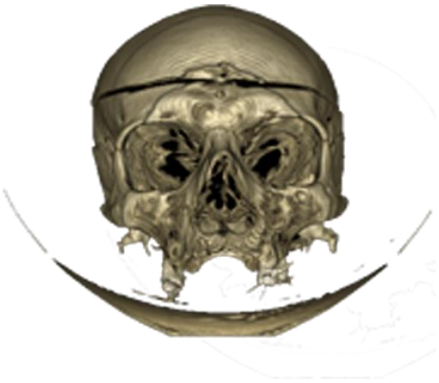
Universidad de las Ciencias Informáticas
Facultad 5

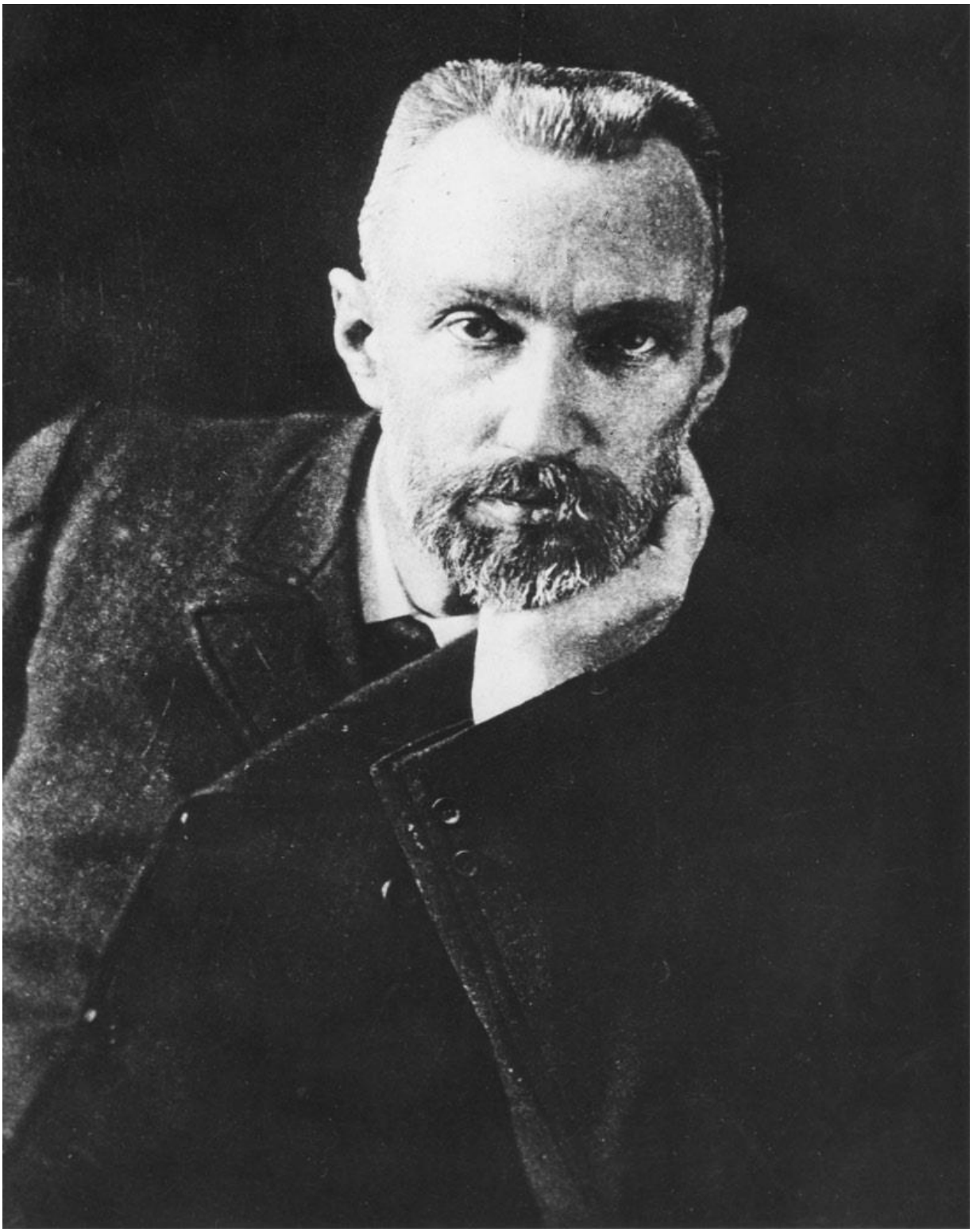
Visualización de estructuras anatómicas 3D para sistemas de neuronavegación guiada por imágenes.



Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Alejandro Ravelo Julian
Tutor: MSc. Ernesto de la Cruz Guevara Ramírez
Ing. Mileydi Moreno Mirabal

"Año 57 de la Revolución"
La Habana, Cuba
Junio 2015



Hay que hacer de la vida un sueño y de un sueño una realidad.

Pierre Curie.

Declaración de autoría

Declaramos ser los únicos autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

Alejandro Ravelo Julian

Autor

_____ _____

MSc. Ernesto de la Cruz Guevara Ramírez

Tutor

Ing. Mileydi Moreno Mirabal

Tutor

Datos de contacto

Tutor:

MSc. Ernesto de la Cruz Guevara.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: elguevara@uci.cu

Graduado como Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2008. Profesor asistente con 8 años de experiencia en el trabajo con Gráficos por Computadoras, Visión por Computadoras y Realidad Aumentada. Actualmente pertenece al Centro de Vertex Entornos Interactivos 3D de la Facultad 5 de la UCI.

Co-tutor:

Ing. Mileydi Moreno Mirabal.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: mmirabal@uci.cu

Graduada como Ingeniera en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2008. Profesor instructor con 8 años de experiencia en el trabajo con Gráficos por Computadoras, Visión por Computadoras y Realidad Aumentada. Actualmente pertenece al Centro de Vertex Entornos Interactivos 3D de la Facultad 5 de la UCI.

Dedicatoria

A mi madre:

Porque las palabras no son suficientes para describir el amor que me profesa. Por su sacrificio, paciencia, cariño y ternura. Porque sin ella todo pierde sentido. Por sus besos, sus abrazos, sus lágrimas. Por ser madre, hermana y amiga. Por ser mi todo y porque quiero serlo todo para ella, porque la amo.

A mis abuelos:

Por estar siempre en mi vida, física y espiritualmente. Por sus consejos y sabiduría, por enseñarme a amar y ser quien soy.

A mi familia:

Porque aún en tiempos difíciles siempre encontraron amor. Por todo el cariño brindado, por apoyarme en todo momento, por quererme y confiar en mí.

A mis amigos:

Por los momentos vividos juntos, los buenos y los malos, por todas las risas y por su apoyo incondicional.

A mis tutores:

Por su sacrificio, por las horas dedicadas, por su confianza y entrega.

A todos los que de una forma u otra me apoyaron en todo este tiempo.

Agradecimientos

Le doy gracias a todas las personas que me apoyaron todo este tiempo y a los que me ayudaron a que este momento se hiciera realidad.

A mi madre Ana María, por ser la luz de mi vida, por tantas noches en desvelo, porque mi sueño es tu sueño. Porque estás conmigo siempre, en mis logros, en mis fracasos, en la felicidad y en esos momentos donde mi sufrimiento se convierte en un infierno para ti. Gracias por ser mis ojos y mi corazón, gracias porque sé que no hay parte de ti que no viva por mí y porque no hay parte de mí que pueda vivir sin ti. No encuentro palabras para decirte cuanto te quiero y que te necesito en mi vida. Gracias por creer en mí, eres lo más grande que tengo en mi vida, gracias por ser especial, gracias por existir, te quiero.

A mi abuelo Fernando por todo su amor, porque fuiste tú quien estuvo siempre a mi lado. Por indicarme el camino y tomarme de la mano para recorrerlo. No tuve tiempo de agradecerte en vida lo suficiente, pero hoy lo hago para que donde sea que te encuentres estés orgulloso de mí. Gracias por todo "papi negro".

A mi abuela Hilda porque su amor es tan grande que alcanza para todos. Gracias por quererme y enseñarme tantas cosas lindas, gracias por todo "mami".

A Amarilis y Cudina porque su cariño no tiene límites, porque me dejaron ser su hijo y me quisieron como tal, porque no tengo como agradecerles todo lo que me han dado, gracias por ser parte de mi vida, los quiero.

A mi tío Fernando, a mi tía Rosario, a mis hermanas Yisel y Yiraelsa, a mis primos Carlitos, Jorgito, Pedrito, Bertica, Katia, Karen, Adrián, Alejandrito, Karina, Karenia, Isabelita, Alexander, Iraima, Neidy y Alfonsito, a Yanetsis, a mis tías Pelusa, Maité, Amalia, Martha, Mirna, Pipe, Irene, Marisol y al resto de la familia por todo su apoyo y cariño, porque son parte de todo lo que he logrado, de mis alegrías y parte de mí.

A mis tutores Ernesto y Mileydi por toda su confianza, por su apoyo incondicional, gracias por todo el tiempo dedicado, gracias por ayudarme a cumplir mi sueño.

A mis amigos Fernando, Renides, Eiler, Ariel, Eliecer, Yoelkys, Yor, Mayrelis y Lili por los momentos juntos, los que ya pasaron y los que están por venir, gracias por apoyarme en todo momento y por estar ahí siempre que los necesité.

A Jesús y a Lía, a Rodrigo y Abel, porque más que amigos son familia, mis hermanos gracias porque sé que aún en la distancia puedo contar con ustedes.

A Yoe y a Rosi, por todo lo que hemos vivido, porque son personas maravillosas, porque ahora no imagino como haber pasado este tiempo sin ustedes, gracias por todo.

A Lili, Nani y Anie por llegar a mi vida para quedarse, por ganarse un pedacito de mi corazón, por confiar en mí, por todos los bellos momentos que me regalaron, gracias.

A mis amigos de la vocacional, los que nunca se olvidan, los que después de tanto tiempo todavía están ahí cuando se les necesita.

A mi grupo por ser los mejores compañeros que se podrían tener, por todo lo que hemos alcanzado juntos y alcanzaremos, porque sé que va a ser imposible que los olvide, gracias por estar siempre ahí, gracias por enseñarme que personas como ustedes pueden existir.

A los profesores que a lo largo de mi carrera han hecho de mí una mejor persona, gracias porque de ustedes me llevo lo mejor. Gracias en especial: A Susej, por ser una persona inigualable, por ser quién eres, no cambies. A Zoraida, porque el amor que siente por sus estudiantes hace que cada uno de nosotros te quiera como una madre, gracias por ser simplemente tú.

A mi equipo de pelota, por los sentimientos creados, porque hemos sufrido juntos, porque si me dieran la oportunidad de escoger a otro equipo los escogería de nuevo, gracias por dejarme ser parte de ustedes.

A todos los del edificio que más que amigos somos familia, porque todavía nos quedan momentos por vivir y reír juntos.

A mis vecinos y amigos que siempre se han preocupado por mí: Denia, Rafelito, Carmen, Cirilo, Landito, Catey, Nirma, Valeria y otros.

Resumen

El avance de la informática ha permitido desarrollar nuevas herramientas de apoyo a los cirujanos en el campo de la imagenología médica y la cirugía asistida por ordenador. Estas herramientas asisten al médico en el diagnóstico, planificación y ejecución de la intervención quirúrgica. Entre ellas se destaca el neuronavegador como herramienta de última generación que ha sustituido procedimientos tradicionales en muchos hospitales del mundo. La presente investigación se realiza a partir de la necesidad de generalizar este procedimiento en todo el país. Este trabajo se enfoca en resolver la problemática de incrementar la información espacial que recibe el cirujano con respecto a la información anatómica del paciente en la etapa de planificación. El objetivo es implementar un componente de software para la herramienta de planificación quirúrgica de un neuronavegador que sea capaz de visualizar un volumen 3D reconstruido a partir de las imágenes DICOM del paciente. Se elaboró un componente de visualización de imágenes 3D y de edición de funciones de transferencia de color y opacidad. Estos componentes se integraron al prototipo de neuronavegación que se desarrolla en la facultad 5. La implementación de este componente logró aumentar la información espacial que recibe el cirujano, así como diseñar funciones de transferencia para personalizar los colores del volumen 3D de forma interactiva. Se desarrolló un caso de estudio utilizando un cráneo de pruebas con su correspondiente estudio de tomografía. Las pruebas realizadas al componente desarrollado arrojaron resultados aceptables, lo que permite comenzar a desarrollar las funcionalidades principales de un neuronavegador quirúrgico.

Palabras Claves: imágenes médicas, neuronavegación, visualización médica, volumen 3D.

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de gráficos

Introducción

Tradicionalmente, en intervenciones neurológicas complejas, era frecuente que el cirujano tuviera que abandonar el campo quirúrgico; observar todas las placas radiográficas, y hacerse una composición imaginaria del lugar donde se encontraba en el momento en que había detenido la intervención. Gracias al avance de la informática, la imagenología médica, y la cirugía asistida por ordenador, se ha evolucionado en el desarrollo de nuevas herramientas de apoyo a los cirujanos. Estas herramientas asisten al médico en el diagnóstico, planificación y ejecución de la intervención quirúrgica; con el objetivo de proporcionar mejores resultados clínicos, disminuir el tiempo en el quirófano y minimizar los riesgos al paciente. Una de las herramientas que se destaca en este campo es el neuronavegador, el cual brinda al cirujano y su equipo profesional, seguridad y confianza para el tratamiento quirúrgico de lesiones cerebrales.

La principal desventaja de la neuronavegación es que necesita de equipos que tienen un alto costo de adquisición en el mercado internacional, así como el costo de mantenimiento y actualización. Las posibilidades técnicas y el desarrollo alcanzado en nuestro país en las ciencias informáticas, sirven como base para la fabricación de equipos como estos. Estos equipos una vez desarrollados podrán distribuirse por los hospitales cubanos y las colaboraciones médicas en el exterior.

Teniendo en cuenta las posibilidades técnicas y el desarrollo alcanzado en nuestro país en las Ciencias Informáticas, la Clínica Central Cira García (CCCG), en colaboración con la Sociedad Cubana de Neurología y Neurocirugía (SCNN) y el Centro Internacional de Rehabilitación Neurológica (CIREN) ha propuesto a la Universidad de las Ciencias Informáticas (UCI) la fabricación de un neuronavegador. El cual una vez desarrollado podrá expandirse por los hospitales nacionales y de las colaboraciones médicas en el exterior.

Los sistemas de navegación guiada por imágenes, específicamente los de neuronavegación, poseen tres componentes principales: visualización de imágenes médicas, sistema de seguimiento tridimensional y registro de correlación.

Existe una etapa previa a la navegación guiada por imágenes en la que el cirujano debe planificar la intervención quirúrgica. En esta etapa, el paso inicial es cargar las imágenes de estudios médicos, las cuales pueden contener información anatómica, morfológica y funcional del paciente. Con estas imágenes el cirujano planifica cómo debe realizar la intervención. Las modalidades de imágenes más utilizadas en los sistemas de navegación guiada por imágenes son la tomografía axial computarizada (TAC), la resonancia magnética (RM) y las tomografías por emisión de positrones (PET) (1). El componente de visualización en un sistema de neuronavegación guiada por imágenes es el responsable de cargar y visualizar los archivos de imágenes médicas, además de los objetos visuales que representan las herramientas quirúrgicas.

En la etapa de planificación es donde el cirujano practica los pasos de la intervención quirúrgica y adquiere la información espacial necesaria de la anatomía del paciente. No obstante, aunque las imágenes en dos dimensiones (2D) ofrecen información de la anatomía del paciente, es frecuente que el cirujano no se ubique con precisión en la región anatómica en la que debe incidir. Para resolver esta problemática se han realizado varios trabajos para visualizar estructuras anatómicas en tres dimensiones (3D) a partir de imágenes médicas.

En el centro Vertex se está desarrollando un software de navegación guiada por imágenes, que cuenta con una herramienta para la planificación de la intervención quirúrgica en una etapa previa al quirófano. Esta herramienta solo brinda las vistas clásicas axial, coronal y sagital y una vista multiplanar, en la que se muestran las vistas anteriores como planos intersectados en 3D. Esta herramienta cuenta con las desventajas anteriormente explicadas que dieron lugar al surgimiento de las técnicas de visualización de volúmenes 3D para representar estructuras anatómicas. Es por esto que se necesita incorporar funcionalidades de visualización de volúmenes a la herramienta de planificación, para incrementar la noción espacial del cirujano con respecto a la información anatómica del paciente.

Teniendo en cuenta lo anteriormente expuesto se plantea el siguiente **problema de científico**: ¿Cómo incrementar la información espacial que recibe el cirujano con respecto a la información anatómica del paciente en la etapa de planificación?

La investigación tiene como **objeto de estudio:** la visualización de imágenes médicas en 3D, constituyendo el **campo de acción:** técnicas de visualización para sistemas de neuronavegación guiada por imágenes.

Para darle solución a este problema se propone el siguiente **objetivo general**: implementar un componente de software para la herramienta de planificación quirúrgica del neuronavegador que muestre la visualización del volumen 3D de las imágenes del paciente.

Para resolver el problema científico y darle cumplimiento al objetivo, se plantearon las siguientes **tareas de investigación**:

1. Análisis de la bibliografía relativa a los sistemas de neuronavegación guiada por imágenes para conocer su estructura y componentes principales.
2. Selección de la técnica de visualización de volumen más eficaz para el cumplimiento del objetivo propuesto.
3. Selección de los algoritmos de visualización tridimensional de imágenes médicas.
4. Implementación de un componente de software para la visualización de estructuras anatómicas en 3D, para la herramienta de planificación quirúrgica del neuronavegador.
5. Validación de los resultados obtenidos al introducir el componente de software implementado al prototipo de neuronavegador.

Además para todo el proceso de investigación y elaboración de este trabajo se tomará en cuenta la utilización de varios **métodos científicos de investigación** como:

- **Histórico-lógico:** método que permitirá conocer los antecedentes y las tendencias actuales referidas a las herramientas de neuronavegación guiada por imágenes, además de conceptos, términos y vocabularios propios del campo que contribuyen en gran medida al entendimiento del trabajo.
- **Analítico-sintético:** mediante este método se podrá estudiar y analizar una serie de documentos y teorías relacionados con los sistemas de neuronavegación guiada por imágenes

y extraer los elementos más importantes de estos, los componentes por los que están conformados, su funcionamiento y la interacción entre ellos.

- **Revisión bibliográfica:** este método permitirá determinar el estado del arte del objeto de investigación.
- **Entrevistas:** método mediante el cual se obtendrá información sobre el tema de la investigación.
- **Monitoreo de proyectos:** se analizarán los posibles resultados y el impacto que tendría consigo la inserción de un producto de este tipo en la sociedad.

El presente trabajo de diploma está estructurado de la siguiente forma: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, bibliografía, glosario de términos y anexos. A continuación se hace una breve descripción del contenido de cada uno de los capítulos:

Capítulo 1 "Fundamentación Teórica": se hace un análisis de los principales temas concernientes al objeto de estudio, se analizan los principales conceptos y términos asociados al problema en cuestión, así como otras soluciones existentes. Se refleja el estado del arte del tema de investigación. Se analizan los principales componentes de un neuronavegador, las principales técnicas de visualización 3D de imágenes médicas que se utilizan en las aplicaciones de neuronavegación guiada por imágenes y las bibliotecas utilizadas para el desarrollo de este tipo de aplicaciones.

Capítulo 2 "Solución Propuesta": se exponen las características de la propuesta de solución. Se realiza un levantamiento de requisitos, se obtienen las historias de usuarios, se realiza el plan de entregas y las tarjetas CRC. Además se exponen los elementos del diseño de la solución propuesta.

Capítulo 3 "Implementación y Pruebas": se realiza la implementación de las historias de usuarios, se chequea el plan de iteraciones y se crean las tareas de ingeniería para implementar exitosamente casa historia de usuario. Además, se demuestra el cumplimiento del objetivo del trabajo a través del análisis del resultado de las pruebas realizadas a la solución.

Capítulo #1. Fundamentación teórica

1.1 Introducción

En este capítulo se abordan los principales elementos teóricos que conforman los sistemas de navegación guiada por imágenes. Se exponen los componentes principales que están presentes en un sistema de neuronavegación. Se identifican las bibliotecas y programas existentes para la visualización 3D de imágenes médicas. Se describe un volumen de datos y las diferentes técnicas de visualización de volumen, así como los algoritmos de visualización. Se caracteriza el formato de ficheros DICOM, las diferentes metodologías de desarrollo de software, el lenguaje de programación y el entorno de desarrollo a utilizar.

1.2 Conceptos y términos asociados al dominio de la investigación

En el presente epígrafe se abordan temáticas comprendidas en el dominio de la investigación. Se muestran una serie de conceptos identificados durante la investigación realizada, así como la descripción de cada uno para facilitar la comprensión de los temas abordados.

Para una mejor comprensión de los temas abordados en este trabajo es necesario conocer algunos conceptos los cuales se exponen en los siguientes epígrafes.

1.2.1 Cirugía asistida por ordenador

La cirugía asistida por ordenador (CAO) surgió por una necesidad de la sociedad (2), la cual demanda una mejora en los tratamientos quirúrgicos. La CAO constituye el conjunto de métodos que utilizan la tecnología informática como apoyo en la realización de la actividad quirúrgica (2). Esta facilita al cirujano herramientas informáticas que le asisten en el diagnóstico, planificación y ejecución de la intervención quirúrgica. El objetivo de la CAO es proporcionar mejores resultados clínicos, con menos procesos y mayor precisión, disminuir el tiempo en el quirófano y minimizar los riesgos para el paciente (2).

Para el apoyo a una CAO se utilizan herramientas informáticas que en muchas ocasiones visualizan volúmenes 3D para lograr representar mayor información de la anatomía del paciente. Con el uso de estas herramientas surge un nuevo concepto clave para el desarrollo de este trabajo y es el concepto de navegación guiada por imágenes.

1.2.2 Navegación guiada por imágenes

La navegación, por definición, es el proceso de determinar y mantener un camino o trayectoria hacia una ubicación objetivo (3). Este término es adoptado en la medicina para expresar la orientación espacial respecto a un volumen anatómico (2). La navegación guiada por imágenes en cirugía consiste en determinar las coordenadas de los instrumentos quirúrgicos en la mesa de operaciones y mostrar su posición en las imágenes de estudios previos para guiar al cirujano en zonas de difícil visibilidad o acceso, lo que ayuda a la toma de decisiones (4).

1.3 Neuronavegación

Watanabe (5) fue el primero en utilizar el término neuronavegación y R. L. Galloway (6) lo definió más tarde como: “localizar la posición quirúrgica en el espacio físico y presentar la posición en el espacio de la imagen”. La neuronavegación consiste en un método médico de orientación espacial intracraneal intraoperatorio, a través de la superposición de los instrumentos quirúrgicos sobre las imágenes médicas preoperatorias, esto permite al neurocirujano una guía interactiva por la imagen (2).

Según el doctor Leonardo Lustgarden (7) la aplicación de la neuronavegación depende de la ubicación de la lesión dentro del cráneo y cerebro, pero generalmente es útil para:

- La correcta y adecuada localización de tumores cerebrales en tiempo real y en forma continua dentro del cerebro, en particular en ubicaciones difíciles.
- Ayuda de forma óptima en la delimitación de tumores cerebrales u otras lesiones ocupantes de espacio.
- Es particularmente útil para aquellas lesiones que radiológicamente muestran una descripción muy parecida a la del cerebro y por ello son difíciles de distinguir.
- Permite la correcta ubicación de las lesiones pequeñas y ayuda a delimitar las lesiones grandes.
- Permite planificar las mejores trayectorias para abordar diferentes lesiones intracerebrales.
- Permite correlacionar y visualizar las estructuras vitales cercanas a la lesión (nervios ópticos, tallo cerebral, entre otros).

Entre las ventajas que la neuronavegación (7) proporciona al paciente, se encuentran:

- Menor tiempo quirúrgico.
- Menor riesgo para las zonas de importancia funcional.
- Mayor radicalidad en las resecciones.
- Menor número de complicaciones postquirúrgicas (infecciones, fístulas, entre otras).
- Corta estancia hospitalaria.
- Menor defecto cosmético.

Para comprender mejor el funcionamiento de un sistema de neuronavegación guiada por imágenes es de vital importancia analizar las características de estos sistemas, sus funcionalidades y componentes principales.

1.3.1 Componentes de un sistema de neuronavegación guiada por imágenes

Los sistemas neuronavegación guiada por imágenes poseen tres componentes principales (2):

- Visualización de imágenes médicas: este es el componente utilizado para cargar y visualizar las imágenes médicas de distintas modalidades.
- Sistema de seguimiento tridimensional: es el encargado de obtener la posición, orientación, y seguir la trayectoria de las herramientas quirúrgicas en el quirófano.
- Registro de correlación: permite establecer una correspondencia entre cada punto de la imagen y su correspondiente anatómico físico o real.

Este trabajo se enfoca en el componente de visualización de imágenes médicas y para una mejor comprensión de este será tratado en el siguiente epígrafe.

1.4 Visualización de imágenes médicas

El primer paso para llevar a cabo la navegación guiada por imágenes, es cargar las imágenes de estudios médicos, las cuales contienen la información anatómica, morfológica y funcional del paciente. Existes diferentes modalidades de imágenes médicas, las cuales se diferencian en cuanto a la naturaleza del principio físico en que se basan. Las modalidades de imágenes más utilizadas en los sistemas de navegación guiada por imágenes son la TAC, la RM y las PET (1). El componente de visualización en un sistema de neuronavegación guiada por imágenes es el responsable de cargar y

visualizar los archivos de imágenes médicas, además de los objetos visuales que representan las herramientas quirúrgicas (2).

1.5 Bibliotecas para la visualización de imágenes médicas

A continuación se mencionan algunas bibliotecas diseñadas para el procesamiento de imágenes o visualización de imágenes específicamente visualización 3D aplicadas a la práctica médica.

1.5.1 Image-Guided Software Toolkit (IGSTK)

IGSTK es una biblioteca multiplataforma de software libre que provee un conjunto de componentes básicos necesarios para desarrollar aplicaciones de cirugía guiada por imágenes (8). Consiste en un conjunto de componentes de alto nivel integrado a un conjunto de bibliotecas de bajo nivel y de interfaz de programación de aplicaciones (APIs). IGSTK está construida sobre *Insight Segmentation and Registration Toolkit* (ITK) (9) y *Visualization Toolkit* (VTK) (10). IGSTK permite al investigador desarrollar rapidamente un prototipo de software para cirugía guiada por imágenes de mínimo acceso (11).

1.5.2 ITK

ITK es una biblioteca de código abierto de segmentación y registro de imágenes, escrita en C++, fue desarrollada para analizar las imágenes de *The Visible Human Project*. El desarrollo de ITK fue financiado por la *National Library of Medicine* y uno de sus principales contribuidores fue la compañía Kitware Inc (12).

ITK no implementa una interfaz gráfica o de visualización, tarea que es llevada a cabo por otras bibliotecas, como VTK. Igualmente, esta biblioteca provee una mínima funcionalidad para el manejo de archivos. Esta biblioteca incluye importantes algoritmos de registro y segmentación en dos y tres dimensiones. También tiene soporte para procesamiento paralelo y multihilo (12).

ITK está basada en una arquitectura de flujo de datos. Esto significa que hay objetos de datos que son procesados por objetos de procesamiento y que ambos están conectados a través de una tubería (12).

1.5.3 VTK

VTK es una biblioteca de código abierto para representaciones visuales por ordenador, procesamiento de imágenes y visualización, usada por cientos de investigadores y desarrolladores en todo el mundo (10).

VTK fue creada inicialmente en 1993 e incluida en el libro "*The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*" por W. Schroeder, K. Martin y B. Lorensen. El desarrollo conceptual de la tubería VTK (similar a la tubería ITK) se muestra en la f**igura 1**. Los datos se leen en el módulo fuente y luego son filtrados por uno o más filtros. Un asignador es usado entonces para crear una representación visual con la que se puede interactuar y transformar por medio de un actor.

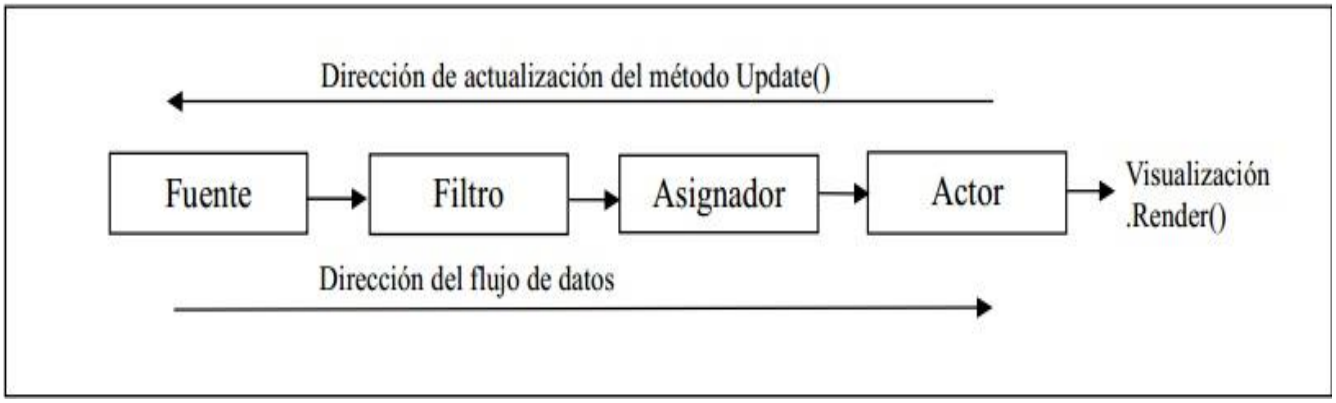


Figura 1. Esquema de tuberías VTK (13)**.**

También otros procesos relacionados con el sistema de funcionamiento y desarrollo de VTK son similares a los de ITK. A continuación se muestran sus principales características:

- El código fuente se ajusta al sistema de plantillas propuesto por el lenguaje C++.
- La herramienta es multi-plataforma, soportada en Windows, MacOS X, y la mayoría de los sistemas GNU\Linux.
- Existen 'envoltorios' (*wrappers*) para varios lenguajes de programación (Tcl, Python y Java).
- Utiliza su propio sistema de referencia para la gestión automática de memoria con apuntadores inteligentes (conocidos como *smart pointers*) (14).

1.5.4 Extensiones de VTK e ITK

1.5.4.1 Medical Imaging Interaction Toolkit (MITK)

MITK es una biblioteca escrita en C++ para el desarrollo de software para la visualización e interacción con imágenes médicas. Provee una capa de abstracción como extensión de las bibliotecas VTK e ITK. Además MITK añade soporte para la sincronización en diferentes ventanas de visualización y permite la construcción y modificación de objetos de datos. MITK puede ser añadida a aplicaciones existentes y permite la construcción de aplicaciones con tareas específicas sin otros complementos innecesarios. También provee extensiones para la biblioteca IGSTK (1).

1.5.4.2 KWWidgets

KWWidgets es una interfaz gráfica de usuario (GUI), que proporciona elementos de interacción de bajo nivel, como botones, campos de texto, menús y similares para bibliotecas como VTK. Es la propuesta de GUI que presenta Kitware para su familia de productos, con opciones avanzadas, aunque carece de editores visuales de GUI (15).

1.5.4.3 OpenGL Volumizer

OpenGL Volumizer es una aplicación comercial multiplataforma de visualización de volúmenes. Se presenta como una API para los mercados energéticos, industriales, médicos y científicos. Es una API diseñada para la visualización interactiva de grandes conjuntos de datos (16).

1.5.4.4 Vascular Modeling Toolkit (VMTK)

VMTK es un conjunto de clases escritas en C++ para la segmentación y el análisis geométrico de vasos sanguíneos u objetos tubulares en general, desarrollado por David Steinman y Luca Antiga (17).

1.5.5 Otros programas de imágenes médicas

1.5.5.1 Volview

VolView es una aplicación de código abierto, es un sistema interactivo para la visualización de volumen que permite a los investigadores explorar y analizar complejos conjuntos de datos médicos o científicos en 3D. Los usuarios pueden cargar y de forma interactiva explorar el conjunto de datos utilizando métodos y herramientas de visualización en 2D y 3D (18).

1.5.5.2 ParaView

ParaView es una aplicación de código abierto, multiplataforma para la visualización y el análisis de datos. Con ParaView los usuarios pueden crear rápidamente visualizaciones para analizar sus datos utilizando técnicas cualitativas y cuantitativas. La exploración de datos se puede realizar de forma interactiva en 3D (19).

1.5.5.3 3D Slicer

3D Slicer es una aplicación de código libre y multiplataforma para la visualización y el análisis de imágenes. Ha sido desarrollada con KWWidgets, TCL, VTK e ITK (20).

1.5.5.4 SCIRun

SCIRun es un programa usado en una amplia gama de aplicaciones incluyendo procesamiento de imágenes y visualización de volúmenes 3D. Su ventaja es la integración de Matlab e ITK (21).

1.5.6 Selección de las bibliotecas para la visualización de volúmenes 3D.

A continuación se representa en la **tabla 1** un resumen de los principales sistemas orientados al procesamiento de imágenes.

Nombre	Lenguaje	Licencia	Propósito	Origen	Desarrollador
VTK	C++	Código abierto	Visualización	1993	Kitware
ITK	C++	Código abierto	Registro y segmentación	1999	Kitware
VGL	C++	Comercial	Visualización de grandes volúmenes de datos	1997	Volume Graphics
OpenGl Volumizer	C++	Comercial	Visualización de grandes volúmenes de datos	2002	SGI
IGSTK	C++	Código abierto	Aplicaciones de cirugía guiada por imágenes	2003	Kitware
MITK	C++	Código abierto	Extensiones ITK y VTK	2004	Kitware

Tabla 1. Resumen de los sistemas de procesamiento de imágenes (13).

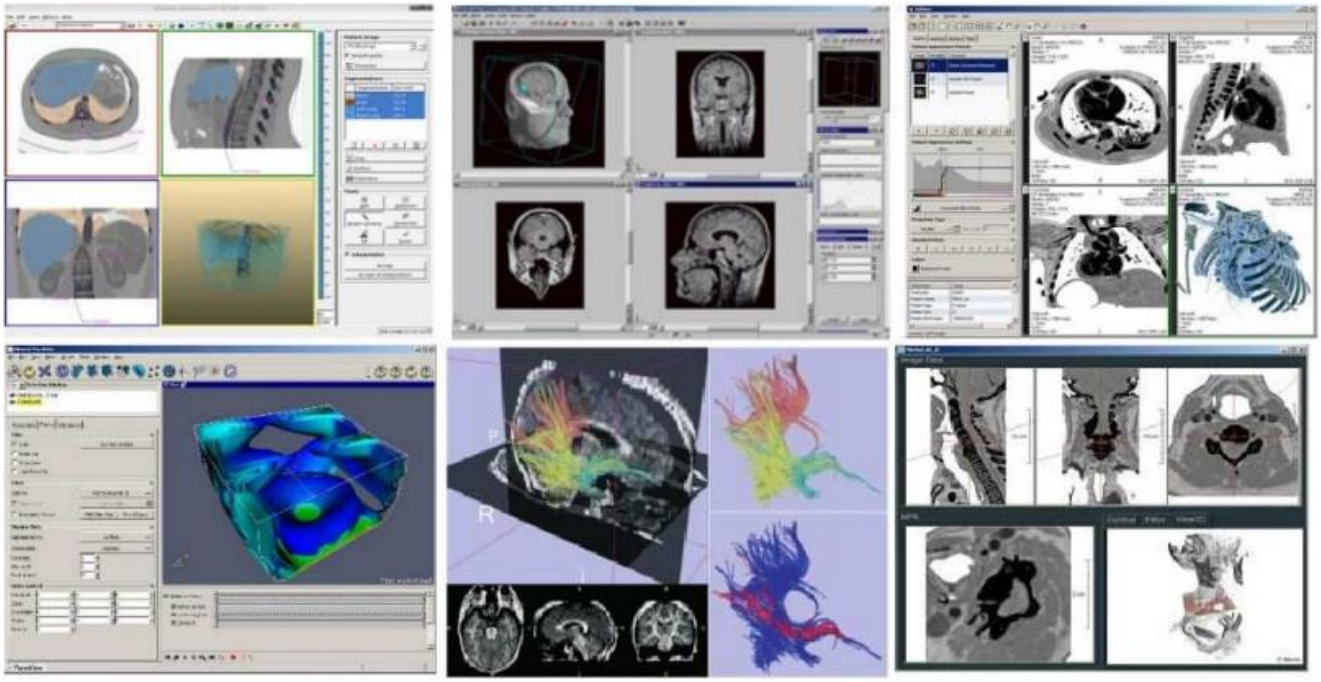


Figura 2. Captura de pantalla de algunos programas de visualización. **MITK** (arriba izquierda), **VGStudio** (arriba centro), **VolView** (arriba derecha), **ParaView** (abajo izquierda), **3D Slicer** (abajo centro) y **MeVisLab** (abajo derecha) (13).

En la **tabla 1** se evidencia a Kitware como una de las compañías especializadas en el desarrollo de aplicaciones de visualización de volúmenes y que cuenta con enorme experiencia en este campo.

Se seleccionó IGSTK como biblioteca para el desarrollo del componente de visualización de imágenes médicas. Esta proporciona los componentes básicos necesarios en una aplicación de navegación quirúrgica, provee una plataforma común para el desarrollo de aplicaciones robustas de cirugía guiada por imágenes. Esta biblioteca está aprobada por la Agencia de Administración de Alimentos y Medicamentos de los Estados Unidos (FDA). Este conjunto de herramientas utiliza la biblioteca ITK para el registro y la segmentación, y VTK para la visualización, ambas bibliotecas están en constante desarrollo y constituyen un estándar en el campo de las aplicaciones de navegación guiada por imágenes.

Se aprecia también en la **tabla 1** que las principales bibliotecas para la visualización de volúmenes 3D están desarrolladas en el lenguaje de programación C++ lo que demuestra las potencialidades de este lenguaje para el desarrollo de aplicaciones de este tipo. En el siguiente epígrafe se exponen algunas características del mismo.

1.6 Lenguaje de Programación

C++ es un lenguaje de programación de propósito general, especialmente indicado para la programación de sistemas por su flexibilidad y potencia. Por sus características es uno de los lenguajes más utilizados por la comunidad de desarrollo de software, incluyendo la programación gráfica, para lo que es ideal por el alto grado de rapidez y optimización que provee (2). Se selecciona C++ como lenguaje de programación a utilizar en el desarrollo de la solución.

1.7 Entorno de desarrollo integrado (IDE)

Qt Creator es un IDE creado para el desarrollo de aplicaciones en C++, es multiplataforma, consta con un depurador visual, un editor avanzado para C++, completamiento automático de código. Cuenta con herramientas para la administración y construcción de proyectos, diseñador de formularios GUI integrado, ayuda sensible al contexto integrada. Tiene soporte para refactorización de código (22). Por las características anteriormente expuestas se selecciona Qt Creator como IDE a utilizar para el desarrollo de las interfaces GUI del componente de visualización de imágenes médicas.

1.8 Volumen de datos

Para la visualización de un volumen es necesaria la adquisición de un conjunto de imágenes que representan cortes del objeto escaneado, una serie de muestras que conforman un ortoedro, donde cada imagen representa una parte del objeto y está compuesta por píxeles que representan los elementos de la imagen. Estos píxeles están organizados en una red bidimensional como se muestra en la parte izquierda de la **figura 3**. Los datos volumétricos combinan imágenes individuales en una red 3D como se muestra en la parte derecha de la **figura 3**. Los elementos individuales ahora son llamados vóxeles que representan los elementos del volumen (23).

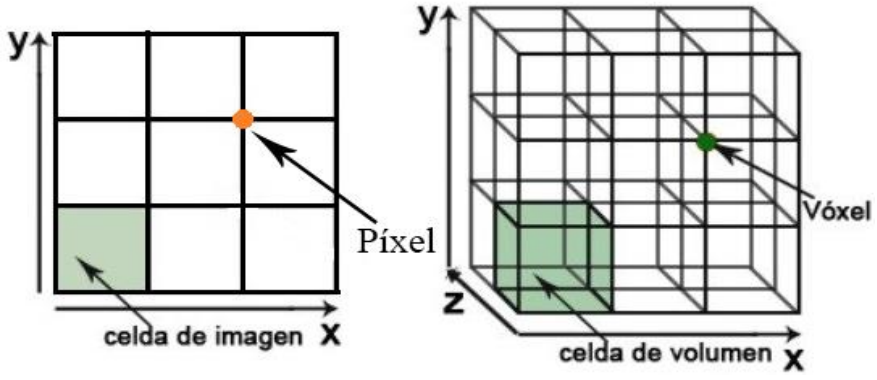


Figura 3. Izquierda: Red de imagen 2D. Derecha: Volumen de datos (23).

1.9 Visualización de volumen

La visualización de volumen consiste en la representación visual del conjunto completo de datos volumétricos, lo que es decir de todas las imágenes al mismo tiempo (24). Los vóxeles individuales deben ser seleccionados, combinados y proyectados sobre el plano de una imagen. Esta imagen actúa literalmente como una ventana de los datos, representando la posición y la dirección desde donde el volumen es examinado por un observador (23).

Existen dos tipos de visualización de volumen, la directa y la indirecta, las cuales serán tratadas a continuación.

1.9.1 Visualización indirecta (VIV)

Mediante la técnica de visualización indirecta de volumen se genera una representación intermedia del volumen de datos que se visualiza posteriormente (24). Los algoritmos de VIV se basan fundamentalmente en el uso de planos (*Planar Rendering*) o superficies geométricas como representación intermedia (*Surface Rendering*) (23).

Actualmente existen varios algoritmos para resolver el problema de la reconstrucción de superficies de un volumen de datos, uno de los más usados en la actualidad es el algoritmo *Marching Cubes* propuesto por William E. Lorensen y Harvey E. Cline en 1987 (23).

1.9.1.1 Marching Cubes

El algoritmo de *Marching Cubes* se publicó en 1987 por Lorensen y Cline y su objetivo es la extracción de mallas poligonales de una isosuperficie a partir de datos escalares tridimensionales (vóxeles). Principalmente se basa en dividir el espacio en vóxeles (cubos) formados por los valores de intensidad de cada una de las 8 esquinas del vóxel que se corresponden con puntos obtenidos de los datos volumétricos procedentes de una TAC. La ventaja de este algoritmo es que es rápido, pero tiene el inconveniente de que no muestra el interior del objeto, por lo que la posibilidad de realizar cortes o disecciones no aporta ninguna información adicional (25).

1.9.2 Visualización directa del volumen (VDV)

Mediante la técnica de VDV, los datos volumétricos se representan mediante su evaluación en un modelo óptico, que describe cómo el volumen emite, refleja, dispersa, absorbe y ocluye la luz. La complejidad de los algoritmos de VDV depende del número de vóxeles de la red y del número de píxeles de la imagen final donde será proyectada la visualización. Los algoritmos de VDV se pueden clasificar en dos grupos: algoritmos basados en imágenes y algoritmos basados en objetos (**figura 4**). Sin embargo, muchas variaciones avanzadas de estos algoritmos no pueden ser clasificadas estrictamente dentro de uno de los grupos, porque fusionan aspectos de ambos grupos en un mismo algoritmo (23).

Figura 4. Visualización directa del volumen.

1.9.2.1 Shear Warp

La idea que persigue el algoritmo de renderizado *Shear Warp* es transformar previamente los datos con los que se va a trabajar, dándoles una orientación concreta a partir de la cual el renderizado pueda ser más rápido. En lugar de proyectar los vóxeles sobre el plano imagen en un ángulo variable, se utiliza una transformación *shear* (cizalla) de forma que se traslada cada rodaja (*slice*) y se muestrea a lo largo de la dirección del rayo. Las rodajas se componen en orden de delante atrás, creando una imagen intermedia, pero en esta ocasión el cálculo es más simplificado puesto que los rayos son perpendiculares a las rodajas del volumen. La desventaja de este algoritmo es que la calidad de imagen conseguida es inferior a la obtenida con métodos como el *Raycasting* (25).

1.9.2.2 Splatting

Splatting fue propuesto por primera vez por Westover (26), es una aproximación orientada al objeto, lo opuesto al *Raycasting* que es orientado a la imagen. Cada vóxel se proyecta sobre el plano imagen o como definía Westover, "*splatted*", como estrellar una bola de nieve, solapando la proyección de los anteriores. Esta proyección se aproxima por un núcleo Gaussiano invariante a la orientación con una amplitud escalada de acuerdo con el valor del vóxel. La imagen es generada al proyectar las funciones base sobre la pantalla. La proyección de estas funciones base con simetría radial puede llevarse a cabo eficientemente mediante el barrido de tablas de huellas precalculadas. Cada una de las entradas de la tabla contiene el núcleo de la función integrada analíticamente a lo largo del rayo que atraviesa el volumen (26).

El método *Splatting* que provee mejor calidad utiliza como función base o función de huella una combinación de un núcleo de reconstrucción elíptica Gaussiana con un filtro paso bajo Gaussiano, por lo tanto la calidad de la imagen es mejorada significantemente. La mayor ventaja del *Splatting* es que los vóxeles que, debido a que son transparentes, no contribuyen a la imagen final no tienen que ser procesados. Esto reduce considerablemente la cantidad de datos a procesar (25).

1.9.2.3 3D Texture Mapping

La aparición del soporte de texturas 3D en las tarjetas gráficas ha permitido su aplicación al renderizado de volumen. La idea básica es interpretar el vector de vóxeles como una textura 3D y entender el mapeo de texturas 3D como la interpolación trilineal del conjunto de datos volumétricos en un punto arbitrario dentro del dominio. Los datos se muestrean sobre planos de corte, que están orientados paralelamente al plano de visión, con los píxeles del plano de corte interpolados trilinealmente a partir de la textura escalar 3D. Esta operación se realiza sucesivamente para múltiples planos que tienen que ser cortados contra el dominio de texturas paramétrico. Los polígonos son renderizados de atrás adelante y las texturas resultantes se componen apropiadamente en el *frame buffer* aproximando de ese modo la integral continua de Low-Albedo (27).

Existen unidades de hardware gráfico dedicadas exclusivamente para la interpolación trilineal dentro de la textura y para la composición de los fragmentos por píxel. Sin embargo, la principal ventaja del renderizado de volumen usando texturas 3D es la utilización de tablas de texturas previas al dibujado. Las muestras escalares reconstruidas a partir de las texturas 3D son convertidas en píxeles RGBA

usando estas tablas. Esto permite manipular directamente la función de transferencia, necesaria para mapear el valor escalar a un valor RGBA, sin la necesidad de recargar la textura entera proporcionándole al usuario una mayor velocidad de interacción. De esta forma se pueden resaltar o eliminar partes arbitrarias de los datos así como utilizar diferentes colores y transparencias (25).

1.9.2.4 Raycasting

Es el algoritmo clásico de visualización directa de volumen. Está basado en el concepto de un rayo de luz que penetra en el volumen e intercepta los vóxeles que encuentra a su paso. Por cada píxel de la ventana a través de la cual se observa la representación tridimensional, se lanza un rayo hacia el interior del volumen, recorriendo una trayectoria rectilínea que comienza en la posición de la cámara virtual y es paralelo al vector direccional de la misma (23). Después de obtener el valor para la muestra, este valor es mapeado a propiedades ópticas a través de una tabla de búsqueda. Esta obtiene como resultado una cuádrupla RGBA que incorpora los coeficientes correspondientes de emisión y absorción para esta ubicación (24). Es muy usado en las implementaciones, que los rayos viajen en sentido contrario al vector direccional de la cámara, proyectando la imagen resultante en la ventana del observador (23).

La mayoría de los métodos de aceleración aplicados al *Raycasting* se centran en evitar cálculos innecesarios, tales como el método de terminación temprana del rayo, en el que el muestreo a lo largo del rayo termina una vez que ha sido alcanzado el máximo de opacidad acumulado, o los espacios en blanco, donde no se procesan áreas totalmente transparentes del volumen de acuerdo a un intervalo de muestreo adaptativo. Hay técnicas que se aprovechan de la coherencia entre píxeles y la coherencia entre vóxeles. La técnica de *supersampling* se basa en aprovechar la coherencia del píxel. En un conjunto de píxeles se trazan los rayos y en el resto se interpola el valor de los píxeles calculados. Para las zonas en las que se ve grandes cambios de gradiente se trazan nuevos rayos. Las ventajas del *Raycasting* es que aporta la mayor calidad de imagen a costa de un procesamiento más lento pero su esquema de cálculo permite explotar cálculos paralelos en las tarjetas gráficas así como estructuras de memoria eficientes para ganar velocidad. Las cuales por la potencia de cálculo ofrecida en la actualidad no demuestra una ralentización apreciable con respecto a las demás técnicas analizadas hasta el momento (25).

1.9.3 Selección de la técnica y el algoritmo de visualización

Los algoritmos de la técnica de visualización indirecta solo reconstruyen la superficie del volumen, tal es el caso del algoritmo *Marching Cubes*. No es conveniente el uso de esta técnica y de sus algoritmos para la visualización del volumen en este trabajo, para la navegación es necesaria la visualización del volumen de datos íntegro. Por estas razones se descarta la técnica de visualización indirecta.

Un estudio realizado por varios autores y publicado en el Simposio de Visualización de Volumen efectuado en Salt Lake en Octubre del 2000 comparó los algoritmos de visualización directa antes expuestos. Para una mejor selección del algoritmo a utilizar, se expondrán los datos obtenidos en el estudio antes mencionado.

Conjunto de datos	Tamaño	Vóxeles Relevantes	Modo de representación	Compactación	Contenido de píxeles
Vaso sanguíneo	256^3	79 442 (0.5%)	isosuperficie opaca	Baja	Bajo
Neghip	64^3	207 872 (79.3%)	moderadamente semitransparente	Alta	Medio
Cráneo	256^3	1 384 817 (8.2%)	isosuperficie opaca	Media	Bajo
Inyección de combustible	64^3	32 768 (12.5%)	semitransparente con el interior estructura opaca	Alta	Medio
Onda expansiva	64^2x512	1 245 184 (59%)	totalmente semitransparente	Alta	Alto

Tabla 2. Conjuntos de datos de referencia y los modos de representación (28).

En la **tabla 2** se exponen los conjuntos de datos y los modos de representación utilizados en el estudio mencionado anteriormente, para este conjunto de datos se obtienen los resultados expuestos en la **tabla 3**.

	Inyección de combustible	Neghip	Cráneo	Vaso sanguíneo	Onda expansiva
Raycasting	4.96	8.15	7.78	12.31	3.02
Splatting	1.41	7.35	11.09	1.87	21.77
Shear Warp	0.09	0.24	0.27	0.09	0.91
3D Texture Mapping	0.06	0.04	0.7	0.7	0.14

Tabla 3. Tiempo medio de fotogramas en segundos para 24 vistas aleatorias (28).

Como se puede observar en la **tabla 3** el algoritmo *Raycasting* obtiene mejores resultados para 3 de los 5 juegos de datos analizados, por lo que nos da una muestra de lo eficaz que puede ser este algoritmo para la visualización de volúmenes de datos. Por lo anteriormente expuesto y porque VTK nos permite su uso, se escoge el algoritmo *Raycasting* como algoritmo para la visualización del volumen 3D.

Uno de los principales aportes de la visualización de volumen, especialmente en aplicaciones médicas, es la habilidad de distinguir diferentes estructuras en el volumen. Este proceso es llevado a cabo por la segmentación o por funciones de transferencia (29). Por la importancia de estas últimas serán tratadas a continuación en el siguiente epígrafe.

1.10 Funciones de transferencia

Las funciones de transferencia asignan propiedades ópticas tales como color y opacidad a los datos extraídos o calculados de un punto dado del volumen. La muestra obtenida del volumen debe ser clasificada por la función de transferencia para calcular su contribución a la imagen final. Esta muestra es evaluada en la función de transferencia, de donde se obtendrá el color que realmente será mostrado en la visualización. Para esta operación es típicamente usado el modelo de color RGB y en adición el complemento de la transparencia, la opacidad; especificando que tan sólido debe aparecer el respectivo color (29).

La visualización del volumen se realiza a partir de la carga de ficheros de imágenes médicas, las imágenes del estudio médico preoperatorio del paciente que se utilizan en el presente trabajo se encuentran en formato DICOM, para una mejor compresión del mismo sus características serán expuestas a continuación.

1.11 Características de los ficheros de formato DICOM

El estándar describe el formato de archivos y la especificación de los datos primordiales de un paciente en la imagen así como el encabezado requerido, describiendo un lenguaje común a distintos sistemas médicos. De esta forma las imágenes vienen acompañadas de mediciones, cálculos e información descriptiva relevante para diagnósticos. Utiliza archivos con extensión *.dcm (30).

1.11.1 Formato de un archivo DICOM

Cada archivo DICOM contiene una cabecera (*header* en inglés) que almacena la información sobre el nombre del paciente, el tipo de exploración, la dimensión de la imagen, así como todos los datos de la imagen que pueden contener la información en tres dimensiones. Después de la cabecera inmediatamente se encuentra un conjunto de datos (*dataset* en inglés) que representa una instancia de la información del estudio realizado a un paciente (**figura 5**). Este conjunto de datos contiene la imagen o las imágenes especificadas (30).

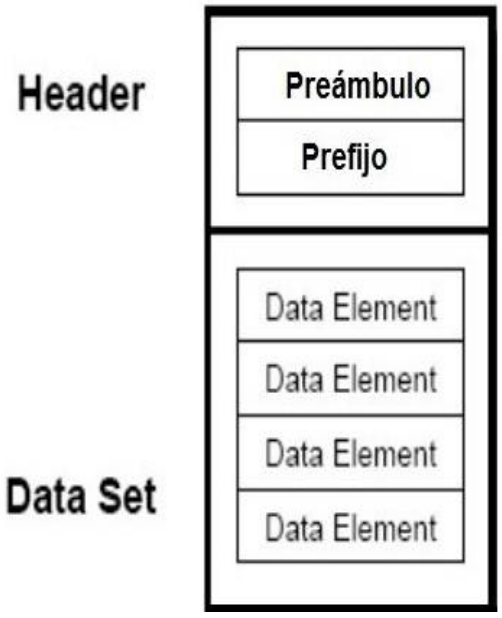


Figura 5. Estructura de un archivo DICOM (*.dcm) (30).

1.12 Metodología para el desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software (31). El objetivo de una metodología de desarrollo de software es garantizar la eficacia y la eficiencia en el proceso de generación de software.

1.12.1 SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto (32).

Los principales beneficios que proporciona Scrum son (32):

- Entrega mensual (o quincenal) de resultados (los requisitos más prioritarios en ese momento, ya completados) lo cual proporciona las siguientes ventajas:

  - ✓ Gestión regular de las expectativas del cliente y basada en resultados tangibles.
  - ✓ Resultados anticipados.
  - ✓ Flexibilidad y adaptación respecto a las necesidades del cliente, cambios en el mercado, etc.

- ✓ Gestión sistemática del Retorno de Inversión.
- ✓ Mitigación sistemática de los riesgos del proyecto.

- Productividad y calidad.
- Alineamiento entre el cliente y el equipo de desarrollo.
- Equipo motivado.

1.12.2 Extreme Programming (XP)

XP es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck en 1999. Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, XP se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los valores originales de la programación extrema son: simplicidad, comunicación, retroalimentación (*feedback*) y coraje. Un quinto valor, respeto, fue añadido en la segunda edición de *Extreme Programming Explained* (33).

Las características fundamentales de XP son (33):

- **Desarrollo iterativo e incremental**: pequeñas mejoras, unas tras otras.
- **Pruebas unitarias continuas**, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- **Programación en parejas**: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. La mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- Frecuente **integración del equipo de programación con el cliente** o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- **Corrección de todos los errores** antes de añadir nueva funcionalidad. Hacer entregas frecuentes.

- **Refactorización del código**, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.

- **Propiedad del código compartida**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- **Simplicidad en el código**: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

1.12.3 Proceso Unificado de Desarrollo (RUP)

El proceso unificado conocido como RUP, es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto.

El proceso de desarrollo constituye un marco metodológico que define en términos de metas estratégicas, objetivos, actividades y artefactos (documentación) requerido en cada fase de desarrollo. Esto permite enfocar esfuerzo de los recursos humanos en términos de habilidades, competencias y capacidades a asumir roles específicos con responsabilidades bien definidas.

RUP se divide en cuatro fases (34):

- Inicio (Define el alcance del proyecto).
- Elaboración (definición, análisis, diseño).
- Construcción (implementación).
- Transición (fin del proyecto y puesta en producción).

Principales características (34)**:**

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

1.13 Selección de la metodología de desarrollo de software

XP está concebida para proyectos con poco tiempo de desarrollo, equipos pequeños y con pocos roles. Propone que el diseño debe ser sencillo, que funcione con todas las pruebas y con el menor número de clases y métodos posible (34). Además, XP ofrece una solución factible para proyectos con requisitos muy cambiantes, propone la aplicación disciplinada de las diferentes prácticas y el uso de tecnologías con un ciclo rápido de realimentación y que soporten fácilmente el cambio. Por lo antes planteado se selecciona XP como metodología de desarrollo de software.

1.14 Consideraciones parciales

En este capítulo se analizaron los conceptos fundamentales sobre el tema en cuestión. Se sentaron las bases necesarias para la comprensión del objeto de estudio y su campo de acción. Se evidenciaron los componentes de un sistema de navegación guiada por imágenes así como las ventajas de este tipo de aplicaciones. Se analizaron las diferentes bibliotecas de visualización de imágenes seleccionándose IGSTK por ser la más conveniente para el desarrollo del componente de visualización, a la vez se analizaron los diferentes algoritmos de visualización de volumen seleccionándose *Raycasting*. Se seleccionó C++ como lenguaje de programación y Qt Creator como IDE de desarrollo. Se evidenciaron las principales modalidades de imágenes médicas. Se compararon diferentes metodologías de desarrollo de software seleccionándose la metodología ágil XP. Todo esto asentó las bases para definir una solución al problema planteado.

Capítulo #2. Solución propuesta

2.1 Introducción

En este capítulo se realiza una descripción general de la solución propuesta al problema de la investigación. Se describen los procesos fundamentales que intervienen en la construcción del componente de visualización y la integración de estos. Además, se especifican las condiciones y pasos a tener en cuenta para el correcto funcionamiento de la solución.

2.2 Soluciones técnicas

La presente investigación pretende obtener como resultado el componente de visualización de imágenes médicas para sistemas de neuronavegación guiada por imágenes, este componente debe visualizar la estructura anatómica 3D a partir de las imágenes médicas del paciente. A su vez debe permitir la navegación quirúrgica mediante la superposición de una representación visual del instrumento quirúrgico sobre las imágenes médicas, de esta manera es posible brindar la retroalimentación visual en tiempo real al médico durante una neurocirugía guiada por imágenes. Para el desarrollo de la propuesta de solución se hizo una estricta selección de las herramientas y lenguajes a utilizar, basada en la investigación realizada. A continuación se menciona cada una de las herramientas seleccionadas y su correspondiente justificación.

2.3 Descripción de la solución

El funcionamiento del componente de visualización para la herramienta de planificación quirúrgica del neuronavegador comienza con el proceso de cargar las imágenes del estudio médico del paciente, estas imágenes deben encontrarse en formato DICOM (**figura 6**).

Con la carga de las imágenes médicas del paciente el componente de visualización muestra al cirujano tres vistas 2D distintas de dichas imágenes: axial, sagital y coronal.

Además de las vistas ya descritas se visualiza la reconstrucción en 3D del volumen a partir de dichas imágenes médicas (**figura 7**).

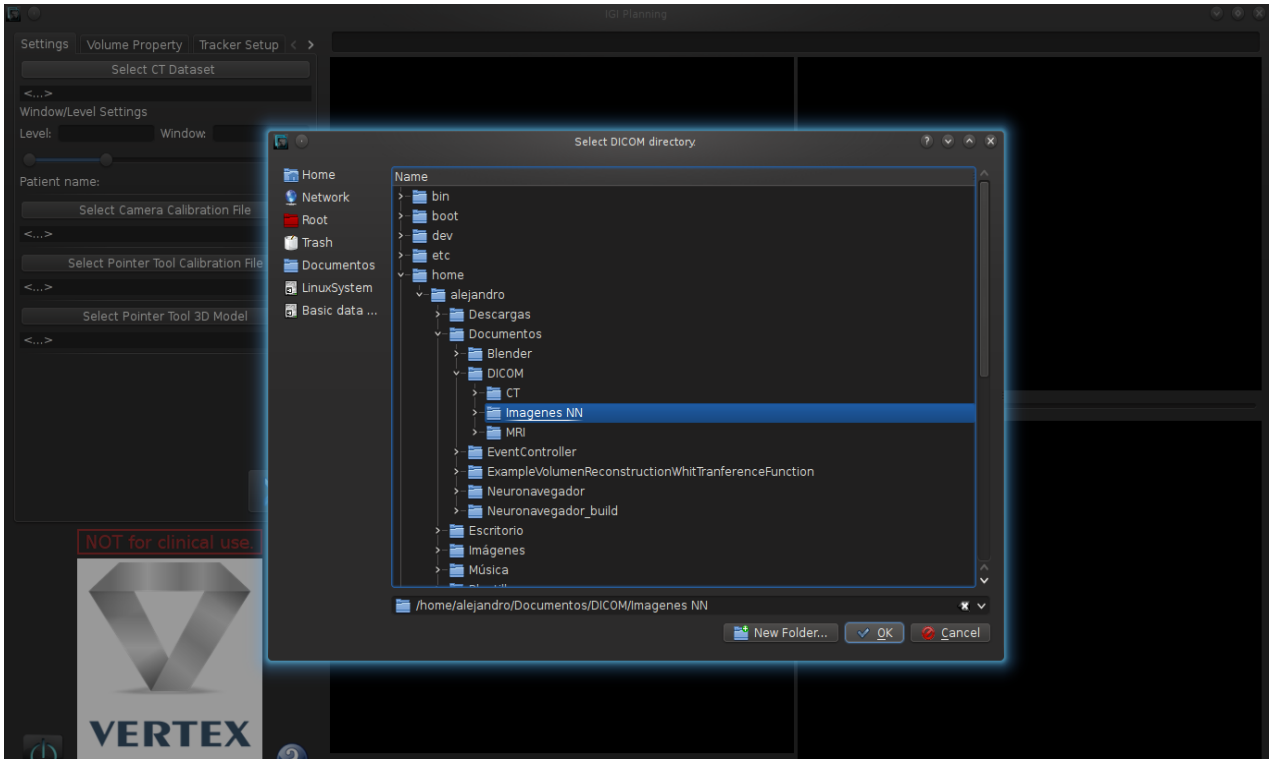


Figura 6. Imagen de la aplicación en el proceso de carga de las imágenes médicas.

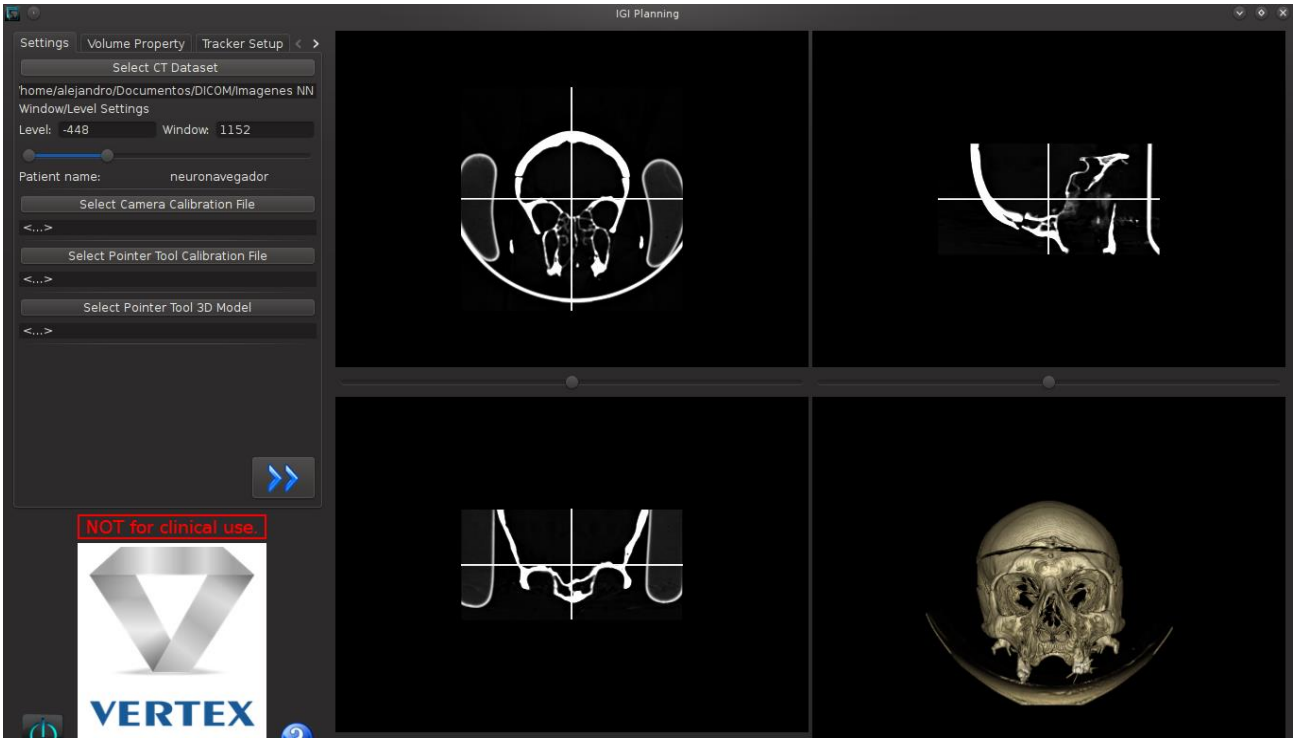


Figura 7. Vista del componente de visualización con las imágenes 2D y el volumen 3D.

Luego de la carga de las imágenes médicas y la correcta visualización tanto de las imágenes 2D como del volumen 3D, el componente de planificación permite cargar un objeto 3D siempre que este objeto posea la extensión *.obj, lo cual implica que puede ser creado en cualquier herramienta de modelado tales como: Blender, 3Ds Max, entre otros.

El objeto 3D cargado representa la herramienta del cirujano y posee una correcta relación entre la posición en el espacio del instrumento quirúrgico en el salón de operaciones con las imágenes del estudio médico del paciente y el volumen creado a partir de las mismas. Este objeto virtual es insertado en la misma escena del volumen para brindar mayor información al cirujano durante la intervención quirúrgica (**figura 8**).

Además el componente cuenta con un editor de funciones de transferencia para el trabajo con los colores, nitidez y la opacidad del volumen 3D (**figura 8**).

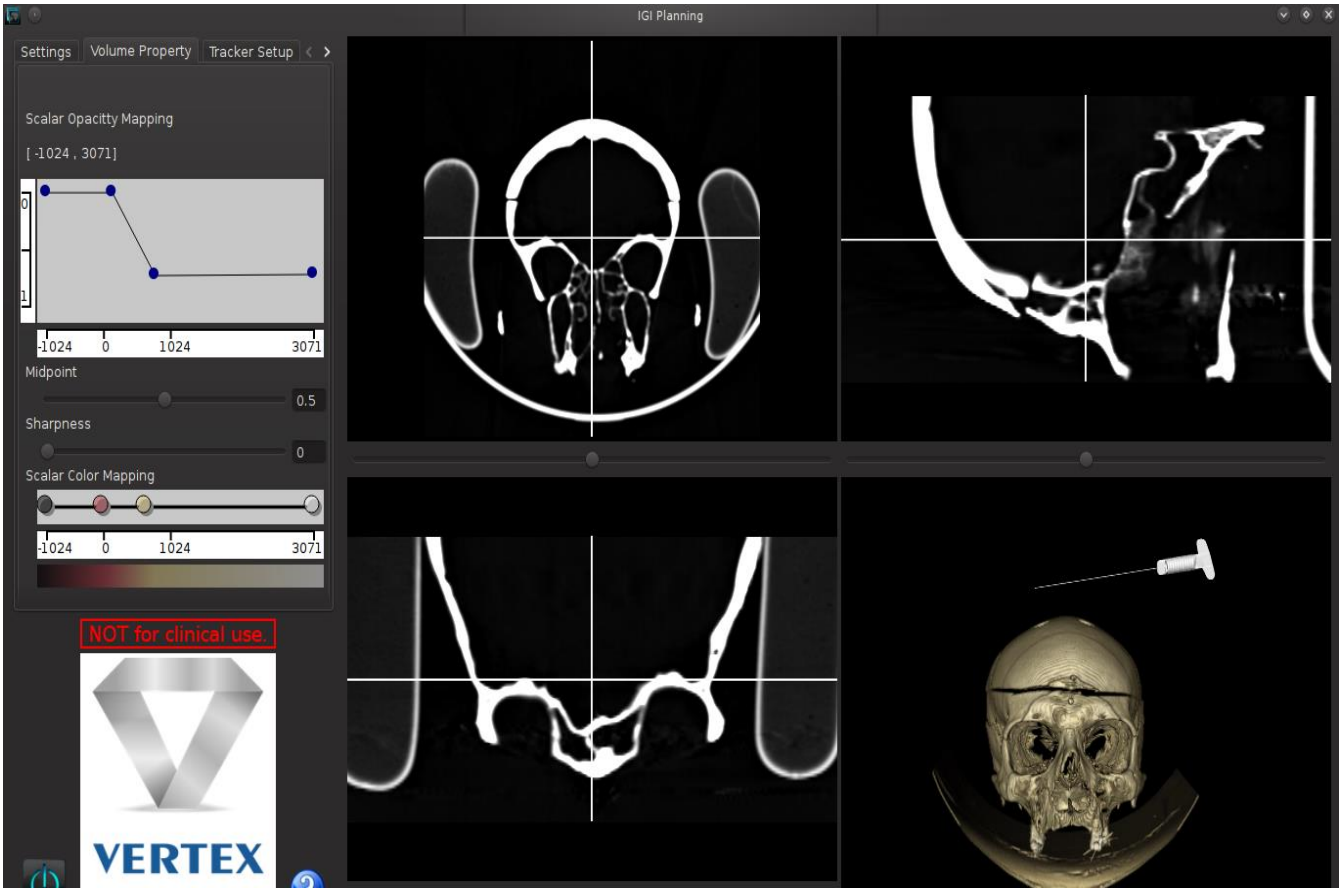


Figura 8. Imagen de la aplicación con el objeto virtual y el editor de funciones de transferencia.

2.4 Fase de Exploración

La metodología XP, comienza en su ciclo de vida con la fase de exploración, proponiendo definir durante esta etapa el alcance general del proyecto; además, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Para esta fase se sugiere una extensión de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.4.1 Historias de Usuario

La metodología XP utiliza la técnica de las Historias de Usuario (HU) para sustituir a los documentos de especificación funcional y a los casos de uso. El tratamiento de las HU es dinámico y flexible, permite que en cualquier momento se puedan romper, reemplazar por otras más específicas o generales, añadirse nuevas o ser modificadas. El tiempo de desarrollo ideal para una HU varía entre 1 y 3 semanas (33). Según Kent Beck cada HU recoge los siguientes aspectos:

Número: Número asignado a la HU.

Nombre de HU: Atributo que contiene el nombre de la HU.

Fecha: Fecha en la cual fue redactada la HU.

Usuario: El usuario del sistema que utiliza o protagoniza la HU.

Prioridad en el negocio: Contiene el nivel de prioridad de la HU en el negocio. Es Alta en caso de que la HU sea indispensable en el negocio, Media en caso de que su realización o no afecte el negocio y Baja cuando no se considera una prioridad.

Riesgo de desarrollo: Contiene el nivel de riesgo en caso de no realizarse la HU. Es Alta, si el riesgo de no realizar la HU incide en el funcionamiento de la plataforma, Media si el riesgo de no realizarla es medianamente importante, y Baja en caso de que no se considere un riesgo tardar en la realización de la HU y no incida en el funcionamiento de la plataforma.

Puntos estimados: Este atributo es una estimación hecha por el equipo de desarrollo sobre el tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo, y un día equivale a 0.2 puntos.

Iteración asignada: Especifica la iteración a la que pertenece la HU correspondiente.

Descripción: Posee una breve descripción de lo que realizará la HU.

Observaciones: Aquellos detalles relevantes que serán resueltos tras la conversación del equipo desarrollador con el cliente.

Historia de usuario	
Número: 1	**Nombre de la HU:** Cargar imágenes médicas
Fecha: 18/1/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Alejandro Ravelo Julian	
Descripción: El usuario selecciona la carpeta donde se encuentran las imágenes del estudio médico del paciente. El sistema debe ser capaz de cargar dichas imágenes para su posterior visualización.	
Observaciones:	

Tabla 4. Historia de usuario # 1.

Historia de usuario	
Número: 2	**Nombre de la HU:** Visualizar imágenes médicas
Fecha: 18/1/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Alejandro Ravelo Julian	
Descripción: Luego de la carga de las imágenes médicas el sistema debe ser capaz de visualizar las imágenes médicas en tres vistas diferentes: axial, sagital y coronal.	
Observaciones:	

Tabla 5. Historia de usuario # 2.

Historia de usuario

Número: 3	**Nombre de la HU:** Reconstruir volumen 3D
Fecha: 18/1/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Alejandro Ravelo Julian	
Descripción: El sistema debe ser capaz de reconstruir el volumen 3D a partir de las imágenes del estudio médico del paciente.	
Observaciones:	

Tabla 6. Historia de usuario # 3.

Historia de usuario	
Número: 4	**Nombre de la HU:** Generar función de transferencia
Fecha: 18/1/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Alejandro Ravelo Julian	
Descripción: El sistema debe ser capaz de variar la opacidad y el color del volumen.	
Observaciones:	

Tabla 7. Historia de usuario # 4.

Historia de usuario	
Número: 5	**Nombre de la HU:** Insertar Objeto quirúrgico 3D
Fecha: 18/1/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Alejandro Ravelo Julian	
Descripción: El sistema debe ser capaz de insertar un objeto 3D en la escena.	
Observaciones:	

Tabla 8. Historia de usuario # 5.

2.5 Fase de Planificación

La planificación es una fase donde el cliente establece la prioridad de cada HU y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Las estimaciones de esfuerzo asociadas a la implementación de las historias la establecen los programadores utilizando como medida el punto, lo que equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. La planificación se puede realizar basándose en el tiempo o el alcance.

2.5.1 Plan de Estimación

Teniendo en cuenta la prioridad que tiene una determinada HU en el desarrollo del componente se decide en que iteración será implementada. Las HU que cuentan con mayor importancia por ser funcionalidades indispensables para a la aplicación deben ser implementadas en las primeras iteraciones del ciclo de desarrollo.

A continuación se muestra mediante una tabla la planificación de las diferentes HU para cada iteración teniendo en cuenta su prioridad.

No	Historia de Usuario	Prioridad	Esfuerzo Estimado
1	Cargar imágenes médicas	Alta	1
2	Visualizar imágenes médicas	Alta	2
3	Reconstruir volumen 3D	Alta	3
4	Generar función de transferencia	Alta	2
5	Insertar objeto quirúrgico 3D	Alta	1

Tabla 9. Estimación de esfuerzos.

2.5.2 Plan de Iteraciones

El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo el cliente es quien decide qué HU se implementarán en cada iteración por lo que no siempre es posible establecer una arquitectura para utilizar durante el resto del proyecto.

Una vez definidas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se decide realizar el sistema en 3 iteraciones, las cuales se describen de manera más detallada a continuación:

2.5.2.1 Iteración 1

Se implementan las HU 1 y 2, estas HU son la base del desarrollo del componente de visualización. Al finalizar la implementación de dichas HU el sistema podrá realizar la carga de las imágenes del estudio médico del paciente y visualizar las imágenes en las diferentes vistas descritas anteriormente. De esta iteración pueden derivarse nuevos requerimientos funcionales. Por último se realizan las pruebas pertinentes a las funcionalidades implementadas.

2.5.2.2 Iteración 2

Se implementa la HU 3 debido a su importancia y complejidad en esta iteración solo se implementará dicha HU. Al finalizar la implementación de dicha HU el sistema deberá ser capaz de reconstruir el volumen 3D a partir de las imágenes médicas cargadas. Luego se pasará a solucionar los errores y no conformidades que fueron detectadas en la iteración anterior. Finalmente, las nuevas funcionalidades implementadas se ponen a prueba.

2.5.2.3 Iteración 3

Se implementan las HU 4 y 5. Al finalizar la implementación de dichas HU el sistema deberá ser capaz de variar la opacidad, el color del volumen reconstruido e insertar un objeto 3D en la escena, dicho objeto representa el instrumento quirúrgico del cirujano. Luego de probar las funcionalidades implementadas en esta iteración se constará con una versión estable del producto con todas las funcionalidades implementadas y probadas.

2.5.3 Plan de duración de las iteraciones

Como parte del ciclo de vida de un proyecto guiado por la metodología de desarrollo XP, se crea el plan de duración de las iteraciones que se llevarán a cabo durante el desarrollo del proyecto. Este plan tiene como objetivo fundamental mostrar la duración de cada una de las iteraciones en las que está dividida la fase de desarrollo del proyecto, así como el orden en que serán implementadas las HU en cada iteración según la prioridad asignada por el cliente.

Iteración	Historias de Usuario	Duración Total de la Iteración
Iteración 1	Cargar imágenes médicas	3
	Visualizar de imágenes médicas	
Iteración 2	Reconstruir volumen 3D	3
Iteración 3	Generar función de transferencia	3
	Insertar Objeto quirúrgico 3D	

Tabla 10. Plan de duración de las iteraciones.

2.5.4 Plan de Entrega

Como resultado de la fase de planificación se genera el plan de entrega que plantea una fecha para la entrega de cada iteración.

Historia de Usuario	Final Iteración 1 (20 de febrero)	Final Iteración 2 (13 de marzo)	Final Iteración 3 (3 de abril)
Cargar imágenes médicas	V1.0		
Visualizar de imágenes médicas	V1.0		
Reconstruir volumen 3D		V1.1	
Generar función de transferencia			V1.2
Insertar Objeto quirúrgico 3D			V1.2

Tabla 11. Plan de entrega.

2.6 Fase de Arquitectura y Diseño

La arquitectura y diseño del software son procesos fundamentales estos indican la estructura, funcionamiento e interacción entre las partes del software. A continuación se presentan las tarjetas

CRC, el patrón arquitectónico, los patrones de diseño y el modelo de datos que se emplearon y ayudaron en el desarrollo del componente de visualización del neuronavegador.

2.6.1 Tarjetas CRC

La utilización de tarjetas CRC (Clase-Responsabilidad-Colaboración) es una técnica de diseño orientado a objetos propuesta por Kent y Ward Cunningham. El objetivo de la misma es hacer, mediante tarjetas, un inventario de las clases que se van a necesitar para implementar el sistema y la forma en que van a interactuar, de esta forma se pretende facilitar el análisis y discusión de las mismas por parte de varios actores del equipo de proyecto con el objeto de que el diseño sea lo más simple posible verificando las especificaciones del sistema (35).

Un esquema típico de tarjeta CRC puede ser aquel en el que se indiquen los siguientes datos:

- Nombre de la clase.
- Las responsabilidades de la clase.
- Las clases con las que va a colaborar para poder realizar las responsabilidades indicadas.

Tarjeta CRC	
Nombre de la clase: NeuronavegadorGUI	
Responsabilidades	**Colaborador**
• Cargar las imágenes médicas. • Visualizar las imágenes médicas. • Construir volumen 3D.	• vtkDICOMImageReader • vtkRenderer • vtkSmartVolumeMapper • vtkColorTransferFunction • vtkVolumeProperty

Tabla 12. Tarjeta CRC de la clase NeuronavegadorGUI.

Tarjeta CRC

Nombre de la clase: VolumeRender	
Responsabilidades	**Colaborador**
• Crea el volumen para su posterior visualización. • Actualiza las propiedades del volumen.	• vtkRenderer • vtkVolume • vtkRendererCollection

Tabla 13. Tarjeta CRC de la clase VolumeRender.

Tarjeta CRC	
Nombre de la clase: ObjectRender	
Responsabilidades	**Colaborador**
• Crea el actor que representa el instrumento quirúrgico del cirujano.	• vtkActor • vtkOBJReader • vtkSmartPointer

Tabla 14. Tarjeta CRC de la clase ObjectRender.

Tarjeta CRC	
Nombre de la clase: Gradiente	
Responsabilidades	**Colaborador**
• Se encarga del trabajo con el gradiente de colores del volumen.	• QGraphicsScene • QPainter

Tabla 15. Tarjeta CRC de la clase Gradiente.

Tarjeta CRC
Nombre de la clase: ColorFunction

Responsabilidades	Colaborador
• Crea la función de transferencia para los colores del volumen. • Envía los cambios a la clase controladora para una nueva visualización.	• QGraphicsScene • QPainter • QGraphicsItem • QBrush

Tabla 16. Tarjeta CRC de la clase ColorFunction.

Tarjeta CRC	
Nombre de la clase: OpacityFunction	
Responsabilidades	**Colaborador**
• Crea la función de transferencia para la opacidad del volumen. • Envía los cambios a la clase controladora para una nueva visualización.	• OpacityScene • OpacityPoint • QGraphicsItem

Tabla 17. Tarjeta CRC de la clase OpacityFunction.

Tarjeta CRC	
Nombre de la clase: OpacityPoint	
Responsabilidades	**Colaborador**
• Representa las propiedades comunes para la representación de un volumen.	• OpacityScene • PointLine • QGraphicsItem

Tabla 18. Tarjeta CRC de la clase OpacityPoint.

2.6.2 Patrón Arquitectónico por Capas

El Patrón de arquitectura por capas es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas de software complicados. Al pensar en un sistema por capas, se puede observar que los principales subsistemas de software están uno encima de otro. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior. Además, normalmente cada capa oculta las capas inferiores de las siguientes superiores a esta (36).

Los beneficios de trabajar un sistema en capas son:

- Se puede entender una capa como un todo, sin considerar las otras.
- Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
- Se minimizan dependencias entre capas.
- Las capas posibilitan la estandarización de servicios.
- Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel.

IGSTK en su implementación propone una arquitectura por capas como se muestra en la siguiente **figura 9** (8). La solución propuesta se adapta a la arquitectura que posee IGSTK que es la base de dicha solución.

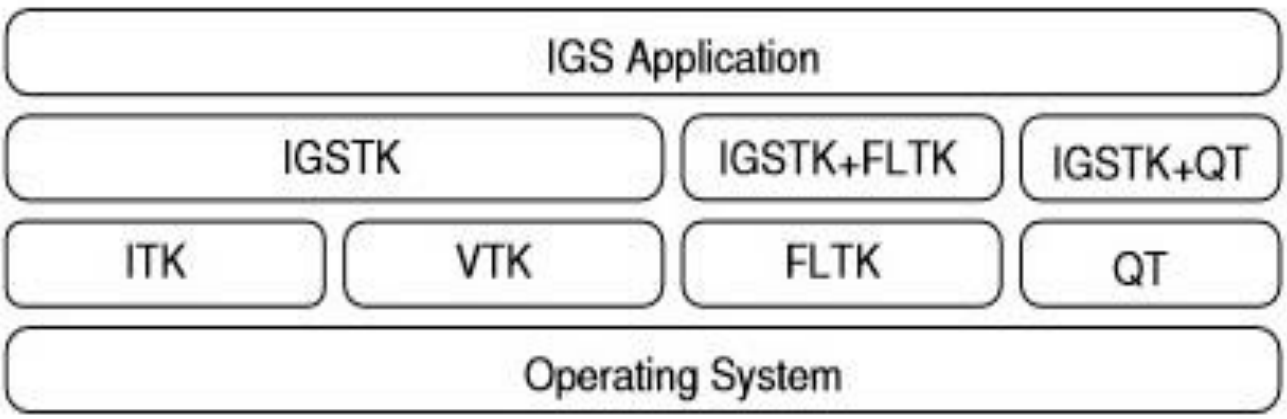


Figura 9. Arquitectura de la IGSTK (8).

2.7 Elementos del diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Para el diseño del componente de software para la visualización de estructuras anatómicas 3D se tuvieron en cuenta los siguientes patrones **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns (**GRASP**): Experto, Bajo acoplamiento, Alta cohesión, Controlador y Creador.

Se hizo uso del patrón **Experto** en la clase VolumeRender. Esta clase es la encargada de todo lo referente al volumen 3D. Contiene la información necesaria para el trabajo con el mismo.

Se evidencia el uso del patrón **Bajo Acoplamiento** y **Alta Cohesión**, debido a que las clases fueron diseñadas de tal forma que existiera la menor relación entre ellas posible, a la vez logrando que la información almacenada en cada clase fuera coherente. Con este diseño se logra que de existir una modificación tenga la mínima repercusión posible en el resto de las clases, se garantiza la reutilización de código y disminuye la dependencia entre clases.

Se utiliza el patrón **Controlador** en la clase NeuronavegadorGUI la cual es la encargada de la separación de la lógica del negocio de la capa de aplicación, aumentando así la reutilización de código. Esta clase es la encargada de controlar todo el proceso de negocio asignando las responsabilidades a las clases correspondientes.

Se utiliza el patrón **Creador** en la clase NeuronavegadorGUI la cual es la encargada de crear instancias de otras clases debido a que posee la información necesaria para este proceso, a la vez es la clase que almacena todas las instancias y controla todo el proceso de negocio.

Capítulo #3. Implementación y prueba del sistema

3.1 Introducción

En este capítulo se analiza la propuesta de solución desde el punto de vista de la calidad, para evaluar cómo se le da cumplimiento al problema de la presente investigación y verificar si fueron implementadas de forma correcta cada una de las HU definidas por el cliente. Se procede a diseñar un conjunto de pruebas correspondientes a la metodología XP, las cuales se dividen en tres partes: carga y estrés, unitarias y de aceptación. Se evalúa el diseño propuesto en aras de comprobar su correcta elaboración. Por otro lado, se describe de forma detallada el proceso de pruebas y se muestra un resultado final, que explica y muestra al usuario que el sistema está implementado correctamente y que cuenta con las funcionalidades descritas en el capítulo anterior.

3.2 Fase de Implementación

En esta fase se realiza la implementación de las HU correspondientes a cada iteración, se chequea el plan de iteraciones por si es necesario realizar modificaciones y se crean las tareas de programación para implementar exitosamente cada HU. A continuación se detallan las tareas de ingeniería de las HU más relevantes de la herramienta de planificación.

Número	Historia de Usuario	Tareas por Historia de Usuario
1	Cargar imágenes médicas	**1.** Implementar un componente que permita escoger la carpeta que contiene las imágenes del estudio médico del paciente. **2.** Implementar una clase que sea la encargada de procesar la información de la carga de las imágenes del estudio médico del paciente.
2	Visualizar imágenes médicas	**1.** Implementar tres componentes para la visualización de las imágenes médicas 2D. **2.** Implementar una clase que reciba los datos procesados en la carga de las imágenes médicas

		y se encargue de visualizar las imágenes médicas 2D.
3	Reconstruir volumen 3D	**1.** Implementar un componente para la visualización del volumen 3D a partir de las imágenes del estudio médico del paciente. **2.** Implementar una clase que reciba la información procesada en la carga de imágenes y sea la encargada de reconstruir el volumen 3D.
4	Generar función de transferencia	**1.** Implementar un componente para la visualización de la función de transferencia para la opacidad del volumen. **2.** Implementar la función de transferencia de la opacidad del volumen. **3.** Implementar un componente para la visualización de la función de transferencia de los colores del volumen. **4.** Implementar la función de transferencia de los colores del volumen. **5.** Implementar una clase que sea la encargada de procesar los cambios realizados en las funciones de transferencia y enviar la información a la clase encargada de la visualización del volumen.
5	Insertar objeto quirúrgico 3D	**1.** Implementar un componente que permita cargar un objeto 3D. **2.** Implementar una clase encargada de construir el objeto 3D.

		3. Insertar el objeto 3D en la escena junto con el volumen 3D de las imágenes del estudio médico del paciente.

Tabla 19. Tareas por HU.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 1
Nombre de la tarea: Implementar un componente que permita escoger la carpeta que contiene las imágenes del estudio médico del paciente.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 2/2/15	**Fecha Fin:** 2/2/15
Descripción: Se implementa un componente que proporcione la opción de escoger una carpeta donde se encuentren las imágenes del estudio médico del paciente.	

Tabla 20. Tarea de Ingeniería 1 de la HU 1.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 1
Nombre de la tarea: Implementar una clase que sea la encargada de procesar la información de la carga de las imágenes del estudio médico del paciente.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 3/2/15	**Fecha Fin:** 3/2/15
Descripción: Se encarga de procesar la información, si el conjunto de imágenes médicas no es válido lanzar un error, de lo contrario mandar la información a las clases encargadas de visualizar las imágenes 2D y reconstruir el volumen 3D.	

Tabla 21. Tarea de Ingeniería 2 de la HU 1.

Tarea de Ingeniería

Número de tarea: 1	**Número de Historia de Usuario:** 2
Nombre de la tarea: Implementar tres componentes para la visualización de las imágenes médicas 2D.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 4/2/15	**Fecha Fin:** 6/2/15
Descripción: Implementar tres componentes que permitan visualizar las imágenes médicas 2D.	

Tabla 22. Tarea de Ingeniería 1 de la HU 2.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 2
Nombre de la tarea: Implementar una clase que reciba los datos procesados en la carga de las imágenes médicas y se encargue de visualizar las imágenes médicas 2D.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 9/2/15	**Fecha Fin:** 20/2/15
Descripción: Se encarga de visualizar las imágenes médicas 2D en 3 vistas: axial, coronal y sagital.	

Tabla 23. Tarea de Ingeniería 2 de la HU 2.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 3
Nombre de la tarea: Implementar un componente para la visualización del volumen 3D a partir de las imágenes del estudio médico del paciente.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 23/2/15	**Fecha Fin:** 24/2/15
Descripción: Implementar un componente que permita la visualización del volumen 3D a partir de las imágenes del estudio médico del paciente.	

Tabla 24. Tarea de Ingeniería 1 de la HU 3.

Tarea de Ingeniería

Número de tarea: 2	**Número de Historia de Usuario:** 3
Nombre de la tarea: Implementar una clase que reciba la información procesada en la carga de imágenes y sea la encargada de reconstruir el volumen 3D.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 2.6
Fecha Inicio: 25/2/15	**Fecha Fin:** 13/2/15
Descripción: Se encarga de reconstruir el volumen 3D a partir de las imágenes del estudio médico del paciente.	

Tabla 25. Tarea de Ingeniería 2 de la HU 3.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 4
Nombre de la tarea: Implementar un componente para la visualización de la función de transferencia para la opacidad del volumen.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 16/3/15	**Fecha Fin:** 18/3/15
Descripción: Un componente que permita la visualización en forma de una gráfica lineal de la función de transferencia para la opacidad del volumen.	

Tabla 26. Tarea de Ingeniería 1 de la HU 4.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 4
Nombre de la tarea: Implementar la función de transferencia de la opacidad del volumen.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 19/3/15	**Fecha Fin:** 19/3/15
Descripción: Se encarga de recibir los cambios realizados en la función de transferencia de la opacidad del volumen.	

Tabla 27. Tarea de Ingeniería 2 de la HU 4.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 4
Nombre de la tarea: Implementar un componente para la visualización de la función de transferencia de los colores del volumen.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 20/3/15	**Fecha Fin:** 24/3/15
Descripción: Un componente que permita la visualización de la función de transferencia de colores para el volumen.	

Tabla 28. Tarea de Ingeniería 3 de la HU 4.

Tarea de Ingeniería	
Número de tarea: 4	**Número de Historia de Usuario:** 4
Nombre de la tarea: Implementar la función de transferencia de los colores del volumen.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 25/3/15	**Fecha Fin:** 25/3/15
Descripción: Se encarga de recibir los cambios realizados en la función de transferencia de los colores del volumen.	

Tabla 29. Tarea de Ingeniería 4 de la HU 4.

Tarea de Ingeniería	
Número de tarea: 5	**Número de Historia de Usuario:** 4
Nombre de la tarea: Implementar una clase que sea la encargada de procesar los cambios realizados en las funciones de transferencia y enviar la información a la clase encargada de la visualización del volumen.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 26/3/15	**Fecha Fin:** 26/3/15

Descripción: Recibe los cambios realizados en las funciones de transferencia los procesa y envía la información a las clase en cargada de visualizar el volumen para una nueva visualización.

Tabla 30. Tarea de Ingeniería 5 de la HU 4.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 5
Nombre de la tarea: Implementar un componente que permita cargar un objeto 3D.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 27/3/15	**Fecha Fin:** 27/3/15
Descripción: Realiza la carga de un objeto 3D y envía la información a la clase encargada de la construcción del objeto.	

Tabla 31. Tarea de Ingeniería 1 de la HU 5.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 5
Nombre de la tarea: Implementar una clase encargada de construir el objeto 3D.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 30/3/15	**Fecha Fin:** 31/3/15
Descripción: Construye el objeto 3D a partir de la información obtenida en la carga del mismo.	

Tabla 32. Tarea de Ingeniería 2 de la HU 5.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 5
Nombre de la tarea: Insertar el objeto 3D en la escena junto con el volumen 3D de las imágenes del estudio médico del paciente.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.6

Fecha Inicio: 1/4/15	**Fecha Fin:** 3/4/15
Descripción: Envía la información del objeto 3D construido a la clase encargada de la visualización del volumen la cual inserta el objeto 3D en la misma escena que el volumen de las imágenes del estudio médico del paciente.	

Tabla 33. Tarea de Ingeniería 3 de la HU 5.

3.3 Pruebas del sistema.

Una vez concluida la fase de implementación comenzó la realización de las pruebas del sistema, las cuales permiten a los desarrolladores y al cliente de la plataforma concluir si la misma tiene la calidad requerida o si cuenta con errores en su funcionamiento. Existen varios tipos de pruebas de software, a continuación se tratan las pruebas realizadas al componente de planificación del neuronavegador.

Pruebas de caja blanca: Están centradas en el código y le permiten al desarrollador comprobar si las funciones o algoritmos implementados en el software tienen la calidad requerida o si tienen algún error. Para su aplicación se analizan los posibles caminos que puede tener una determinada función o algoritmo y luego es probada para comprobar si el resultado esperado es el obtenido.

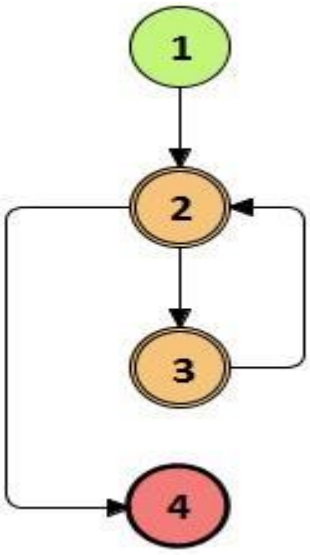
Pruebas de aceptación: Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, buscan una cobertura de la especificación de requisitos y del manual del usuario, las mismas permiten al cliente evaluar el sistema desarrollado.

3.3.1 Pruebas de caja blanca

A continuación se realizaron pruebas de caja blanca a las principales funcionalidades del sistema. Se escogieron dichas funcionalidades porque representan la mayor complejidad en la solución, además son las funcionalidades más utilizadas y las que marcan un hito en el desarrollo de la solución lo que trae consigo que un error en dichas funcionalidades conlleva a que la solución no funcione correctamente o no cumpla con los requisitos planteados por el cliente. Los métodos escogidos para las pruebas de caja blanca son: el método Render de la clase VolumeRender, los métodos LoadObject3D, MovingOpacityPoint y GetValueSliderMidpointChange de la clase NeuronavegadorGUI.

Método: Render	**Grafo Resultante:**

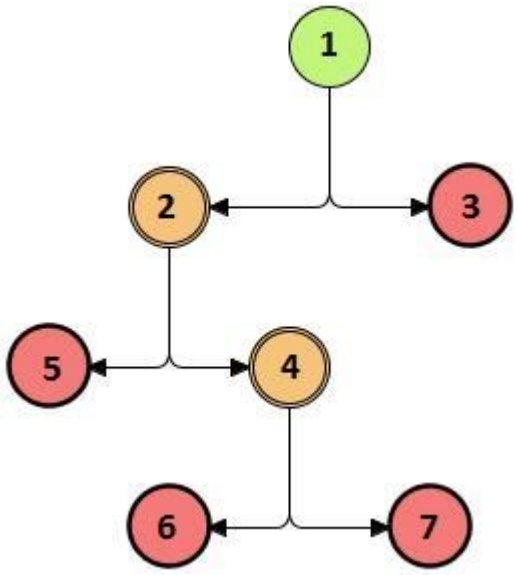
```
vtkSmartPointer<vtkRenderer> VolumeRender::Render(QString pdirectory,
QList<QGraphicsItem*> opacityPointList)

{

(1) this->directory = pdirectory;

(1) QByteArray dir = directory.toLocal8Bit();

(1) char *dirname = dir.data();

(1) bool independentComponents=true;

(1) dicomReader->SetDirectoryName(dirname);

(1) dicomReader->Update();

(1) input=dicomReader->GetOutput();

(1) reader=dicomReader;

(1) int dim[3];

(1) input->GetDimensions(dim);

(1) mapper->SetInputConnection( reader->GetOutputPort() );

(1) property->SetIndependentComponents(independentComponents);

(1) property->SetColor( colorFun );

(1) property->SetScalarOpacity( opacityFun );

(1) property->SetInterpolationTypeToLinear();

(1) volume->SetProperty( property );

(1) volume->SetMapper( mapper );

(2) for (int i=0; i<opacityPointList.size();i++)

{

(3)opacityindex->append(opacityFun-
>AddPoint((double)((((OpacityPoint*)opacityPointList.at(i))->pos().x()-
10)*15)-1024, (double)(((OpacityPoint*)(opacityPointList.at(i)))->pos().y()-
```



12)/100, (double)((OpacityPoint*)opacityPointList.at(i))->GetMidpoint(), (double)((OpacityPoint*)opacityPointList.at(i))->GetSharpness())); (3) int r,g,b; (3) ((OpacityPoint*)opacityPointList.at(i))->GetColor().getRgb(&r,&g,&b); (3) colorindex->append(colorFun->AddRGBPoint((double)((((OpacityPoint*)opacityPointList.at(i))->pos().x()-10)*15)-1024, (double)r/255,(double)g/255,(double)b/255, (double)((OpacityPoint*)opacityPointList.at(i))->GetMidpoint(), (double)((OpacityPoint*)opacityPointList.at(i))->GetSharpness())); } (4) emit SendIndex(opacityindex,colorindex); (4) mapper->SetBlendModeToComposite(); (4) property->ShadeOn(); (4) property->SetAmbient(0.1); (4) property->SetDiffuse(0.9); (4) property->SetSpecular(0.2); (4) property->SetSpecularPower(10.0); (4) property->SetScalarOpacityUnitDistance(0.8919); (4) renderer->AddVolume(volume); (4) renderer->ResetCamera(); (4) emit SendVolume(volume); (4) return renderer; }	
Complejidad Ciclomática:	**Caminos Básicos:**
V (G)= # de regiones = 3 V (G)=A-N+2 = 4-4+2 = 2 V (G)= P+1= 1+1 = 2	{1,2,4} {1,2,3,4}

Tabla 34. Prueba de Caja Blanca al Método Render.

Método: LoadObject3D	**Grafo Resultante:**

void NeuronavegadorGUI::LoadObject3D(){
(1) if (!directory.isEmpty()){
(2) handleMessage("NeuronavegadorGUI::LoadObjectProcessing called...\n", 0);
(2) QString path = m_DatatDir + "/CT";
 	
(2) objectdirectory = QFileDialog::getOpenFileName(this, "Select DICOM directory.",path, tr("Images (*.obj)"));
 	
(2) if(objectdirectory.isNull())
(3) return;
(4) if (objectdirectory.isEmpty()){
 	
(5) handleMessage("No directory was selected.\n",1);
 	
(5) return; }
 	
(6) m_GUI.toolMeshPathEdit->setText(objectdirectory);
 	
(6) actor = objectinstance->RenderObject(objectdirectory);
 	
(6) vtkRenderer *renderer = vtkRenderer::New();
 	
(6) renderer->AddVolume(volume);
 	
(6) renderer->AddActor(actor);
 	
(6) m_GUI.Display_3D->GetRenderWindow()->AddRenderer(renderer);
(6) m_GUI.Display_3D->update(); } else
(7) handleMessage("Can´t load object", 0);}
	

Complejidad Ciclomática:	**Caminos Básicos:**
V (G)= # de regiones = 3 V (G)=A-N+2 = 6-7+2 = 1 V (G)= P+1= 3+1 = 4	{1,3} {1,2,5} {1,2,4,6} {1,2,4,7}

Tabla 35. Prueba de Caja Blanca al método LoadObject3D.

<table>
<tr><th>Método: MovingOpacityPoint</th><th>Grafo Resultante:</th></tr>
<tr><td>void NeuronavegadorGUI::MovingOpacityPoint(OpacityPoint *point)

{

(1) for (int i=0;i<opacityPointList.size();i++)

(2) if (((OpacityPoint*)(opacityPointList.at(i)))->GetIdentificador() == point->GetIdentificador())

(3) opacityPointList.replace(i,point);

(4) for (int i=0; i<opacityPointList.size();i++)

(5) ((OpacityPoint*)opacityPointList.at(i))->SetColor(gradiente->GetColor(((OpacityPoint*)opacityPointList.at(i))->pos().x()));

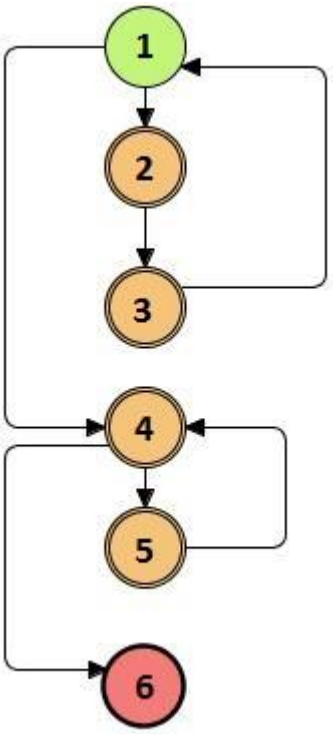
(6) volume = instance->RefreshRender(volume,opacityPointList);

(6) vtkRenderer *renderer = m_GUI.Display_3D->GetRenderWindow()->GetRenderers()->GetFirstRenderer();

(6) renderer->AddVolume(volume);

(6) m_GUI.Display_3D->GetRenderWindow()->AddRenderer(renderer);

(6) m_GUI.Display_3D->update();

}</td><td> 
</td></tr>
<tr><th>Complejidad Ciclomática:</th><th>Caminos Básicos:</th></tr>
<tr><td>V (G)= # de regiones = 3

V (G)=A-N+2 = 7-6+2 = 3

V (G)= P+1= 2+1 = 3</td><td>{1,4,6}

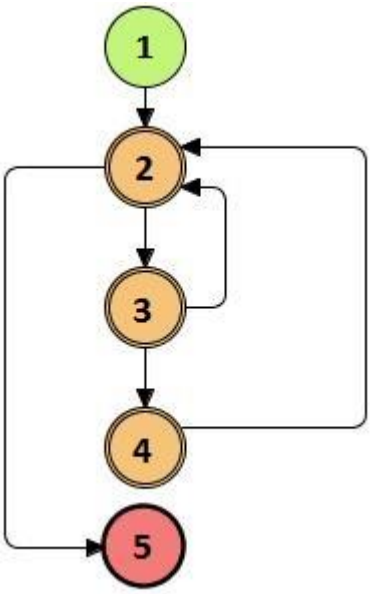
{1,2,3,4,6}

{1,4,5,6}

{1,2,3,4,5,6}</td></tr>
</table>

Tabla 36. Prueba de Caja Blanca al método MovingOpacityPoint.

Método: GetValueSliderMidpointChange	**Grafo Resultante:**

void NeuronavegadorGUI::GetValueSliderMidpointChange(int value){
 	
(1) linemidpoint->setText(QString::number((double)value/100));
 	
(2) for (int i=0; i<opacityPointList.size();i++)
 	
{
 	
(3) if (((OpacityPoint*)opacityPointList.at(i))-
>GetIdentificador()==pointactive)
 	
(4) ((OpacityPoint*)opacityPointList.at(i))-
>SetMidpoint((qreal)value/100);
 	
}
 	
(5)
 	
}
	

Complejidad Ciclomática:	**Caminos Básicos:**
V (G)= # de regiones = 3 V (G)=A-N+2 = 6-5+2 = 3 V (G)= P+1= 1+1 = 2	{1,2,5} {1,2,3,5} {1,2,3,4,5}

Tabla 37. Prueba de Caja Blanca al método GetValueSliderMidpointChange.

Tras haber identificado los casos de prueba de Caja Blanca se le aplicaron las pruebas a cada camino identificado de cada caso de prueba:

Para un total de 13 caminos se le aplicaron dos pruebas a cada uno para un total de 26 pruebas. En una primera iteración de pruebas para un total de 26, 14 fueron satisfactorias y 12 no satisfactorias, lo cual trajo consigo que en una segunda iteración de pruebas para un total de 26, 22 fueron satisfactorias y 4 no satisfactorias, las cuales fueron corregidas inmediatamente y en la tercera iteración de pruebas para un total de 26 pruebas, 26 fueron satisfactorias.

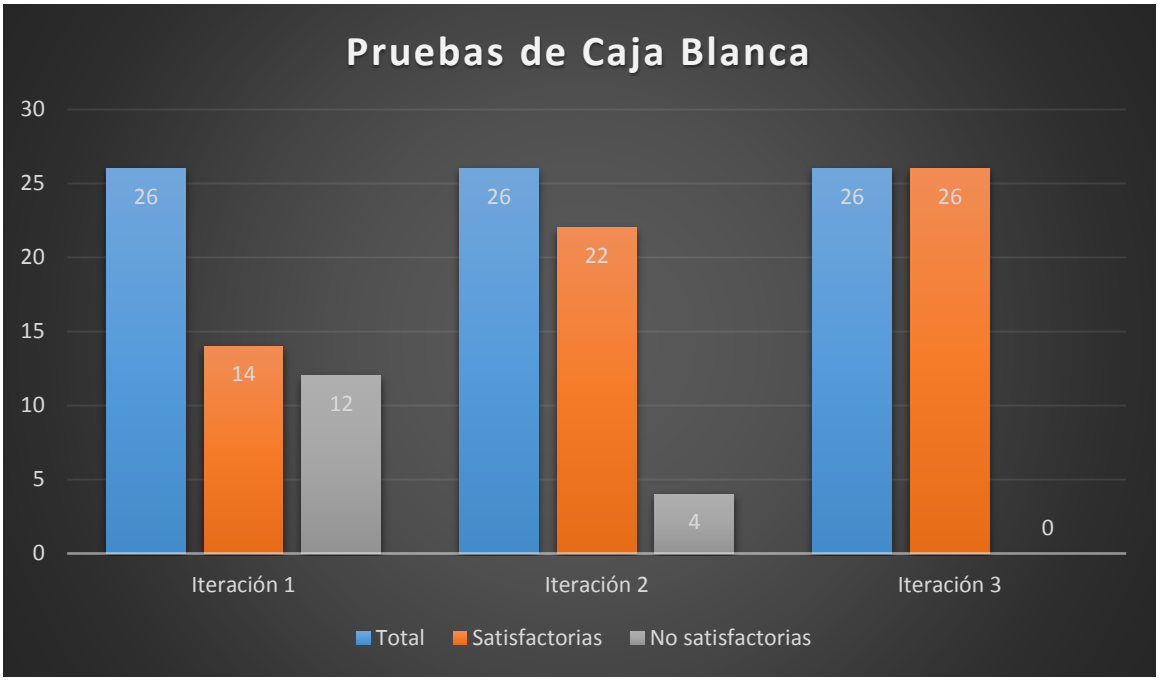


Gráfico 1. Estadísticas de las pruebas de caja blanca.

3.3.2 Pruebas de aceptación

A continuación se muestran las pruebas de aceptación realizadas a las principales funcionalidades del sistema. Para las pruebas de aceptación son escogidas todas las funcionalidades del sistema. Debido a que la prioridad de cada una de estas historias de usuario es alta es necesario comprobar que todas las funcionalidades del sistema sean aceptadas por el cliente y de no ser así pasar a corregirlas inmediatamente.

Caso de Prueba de Aceptación	
Código: H1_P1	**Historia de Usuario:** 1
Nombre: Cargar imágenes médicas	
Descripción: Es el paso principal para la visualización de las imágenes 2D y la reconstrucción del volumen 3D. Carga el conjunto de imágenes pertenecientes al estudio médico del paciente.	
Condiciones de ejecución: No presenta ninguna condición para su ejecución.	

Entrada/Pasos de ejecución: **1.** Click en el botón “Select CT Dataset”. **2.** Buscar y escoger la carpeta que contiene el conjunto de imágenes médicas. **3.** Click en el botón “Ok”.
Resultado esperado: Comprueba que las imágenes cargadas sean válidas y de ser así guarda la información relacionada con las imágenes médicas del paciente.
Resultado obtenido: Se comprobó que el conjunto de imágenes fuera válido y se guardó la información relacionada con las mismas.
Evaluación de la prueba: Satisfactoria.

Tabla 38. Prueba de aceptación para la HU “Cargar imágenes médicas”.

Caso de Prueba de Aceptación	
Código: H2_P1	**Historia de Usuario:** 2
Nombre: Visualizar imágenes médicas.	
Descripción: Visualiza las imágenes 2D en tres vistas: axial coronal y sagital. Visualiza el volumen 3D.	
Condiciones de ejecución: Las imágenes médicas deben haber sido cargadas con anterioridad.	
Entrada/Pasos de ejecución: **1.** Recibe la dirección donde se encuentra el conjunto de imágenes del estudio médico del paciente. **2.** Visualiza las imágenes 2D. **3.** Envía la información necesaria para la reconstrucción del volumen 3D a la clase encargada de hacerlo.	
Resultado esperado: Visualiza las imágenes 2D en tres vistas: axial, coronal y sagital. Visualiza el volumen 3D.	

Resultado obtenido: Se visualizaron las imágenes 2D en las diferentes vistas y se visualizó el volumen 3D.
Evaluación de la prueba: Satisfactoria.

Tabla 39. Prueba de aceptación para la HU “Visualizar imágenes médicas”.

Caso de Prueba de Aceptación	
Código: H3_P1	**Historia de Usuario:** 3
Nombre: Reconstruir el volumen 3D.	
Descripción: Reconstruye el volumen 3D a partir de las imágenes del estudio médico del paciente.	
Condiciones de ejecución: Las imágenes médicas deben haber sido cargadas con anterioridad.	
Entrada/Pasos de ejecución: **1.** Recibe la dirección donde se encuentra el conjunto de imágenes del estudio médico del paciente. **2.** Construye el volumen 3D a partir de dichas imágenes médicas. **3.** Envía el volumen a la clase encargada de visualizar el volumen.	
Resultado esperado: La construcción de un volumen.	
Resultado obtenido: Se construyó correctamente un volumen 3D.	
Evaluación de la prueba: Satisfactoria.	

Tabla 40. Prueba de aceptación para la HU “Reconstruir volumen 3D”.

Caso de Prueba de Aceptación	
Código: H4_P1	**Historia de Usuario:** 4
Nombre: Generar función de transferencia.	
Descripción: Genera funciones de transferencia para los colores y la opacidad del volumen 3D.	
Condiciones de ejecución: El volumen debe haber sido creado y visualizado correctamente.	

Entrada/Pasos de ejecución: **1.** Crea una nueva pestaña en la aplicación que posee las herramientas para el trabajo con el volumen. **2.** Crea la función de transferencia para la opacidad y la función de transferencia para los colores del volumen 3D. **3.** Recibe cualquier cambio en ambas funciones de transferencia y envía dichos cambios a la clase encargada de reconstruir el volumen para una nueva visualización.
Resultado esperado: Construir una nueva pestaña en la aplicación con las funciones de transferencia para los colores y la opacidad del volumen 3D.
Resultado obtenido: Se construyó una nueva pestaña en la aplicación con las herramientas para realizar cambios en las funciones de transferencias y envió los cambios a la encargada de reconstruir el volumen 3D con lo que se logró una nueva visualización del volumen con los cambios realizados.
Evaluación de la prueba: Satisfactoria.

Tabla 41. Prueba de aceptación para la HU “Generar función de transferencia”.

Caso de Prueba de Aceptación	
Código: H5_P1	**Historia de Usuario:** 5
Nombre: Insertar Objeto quirúrgico 3D.	
Descripción: Inserta un objeto 3D en la escena del volumen 3D que representa el objeto quirúrgico del cirujano.	
Condiciones de ejecución: El volumen debe haber sido creado y visualizado correctamente.	
Entrada/Pasos de ejecución: **1.** Click en el botón “Select Pointer Tool 3D Model”. **2.** Buscar y escoger el archivo *.obj que contiene la información del objeto 3D a insertar. **3.** Click en el botón “Open”.	

Resultado esperado: Inserción de un objeto 3D en la escena junto con el volumen.
Resultado obtenido: Se insertó correctamente un objeto 3D en la escena junto con el volumen visualizado anteriormente.
Evaluación de la prueba: Satisfactoria.

Tabla 42. Prueba de aceptación para la HU “Insertar Objeto quirúrgico 3D”.

Al realizar las pruebas de aceptación en una primera iteración de un total de 5 pruebas, 5 fueron satisfactorias por lo que no fue necesaria una segunda iteración de pruebas.

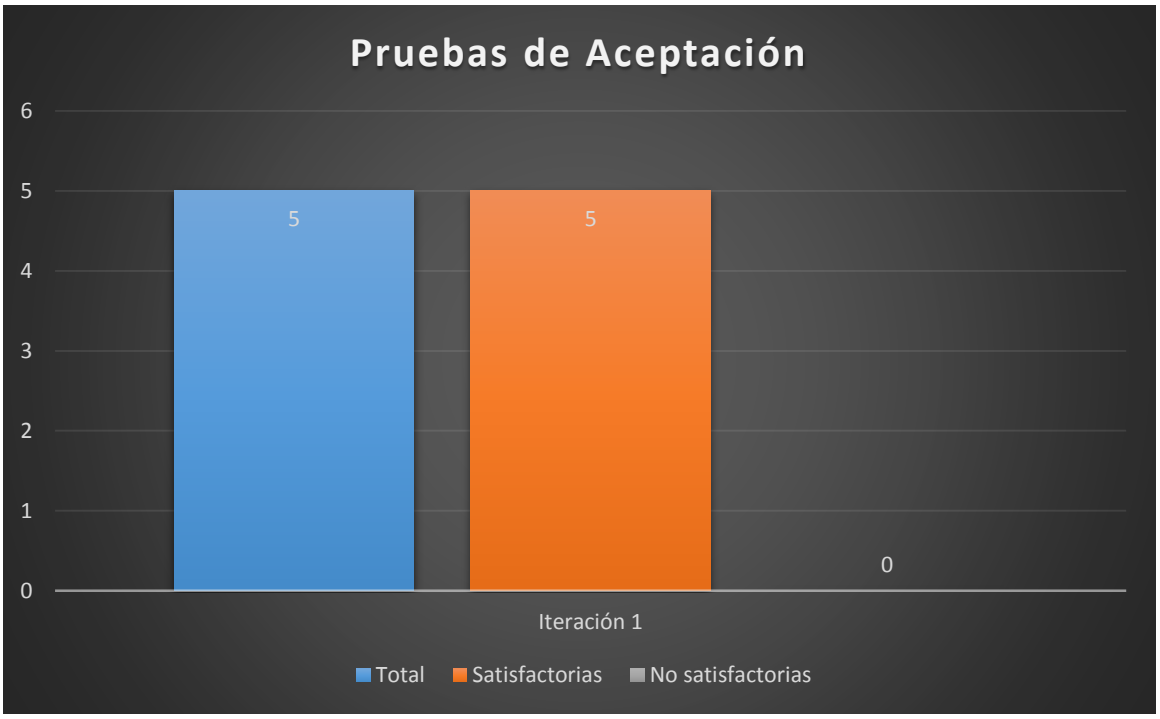


Gráfico 2. Estadísticas de las pruebas de aceptación.

Conclusiones generales

Con la realización de este trabajo se desarrolló un componente de software para la visualización del volumen contenido en imágenes médicas de formato DICOM. De esta forma se da cumplimiento al objetivo propuesto al inicio de la investigación, además se comprobó que:

- La implementación del editor de funciones de transferencia facilitó el trabajo con la visualización del volumen.
- La integración del componente realizado al prototipo de planificador para sistemas de neuronavegación guiada por imágenes logró aumentar la información espacial que recibe el cirujano en el salón de operaciones al lograr combinar la información mostrada por las imágenes 2D con la información que proporciona el volumen 3D.
- Al adicionar a la escena del volumen 3D un objeto virtual que representa la herramienta quirúrgica del cirujano, permitió ofrecer una retroalimentación visual al cirujano de la posición y orientación de dicha herramienta con respecto al estudio de imágenes médicas del paciente.

Recomendaciones

A continuación se enuncian algunas recomendaciones orientadas a trabajos futuros:

- Incorporar el histograma del volumen 3D al editor de funciones de transferencia.
- Incorporar al editor de transferencia una paleta de funciones preestablecidas con valores conocidos en otras aplicaciones y en la literatura consultada.
- Incorporar a la escena junto con el volumen 3D los planos de las diferentes vistas 2D.

Bibliografía

1. **Wolf, Ivo.** *The medical imaging interaction toolkit. Medical image analysis.* 2005.

2. **Pérez, Fidel Martínez.** *Sistema de Registro para Software de Navegación Quirúrgica.* La Habana : s.n., 2012.

3. **Mallot, Franz.** *Robotics and Autonomous System.* 2000.

4. **Gómez, Alvaro y Randall, Gregory.** *Navegación guiada por imágenes en Neurocirugía.*

5. **Watanabe, E, y otros.** *Three-dimensional digitizer (Neuronavigator): new equipment for CT-Guided stereotaxy surgery.* 1987.

6. **Galloway, R L y Maciunas, R J.** *Interactive image-guided neurosurgery.* 1992.

7. **Lustgarden, Leonardo.** Leonardo Lustgarden. *Dr Leonardo Lustgarten - Neurocirugia y Radiocirugia.* [En línea] 8 de Febrero de 2011. [Citado el: 16 de Diciembre de 2014.] http://www.leonardolustgarten.com/2010/02/neuronavegacion-cerebral/.

8. **Cleary, Kevin, y otros.** *IGSTK: The Book.* Maryland : Signature Book Printing, 2009.

9. **Kitware Inc.** Insight Segmentation and Registration Toolkit (ITK). [En línea] Kitware Inc., 2015. [Citado el: 15 de 01 de 2015.] http://itk.org/.

10. —. Visualization Toolkit (VTK). [En línea] Kitware Inc., 2015. [Citado el: 15 de 01 de 2015.] http://www.vtk.org/.

11. **Cleary, Kevin, y otros.** *IGSTK: a software toolkit for image-guided surgery applications.* Washington : s.n., 2004.

12. **Johnson, Hans J., McCormick, Matthew M. y Ibañez, Luis.** *The ITK Software Guide Book 1: Introduction and Development Guidelines Fourth Edition.* s.l. : Kitware Inc., 2015.

13. **Departamento de Teoría de la Señal y Comunicaciones.** *Programas de procesamiento de imágenes médicas en la actualidad.* Sevilla : Universidad de Sevilla, 2015.

14. **Avila, Lisa S.** *The VTK User´s Guide. Updated for version 5.* s.l. : Kitware Inc., 2006.

15. **Caban, Jesus J, Joshi, Alark y Nagy, Paul.** *CABAN, Jesus J.; JOSHI, Alark; NAGY, Paul. Rapid development of medical imaging tools with open-source libraries.* s.l. : Journal of Digital Imaging, 2007.

16. **BHANIRANTKA, P y DEMANGE, Yves.** *OpenGL Volumizer: A toolkit for high quality volume rendering of large data sets.* s.l. : IEEE/ACM SIGGRAPH Symposium on. IEEE, 2002.

17. **Steiman, David A, Piccinelli, Marina y Antiga, Luca.** *PICCINELLI, Marina, et al. A framework for geometric analysis of vascular structures: application to cerebral aneurysms.* 2009.

18. **Kitware Inc.** Volview. [En línea] Kitware Inc., 2015. [Citado el: 01 de 03 de 2015.] http://www.kitware.com/opensource/volview.html.

19. —. ParaView. [En línea] Kitware Inc., 2015. [Citado el: 01 de 03 de 2015.] http://www.paraview.org/.

20. —. 3DSlicer. [En línea] Kitware Inc., 2015. [Citado el: 01 de 03 de 2015.] http://slicer.org/.

21. **Parker, Steven G y Johnson, Christopher R.** *SCIRun: a scientific programming environment for computational steering.* s.l. : Proceedings of the 1995 ACM/IEEE conference on Supercomputing, 1995.

22. **Figueredo Falcón, Yehimy y Rodríguez Pajares, Ramiro.** *FALCÓN, Yehimy Figueredo; PAJARES, Ramiro Rodríguez. Plugin para el particionado de tablas en la herramienta de administración HABD.* La Habana : Serie Científica, 2012, 2012.

23. **Rojas, Luis Guillermo Silva.** *Visualización directa de volumen para endoscopias virtuales .* La Habana : s.n., 2011.

24. **Núñez, Rubén Alcolea.** *Módulo de iluminación para visualización directa de volumen.* La Habana : s.n., 2011.

25. **Treviño, David Anaya.** *Generación de imágenes volumétricas de datos biomédicos en tiempo real.* Zaragoza : s.n., 2009.

26. **Westover, Lee Alan.** *Splatting: a parallel, feed-forward volume rendering algorithm.* North Carolina : University of North Carolina, 1991.

27. **Drebin, Robert A, Carpenter, Loren y Hanrahan, Pat.** *Volume rendering.* 1988.

28. **Meibner, Michael, y otros.** *A Practical Evaluation of Popular Volume Rendering Algorithms.* Salt Lake : s.n., 2000.

29. **Urra, Leonel.** *FUNCIONES DE TRANSFERENCIA PARA VISUALIZACIÓN DIRECTA DE VOLUMEN.* La Habana : s.n., 2011.

30. **Pereira Barzaga, Osvaldo y Kindelan Nunez, Rolando.** *Reconstrucción Tridimensional de Modelos Anatómicos a partir de Imágenes Médicas Digitales.* La Habana : s.n., 2008.

31. **Commons, Creative.** Portal Web de la Universidad de Murcia. [En línea] 30 de 12 de 2006. [Citado el: 15 de 01 de 2015.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

32. **IBM.** IBM developerWorks. *Rational Team Concert for Scrum Projects > SCRUM como metodología .* [En línea] 22 de 11 de 2010. [Citado el: 12 de 03 de 2015.] https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Rational%20Team%20Concert%20for%20Scrum%20Projects/page/SCRUM%20como%20metodolog%C3%ADa.

33. **Beck, Kent y Andres, Cynthia.** *Extreme Programming Explained. Second Edition. Embrace Change.* Boston : Addison-Wesley, 2012.

34. **Wells, Don.** Extreme Programming: A gentle introduction. [En línea] 08 de 10 de 2013. [Citado el: 12 de 03 de 2015.] http://www.extremeprogramming.org/.

35. **Jummp's Blog.** jummp.wordpress. *Gestión de proyectos y desarrollo de software.* [En línea] wordpress, 10 de 01 de 2012. [Citado el: 19 de 03 de 2015.] https://jummp.wordpress.com/2012/01/10/desarrollo-de-software-tarjetas-crc/.

36. **Valdez, José.** *Introducción a Patrones de Arquitectura por Capas.* [En línea] Microsoft Virtual Academy, 2013. [Citado el: 19 de 03 de 2015.] https://todojosevaldez.wordpress.com/desarrollo-de-software/desarrollo-net/capitulo-iv-modelado-dinamico/introduccion-a-patrones-de-arquitectura-por-capas/.

37. **Preim, Bernhard y Bartz, Dirk.** *Visualization in Medicine. Theory, algorithms and applications.* Burlington : Elsevier Inc., 2007.

38. *Estereotaxia: Historia, generalidades y actualidades.* **Tejera del Valle, José Ramón, Piñeiro Martí, Juan Francisco y Morales Sabina, Osmany.** 1727-879X, Cienfuegos : s.n., 2005, Vol. 3.

39. **Verdecia Álvarez, Yadira y Peña Guerra, Liannet.** *Herramienta de autor para la creación de contenido de realidad aumentada.* La Habana : s.n., 2010.

40. **Project, Qt.** Qt Project. *Q Project.* [En línea] 2014. [Citado el: 15 de 01 de 2015.] http://www.qt.io/qt-framework/.

41. **WIKILIBROS.** Programación con Qt4. *Programación con Qt4.* [En línea] WIKILIBROS, 06 de 08 de 2013. [Citado el: 15 de 01 de 2015.] http://es.wikibooks.org/wiki/Programaci%C3%B3n_con_Qt4.

42. **Santiago Zaragoza, Mtra. María de Lourdes.** Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado (RUP). *Proceso Unificado de Rational.* [En línea] [Citado el: 12 de 03 de 2015.] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

Anexos

Anexo 1: Propuesta de construcción en Cuba de un Neuronavegador.

La Habana, 5 de diciembre, 2011

"Año 53 de la Revolución"

Propuesta de construcción en Cuba de un Neuronavegador para aplicación en cirugía intracraneal y espinal.

La Neuronavegación es un procedimiento que ha sido posible gracias al desarrollo de la computación y la imagenología aplicada a la medicina. Su objetivo es: ubicar o visualizar **"**en tiempo real**",** cualquier instrumento quirúrgico durante una intervención, en imágenes obtenidas por Resonancia Magnética Nuclear (RMN) o Tomografía Axial Computadorizada (TAC) del paciente, donde sería imposible lograrlo con nuestros órganos de los sentidos (visión o tacto).

A modo de comparación: cuando se pilotea un avión a expensas de los órganos de los sentidos puede estimarse la altura a la que se está volando, ver las montañas que tiene delante para esquivarlas, observar la línea del horizonte para ver el ángulo que llevan sus alas, estimar la velocidad con que se mueve. De noche es prácticamente imposible hacerlo. Es necesario navegar con aparatos. Equipos que le dicen la altura, radares que le indican las montañas que tiene delante con la distancia y ángulo a la que se encuentran, otros que le indican la velocidad a la que se mueve, "líneas del horizonte" donde se puede conocer la angulación de las alas del avión con relación al piso. Y todo esto "en tiempo real". Un neuronavegador es un similar. Con este podemos "ver" en cortes axiales, coronales, sagitales, 3D o en alguno otro plano previamente definido de la RMN o TAC; la ubicación de los instrumentos neuroquirúrgicos mucho más allá de lo permisible por los órganos de los sentidos del neurocirujano. Por ejemplo: observar (todavía con la cabeza cerrada) el sitio real donde se encuentra una lesión colocando el instrumental con el neuronavegador para poder abordarla directamente desde un inicio sin equivocar la trayectoria. Es decir, el médico se ayuda en este caso de equipos que le permiten hacer lo que es imposible con la visión o tacto.

Los neuronavegadores comenzaron a construirse con equipos mecánicos alámbricos altamente complejos que permitían censar los ángulos y posiciones de muchas de sus articulaciones y, a distancia, poder ubicar el instrumental con relación a la anatomía quirúrgica. Pero desde hace menos de 20 años se han desarrollado neuronavegadores inalámbricos guiados por rayos infrarrojos. Los sistemas inalámbricos revolucionaron la Neuronavegación por prescindir de los altamente complejos sistemas articulados con múltiples sensores. Los sistemas inalámbricos actuales consisten en:

1. Computadora de gran velocidad y memoria virtual (fácil de obtener en el mercado con costo menor de 2000 USD).
2. Cámaras que emiten y recepcionan rayos infrarrojos para localizar puntos de referencia fijados en la cabeza del paciente (fáciles de obtener las cámaras en el mercado probablemente con costos menores de 2000 USD).
3. Tarjeta interfaz que convierte la señal infrarroja en digital (fácil de obtener en el mercado probablemente con costo inferior a 1500 USD).
4. Software para hacer interactuar las imágenes obtenidas de los estudios de RMN o TAC previos a la cirugía, con información recibida a través de las cámaras infrarrojas desde los puntos de referencia en la cabeza del paciente (software altamente costoso e imposible de obtener en el mercado sin comprar el equipo).

Por otra parte, los neuronavegadores actuales tienen un costo en el mercado que oscila entre los 400 000 y los 600 000 USD. Son fabricados por firmas como Brain Lab, Medtronic, Bbraun; algunas de origen norteamericano. Es decir, estos desorbitantes precios de los equipos están dados por:

1. El Software.
2. Por la factibilidad de un mercado en el mundo neuroquirúrgico (elitista dentro de las especialidades médicas a nivel internacional) que permite su adquisición aunque tenga estos precios. Conociendo las posibilidades técnicas y el desarrollo alcanzado en nuestro país en las Ciencias Informáticas la fabricación de equipos como estos pueden llevarse a cabo, expandirse por los hospitales cubanos y de las colaboraciones médicas en el exterior, así como lograr invadir el mercado internacional, especialmente América Latina, África y Asia, con precios que pudieran acercarse mucho más a su costo real de producción. Todo esto con elevada calidad tecnológica.

Esta propuesta la hicimos en el año 1998, fue aceptada y se comenzaron los primeros intentos en EICISOFT. Semanas más tarde fue desechada la idea por falta de prioridad en dicho centro. En estos momentos volvemos a proponer sea analizada la posibilidad de emprender la fabricación de Neuronavegadores en nuestro país utilizando la posibilidades tecnológicas y organizativas de la Universidad de las Ciencias de la Informática.

Nota: podemos adjuntar video relacionados con la funcionalidad de los neuronavegadores. Fraternalmente,

Dr. Enrique de Jongh Cobo

Presidente de la Sociedad Cubana de Neurología y Neurocirugía Clínica Central “Cira García”.

Anexo 2: Imágenes de la aplicación.

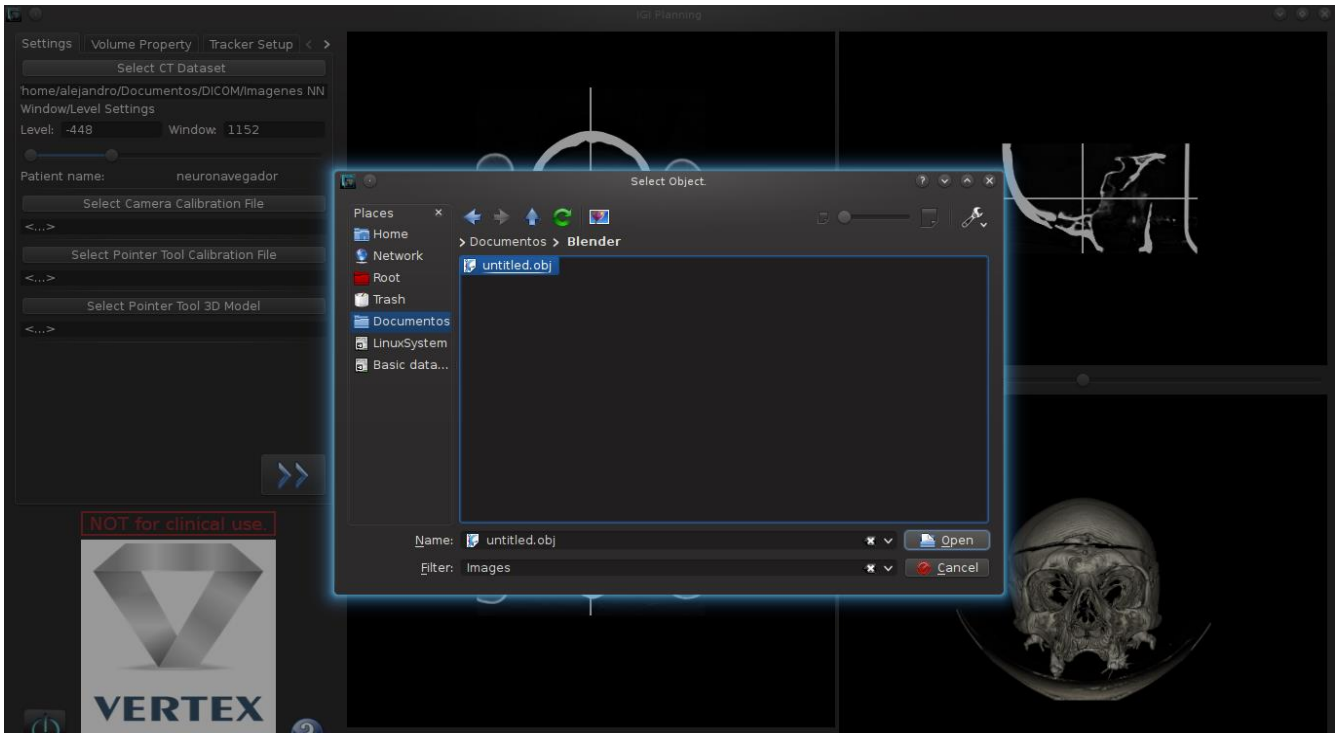


Figura 10. Imagen de la aplicación en el proceso de carga del objeto quirúrgico 3D.

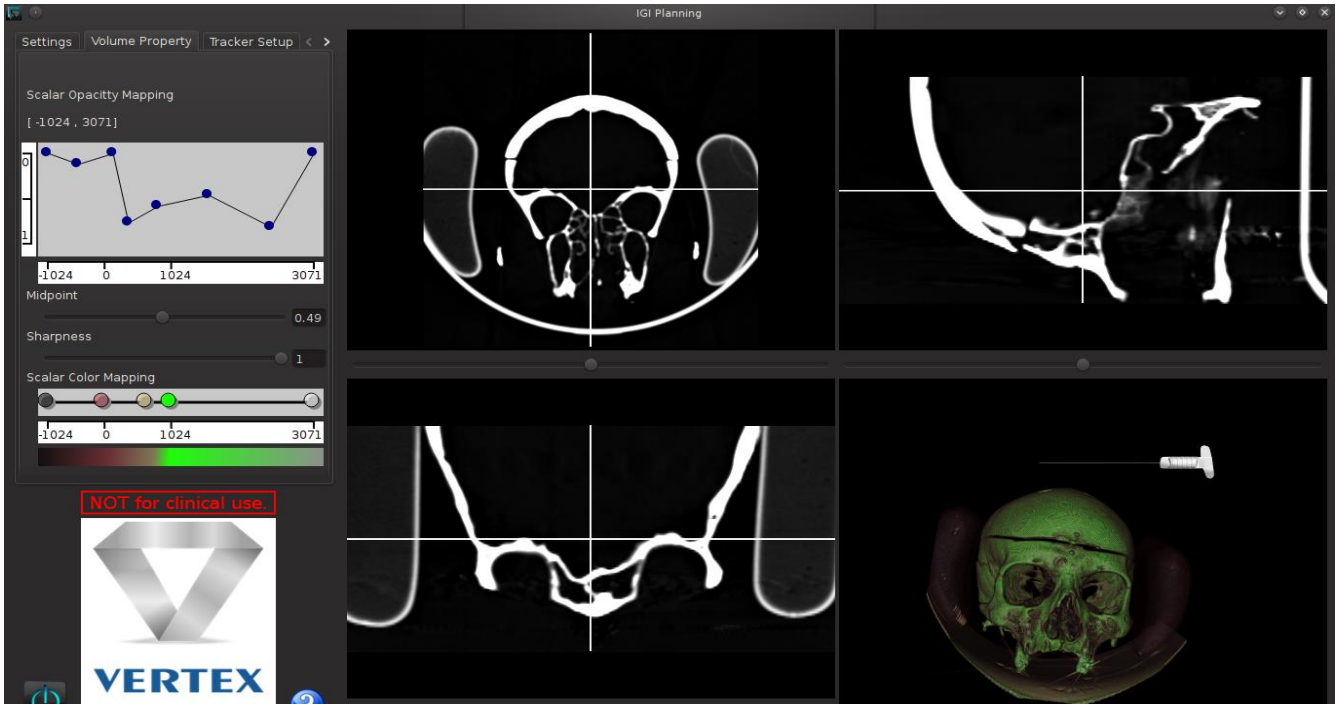


Figura 11. Imagen de la aplicación luego de unos cambios en las funciones de transferencia de la opacidad y los colores del volumen 3D