



UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MÁSTER
INGENIERÍA EN INFORMÁTICA

3DCurator

Sistema gráfico de ayuda al diagnóstico e intervención de esculturas mediante datos médicos volumétricos

Autor

Francisco Javier Bolívar Lupiáñez

Director

Francisco Javier Melero Rus



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, 10 de junio de 2018



3DCurator

Sistema gráfico de ayuda al diagnóstico e intervención de esculturas mediante datos médicos volumétricos

Autor

Francisco Javier Bolívar Lupiáñez

Director

Francisco Javier Melero Rus

3DCurator: Sistema gráfico de ayuda al diagnóstico e intervención de esculturas mediante datos médicos volumétricos

Francisco Javier Bolívar Lupiáñez

Palabras clave: informática gráfica, renderizado de volúmenes, tomografía computarizada, escultura policromada de madera, conservación y restauración, restaurador de arte

Resumen

Lala

3DCurator: Graphic system to aid the diagnosis and intervention of sculptures using volumetric medical data

Francisco Javier Bolívar Lupiáñez

Keywords: Computer Graphics, Volume Rendering, Computed Tomography, Polychromed Wood Sculpture, Conservation-Restoration of Cultural Heritage, Art Curator

Abstract

Lala

Yo, **Francisco Javier Bolívar Lupiáñez**, alumno de la titulación Máster en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación**, con DNI 75926571Y, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Francisco Javier Bolívar Lupiáñez

Granada a 10 de junio de 2018.

D. **Francisco Javier Melero Rus**, Profesor del Área de Lenguajes y Sistemas Informáticos del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado *3DCurator, Sistema gráfico de ayuda al diagnóstico e intervención de esculturas mediante datos médicos volumétricos*, ha sido realizado bajo su supervisión por **Francisco Javier Bolívar Lupiáñez**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 10 de junio de 2018.

El director:

Francisco Javier Melero Rus

Agradecimientos

Lala

Índice general

1. Introducción	1
1.1. Tomografía Computarizada	2
1.1.1. Historia	2
1.1.2. Generaciones	2
1.2. Esculturas de madera policromadas	7
1.2.1. Historia	7
1.2.2. Maderas más utilizadas	7
1.2.3. Defectos de la madera	8
1.2.4. Proceso de tallado	9
1.3. Estado del arte	10
1.4. Motivación	11
2. Especificación de requisitos	13
2.1. Introducción	13
2.1.1. Propósito	13
2.1.2. Ámbito del sistema	13
2.1.3. Definiciones, acrónimos y abreviaturas	13
2.1.4. Visión general del documento	16
2.2. Descripción general	17
2.2.1. Perspectiva del producto	17
2.2.2. Funciones del producto	17
2.2.3. Características de los usuarios	17
2.2.4. Restricciones	18
2.2.5. Suposiciones y dependencias	18
2.3. Requisitos específicos	18
2.3.1. Interfaces	18
2.3.2. Funciones	20
2.3.3. Requisitos de rendimiento	22
2.3.4. Restricciones de diseño	22
2.3.5. Atributos del software	22

3. Planificación	23
3.1. Planificación inicial	23
3.1.1. Diagrama de Gantt	24
3.2. Metodología de trabajo	24
3.2.1. <i>GitHub</i>	24
3.2.2. <i>Issues</i>	24
3.2.3. <i>Milestones</i>	25
3.2.4. <i>Branches</i>	26
3.3. Resultados	28
4. Análisis	29
4.1. Historias de usuario	29
4.1.1. <i>Product backlog</i>	29
5. Diseño	31
5.1. Arquitectura del software	31
5.2. Diagrama de paquetes	32
5.3. Diagramas de clases	32
5.3.1. <i>Chart</i>	33
5.3.2. <i>Core</i>	33
5.3.3. <i>Documentation</i>	34
5.3.4. <i>GUI</i>	35
5.3.5. <i>Interactor</i>	37
5.3.6. <i>Segmentation</i>	37
5.3.7. <i>Util</i>	37
5.3.8. <i>Widget</i>	38
6. Implementación	39
7. Ejemplos	41
8. Conclusiones y trabajos futuros	43
Bibliografía	46

Índice de figuras

1.1.	Primera generación de aparatos de tomografía computarizada [14]	3
1.2.	Segunda generación de aparatos de tomografía computarizada [14]	3
1.3.	Tercera generación de aparatos de tomografía computarizada [14]	4
1.4.	Cuarta generación de aparatos de tomografía computarizada [14]	5
1.5.	Cuarta generación de aparatos de tomografía computarizada [14]	5
1.6.	Diferencias entre TC helicoidal multicorte (B) y monocorte (A) [20]	6
1.7.	Equipo TC con doble fuente [20]	7
1.8.	Obtención de datos volumétricos de una escultura posteriormente examinada	12
2.1.	Boceto a mano alzada de la posible distribución de los elementos en la GUI principal	19
2.2.	Boceto a mano alzada de la posible distribución de los elementos en la GUI de selección de filtros	20
3.1.	Diagrama de Gantt con la planificación inicial del proyecto generado con <i>Microsoft Office Project 2016</i>	24
3.2.	Vista con la lista de <i>issues</i> abiertos con sus correspondientes etiquetas y el <i>milestone</i> al que pertenecen	25
3.3.	Resumen de los <i>milestones</i> con su porcentaje de elaboración .	26
3.4.	Ejemplo de <i>commits</i> en la rama <i>develop</i>	27
3.5.	Gráfico de los <i>commits</i> realizados en la rama <i>develop</i> . El gráfico es orientativo porque hay <i>commits</i> con poca funcionalidad y otros con mucha, pero ayuda a ver las etapas donde más se trabajó	28
5.1.	Diagrama de paquetes de 3DCurator	32
5.2.	Diagrama de clases del paquete <i>Chart</i>	33

5.3.	Diagrama de clases del paquete <i>Core</i>	34
5.4.	Diagrama de clases del paquete <i>Documentation</i>	35
5.5.	Diagrama de clases del paquete <i>GUI</i>	36
5.6.	Diagrama de clases del paquete <i>Interactor</i>	37
5.7.	Diagrama de clases del paquete <i>Segmentation</i>	37
5.8.	Diagrama de clases del paquete <i>Util</i>	38
5.9.	Diagrama de clases del paquete <i>Widget</i>	38

Índice de cuadros

3.1. Planificación inicial	24
4.1. Historias de usuario	30
5.1. Arquitectura del software	32

Capítulo 1

Introducción

Este trabajo de fin de máster es una continuación del trabajo de fin de grado que realicé en el que se desarrolló un *software* (3DCurator) para visualizar un conjunto de datos volumétricos, en formato DICOM, de esculturas de madera policromada.

Con este *software*, los expertos en la materia como restauradores o historiadores del arte podían inspeccionar el interior de las esculturas sin dañarlas para un posterior proceso de estudio, restauración y/o conservación.

En este trabajo de fin de máster se llevarán a cabo distintas tareas de desarrollo que se integrarán a 3DCurator así como un estudio teórico más completo del proceso de obtención de datos volumétricos en los objetos que abarcan el campo de estudio en cuestión.

Las tareas de desarrollo que se integrarán con el *software* desarrollado se dividen en tres bloques:

- **Pre-procesamiento de datos:** Se estudiarán los distintos filtros disponibles para ver cuáles ofrecen mejores resultados en la tarea de reducción de ruido.
- **Subdivisión de piezas de madera:** Las esculturas suelen estar formadas por distintas piezas de madera. Se estudiará la forma de segmentarla probando, en primer lugar, los algoritmos ya existentes utilizados principalmente en medicina. Si ninguno ofrece los resultados que esperamos, se pasará a desarrollar uno propio.
- **Herramientas de documentación:** Se incluirán herramientas para ayudar a los usuarios en la tarea de documentación permitiendo, por ejemplo, incluir distintas anotaciones en puntos de interés.

A parte se van a realizar mejoras en el código para que sea más legible

y mantenible como la internacionalización pues hasta ahora todos los textos estaban *hard-coded*.

Además de las librerías que ya se utilizaron: VTK [10] (visualización), Qt [7] (GUI) y Boost [1] (XML); se utilizarán las librerías ITK [5] y OpenCV [6] para el análisis de imágenes y la visión por computador respectivamente. Haciendo uso de CMake [2] para pre-compilarlas todas juntas.

Antes de empezar a profundizar en aspectos técnicos se realizará una introducción al proceso de obtención de datos volumétricos usando una Tomografía Computarizada así como de los objetos que se quieren analizar con esta técnica de obtención de datos: las esculturas de madera policromada.

1.1. Tomografía Computarizada

1.1.1. Historia

La tomografía computarizada (TC) es una técnica de obtención de imágenes muy utilizado en el campo de la medicina para, por ejemplo, localizar y ver el tamaño de tumores.

Sus orígenes se remontan a los años 60 cuando en 1967 Goodfrey Newblod Hounsfield propuso la elaboración del que llamó escáner EMI, base para desarrollar el Tomógrafo Axial Computarizado (TAC). El objetivo era “*crear una imagen tridimensional de un objeto tomando múltiples mediciones del mismo con la misma fuente de rayos X desde diferentes ángulos y utilizar un ordenador que permita reconstruir a partir de cientos de ‘planos’ superpuestos y entrecruzados*” [21].

Cuatro años más tarde, en 1971, se realizó con éxito el primer escáner cerebral usando este tomógrafo. En 1972 se instaló permanentemente en el hospital donde realizaron las pruebas y al año siguiente ya era solicitado por hospitales alrededor de todo el mundo.

1.1.2. Generaciones

El sistema de tomografía computarizada ha pasado por cuatro generaciones [20]:

Primera generación

La adquisición de datos en la primera generación se basaba en la geometría del haz de rayos X paralelo y traslación-rotación en un tubo de rayos X y un solo detector (Figura 1.1). El haz de rayos X se colimaba en di-

mensiones de 2 x 13mm. Estos 13mm correspondían al grosor del corte. Se tomaba una medida por cada 160 rotaciones durante 180 translaciones dando un total de 28.800 medidas. El proceso era lento, tardaba unos 5 minutos, y el movimiento del paciente afectaba muy negativamente a la calidad de la imagen, por lo que su uso se veía reducido al escaneo de zonas que podían mantenerse inmóviles como la cabeza.

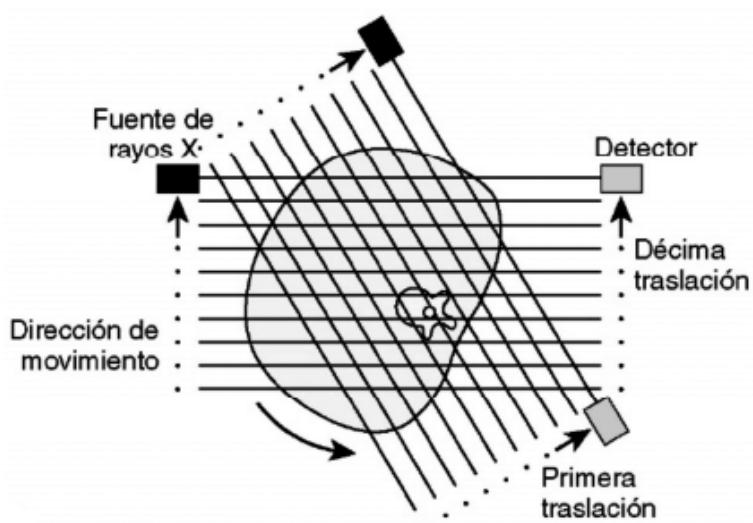


Figura 1.1: Primera generación de aparatos de tomografía computarizada [14]

Segunda generación

En esta segunda generación se aumentó el número de detectores (de 5 a 30) por lo que se vio disminuido el tiempo de la exploración a unos 18 segundos (Figura 1.2).

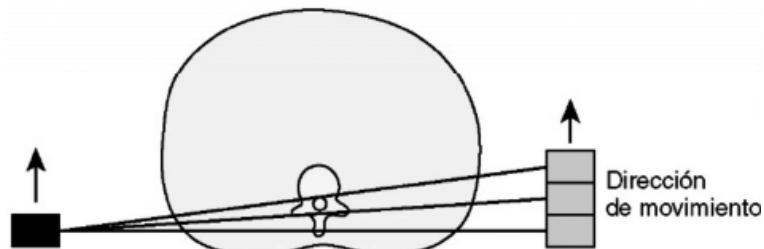


Figura 1.2: Segunda generación de aparatos de tomografía computarizada [14]

Tercera generación

La tercera generación supuso un gran cambio y se ha convertido en la configuración estándar utilizada en la mayoría de los sistemas de escáner. Se utiliza una geometría de haz en abanico de gran angular (50° a 55°), un arco de detectores y un tubo de rayos X. Estos elementos giran 360° alrededor del paciente (Figura 1.3). El número de detectores se encuentra entre 600 y 900. Con este sistema el tiempo de barrido oscila entre 3 y 10 segundos.

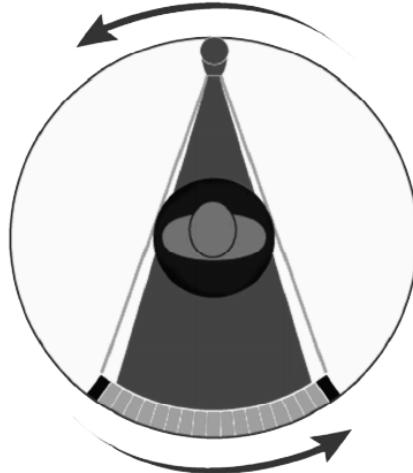


Figura 1.3: Tercera generación de aparatos de tomografía computarizada [14]

Cuarta generación

La cuarta generación es muy parecida a la tercera solo que añade una configuración de giro estacionario (Figura 1.4).

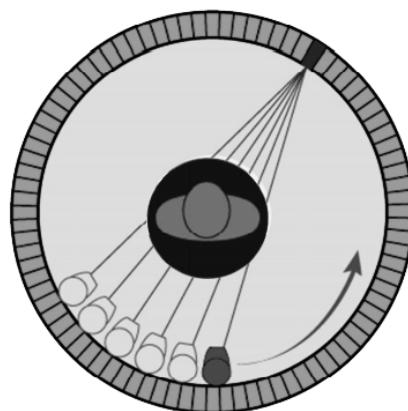


Figura 1.4: Cuarta generación de aparatos de tomografía computarizada [14]

Nuevas tecnologías

- **TC helicoidal (TCH):** Hasta finales de los años 80, los aparatos de TC adquirían los datos en cortes según un método conocido como exploración axial (de ahí el nombre de TAC). Con los sistemas de tipo helicoidal los datos se obtienen de forma continua mientras se avanza la mesa a través del *gantry* haciendo que el tubo de rayos X describa una trayectoria helicoidal (Figura 1.5).

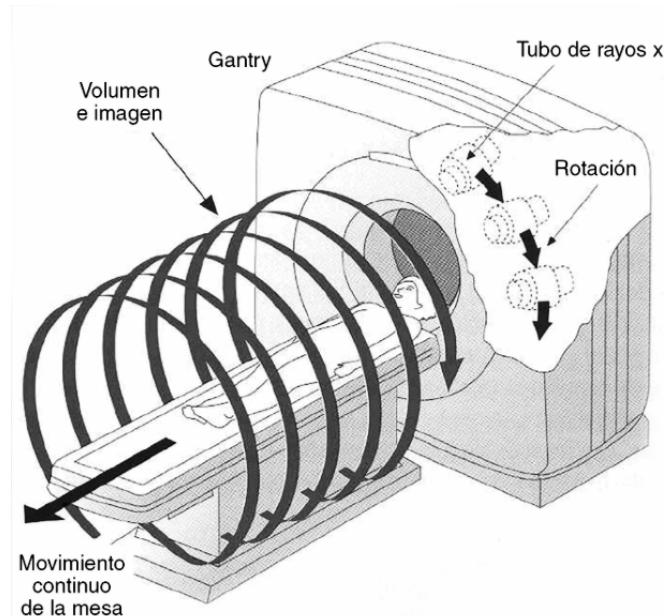


Figura 1.5: Cuarta generación de aparatos de tomografía computarizada [14]

- **TC helicoidal multicorte (TCM):** En el lugar donde había una fila de detectores, se colocan múltiples filas. Los primeros tenían 4 filas contiguas, pero posteriormente se ha pasado a alrededor de 16 y 64 filas (Figura 1.6). Por cada rotación se estudia un mayor volumen aumentando así la velocidad de rotación y por tanto los tiempos de exposición obteniendo imágenes de mayor calidad.

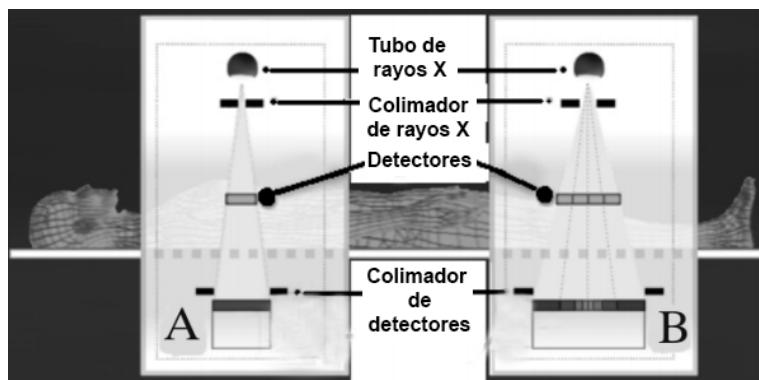


Figura 1.6: Diferencias entre TC helicoidal multicorte (B) y monocorte (A) [20]

- **TC de doble fuente (TCED):** Es uno de los equipos más novedosos pues permiten realizar estudios con diferentes espectros de rayos X. Utilizan dos tubos de rayos X colocados de forma perpendicular en el *gantry* (Figura 1.7). Se obtiene por tanto una resolución temporal equivalente a un cuarto del tiempo de rotación del *gantry*.

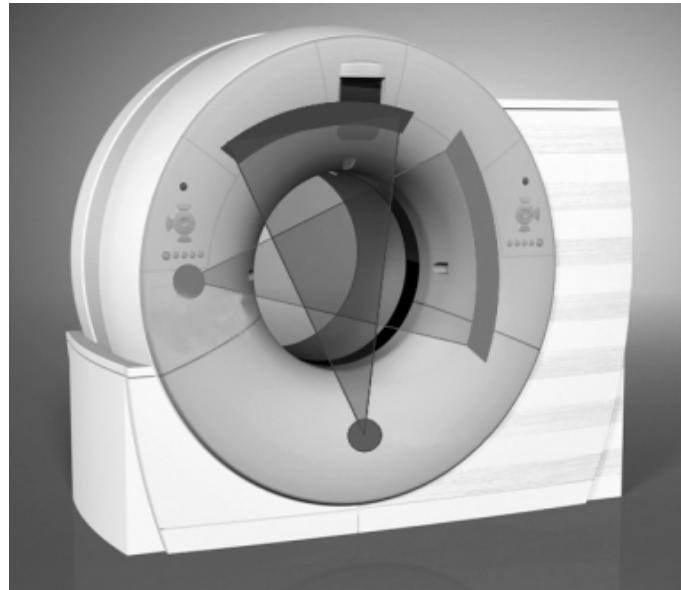


Figura 1.7: Equipo TC con doble fuente [20]

1.2. Esculturas de madera policromadas

1.2.1. Historia

El tallado es el método de elaboración de esculturas más antiguo conocido. Se ha tallado en distintos materiales (madera, piedra, marfil...). Pero la madera, por condiciones como su ligereza o la facilidad de ensamblado entre distintas piezas, ha sido uno de los materiales más utilizados.

Se conoce que desde el Antiguo Egipto ya se realizaban esculturas de madera pero es a partir del siglo XI cuando comienza su proliferación. Y desde este momento comienzan a producirse mejoras en las técnicas y herramientas utilizadas durante el proceso del tallado [20].

1.2.2. Maderas más utilizadas

Dependiendo del tipo de escultura se utilizan maderas blandas o duras. Si la escultura es más pequeña y contiene más detalles se utilizará un tipo de madera más dura.

No obstante, en la elección también tiene mucha influencia la situación geográfica al utilizarse maderas autóctonas [20]:

- **Italia:** Álamo y chopo.

- **Francia:** Nogal y castaño.
- **Países Bajos:** Roble y encina.
- **España:** Pino de Flandes, cedro de la Habana, castaño, tejo, álamo, nogal, ciprés, boj, pino silvestre y algunos frutales como el peral.

Además de la madera, una escultura de madera policromada, puede tener varios materiales como el estuco o el metal de los clavos utilizados.

1.2.3. Defectos de la madera

Entre los defectos de la madera se pueden encontrar [20]:

Grietas o fendas

Según la UNE-EN 844-9 se denomina grieta o fenda a “toda separación de las fibras (raja o hendidura) en dirección longitudinal”. Según su origen, pueden ser de distintos tipos:

- **Acebolladuras o colainas:** Hay una discontinuidad entre los anillos de crecimiento.
- **Superficiales o de desecación:** Producidas por el calor, provocan un deterioro en las zonas externas del tronco del árbol dejando la madera desprotegida. Provocan grietas en sentido longitudinal.
- **De heladura:** Producidas por una helada dañan la superficie e interior del tronco. Provocan grietas radiales.
- **De viento:** Originadas por la acción de un fuerte viento. Provocan grietas longitudinales y transversales.

Además de estos procesos naturales, se pueden producir grietas durante procesos como el secado que provocan una separación de las fibras.

Fibras reviradas y entrelazadas

Las fibras se encuentran normalmente orientadas en paralelo al eje principal del tronco, pero en ocasiones pueden presentar nudos que alteran la dirección de éstas.

Nudos

La UNE 56.521 define nudo como “anomalía local de la estructura de la madera producida por la parte inferior de una rama que va quedando englobada en el tronco a medida que se producen los crecimientos de este”. Existen distintos tipos:

- **Adherente, vivo, fijo o sano:** Definido por la UNE 56.521 como “aquel cuyos tejidos son solidarios con los de la madera que los rodea debido a ser formado por una rama viva”.
- **Suelto, saltadizo, muerto o seco:** Definido por la UNE 56.521 como “aquel en que los tejidos de la rama que lo producen no son solidarios con los de la madera que los rodea y suelen separarse”.

Núcleos de resina

Son cavidades entre los anillos de crecimiento producidos frecuentemente por nudos.

Factores de deterioro de tipo biótico

Además de las alteraciones que ya presenta la madera, existen otros factores que también influyen como la humedad y la temperatura o el ataque de insectos xilófagos y hongos.

Los insectos xilófagos se nutren de madera seca y favorecen a su desarrollo una humedad relativa y temperaturas no muy bajas. Estos producen daños rompiendo las fibras de la madera.

Los hongos son microorganismos que pueden desarrollarse en la superficie o en el interior de la madera, haciendo que pierda humedad, reduciendo su tamaño y deformándose. Existen tres tipos distintos de degradación tras un ataque de hongos: pudrición blanca, parda o seca y blanda.

1.2.4. Proceso de tallado

El proceso de tallado se podría definir como una técnica sustractiva en la que a partir de una pieza se obtiene una forma concreta.

Un método puede ser utilizar un único bloque de madera. En ocasiones ahuecado por el reverso para contrarrestar fuerza y movimiento de la madera, y, en cierto modo, aligerando el peso.

A partir del siglo XVI empieza a utilizarse con más frecuencia otro método en el que a partir de un bloque principal se ensamblan diferentes piezas

generando un bloque más grande, denominado embón, con la forma y el tamaño de la imagen a esculpir. A partir de éste se comienza el tallado.

A partir del Barroco se mejora esta técnica realizando ensamblados en hueco para evitar el posterior ahuecado [20].

1.3. Estado del arte

Como ya se ha comentado anteriormente, este trabajo es la continuación del trabajo fin de grado que realicé. Durante este trabajo fin de grado resolví el problema de la visualización de datos volumétricos de esculturas de madera policromada [12]. Pero no realizaba ningún pre-procesamiento, tan solo se visualizaban permitiendo explorar la escultura en su interior. Toda la información que obtenía el experto que usase el programa debía ser anotada a mano a parte pues tampoco se proporcionaba funcionalidad para almacenar esta información para poder ser rescatada posteriormente.

Este trabajo se divide en tres bloques: pre-procesamiento, documentación y segmentación.

Los dos primeros están incluidos en la mayoría de los programas disponibles en la web de visualización de datos volumétricos. Tales como AMILab [16], RadiAnt [8], Slicer [13] o el usado por María Francisca en su estudio [20] OsiriX [19] en el que me he inspirado para realizar mi implementación. Todas estas herramientas son muy completas, pero los resultados obtenidos al visualizar los datos volumétricos de esculturas de madera policromadas dejaban mucho que desear y el gran número de funciones específicas para la medicina los hacen complejos de más, y es que por esta razón surgió la idea de crear un software específico de visualización de datos volumétricos de esculturas.

El tercer bloque, el de la segmentación, es totalmente novedoso pues no hay ningún tipo de segmentación específica para separar distintos trozos de madera. Sin embargo, antes de programar un algoritmo específico habría que probar con los que ya hay disponibles que además son muy usados en el campo de la medicina.

Existen varias aproximaciones para llevar a cabo la segmentación en datos volumétricos. Pueden clasificarse en:

- Segmentación manual
- Segmentación semi-automática
- Segmentación basada en umbrales (*thresholding*)
- Segmentación basada en crecimiento de regiones (*region-growing*)

- Segmentación basada en bordes (*edge*)

La segmentación manual la podríamos descartar automáticamente. Pese a que es un método que siempre se puede aplicar, tener a una persona recortando manualmente las distintas piezas corte por corte puede ser muy costoso en cuanto a tiempo.

Los métodos de segmentación basada en umbrales [18] se basan en definir un umbral en los valores escalares de cada voxel para diferenciar las distintas partes del volumen. Estos no nos son útiles pues las distintas piezas de madera suelen ser del mismo material y coincidir en estos valores de densidad.

Los métodos de segmentación basada en crecimiento de regiones que utilizan umbrales [15] tampoco nos van a servir pues las distintas piezas de madera se encuentran juntas. Otros métodos de segmentación basada en regiones como la transformación divisoria (*watershed*) [11] tampoco nos van a servir pues se definirían muchas cuencas debido a los anillos que presentan los cortes de la madera y si se empiezan a inundar para obtener regiones más grandes obtendremos un resultado similar al que obtenemos con el de crecimiento de regiones con umbrales.

Los métodos de segmentación basados en bordes como el *livewire* [17] podrían resultar efectivos pues el borde que separa dos piezas de madera es visualmente diferenciable. No obstante, este método necesita de una supervisión humana y, aunque sea más preciso y rápido que una segmentación manual, se debería seguir realizando corte a corte.

Como vemos, ninguno de los métodos que no requieren de supervisión humana son efectivos a la hora de resolver nuestro problema. Es por ello que, definitivamente, hay que desarrollar uno propio.

1.4. Motivación

La exploración interna de una escultura mediante los datos obtenidos con una TC 1.8 son mucho más completos que los obtenidos con otras técnicas como las radiografías tradicionales [20]. La necesidad de una herramienta con la que poder visualizar estos datos se vio satisfecha con 3DCurator [12], sin embargo, se trataba de un software todavía incompleto pues no contaba con técnicas de pre-procesamiento ni documentación como otros software del mercado especializados en visualizar datos médicos.

La extracción de los distintos trozos de madera que forman parte de la escultura podrán ayudar enormemente a su estudio por separado además de poder ser extraídos en un formato como el STL para ser imprimidos en 3D.



Figura 1.8: Obtención de datos volumétricos de una escultura posteriormente examinada

Capítulo 2

Especificación de requisitos

Este capítulo es una Especificación de Requisitos Software para el software que se va a realizar siguiendo las directrices dadas por el estándar IEEE830 [3].

2.1. Introducción

2.1.1. Propósito

Este capítulo de especificación de requisitos tiene como objetivo definir las especificaciones funcionales y no funcionales para el desarrollo de un software que permitirá pre-procesar datos DICOM obtenidos al someter a una escultura de madera policromada a una TC para que posteriormente los usuarios puedan realizar labores de documentación al explorar la figura así como realizar una segmentación de los distintos trozos de madera que la componen.

2.1.2. Ámbito del sistema

En la actualidad, los datos DICOM obtenidos tras una TC se utilizan, principalmente, en el campo donde surgieron, la medicina. Con este software llamado 3DCurator, se tratará de trasladar esta técnica al campo de la restauración de bienes culturales y poder pre-procesar, visualizar, interactuar y documentar los datos DICOM obtenidos con esculturas de madera policromada.

2.1.3. Definiciones, acrónimos y abreviaturas

- **ERS** (Especificación de Requisitos Software).

- **GUI** (Interfaz Gráfica de usuario).
- **Widget**: Elemento de la GUI.
- **DICOM** (*Digital Imaging and Communication in Medicine*): Formato de datos volumétricos donde se obtienen las imágenes.
- **CT o TC** (Tomografía Computarizada): Técnica de extracción de datos volumétricos que utiliza radiación X para obtener cortes de objetos.
- **MRI o IRM** (Imagen por Resonancia Magnética): Técnica de extracción de datos volumétricos que utiliza el fenómeno de resonancia magnética nuclear.
- **PET** (Tomografía por Emisión de Positrones): Técnica de extracción de datos volumétricos capaz de medir la actividad metabólica del cuerpo humano.
- **CPU** (*Graphic Processor Unit*): Microprocesador.
- **GPU** (*Graphic Processor Unit*): Tarjeta gráfica.
- **UML** (Lenguaje unificado de modelado): Lenguaje de modelado de sistemas software.
- **Historia de usuario**: Representación de un requisito utilizando el lenguaje común del usuario.
- **Product Backlog**: Listado de historias de usuario del proyecto.
- **C++**: Lenguaje de programación que se usará.
- **XML** (*Extension Markup Language*): Meta lenguaje que se usará para exportar ficheros que después podrán ser importados.
- **CMake** (*Cross platform Make*): Herramienta para generar código compilable en distintas plataformas.
- **Qt**: Librería que se utilizará para realizar la GUI.
- **VTK** (*The Visualization ToolKit*): Librería gráfica que se utilizará para la visualización de volúmenes.
- **ITK** (*Insight Segmentation and Registration Toolkit*): Librería de procesamiento de imágenes que se utilizará.
- **OpenCV**: Librería de visión por computador que se utilizará.
- **Boost**: Conjunto de algoritmos para C++ de los que se usarán la gestión de ficheros XML.

- **Volumen:** Conjunto de datos en los que para cada posición XYZ se tiene un valor determinado.
- **Voxel (Volumetric Pixel):** Celda en la matriz 3D del conjunto de datos del volumen.
- **Vecinadario:** Celdas alrededor de un voxel.
- **Adyacencia de voxels:** Voxels vecinos que satisfacen un criterio de similitud.
- **Conectividad de voxels:** Voxels entre los cuales hay un camino de voxels adyacentes.
- **Malla:** Estructura de datos con la información de una superficie tridimensional.
- **STL (Standard Triangle Language):** Formato que define mallas 3D.
- **Corte:** Vista de la figura a través de un plano. Por ejemplo, al cortar con una sierra un tronco por la mitad, se puede ver cómo es por dentro en esa posición por donde se ha cortado.
- **TF (Función de transferencia):** Función utilizada para visualizar los datos deseados de un volumen.
- **Preset:** Función de transferencia previamente configurada.
- **Direct Volume Rendering:** Visualización directa de volúmenes en la que cada valor del volumen se mapea con un determinado color y opacidad dado por una función de transferencia.
- **Ray Casting:** Técnica de Direct Volume Rendering utilizada para la visualización de volúmenes.
- **Marching Cubes:** Técnica para generar malla de polígonos a partir de un volumen y un valor de isosuperficie.
- **HU (Hounsfield Units):** Unidad de medida escalar del valor de densidad en un voxel del volumen.
- **ROD (Región de documentación):** Corte en el que se documentará.
- **Regla:** Widget utilizado para realizar mediciones de distancias.
- **Transportador de ángulos:** Widget utilizado para realizar mediciones de ángulos.
- **Nota:** Widget utilizado para realizar anotaciones sobre la figura.

- **Segmentación:** Método por el cual se distinguen distintas partes del volumen.
- **Thresholding:** Segmentación basada en umbrales.
- **Region-growing:** Segmentación basada en crecimiento de regiones.
- **Semilla:** Coordenada donde se comienza el *region-growing*.
- **Watershed:** Técnica de segmentación basada en crecimiento de regiones por inundación.
- **Canny Edge Detection:** Algoritmo para resaltar los bordes de una imagen.
- **Transformada de Hough:** Técnica utilizada para detectar líneas rectas.
- **Filtro gaussiano:** Filtro que utiliza la distribución gaussiana del vecindario.
- **Filtro media:** Filtro que utiliza la media del vecindario.
- **Filtro mediana:** Filtro que utiliza la mediana del vecindario.
- **Embón:** Ensamblado de distintos trozos de madera que hacen de base para la escultura.
- **Estuco:** Pasta de grano fino utilizada para realizar correcciones en la escultura.
- **Policromía:** Capa de pintura de las esculturas.
- **Blanco de plomo:** Pigmento de color blanco creado a partir del plomo.
- **Pan de oro:** Lámina muy fina de oro.

2.1.4. Visión general del documento

Este capítulo consta de tres secciones:

- En la primera sección se realiza una introducción a éste y se proporciona una visión general de la ERS.
- En la segunda sección se realiza una descripción general a alto nivel del software, describiendo los factores que afectan al producto y a sus requisitos y con el objetivo de conocer las principales funcionalidades de éste.

- En la tercera sección se definen detalladamente los requisitos que deberá satisfacer el software.

2.2. Descripción general

2.2.1. Perspectiva del producto

El software 3DCurator tiene como objetivo interactuar con datos DICOM, pero no es el encargado de generarlos. Para generarlos se deberá utilizar algún escáner de TC.

Una vez obtenidos, se podrán pre-procesar aplicando una serie de filtros, segmentar en los distintos trozos de madera que forman la escultura y documentar añadiendo notas y mediciones de distancias y ángulos.

2.2.2. Funciones del producto

Las principales funcionalidades de este sistema serán:

- Pre-procesar la imagen aplicando filtros de reducción de ruido:
 - Media.
 - Mediana.
 - Gaussiano.
- Documentar la escultura:
 - Agregar anotación.
 - Agregar regla.
 - Agregar transportador de ángulos.
- Segmentar la escultura por los distintos trozos de madera.
- Exportar el volumen.
- Importar el volumen.

2.2.3. Características de los usuarios

Solo existe un tipo de usuario, que es la persona que desee interactuar con los datos DICOM de una escultura. Esta persona no tiene por qué tener habilidad con un equipo informático, por lo que 3DCurator deberá tener una GUI intuitiva y fácil de utilizar.

2.2.4. Restricciones

Se llevará a cabo un desarrollo evolutivo basado en un prototipo funcional sobre un software de visualización de datos volumétricos de esculturas de madera ya existente.

Se programará en C++ usando las librerías VTK para la visualización de gráficos, ITK para el filtrado de imágenes, OpenCV para la visión por computador, Boost para la gestión de ficheros XML y Qt para la GUI.

Al usar estas librerías se contará con restricciones funcionales con las que éstas cuentan que se solventarán reescribiendo el código que sea posible y necesario.

Aprovechando que se debe usar CMake para compilar las librerías mencionadas, se utilizará también para generar el proyecto, pues se puede generar código compilable en distintas plataformas.

2.2.5. Suposiciones y dependencias

El software se utilizará para poder interactuar con esculturas de madera por lo que se tendrán en cuenta los materiales con los que están hechas la mayoría de estas (madera, estuco y metal). Si se introducen los datos DICOM de cualquier otra cosa con materiales distintos a los utilizados en las esculturas no se visualizará correctamente. Sin embargo se podrá editar la función de transferencia para su correcta visualización.

La única funcionalidad que se vería afectada sería la segmentación en las distintas piezas de madera que la forman pues usaría como dato el valor escalar en HU de la madera.

2.3. Requisitos específicos

2.3.1. Interfaces

Se adaptará la interfaz actual de 3DCurator para agregar las nuevas funcionalidades (Figura 2.1):

- Se crearán botones para importar y exportar volumen, segmentar y aplicar filtro. Éste último lanzará una ventana modal (Figura 2.2) donde se seleccionará el filtro y parámetros deseados.
- A la derecha se colocará la parte de documentación con la que se podrán agregar y eliminar ROD, reglas, transportadores de ángulos y anotaciones.

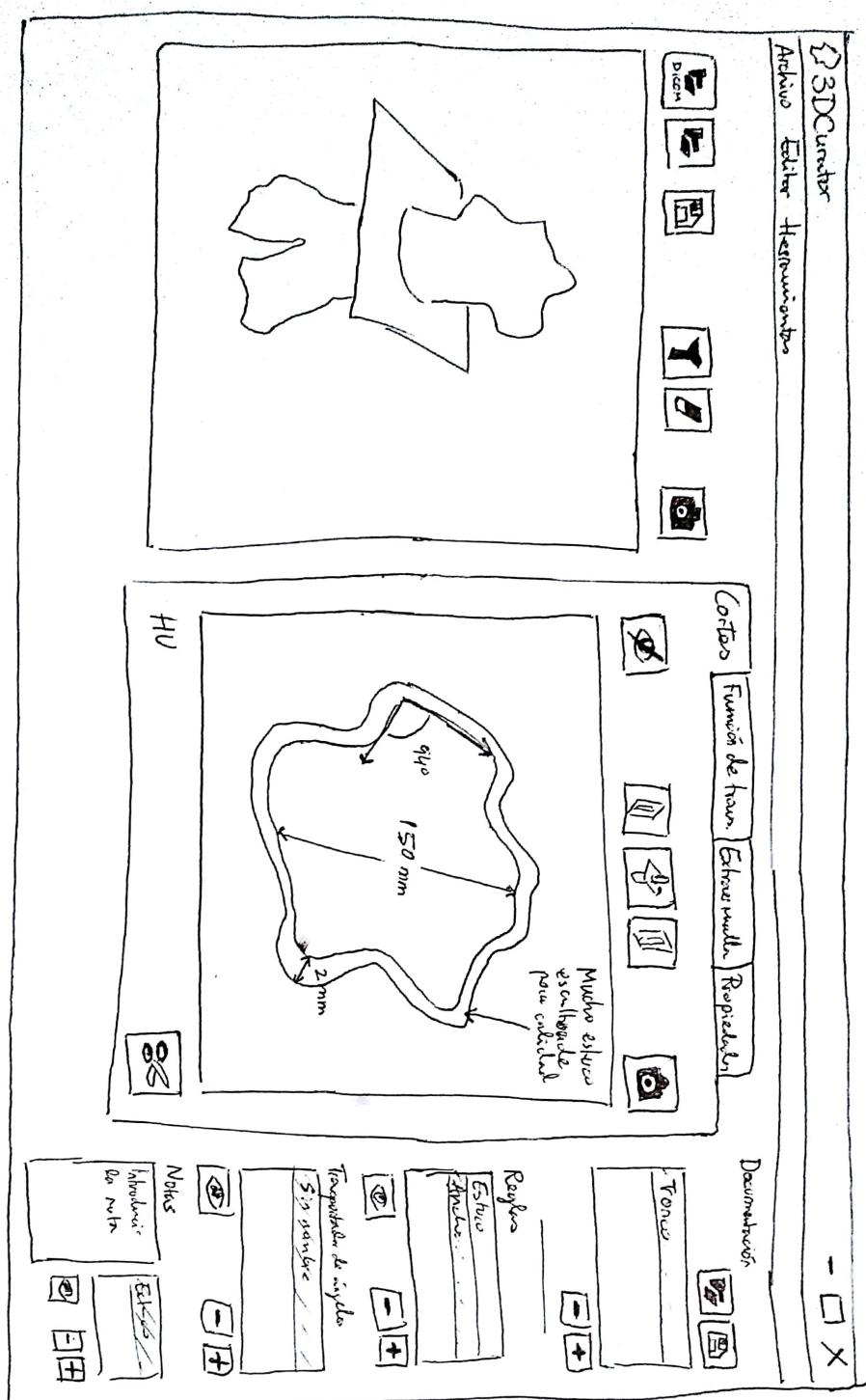


Figura 2.1: Boceto a mano alzada de la posible distribución de los elementos en la GUI principal

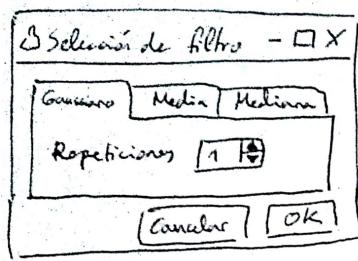


Figura 2.2: Boceto a mano alzada de la posible distribución de los elementos en la GUI de selección de filtros

2.3.2. Funciones

El sistema tendrá que realizar distintas funciones que se comentaron anteriormente pero se profundizará en esta sección. Estas funciones se han estructurado por módulo.

Auxiliar

- **Exportar volumen:** El usuario seleccionará la carpeta y el nombre donde deseará guardar el volumen que está visualizando.
- **Importar volumen:** En lugar de cargar directamente los datos DICOM, el usuario podrá cargar el volumen que haya exportado anteriormente el cual puede tener aplicado los filtros de pre-procesamiento o segmentación.

Pre-procesamiento

- **Filtro gaussiano:** El usuario seleccionará el número de repeticiones que desea aplicar y se aplicará el filtro al volumen cargado.
- **Filtro media:** El usuario seleccionará el tamaño del vecindario y se aplicará el filtro al volumen cargado.
- **Filtro mediana:** El usuario seleccionará el tamaño del vecindario y se aplicará el filtro al volumen cargado.

Segmentación

- **Segmentar:** Desde un corte, el usuario seleccionará la pieza de madera que desea segmentar. Una vez realizado este proceso se mostrará en pantalla el volumen segmentado y se dará la posibilidad de exportarlo.

Documentación

- **Crear ROD:** El usuario podrá agregar un ROD al que podrá dar un nombre.
- **Eliminar ROD:** El usuario podrá eliminar un ROD seleccionado.
- **Exportar ROD:** El usuario podrá exportar el ROD seleccionado.
- **Importar ROD:** El usuario podrá importar un ROD previamente exportado.
- **Cambiar ROD activo:** El usuario podrá cambiar de ROD y se cargará la posición del plano con los elementos de éste. Si el usuario mueve el plano se dejará de visualizar el ROD actual.
- **Crear regla:** El usuario podrá agregar una regla a la que podrá dar un nombre.
- **Eliminar regla:** El usuario podrá eliminar una regla seleccionada.
- **Ocultar regla:** El usuario podrá ocultar una regla seleccionada.
- **Mostrar regla:** El usuario podrá ocultar una regla seleccionada oculta.
- **Cambiar regla activa:** El usuario podrá cambiar la regla activa.
- **Crear transportador de ángulos:** El usuario podrá agregar un transportador de ángulos al que podrá dar un nombre.
- **Eliminar transportador de ángulos:** El usuario podrá eliminar un transportador de ángulos seleccionado.
- **Ocultar transportador de ángulos:** El usuario podrá ocultar un transportador de ángulos seleccionado.
- **Mostrar transportador de ángulos:** El usuario podrá ocultar un transportador de ángulos seleccionado oculto.
- **Cambiar transportador de ángulos activa:** El usuario podrá cambiar el transportador de ángulos activo.
- **Crear nota:** El usuario podrá agregar una nota a la que podrá dar un nombre y agregar un texto.
- **Eliminar nota:** El usuario podrá eliminar una nota seleccionada.
- **Ocultar nota:** El usuario podrá ocultar una nota seleccionada.
- **Mostrar nota:** El usuario podrá ocultar una nota seleccionada oculta.
- **Cambiar nota activa:** El usuario podrá cambiar la nota activa.

2.3.3. Requisitos de rendimiento

Al ser una aplicación de escritorio donde se mostrará y se interactuará con datos, los requisitos de rendimiento no se centrarán en la concurrencia de acceso ni en el almacenamiento, como lo podrían estar en una aplicación web.

Sin embargo, hay que tener en cuenta otros factores, como pueden ser el uso eficiente de memoria y no tener cargadas todos los volúmenes que se han estado visualizando, desecharando el anterior cuando se carga uno nuevo.

El rendimiento gráfico también es importante, por eso y para obtener imágenes de mayor calidad, se utilizarán técnicas de DVR como el *ray casting*. Estas técnicas necesitan una gran cantidad de procesamiento, pero la velocidad de procesamiento de las GPU actuales no deberían resultar un problema.

En cuanto a la segmentación que es la función más compleja y costosa computacionalmente, habrá que tener especial cuidado con la gestión de memoria intentando usar los almacenes de datos más adecuados en cada caso.

2.3.4. Restricciones de diseño

Al utilizar librerías como VTK, ITK, OpenCV o Boost, se seguirá la estructura de las mismas a la hora de construir el software, y se tendrán restricciones en cuanto a funcionalidad que se pueda construir con éstas. No obstante son librerías muy completas y no se debería encontrar casi ninguna restricción a excepción de la parte del pre-procesamiento y segmentación donde quizás haya que generar algoritmos propios.

2.3.5. Atributos del software

Al usar CMake, se podrá crear un software multiplataforma que funcione en cualquier sistema operativo.

El software generado deberá ser fiable, porque aunque no trabaje con datos sensibles cuya pérdida pueda ser grave, siempre resulta molesto utilizar un software con fallos que interrumpan durante su uso.

También se debe tener en cuenta que el software sea mantenable pues, en el futuro, otros desarrolladores pueden colaborar y debe estar bien documentado para que esto sea una tarea fácil.

Se tratará de un software de código abierto que se alojará en un repositorio de GitHub.

Capítulo 3

Planificación

En este capítulo se comentará la planificación inicial de tiempo en la que se llevará a cabo este TFM, la estimación en horas de cada módulo de tareas, la metodología de trabajo y los resultados.

3.1. Planificación inicial

La fecha de inicio trabajando en el proyecto que me fijé es el 1 de diciembre de 2016 y se espera llegar a la convocatoria de julio de 2017 por lo que se cuentan con siete meses. No obstante, y como es lógico, no voy a trabajar exclusivamente en este proyecto durante estos meses pues tengo que cursar el resto de asignaturas del máster, además debería tener un mes de margen para pulir la memoria desarrollando algunas secciones y organizando lo que haya ido apuntando durante el resto de meses. Por lo que serían 6 meses de desarrollo.

Para organizarme he decidido dedicar un número de horas semanales al desarrollo de este proyecto. Inicialmente me he fijado 15 horas a la semana dando un total de 360 horas de desarrollo puro excluyendo la redacción de la memoria.

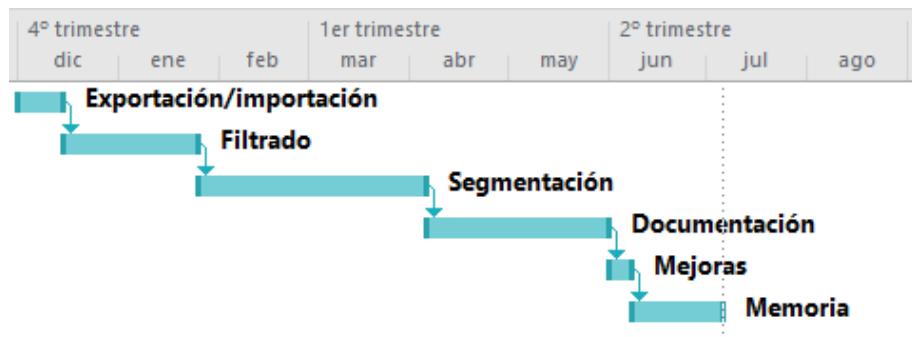
Ordenando los módulos por prioridades y dependencias se tendría la siguiente planificación:

Tarea	Duración	Comienzo	Fin
Exportación/importación	2 semanas	01/12/2016	14/12/2016
Filtrado	6 semanas	15/12/2016	25/01/2017
Segmentación	10 semanas	26/01/2017	05/04/2017
Documentación	8 semanas	06/04/2017	31/05/2017
Mejoras	1 semana	01/06/2017	07/06/2017

Tarea	Duración	Comienzo	Fin
Memoria	4 semanas	08/06/2017	05/07/2017

Cuadro 3.1: Planificación inicial

3.1.1. Diagrama de Gantt

Figura 3.1: Diagrama de Gantt con la planificación inicial del proyecto generado con *Microsoft Office Project 2016*

3.2. Metodología de trabajo

3.2.1. GitHub

Aprovechando que se está usando *git* como sistema de control de versiones en un repositorio de *GitHub*, voy a aprovechar todos los recursos que nos ofrece esta plataforma para organizarme.

3.2.2. Issues

En lugar de tener por un lado un tablero *kanban* (por ejemplo *Trello*), yo voy a utilizar los *issues* (Figura 3.2) de mi repositorio en *GitHub* para organizar las tareas, así como ir añadiendo tareas nuevas que vaya necesitando y tener un registro de qué he hecho y cuándo, que problemas han surgido, etc.

The screenshot shows a GitHub repository page for 'fblupi / 3DCurator'. The top navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore' buttons. On the right, there are icons for 'Unwatch', 'Star', 'Fork', and user profile. Below the header, a search bar says 'Search or jump to...'. A sidebar on the left has tabs for 'Code', 'Issues 11', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. A 'Filters' dropdown is set to 'is:issue is:open'. A search input field contains 'is:issue is:open'. Buttons for 'Labels' and 'Milestones' are present, along with a green 'New issue' button. The main area displays a table of 11 open issues:

	Author	Labels	Milestone	Comments
Change from QVTKWidget to QVTKOpenGLWidget	fblupi	enhancement		#71 opened 28 days ago by fblupi Set-up environm...
Set-up environment for Mac OS X	fblupi	enhancement, resources needed		#65 opened 28 days ago by fblupi Set-up environm...
Planes detection	fblupi	enhancement		#64 opened on 7 Sep 2017 by fblupi Automatic Segm...
Lines detection	fblupi	enhancement		#63 opened on 7 Sep 2017 by fblupi Automatic Segm...
Special characters in filenames	fblupi	bug		#58 opened on 5 Aug 2017 by fblupi GUI Improvements
3D Region Growing	fblupi	enhancement, question		#54 opened on 31 Jul 2017 by fblupi Semi-automatic ...
Show degree symbol in protractor	fblupi	enhancement, help wanted		#46 opened on 24 Jul 2017 by fblupi GUI Improvements
Show histogram in color and scalar opacity charts	fblupi	enhancement, help wanted		#39 opened on 13 Jul 2017 by fblupi GUI Improvements
Segmentation using more than one line	fblupi	enhancement		#15 opened on 2 May 2017 by fblupi Semi-automatic ...
Parse plane coordinates into volume coordinates	fblupi	enhancement, help wanted		#4 opened on 28 Mar 2017 by fblupi Semi-automatic ...
Set-up environment for Linux	fblupi	enhancement, help wanted		#2 opened on 1 Nov 2016 by fblupi Set-up environm...

Figura 3.2: Vista con la lista de *issues* abiertos con sus correspondientes etiquetas y el *milestone* al que pertenecen

Una de las ventajas de utilizar los *issues* de *GitHub* antes que otra plataforma como *Trello* es la posibilidad de referenciarlos en los *commit* agregando `#ref` en el mensaje. Y dentro del propio *issue* se vería una lista de todos los *commit* donde se ha trabajado en éste. Además, si antes de escribir la referencia del *issue* se añade *closes*, el *issue* se cerrará automáticamente sin necesidad de hacerlo de forma manual.

3.2.3. *Milestones*

Voy a clasificar los *issues* por *milestones* (Figura 3.3), uno por cada módulo y al mismo tiempo voy a agregarles *tags* para etiquetarlos según sean *bugs*, mejoras, bloqueantes por necesidad de recursos o por ayuda necesitada, etc.

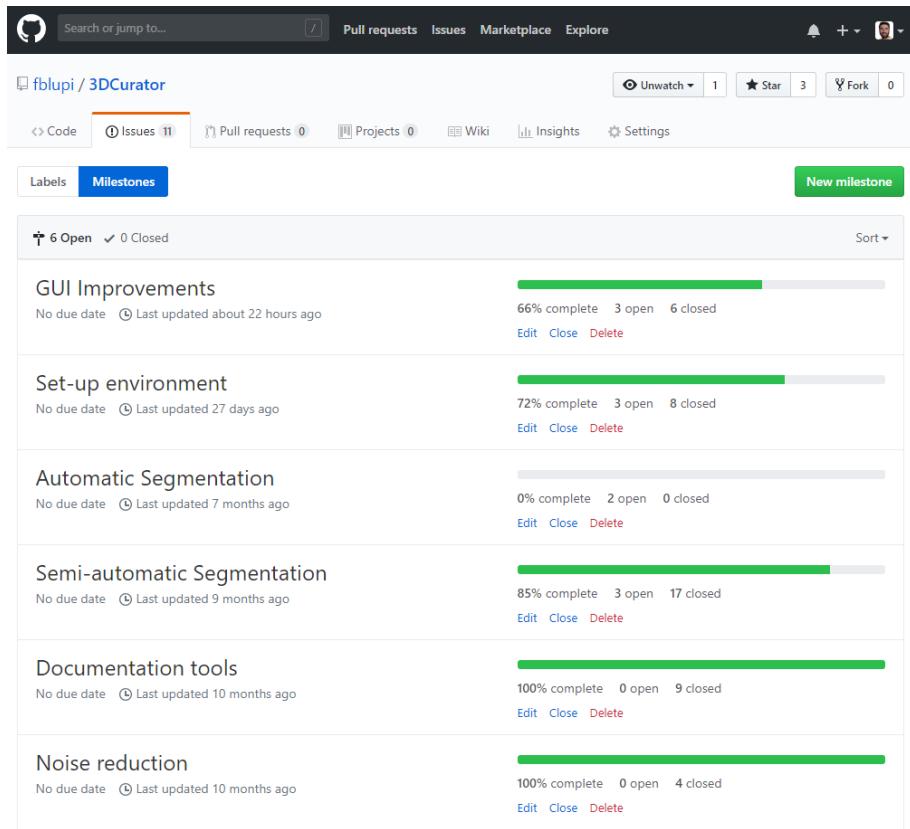


Figura 3.3: Resumen de los *milestones* con su porcentaje de elaboración

3.2.4. *Branches*

La organización que voy a llevar en cuanto a las ramas (*branches*), es la siguiente, basada en *Gitflow* [4]:

- **master**: Esta rama tendrá la última *release* del programa. Debe ser completamente estable, pues es de donde se generan los instaladores finales de la aplicación.
- **develop**: Esta es la rama principal con los últimos cambios desarrollados (Figura 3.4). Cuando se tiene una batería de cambios suficiente como para generar una nueva versión del *software* se realizará un *pull request* a *master* por lo que esta rama debe ser también estable.
- **features**: Estas son ramas de trabajo que no tienen por qué ser estables. Por ejemplo, si se va a agregar un filtro *gaussiano* se podría crear una rama con el nombre *feature/gaussian-filter* y trabajar en esta rama hasta que se validase esta nueva funcionalidad. Entonces se

haría un *rebase* con *develop* por si ha habido algún cambio durante el transcurso de este desarrollo para resolver conflictos y finalmente un *pull request* a *develop* para mezclar los cambios.

Además cuento con otras ramas auxiliares que no intervienen en el flujo de desarrollo pero donde almaceno información útil.

- **design:** En esta rama se encuentran los diagramas de paquetes y clases actualizados con la versión del software en la rama *develop*.
- **assets:** En esta rama se almacenan recursos como logos e imágenes utilizados en la aplicación.
- **user-manual:** En esta rama se encuentra el manual de usuario de la aplicación.
- **gh-pages:** En esta rama se encuentra la *landing page* de la aplicación. Pues *GitHub* ofrece gratuitamente alojamiento para una página web estática por repositorio.

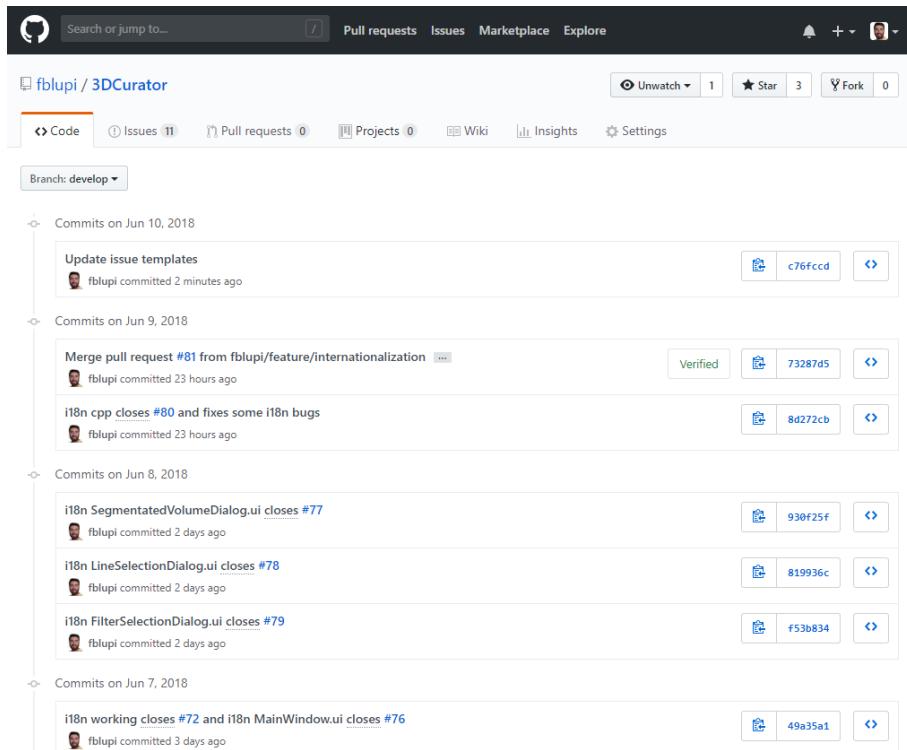


Figura 3.4: Ejemplo de *commits* en la rama *develop*

3.3. Resultados

Los resultados no tienen nada que ver con la planificación inicial que se hizo. La carga del trabajo del resto de asignaturas del máster han hecho que le dedicase muchas menos horas a la semana que las que tenía pensado y durante algunos periodos lo he tenido que tener apartado.

Cuando llegó junio y vi que no llegaba a la convocatoria, recalculeé tiempos para llegar a septiembre. Pero otra vez se vieron trastocados pues en agosto empecé a trabajar a jornada completa y las horas que le pude echar se volvieron a ver reducidas. No obstante llegué a tenerlo todo a excepción de la documentación. Que ya, una vez matriculado de nuevo en la asignatura la tomé con calma.

Además, me permití el lujo de añadir a última hora internacionalización al proyecto para poder tener la aplicación también en inglés además de la versión ya existente en español.



Figura 3.5: Gráfico de los *commits* realizados en la rama *develop*. El gráfico es orientativo porque hay *commits* con poca funcionalidad y otros con mucha, pero ayuda a ver las etapas donde más se trabajó

En total he registrado 312 horas de desarrollo frente a las 360 planificadas. Esto quiere decir que la planificación de tareas no ha fallado como tal. La que ha fallado ha sido la planificación de disponibilidad de recursos humanos.

Capítulo 4

Análisis

En este capítulo se describirán las distintas historias de usuario que han sido implementadas en el software.

4.1. Historias de usuario

4.1.1. *Product backlog*

A continuación se muestra el listado de historias de usuario (*Product Backlog*) completo separadas por módulo, y para cada historia de usuario sus dependencias, estimación (en puntos de historia) y prioridad.

#	Descripción	Dep.	Est.	Prio.
Auxiliar				
1.1.	Exportar volumen	-	12	1
1.2.	Importar volumen	1.1.	8	5
Pre-procesamiento				
2.1.	Filtro <i>gaussiano</i>	-	24	1
2.2.	Filtro media	-	10	3
2.3.	Filtro mediana	-	8	3
Segmentación				
3.1.	Segmentar pieza de madera	1.1.	64	1
Documentación				
4.1.	Crear ROD	-	4	1
4.2.	Eliminar ROD	4.1.	4	3
4.3.	Exportar ROD	4.1.	2	2
4.4.	Importar ROD	4.3.	3	2
4.5.	Cambiar ROD	4.1.	3	1
4.6.	Crear regla	-	3	2

#	Descripción	Dep.	Est.	Prio.
4.7.	Eliminar regla	4.6	3	4
4.8.	Editar regla	4.6	1	5
4.9.	Ocultar regla	4.6	2	6
4.10.	Mostrar regla	4.9	1	6
4.11.	Crear transportador de ángulos	-	3	3
4.12.	Eliminar transportador de ángulos	4.11.	3	5
4.13.	Editar transportador de ángulos	4.11.	1	6
4.14.	Ocultar transportador de ángulos	4.11.	2	7
4.15.	Mostrar transportador de ángulos	4.14.	1	7
4.16.	Crear nota	-	4	2
4.17.	Eliminar nota	4.16.	2	4
4.18.	Editar nota	4.16.	3	5
4.19.	Ocultar nota	4.16.	2	6
4.20.	Mostrar nota	4.19.	1	6
Mejoras en código				
5.1.	Internacionalización	-	6	8

Cuadro 4.1: Historias de usuario

Se han estimado un total de 180 PH (Puntos de historia) y se habían planificado unas 360 horas de trabajo. Por lo que cada PH corresponde aproximadamente a 2 horas.

Al haber registrado 312 horas finalmente, he determinado que mi velocidad ha sido aproximadamente 1,73 horas/PH, un poco más rápido que lo planificado.

Capítulo 5

Diseño

En este capítulo se mostrarán los diagramas arquitectónico, de paquetes y de clases de la aplicación.

Estos diagramas UML han sido generados con la aplicación StarUML [9] y están disponibles en la rama de diseño del repositorio del proyecto alojado en GitHub: <https://github.com/fblupi/3DCurator/tree/design> por si es necesario verlos a mayor tamaño.

5.1. Arquitectura del software

El software está íntegramente escrito en C++ a excepción de las interfaces gráficas que usan un formato específico basado en XML.

Se hace uso a nivel hardware tanto de la CPU como de la GPU usando OpenGL como librería gráfica de bajo nivel pese a no programar directamente con esta librería. En su lugar se utiliza la librería gráfica de alto nivel VTK que abstrae de la complejidad de OpenGL.

Para gestionar la interfaz de usuario se utiliza Qt que permite una buena integración con VTK.

Se usan también, como librerías adicionales, ITK para aplicar algoritmos de filtros a los volúmenes, OpenCV para los algoritmos de visión por computador y Boost para la gestión de ficheros XML.

Para generar el proyecto pre-compilando todas estas librerías citadas anteriormente, así como para hacerlo multiplataforma y poder compilarlo en cualquier sistema operativo se usa CMake.

Librerías de alto nivel	Boost	VTK	ITK	OpenCV	Qt
Librerías de bajo nivel	OpenGL				
Lenguaje de programación	C++				
Nivel Hardware	CPU		GPU		

Cuadro 5.1: Arquitectura del software

5.2. Diagrama de paquetes

En el siguiente diagrama (Figura 5.1) se muestran las dependencias entre los distintos paquetes cuyas clases se detallarán en el siguiente apartado.

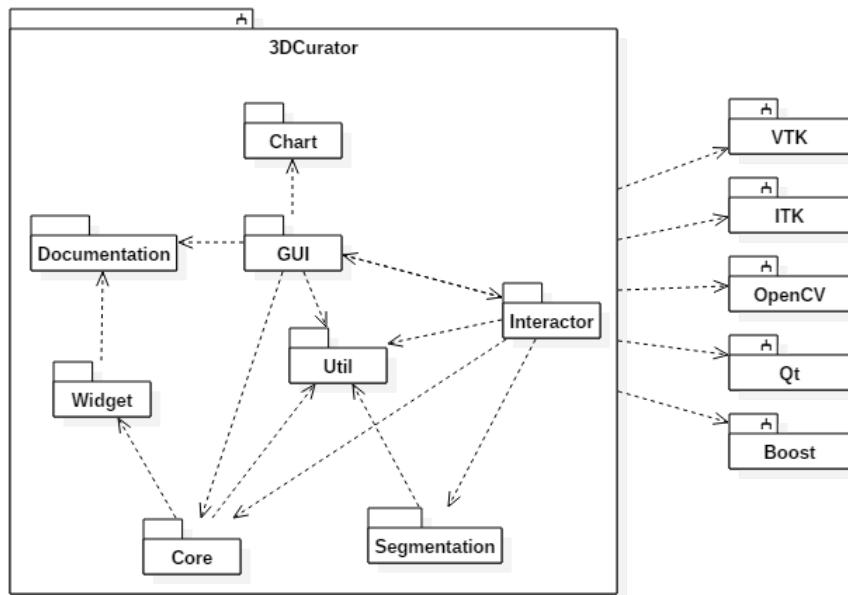


Figura 5.1: Diagrama de paquetes de 3DCurator

5.3. Diagramas de clases

Se presentan las distintas clases de los distintos módulos que contiene 3DCurator.

5.3.1. *Chart*

Este módulo contiene las clases auxiliares para controlar las gráficas utilizadas para visualizar y editar la función de transferencia:

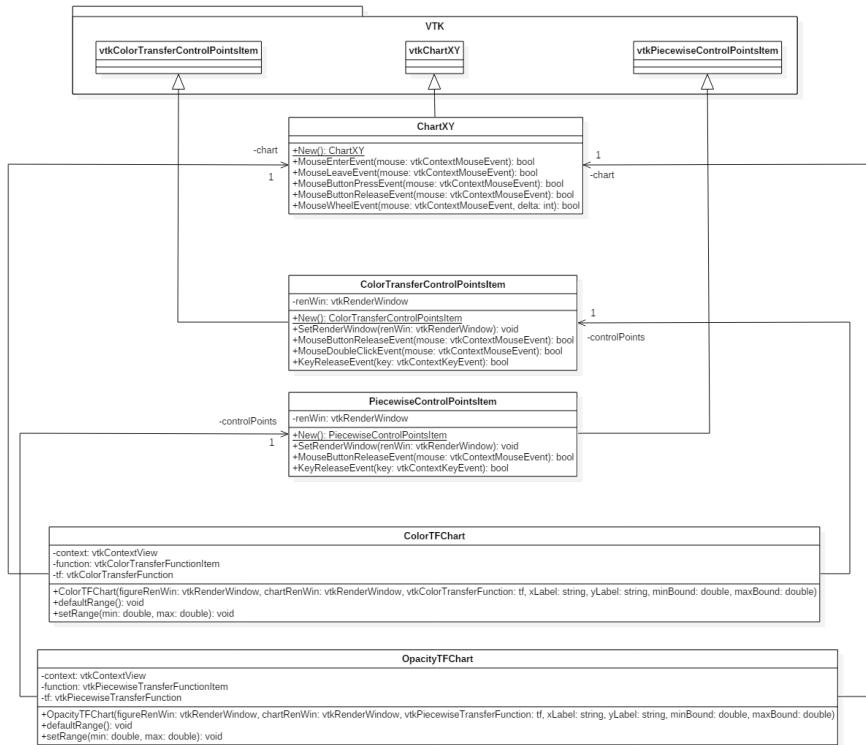
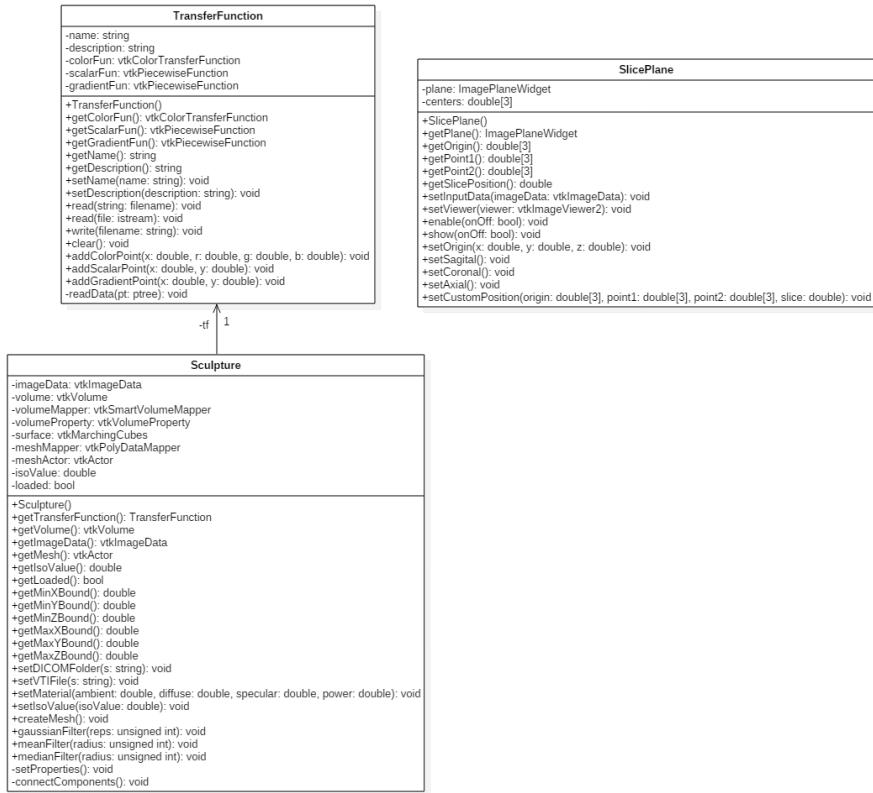


Figura 5.2: Diagrama de clases del paquete *Chart*

5.3.2. *Core*

Este es el módulo principal de 3DCurator. Contiene las clases que gestionan los datos del volumen (*Sculpture*), la función de transferencia (*Transfer Function*) y plano de corte (*SlicePlane*):

Figura 5.3: Diagrama de clases del paquete *Core*

5.3.3. Documentation

En este módulo se encuentra la clase que almacena los distintos elementos utilizados para la documentación:

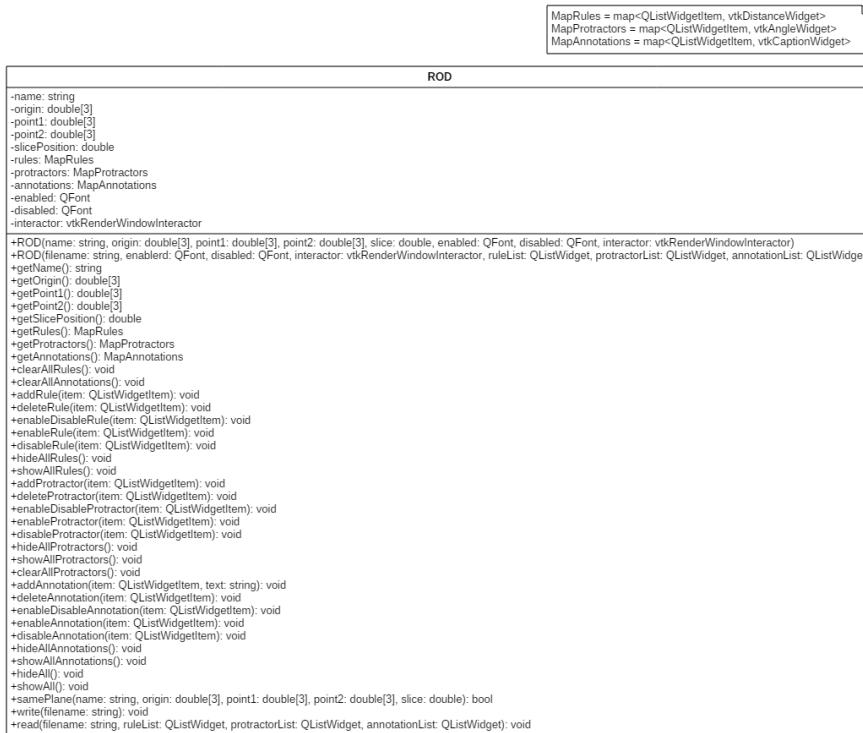


Figura 5.4: Diagrama de clases del paquete *Documentation*

5.3.4. GUI

Este módulo contiene las distintas clases que gestionan las ventanas de la GUI:

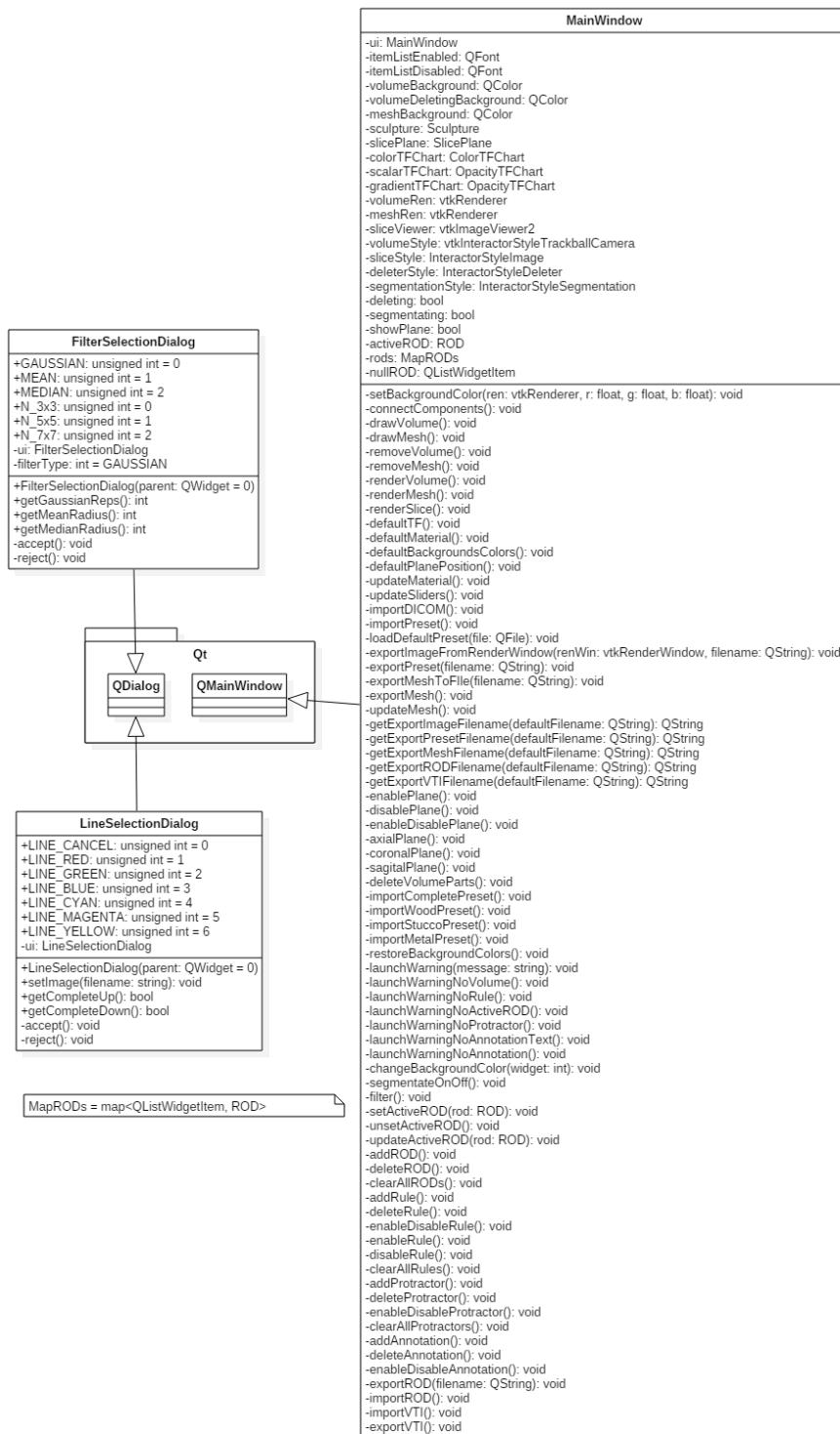


Figura 5.5: Diagrama de clases del paquete *GUI*

5.3.5. *Interactor*

Este módulo contiene los distintos interactuadores con las ventanas de visualización:

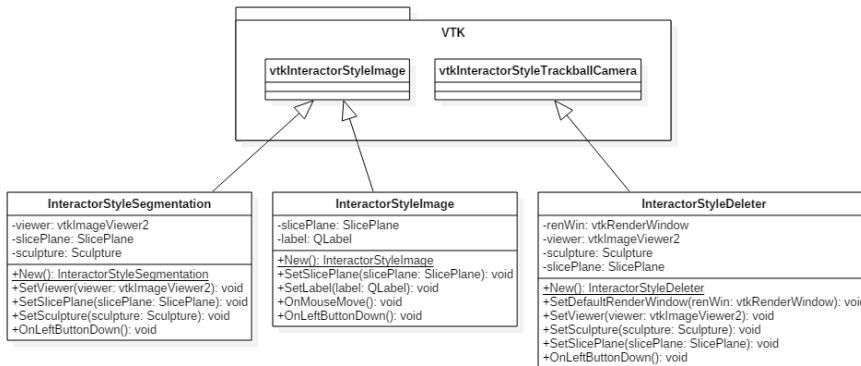


Figura 5.6: Diagrama de clases del paquete *Interactor*

5.3.6. *Segmentation*

En este módulo se encuentran las clases utilizadas para realizar segmentaciones en el volumen:

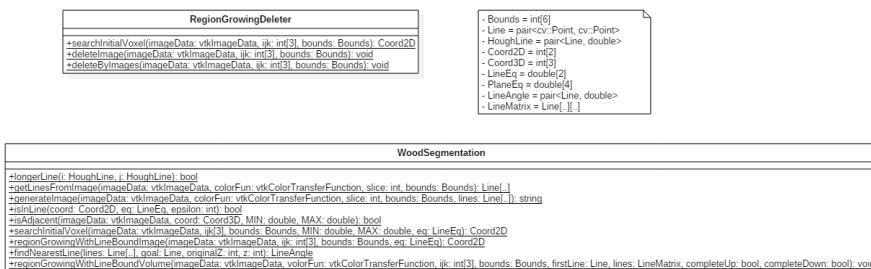
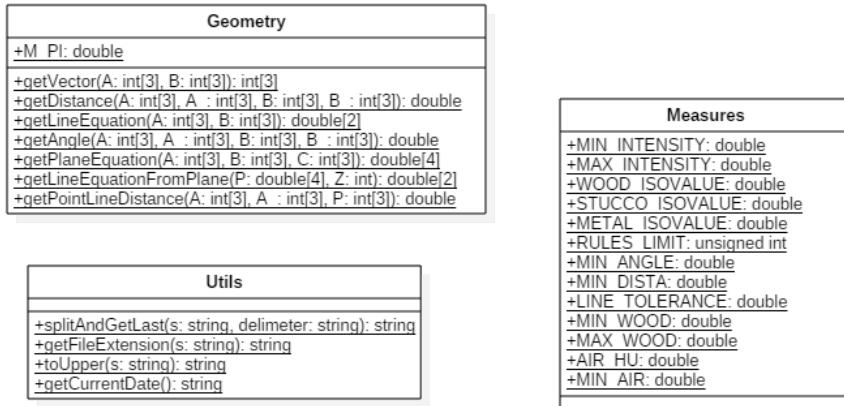


Figura 5.7: Diagrama de clases del paquete *Segmentation*

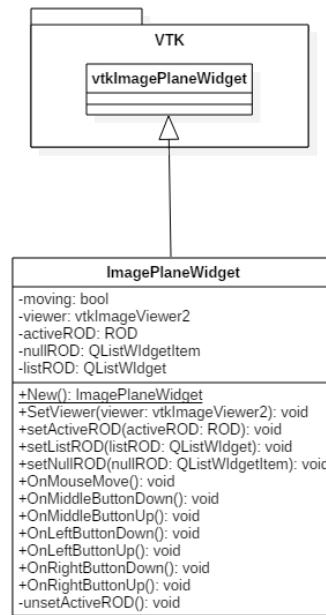
5.3.7. *Util*

Este módulo contiene funciones utilizadas frecuentemente por distintos módulos del programa:

Figura 5.8: Diagrama de clases del paquete *Util*

5.3.8. *Widget*

Este módulo contiene el *widget* personalizado de VTK que se utiliza:

Figura 5.9: Diagrama de clases del paquete *Widget*

Capítulo 6

Implementación

Lala

Capítulo 7

Ejemplos

Lala

Capítulo 8

Conclusiones y trabajos futuros

Lala

Bibliografía

- [1] Boost. <http://www.boost.org/>.
- [2] Cmake. <https://cmake.org/>.
- [3] Especificación de requisitos según el estándar de iee 830. https://www.fdi.ucm.es/profesor/gmendez/docs/is0809_ieee830.pdf.
- [4] Gitflow workflow. <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>.
- [5] Itk. <https://itk.org/>.
- [6] Opencv. <http://opencv.org/>.
- [7] Qt. <https://www.qt.io/>.
- [8] Radiant dicom viewer. <https://www.radiantviewer.com/>.
- [9] Staruml. <http://staruml.io/>.
- [10] Vtk. <http://www.vtk.org/>.
- [11] Serge Beucher and Christian Lantuéjoul. Use of watersheds in contour detection. 1979. https://www.researchgate.net/publication/230837989_Use_of_Watersheds_in_Contour_Detection.
- [12] Francisco Javier Bolívar and Francisco Javier Melero. 3dcurator: A 3d viewer for cts of polychromed wood sculptures. In Alejandro García-Alonso and Belen Masia, editors, *Spanish Computer Graphics Conference (CEIG)*. The Eurographics Association, 2016. <http://dx.doi.org/10.2312/ceig.20161311>.
- [13] Andriy Fedorov, Reinhard Beichel, Jayashree Kalpathy-Cramer, Julien Finet, Jean-Cristophe Fillion-Robin, Sonia Pujol, Christian Bauer, Dominique Jennings, Fiona Fennessy, Milan Sonka, John Buatti, Stephen Aylward, James Miller, Steve Pieper, and Ron Kikinis. 3d slicer as an

- image computing platform for the quantitative imaging network. *Elsevier*, pages 1323–41, November 2012. <http://dx.doi.org/10.1016/j.mri.2012.05.001>.
- [14] Robert Iván García Zavaleta. La historia y las generaciones de la tomografía computarizada. 2014. https://issuu.com/ivangz/docs/la_historia_de_la_tomografia_y_sus_
 - [15] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985. <http://dx.doi.org/10.1117/12.948400>.
 - [16] Karl Krissian, Francisco Santana-Jorge, Daniel Santana-Cedrés, Carlos Falcón-Torres, Sara Arencibia, Sara Illera, Agustín Trujillo, Claire Chalopin, and Luis Alvarez. Amilab software: Medical image analysis, processing and visualization. In *MMVR*, pages 223–237, January 2012. http://researchgate.net/publication/221853670_AMILab_software_medical_image_analysis_processing_and_visualization.
 - [17] Eric N Mortensen and William A Barrett. Intelligent scissors for image composition. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 191–198. ACM, 1995. <http://dx.doi.org/10.1145/218380.218442>.
 - [18] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979. <http://dx.doi.org/10.1109/TSMC.1979.4310076>.
 - [19] Antoine Rosset, Luca Spadola, and Osman Ratib. Osirix: An open-source software for navigating in multidimensional dicom images. *Journal of Digital Imaging*, pages 205–216, September 2004. <http://dx.doi.org/10.1007/s10278-004-1014-6>.
 - [20] María Francisca Sarrió Martín. *Aplicación de la tomografía computarizada médica para el análisis y estudio en escultura policromada en madera*. November 2015. <http://hdl.handle.net/10251/61986>.
 - [21] J. Gonzales Vasquez. *Manual Práctico de Tomografía*. 2011. https://www.academia.edu/10780497/MANUAL_PRACTICO_DE_TOMOGRAFIA.