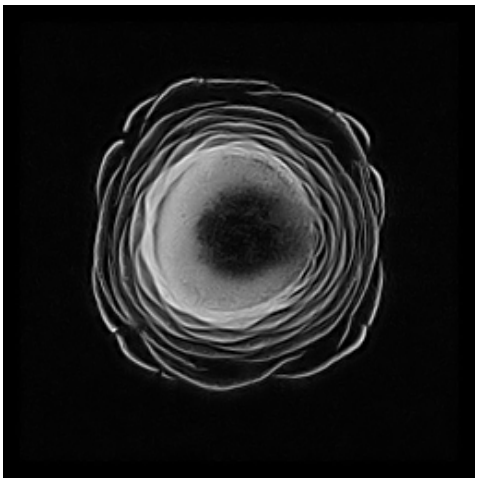


Figura 11: Conversión de `DicomImage1.dcm` a PNG.

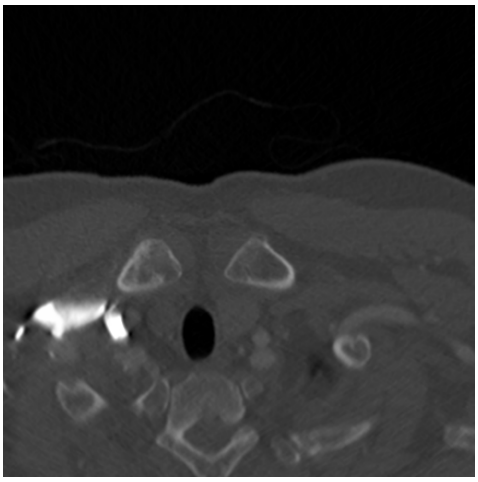


Figura 12: Conversión de `DicomImage2.dcm` a PNG.

Conclusiones

Este ejercicio demuestra cómo transformar imágenes médicas del estándar DICOM a un formato visual generalista, útil tanto para fines diagnósticos preliminares como para generación de informes o análisis posteriores con herramientas no especializadas.

Ejercicio 2: Lectura de una serie DICOM 2D y construcción de un volumen 3D

Objetivo

El objetivo de este ejercicio es agrupar una serie de imágenes DICOM bidimensionales (`IMG113.dcm` hasta `IMG130.dcm`) en un único volumen 3D y guardarlo como un archivo en formato `MetaImage` (`.mhd + .raw`). Esta operación es fundamental en flujos de trabajo donde las imágenes obtenidas en cortes axiales, sagitales o coronales deben combinarse en una estructura volumétrica común para análisis tridimensional o visualización médica.

Funcionamiento del programa

El script desarrollado permite:

1. Leer una secuencia ordenada de imágenes DICOM con nombres como `IMG113`, `IMG114`, ..., `IMG130`, aceptando múltiples extensiones posibles (por ejemplo, con o sin `.dcm`).
2. Cargar las imágenes como slices 2D mediante `ImageSeriesReader`, que las combina en un único volumen tridimensional.
3. Imprimir en consola los metadatos del volumen creado para comprobar que se han conservado correctamente durante el proceso de agregación.
4. Guardar el volumen final en formato `.mhd`, listo para ser usado con visores volumétricos o algoritmos de procesamiento.

Metadatos verificados

Tras generar el volumen, se imprimieron las claves de metadatos asociadas. Entre ellas destacan:

- `0010|0010` – Nombre del paciente.
- `0008|0060` – Modalidad de adquisición (CT, MR, etc.).
- `0028|0030` – Spacing de píxeles.
- `0020|0032` – Posición del paciente.

La correcta conservación de estos metadatos es esencial para asegurar una interpretación clínica válida y para mantener el contexto original del estudio.

Visualización del volumen

El volumen resultante fue posteriormente visualizado utilizando el script desarrollado en la **Tarea 7 (7.2)**, que permite mostrar en una cuadrícula todas las slices del volumen. La visualización permite comprobar la continuidad entre las rebanadas y validar visualmente que el ensamblado ha sido correcto.

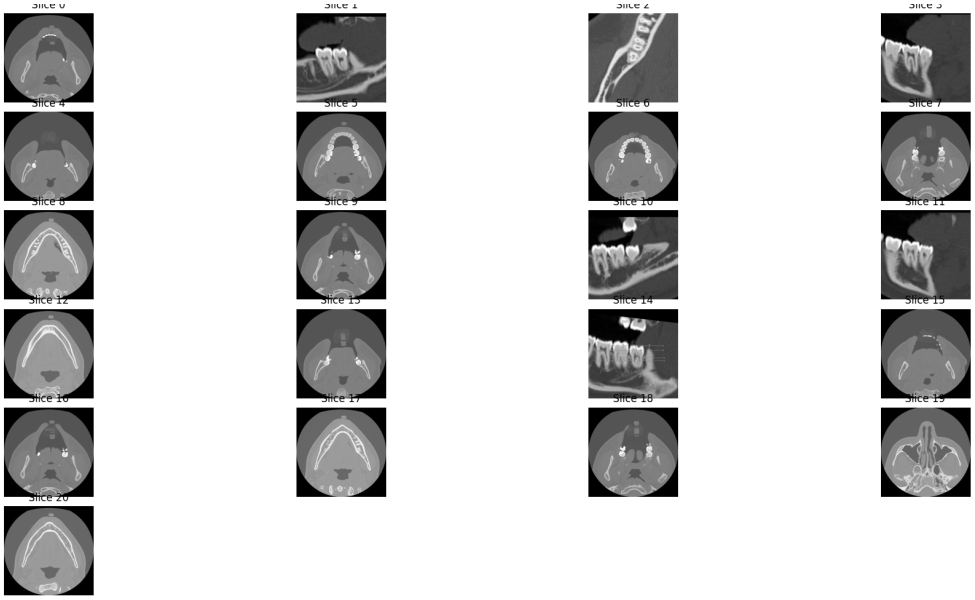


Figura 13: Visualización del volumen guardado en formato `.mhd` (MetaImage).

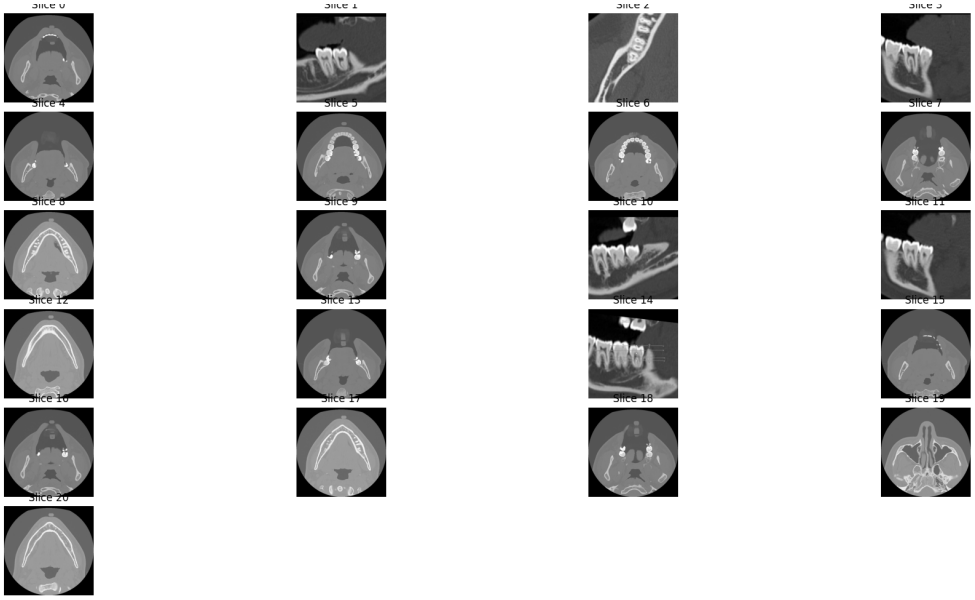


Figura 14: Visualización del volumen reconstruido desde archivo `.raw` con cabecera asociada.

Conclusión

Este ejercicio demuestra cómo construir un volumen tridimensional a partir de una serie de imágenes DICOM bidimensionales numeradas. Este proceso es ampliamente utilizado en imagen médica para convertir datos adquiridos por planos en representaciones volumétricas útiles para diagnóstico, segmentación o navegación quirúrgica. La correcta lectura de los archivos, conservación de los metadatos y escritura en formato estándar aseguran la interoperabilidad del volumen generado con otros sistemas y herramientas.