Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

Navegador quirúrgico para ablación de tumores con radiofrecuencia

T E S I S

que para obtener el grado de:

Maestro en Ingeniería
(Computación)

P R E S E N T A

Claudio Alcérreca Alcocer

Director: Dr. Fernando Arámbula Cosío

Ciudad Universitaria, México D.F., 2007

Contenido

Índice de figuras

Resumen

Desde 1990 se utiliza la terapia de ablación con radiofrecuencia (RF) para tratar tumores malignos en diversas partes del cuerpo. La parte más delicada de este procedimiento es la inserción de una aguja, que contiene un electrodo que se debe colocar en el centro del tumor. El éxito del tratamiento depende de la precisión con que se coloque esta aguja.

En este proyecto se desarrolla un navegador quirúrgico que asista a un médico durante la planeación y ejecución de un tratamiento de tumores en el sistema musculoesquelético con RF. El sistema se basa en imágenes de Resonancia Magnética Nuclear (MRI) para construir un modelo tridimensional (3D) del tumor y la anatomía de interés del paciente. Durante la cirugía, se registra el modelo 3D preoperatorio con la anatomía del paciente utilizando imágenes de ultrasonido (US). El modelo 3D y la aguja de RF se presentan en un monitor de computadora. El sistema utiliza un seguidor óptico NDI Polaris, que se basa en un par estéreo de imágenes para calcular, en tiempo real, la posición y orientación de la aguja con respecto al tumor. Así, el médico puede ver un modelo de lo que hay dentro del cuerpo del paciente y, por lo tanto, realizar la ablación con la mayor eficacia y eficiencia posible.

En este proyecto de tesis se implementaron las herramientas de construcción de modelos 3D y de registro, además de un prototipo del navegador para ablación de tumores, que se basa únicamente en imágenes de US para guiar al cirujano. Este navegador permite agregar funcionalidad, de manera que los demás elementos se agreguen más tarde, junto con herramientas de segmentación, que son necesarias para la construcción de modelos. Este sistema es funcional y por sí mismo presenta ventajas sobre otros navegadores similares, como por *Ultraguide 1000* o Vector Vision, de BrainLAB.

El navegador quirúrgico se implementó utilizando el marco de trabajo CAMPAR, desarrollado por el grupo de Procedimientos Médicos Asistidos por Computadora de la Universidad Tecnológica de Múnich, en Alemania. El mismo

marco de trabajo se usó para la calibración del sistema, que es fundamental para el funcionamiento. El sistema presenta 4 vistas que, en conjunto, ayudan al cirujano a ubicar el tumor y a insertar la aguja de RF en el centro de éste.

Se realizaron diversos experimentos para probar tanto el navegador como las herramientas de construcción y registro de modelos. Se demostró que el navegador es muy confiable y preciso.

1. Introducción

1.1 Cirugía asistida por computadora y robótica quirúrgica

Desde mediados de los años ochenta se desarrollan sistemas de cirugía asistida por computadora (CAS, por sus siglas en inglés) y terapia asistida por computadora (CAT) para diferentes procedimientos quirúrgicos. La idea principal de este tipo de sistemas es aprovechar las tecnologías actuales para ayudar a los médicos a tratar a sus pacientes de manera más eficiente y eficaz. Estos sistemas ayudan a una mejor planeación, transferencia y ejecución de los planes quirúrgicos, minimizando la invasividad de los tratamientos e incrementando la seguridad en intervenciones delicadas.

Un procedimiento quirúrgico asistido por computadora se puede dividir en diversas etapas. En la fase preoperatoria, se crean modelos y, con ellos, planes de tratamiento. Éstos se basan en imágenes médicas, información del paciente e información de otras fuentes, tales como atlas de anatomía **Error! Reference source not found.**. Los sistemas de cómputo ayudan al cirujano a realizar planes óptimos al proporcionar herramientas de análisis de información.

Durante la intervención, los sistemas computacionales ayudan a ejecutar los planes. Para esto, se puede pasar por cuatro etapas de forma cíclica: obtención de información e imágenes médicas transoperatorias, actualización de modelos, actualización del plan y ejecución asistida por computadora. La información que se obtiene, generalmente proviene de sistemas de obtención de imágenes transoperatorias, tales como fluoroscopios, brazos C y ultrasonido (US), además de uno o varios sistemas de rastreo. Con esto, se hace un registro **Error! Reference source not found.** de los modelos preoperatorios con el paciente, de manera que los modelos pueden actualizarse, causando cambios en los planes. Finalmente, el sistema debe ayudar a guiar al cirujano para realizar la intervención con la mayor precisión

posible. Durante la fase postoperatoria, los sistemas pueden ayudar a validar el tratamiento, así como a mantener la información pertinente para dar seguimiento al paciente.

Generalmente, la interfaz entre el cirujano y el sistema CAS es un navegador o un robot quirúrgico, que aprovecha información preoperatoria (principalmente imágenes médicas preprocesadas) y un plan específico diseñado para la intervención. Se puede clasificar a los sistemas CAS, de acuerdo al nivel de participación o actividad que tienen durante el tratamiento, en tres grandes grupos: pasivos, activos y semiactivos (o sinérgicos). Los sistemas pasivos se utilizan para guiar al cirujano durante la realización manual del procedimiento quirúrgico, los sistemas activos realizan parte de la intervención automáticamente, y los sistemas semiactivos realizan actividades conjuntas con el operador humano, tales como restringir sus movimientos en uno o más grados de libertad **Error! Reference source not found.**. Los navegadores son sistemas pasivos, mientras que los robots pueden ser activos o semiactivos. Taylor *et al.* **Error! Reference source not found.** mencionan algunos de los robots quirúrgicos que se utilizan actualmente.

Otra forma de clasificar los sistemas CAS es de acuerdo al tipo de imágenes que se utilizan para guiar al cirujano. Así, pueden estar basados en imágenes preoperatorias, transoperatorias o no estar basados en imágenes. En el caso de cirugía ortopédica, las imágenes preoperatorias son suficientes, ya que los huesos no cambian considerablemente desde el estudio hasta la cirugía, como sucede con otros órganos. Para algunos tipos de cirugía no se realizan estudios de imágenes, por lo que se recomienda el uso de sistemas CAS guiados por plantillas u otros sistemas no basados en imágenes **Error! Reference source not found.**.

En los últimos años se han desarrollado diversos navegadores para ayudar a los cirujanos a realizar diversos tipos de intervención. Un ejemplo de estos sistemas es el desarrollado por Ali Khamene *et al.* **Error! Reference source not found.**, que utiliza un sistema de realidad aumentada para superimponer imágenes de US con video en tiempo real. Con estas imágenes, el sistema ayuda al médico a realizar una biopsia,

para lo que se introduce una aguja en el centro de un tumor y se toma una muestra del tejido. Para esto, el sistema utiliza tres cámaras y dos pantallas a color fijas a la cabeza del usuario. Este sistema es un ejemplo de navegador quirúrgico para colocación de aguja.

También existen navegadores comerciales que se utilizan actualmente en muchos hospitales alrededor del mundo. Uno de los ejemplos con mayor éxito es el sistema *VectorVision*, de BrainLAB **Error! Reference source not found.**, que se muestra en la Figura 1. Este producto incluye las herramientas de *software* y *hardware* necesarias para una gran variedad de intervenciones quirúrgicas. Entre las aplicaciones de *VectorVision*, se encuentra la colocación de agujas, principalmente para biopsias. El problema de este sistema es que requiere de puntos de referencia para el registro de un estudio de imágenes tridimensionales (3D) preoperatorio con el paciente. Así, es necesario que el cirujano toque diversos puntos que se identifiquen claramente en el estudio previo, con una herramienta rastreada. Estos puntos pueden ser características anatómicas o marcadores fiduciales. Este método no es ideal para todas las partes del cuerpo, ya que puede ser muy difícil identificar los puntos.

Figura 1. Navegador quirúrgico VectroVision sky, de BrainLAB. Error! Reference source not found.

Otro navegador utilizado para inserciones de aguja es el *UltraGuide 1000* **Error! Reference source not found.**. Este sistema utiliza una sonda

de US y una aguja. Ambos rastreados magnéticamente. Para conocer la posición de la punta de la aguja, el usuario tiene que ingresar la distancia desde el sensor hasta ésta, ya que el sistema no incluye opciones para calibración. La interfaz de usuario incluye una vista de las imágenes de US con una proyección de la aguja, como se muestra en la Figura 2.

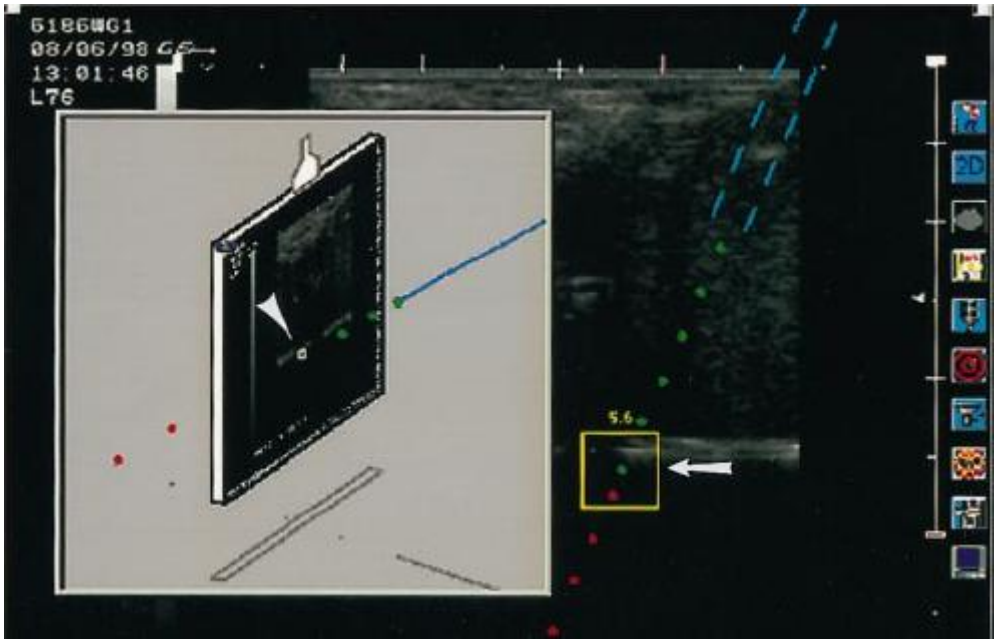

Figura 2. Vista del sistema *Ultraguide 1000*. Error! Reference source not found.

1.1.1. Cirugía guiada por imágenes

La primera vez que se utilizó un sistema de imágenes médicas para guiar un procedimiento quirúrgico fue en 1895, cuando John Cox **Error! Reference source not found.** localizó y extrajo una bala de la pierna de un paciente basándose en imágenes de rayos X. Desde entonces se han utilizado una gran variedad de métodos mínimamente invasivos para visualizar el cuerpo humano con el fin de diagnosticar y de guiar a los médicos en la aplicación de tratamientos y, en particular, durante la realización de cirugías.

Una limitante importante de los rayos X es que sólo permiten la obtención de una proyección del cuerpo, por lo que se está perdiendo mucha información que puede ser necesaria para diagnosticar y guiar intervenciones.

Una aplicación un poco más moderna de las imágenes de rayos X son las angiografías, que presentan una zona del árbol vascular. Para esto, el médico inyecta un medio de contraste en la sangre, de forma que las venas y arterias aparecen en la imagen obtenida. Posteriormente, se desarrolló la Angiografía de Substracción Digital (DSA). Ésta fue la primera modalidad de imágenes médicas en utilizar la computadora para procesar los datos. Para obtener una imagen de DSA simplemente se adquieren dos imágenes de rayos X, una con el medio de contraste y una sin éste. Después, se sustrae una imagen de la otra, de forma que lo único que queda es el árbol vascular. Estas imágenes son muy útiles para diagnosticar una gran variedad de lesiones. **Error! Reference source not found.**

Actualmente se utilizan sistemas de obtención de imágenes médicas muy sofisticados, que proveen información 3D de la anatomía y función del cuerpo. Entre las más usadas están la Tomografía Computarizada (CT), las imágenes de Resonancia Magnética Nuclear (MRI) y la Tomografía por Emisión de Positrones (PET). Cada modalidad se utiliza en diferentes situaciones y para diagnosticar diferentes problemas.

El sistema de imágenes médicas más utilizado transoperatoriamente en ortopedia en la actualidad es la fluoroscopía, que está basada en rayos X. Otro sistema que se utiliza cada vez más son las imágenes de US, que proveen una gran flexibilidad y bajo costo. Otra ventaja importante del US es que no emite radiactividad, por lo que se puede utilizar sin ningún riesgo en todos los pacientes.

El flujo de trabajo médico es la serie de pasos o etapas que se realizan antes, durante y después de una intervención quirúrgica. Cada tipo de intervención tiene un flujo de trabajo médico definido. En particular, cualquier cirugía guiada por imágenes siempre incorpora al menos una etapa de obtención de imágenes médicas. Gracias a su flexibilidad y bajo costo, los sistemas de US pueden integrarse al flujo de trabajo fácilmente. Sin embargo, como la calidad de las imágenes obtenidas no es tan buena como con otras técnicas, en muchos casos será necesario el uso de sistemas de cómputo que ayuden a aprovecharlas al máximo.

1.2 Ablación de tumores con radiofrecuencia

La ablación es la destrucción de un tumor mediante la aplicación directa de un agente térmico (frío o calor) o químico. Los agentes químicos que se han utilizado más comúnmente son el etanol (alcohol) y el ácido acético. La crioablación implica el uso de una criosonda que deposita argón o nitrógeno a temperaturas bajo cero grados Celsius directamente sobre o dentro del tumor. La ablación por calor se puede producir mediante radiofrecuencia (RF), microondas, láser o ultrasonido. Todos estos procedimientos causan la muerte celular en zonas localizadas. Otra ventaja de estos métodos es que se pueden utilizar en casos en que no es posible operar, además de ser poco traumáticos, por lo que el período de recuperación y las molestias del paciente son mínimos.

Actualmente los pacientes con tumores en el sistema muscoloesquelético se tratan principalmente con tres técnicas: quimioterapia, radioterapia, extracción quirúrgica o una combinación de éstas. La primera consiste en la administración intravenosa de medicamentos, la segunda, en la exposición de la zona afectada a radiación con frecuencias de alta energía y la tercera en extraer el tumor mediante cirugía. Todas estas terapias son altamente traumáticas y pueden causar muchas molestias en los pacientes.

El tratamiento de tumores con RF es una técnica relativamente nueva y muy prometedora. El primer reporte de su aplicación se publicó en 1990 **Error! Reference source not found.**. En su artículo, McGahan *et al.* aplicaron este tipo de terapia a tumores en el hígado. Actualmente, ésta es la principal aplicación de la ablación con RF. Más recientemente se ha comenzado a aplicar este tratamiento a tumores en el sistema musculoesquelético **Error! Reference source not found.** con resultados muy prometedores.

En la ablación con RF se emite una corriente eléctrica alterna de alta frecuencia(de 460 a 500 kHz) desde un electrodo en el tejido que se desea eliminar. La corriente fluye hasta uno o más electrodos de salida que cierran el circuito eléctrico, colocados generalmente en los muslos. Las células cercanas al punto de emisión de

energía se calientan debido a la fricción causada por el flujo de electrones. Las células humanas mueren en minutos a una temperatura mayor a 49° C. Cuando la temperatura supera los 105° C, el agua en las células se evapora, y el tejido se carboniza. El gas y el tejido carbonizado inhiben el flujo de energía, limitando el efecto de la terapia de RF. Debido a esto, la temperatura deseable en la punta de una aguja de RF durante el tratamiento debe mantenerse entre los 50° y los 105° C. **Error! Reference source not found.**

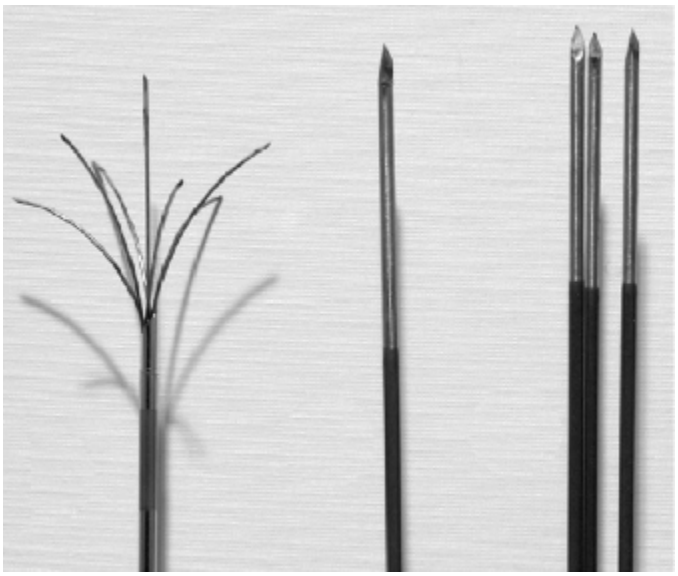

Figura 3. Electrodos en la punta de algunas agujas de RF. Error! Reference source not found.

Un problema común con los primeros generadores de RF es que la aguja se calentaba demasiado, por lo que las células más cercanas se vaporizaban instantáneamente, evitando que el calor llegara a células más lejanas. Esto limitaba en gran medida el volumen de tratamiento (esferas con un diámetro de hasta 1.6 cm). Para aumentar el volumen de acción de la RF se han utilizado diversos métodos, entre los que destacan el uso de arreglos de electrodos, inyección de soluciones salinas para aumentar la conductividad del tejido, agujas con múltiples extensiones retráctiles, el enfriamiento interno de las agujas y la aplicación de RF en pulsos. En la Figura 3 se muestran algunos ejemplos de puntas de agujas comerciales. Con estos avances, se han logrado volúmenes de ablación superiores a 7 cm de diámetro **Error! Reference source not found.**. En la Figura 4 se muestran un sistema de ablación de tumores con

RF moderno, que incorpora las tecnologías de refrigeración y de pulsos antes mencionadas.

Figura 4. Sistema de ablación por RF Cool-tip, de Valleylab.[1]

1.2.1. Flujo de trabajo médico

Para diseñar un sistema que sea útil para los médicos es necesario analizar el flujo de trabajo médico, de manera que el sistema facilite la ejecución de alguna o varias etapas de éste. Si el sistema no es capaz de incorporarse transparentemente al flujo de trabajo, es muy difícil lograr que sea adoptado por los médicos, por lo que muy probablemente fracasaría.

La ablación con RF de tumores en el sistema musculoesquelético tiene un proceso relativamente simple. Primero es necesario realizar el diagnóstico. Para esto se recurre, entre otras herramientas, a un estudio de MRI, que proporciona la forma y tamaño del tumor con mucha precisión. A partir de esta información se realiza el plan de intervención, en el que se detalla el número de agujas necesarias, de acuerdo al tamaño del tumor, y los parámetros del generador de RF (tiempo y potencia). También se decide la trayectoria que deberá seguir cada aguja, de forma que no esté obstruida por arterias u otros órganos delicados. La trayectoria debe, además, respetar los

[1] Valleylab. http://www.valleylab.com, 2007

principios de cirugía oncológica, que incluyen realizar el procedimiento en un solo compartimiento para evitar la dispersión de células cancerosas.

Durante la intervención, se localiza el tumor mediante imágenes de US. Posteriormente, se coloca el plano de adquisición del sistema de US de manera que intersecte la trayectoria planeada. Después, se inserta la aguja en la dirección del tumor, guiándola con el US. Como el plano de adquisición intersecta la trayectoria de la aguja, ésta aparece en las imágenes, como se muestra en la Figura 5. Esta imagen se tomó con la aguja sumergida en agua, por lo que prácticamente todo el ruido que aparece es causado por la aguja. Claramente la calidad de este tipo de imágenes no es muy alta, aunque en muchos casos es suficientemente buena para ubicarla con precisión.

Cuando la punta de la aguja se encuentra en el centro del tumor, se prosigue de la misma manera con las demás agujas, en caso de haberlas. El siguiente paso es activar el generador de RF con los parámetros definidos en el plan. Por último, se retiran las agujas.

Después de la intervención, generalmente se realizan estudios, que pueden incluir PET, para confirmar que se eliminaron todas las células dañadas.

Figura 5. Vista con US de una aguja de RF sumergida en agua.

Generalmente, un navegador quirúrgico ayuda durante la etapa de planeación y posteriormente ayuda a ejecutar el plan realizado. En el caso de ablación de tumores, lo más importante que debe hacer el navegador quirúrgico es ayudar a colocar las agujas con la mayor precisión posible. El sistema debe estar disponible cuando el médico lo necesite durante el tratamiento. De esta manera, se simplifica la etapa de inserción de la aguja, haciendo el tratamiento más sencillo, reduciendo el tiempo de intervención y aumentando la confiabilidad.

1.3 Objetivos y alcance

Para la aplicación del tratamiento de tumores con RF es necesario que el cirujano pueda ubicar la posición y forma del tumor con la mayor precisión posible, como se explicó previamente. El objetivo principal de este trabajo es desarrollar un navegador quirúrgico para ayudar al médico a realizar este tipo de intervenciones en tumores blandos del sistema musculoesquelético de una manera segura y confiable. El sistema mostrará la anatomía del paciente y la posición de la aguja de RF en un

ambiente virtual, de forma que el médico pueda ver en tiempo real exactamente en dónde está aplicando la terapia.

El sistema debe ayudar en la construcción de un modelo 3D preoperatorio de los huesos adyacentes y del tumor que se desea tratar. Después, durante la operación se alineará este modelo con el paciente real, utilizando imágenes transoperatorias de US. Para medir la posición de la aguja de RF con respecto al paciente se utilizará un seguidor óptico comercial.

Se desarrollaron las herramientas de *software* necesarias para la construcción de modelos 3D preoperatorios que incluyen los huesos adyacentes y el tumor, a partir de imágenes de MRI, así como para el registro transoperatorio del modelo 3D basado en imágenes de US **Error! Reference source not found.**,**Error! Reference source not found.**.

El sistema asume que todas las imágenes fueron segmentadas previamente, de forma que estén diferenciados los huesos, el tumor y el resto de la imagen (el fondo). Nuestro sistema calcula las transformaciones geométricas necesarias para pasar del sistema de referencia del modelo preoperatorio al sistema de referencia de las imágenes de US transoperatorias.

Para esto se utilizó el algoritmo *Marching Cubes* **Error! Reference source not found.**, para el modelado 3D, e *Iterative Closest Point* (ICP) **Error! Reference source not found.**, para el registro. Además se diseñó un sistema computacional robusto que puede integrarse adecuadamente a un sistema mayor que agilice y aumente la eficacia de todo el proceso de cirugía: desde el entrenamiento hasta la intervención quirúrgica.

El sistema final es un asistente computarizado para el tratamiento de tumores con RF. Este sistema debe recibir datos de imágenes de MRI, de US y de un sistema de seguimiento 3D que deberá informar sobre la posición y dirección de la aguja de RF y la sonda de US. Así, el sistema debe reportar con un grado muy alto de confiabilidad y en tiempo real, el estado del tratamiento con la ubicación de los

elementos principales que se toman en cuenta durante el procedimiento: hueso, arterias, nervios, tumor y agujas.

La parte del sistema que se realizó en este trabajo consiste en el *software* para navegar la escena durante la intervención, que incluirá la visualización de las imágenes de US, además de un modelo de aguja y de tumor en un ambiente virtual. Además, se implementarán las herramientas para construir los modelos 3D que incluyan al menos huesos y tumor(es), y para alinear (registrar) estos modelos. Éstos estarán basados en imágenes tomadas mediante MRI y US, respectivamente. El *software* debe proporcionar un grado suficiente de confiabilidad, ya que servirá como base para el resto de las etapas en el tratamiento y, de la misma forma, de los otros componentes del sistema.

1.4 Descripción de la Tesis

En el siguiente capítulo se estudia la construcción y registro de modelos desde un punto de vista teórico, incluyendo descripciones de los sistemas de imágenes médicas utilizados, las transformaciones geométricas implicadas en el sistema y se explican las características principales de los sistemas de rastreo.

En el capítulo 3 se describe la implementación del navegador quirúrgico y los sistemas para construcción y registro de modelos 3D. Se da una revisión de las principales herramientas de *software* usadas y una descripción detallada de la funcionalidad del navegador.

El capítulo 4 describe los experimentos realizados para probar los diferentes sistemas desarrollados. Para cada experimento se da una breve introducción y justificación, métodos y materiales, y se presentan los resultados y conclusiones de cada experimento.

Finalmente, se presentan las conclusiones del trabajo, seguidas de una lista con las principales fuentes bibliográficas utilizadas.

2. Construcción y Registro de Modelos Tridimensionales

Durante la etapa de diagnóstico del tratamiento de tumores, el médico debe realizar un estudio con imágenes que muestren las lesiones en el cuerpo del paciente. Esto se hace para saber con precisión la posición y forma del tumor a tratar. En nuestra metodología, se utilizan imágenes de MRI. A partir de estas imágenes, generamos un modelo 3D con los bordes de las estructuras de interés, que en este caso son el tumor y los huesos adyacentes. Durante la intervención, se utilizarán imágenes de US para generar parcialmente un segundo modelo 3D de baja resolución, que será registrado con el primer modelo. El transductor de US será rastreado y calibrado, así como la aguja de RF. Así, es posible mostrar al cirujano el tumor y anatomía del paciente junto con la aguja de RF en un mismo ambiente virtual. A continuación se explica con mayor detalle este proceso.

2.1 Imágenes de resonancia magnética nuclear

La MRI se ha vuelto la modalidad de imágenes médicas predilecta para el diagnóstico de una gran cantidad de tipos de lesiones de tejido blando y la planeación de su tratamiento. Esto se debe a su capacidad de mostrar diferencias sutiles en las características de tejidos. En estas imágenes, el mapeo de estos píxeles a puntos en el cuerpo se realiza mediante la detección de pequeños cambios en la frecuencia y fase de las señales de RF emitidas por los núcleos de hidrógeno en resonancia. Sin embargo, cualquier distorsión en el campo magnético que se genera para la obtención de las imágenes puede llevar a deformaciones geométricas en éstas.

Se eligió el uso de imágenes de MRI por dos razones principales: es una modalidad de imágenes mínimamente traumática para el paciente y las imágenes obtenidas tienen un alto nivel de resolución y contraste en los bordes de las estructuras que deseamos visualizar. Otra ventaja de usar estas imágenes es que no afecta el flujo

de trabajo que se utiliza actualmente, ya que actualmente se utilizan para la etapa de diagnóstico. Gracias a esto es posible distinguir con claridad el tejido afectado del tejido sano en las imágenes obtenidas.

El sistema de imágenes de MRI almacena los resultados en archivos con formato DICOM (*Digital Imaging and Communications in Medicine*) **Error! Reference source not found.**. Estos archivos son documentos multipartes, que permiten almacenar imágenes y objetos de información relacionados. El formato no especifica nada sobre la implementación ni sobre el uso que se le dé a los archivos, por lo que cada compañía tiene diferentes implementaciones con archivos sustancialmente diferentes. Para las imágenes de MRI, el encabezado de los archivos DICOM contiene información sobre la localización y escala de las imágenes, además de información sobre el paciente, el médico y la institución en que se realizó el estudio.

El siguiente paso es elegir las imágenes que muestren con mayor claridad la forma y posición del tumor. Estás imágenes deben ser segmentadas por el médico con ayuda de un sistema de cómputo, de manera que se tengan imágenes binarias en las que se tenga un valor de objeto 1 en las zonas internas de huesos y tumor, y un valor de fondo 0 en el resto de la imagen, como se ilustra en la Figura 6.

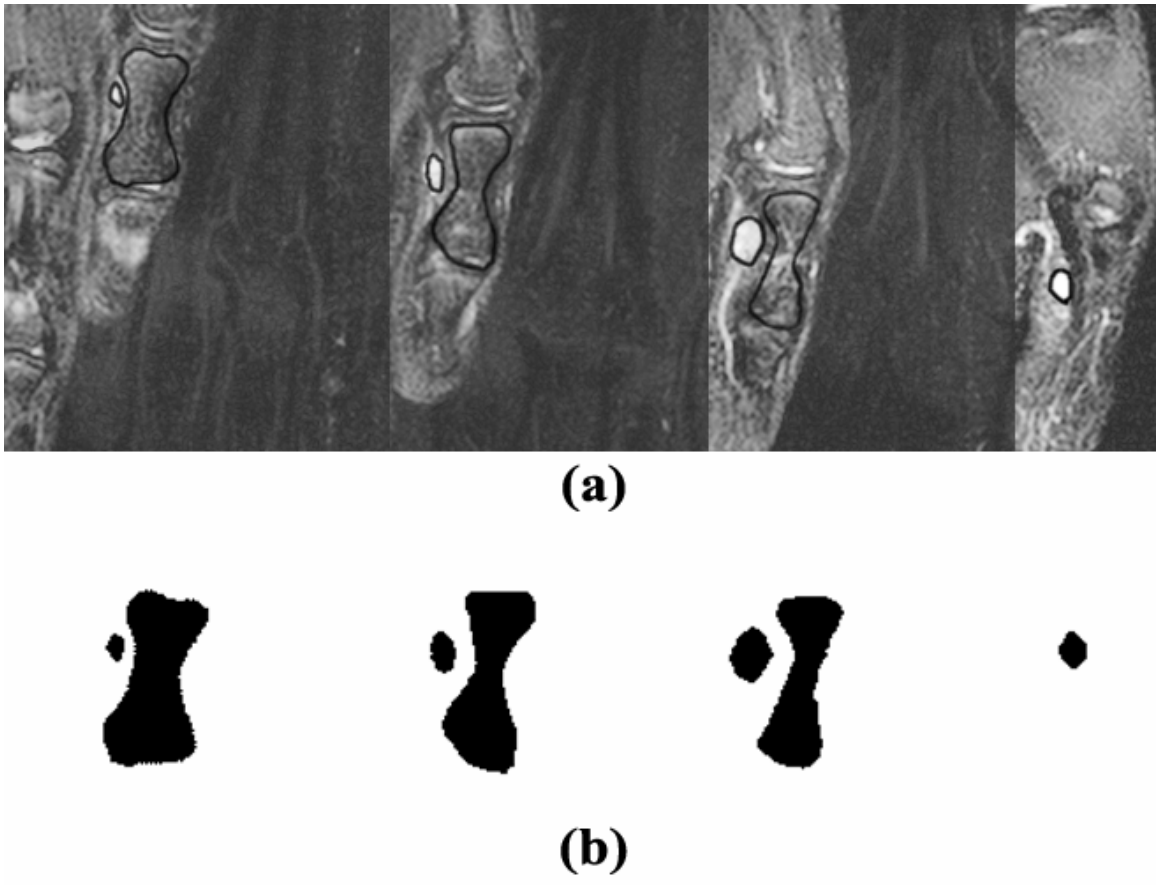

Figura 6.Serie de imágenes con un hueso del pulgar y tumor adyacente (a) marcadas y (b) segmentadas.

2.2 Mallado del modelo de voxeles

El mallado es el proceso de obtención de una malla poligonal que represente la superficie de un volumen dado. En este caso, el volumen está dado por los elementos de volumen (voxeles) definidos a partir de las imágenes segmentadas de MRI. Este método permite visualizar el borde de una estructura 3D utilizando algoritmos de despliegue implementados en *hardware* en cualquier computadora personal moderna.

Para realizar este proceso se eligió una versión modificada del algoritmo *marching cubes*, presentado originalmente por Lorensen *et al.* **Error! Reference source not found.**. Este algoritmo toma una serie de cortes, o imágenes binarias, y regresa una serie de polígonos (en este caso, triángulos) en el espacio 3D que representan la superficie de los objetos representados con voxeles en los cortes. La idea es recorrer el volumen completo, tomando una vecindad de 2x2x2 voxeles, o

cubo que marcha, como se muestra en la Figura 7, tomada del artículo original de Lorensen *et al.* **Error! Reference source not found.**. En esta imagen, los voxeles se representan con puntos en el espacio y el cubo que marcha abarca 8 de estos puntos. Para cada posición del cubo, se detectan las intersecciones con el borde del volumen y se agregan triángulos a la malla. El algoritmo se basa en una interpolación lineal sobre los bordes de los voxeles, que es la mejor interpolación que se puede realizar tomando una vecindad de estas dimensiones. Así, para cada una de las 256 (2^8: dos posibles valores en 8 posibles posiciones) configuraciones de voxeles existe una configuración de triángulos que se adapta para formar la malla. Gracias a las cualidades de simetría del cubo y del problema mismo, es posible reducir los 256 casos a sólo 14 patrones.

La versión modificada de *marching cubes* que se usó **Error! Reference source not found.** obtiene, además de los triángulos de la superficie, las vecindades entre estos triángulos y el conjunto de vértices que los forman sin repeticiones. En la Figura 8 se muestra un ejemplo de modelo 3D en voxeles y en malla.

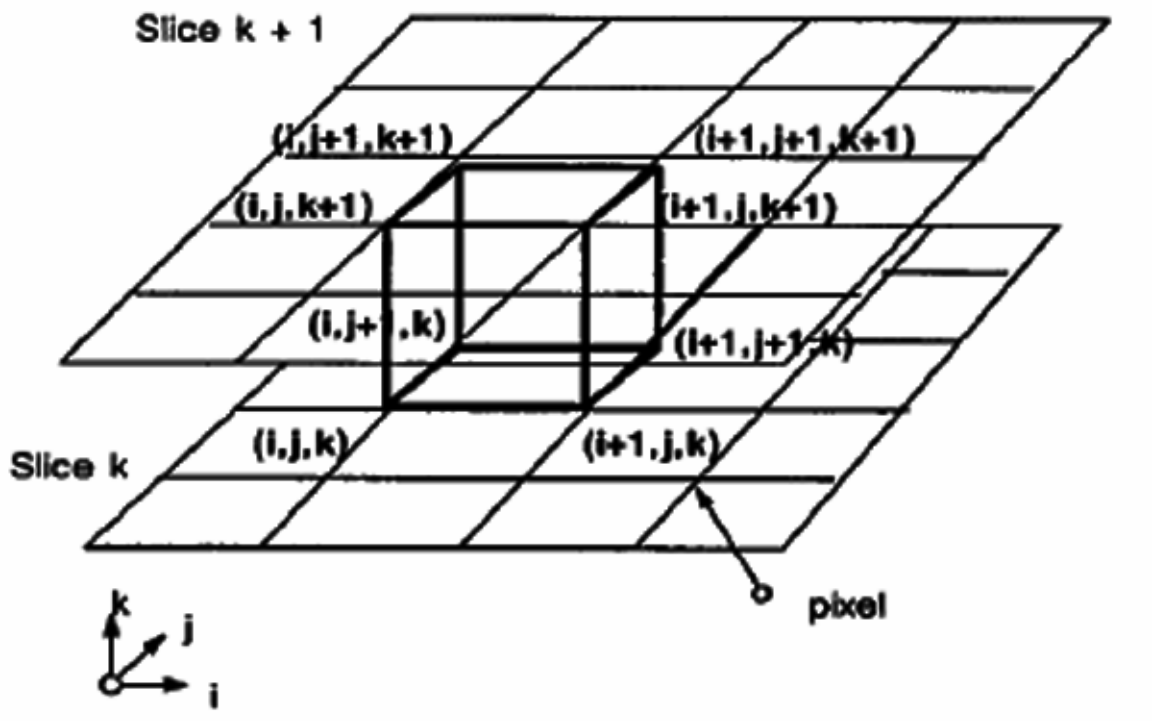

Figura 7.Cubo "que marcha" original. Lorensen *et al.* Error! Reference source not found..

Figura 8.Modelo 3D representado con voxeles y con malla (http://www.effectware.com).

2.3 Registro de Modelos

El modelo 3D construido a partir de imágenes de MRI debe ayudar al cirujano a posicionar la aguja de RF durante la intervención. Para esto es necesario presentar el modelo y la aguja en un mismo ambiente virtual, por lo que deben compartir un sistema de coordenadas. En general, todos los elementos que se modelen dentro del navegador quirúrgico deben compartirlo. Las imágenes de MRI se adquieren con un sistema de coordenadas propio. Las imágenes de US se adquieren durante la cirugía con una sonda rastreada, por lo que es posible crear un modelo en el sistema de coordenadas del quirófano, definido por el sistema de rastreo. Así, para mostrar el modelo preoperatorio durante la cirugía en la posición correcta es necesario alinearlo con el modelo basado en US. Este proceso se conoce como registro de modelos. El resultado debe ser una transformación geométrica para mover el modelo de un sistema de coordenadas a otro.

La manera más simple y precisa de registrar un modelo preoperatorio con la anatomía del paciente se basa en el uso de marcadores fiduciales que se fijan rígidamente a los huesos del paciente antes de la adquisición de imágenes de

diagnóstico. Posteriormente, durante la intervención, se utilizan estos marcadores para realizar el registro **Error! Reference source not found.**. Sin embargo, éste es un sistema altamente invasivo, que requiere de una intervención quirúrgica anterior al estudio de MRI y que causa dolor postoperatorio en el paciente. Este tipo de marcadores se utiliza comúnmente para neurocirugías. En este caso se justifica la intervención preoperatoria necesaria para fijar los marcadores, ya que el tratamiento en sí es muy traumático y requiere de un muy alto grado de precisión.

Una alternativa es utilizar marcadores superficiales, sobre la piel, pero como éste es un órgano muy flexible y en general está lejos de la zona de interés, se pierde mucha precisión. También se pueden utilizar puntos conocidos en el cuerpo para realizar el registro. El problema de esta aproximación es que puede ser difícil reconocer puntos característicos sobre la piel en imágenes de MRI, además de que se tiene el mismo problema que con los marcadores superficiales.

En nuestro sistema la sonda de US y la aguja de RF se rastrean con el mismo sistema de coordenadas, por lo que teniendo el registro de modelos es posible tener todos los elementos de interés en el mismo ambiente virtual. La Figura 9 muestra gráficamente las transformaciones y puntos de interés del navegador. F_W representa el sistema de coordenadas del mundo, dado por el sistema de rastreo; $F_{U,}$ el de la imagen de US; F_P, el de la sonda de US; F_I, el de la imagen de la cámara; F_C, el de la cámara; F_N, el de la aguja de RF; ${}^N\boldsymbol{B}$ y ${}^N\boldsymbol{P}$ representan la posición de la base y la punta de ésta. Cada flecha en la figura representa una transformación geométrica rígida (rotación y traslación). Utilizando estas transformaciones es posible conocer cualquier punto de interés en el sistema de coordenadas común: F_W.

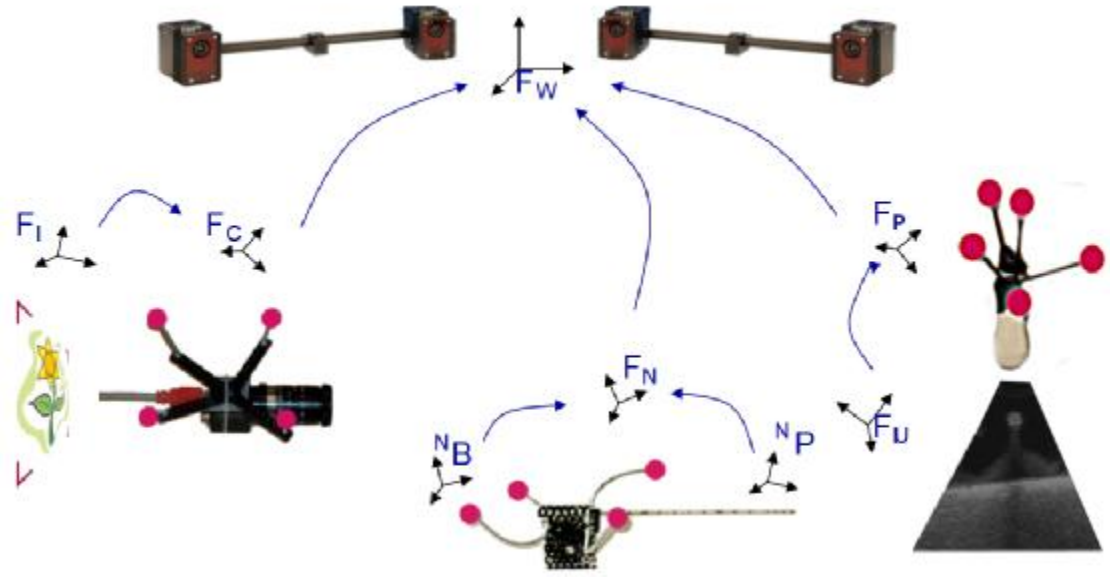

Figura 9.Transformaciones geométricas involucradas en el navegador quirúrgico.

2.4 Sistemas de rastreo (trackers)

Existen diversos métodos para rastrear objetos en el espacio. Los sistemas más usados se dividen en rastreadores mecánicos, electromagnéticos y ópticos. **Error! Reference source not found.**

Los rastreadores mecánicos se basan en un brazo al que se fija la herramienta de trabajo. El brazo tiene sensores que miden el desplazamiento de cada eje de movimiento o articulación, de manera que es posible calcular la posición y dirección de la herramienta quirúrgica, fija a la última sección rígida del brazo. Para esto es necesario que cada sección del brazo sea rígida. Estos rastreadores tienen una alta precisión y no requieren línea de visión para operar, pero pueden resultar muy estorbosos en la sala de operaciones, por lo que es difícil que sean adoptados por muchos cirujanos **Error! Reference source not found.**. Una ventaja de estos sistemas es que pueden incorporar motores, de forma que el sistema CAS puede incluir elementos activos o semi-activos. La Figura 10 muestra un ejemplo de rastreador mecánico de 6 grados de libertad. El brazo está formado por 6 secciones rígidas y 6 articulaciones con sensores.

Figura 10.Dispositivo de rastreo mecánico de 6 grados de libertad[1].

Los rastreadores electromagnéticos funcionan con un generador de campos electromagnéticos y objetivos de rastreo formados por 3 sensores magnéticos ortogonales. Los sensores detectan su posición dentro del campo, de manera que se puede calcular la posición y orientación del objetivo de rastreo. Las principales ventajas de estos sistemas es que no requieren campo de vista (funcionan dentro del cuerpo) y que los objetivos de rastreo son muy pequeños (1x1x2 mm^3). El problema es que el campo electromagnético es muy sensible a distorsiones causadas por la presencia de metales, por lo que llegan a tener errores de más de un centímetro, que no es tolerable para la mayoría de las aplicaciones clínicas. Otro problema es que el volumen de rastreo es relativamente pequeño (30 x 30 x 30 cm^3 aprox.). La Figura 11 muestra un dispositivo comercial de rastreo electromagnético.

[1] The FARO arm, Faro Industries Inc, FL, EUA.

Figura 11.Rastreador electromagnético *miniBIRD*.[1]

El sistema que se utilizó en este proyecto, debido a sus características, es un sistema óptico. Este tipo de rastreadores funciona con un arreglo de al menos dos cámaras calibradas. Con esto, al reconocer un objeto en una imagen es posible calcular una línea recta imaginaria en el espacio que lo cruce. Al haber dos cámaras que detectan el mismo objeto, las líneas imaginarias calculadas para cada una se cruzan en un punto del espacio. Este punto es la posición 3D del objeto observado por ambas cámaras. Un sistema con más de dos cámaras es más robusto, ya que es capaz de seguir rastreando los marcadores cuando alguna cámara no los registra debido a problemas de oclusión. Los seguidores ópticos son más precisos y confiables que los basados en campos electromagnéticos. El problema es que necesita línea de visión entre las cámaras y los marcadores, por lo que no se puede rastrear objetos dentro del cuerpo. El otro problema es que los marcadores y configuraciones de éstos son relativamente grandes, por lo que pueden resultar incómodos. Los marcadores pueden ser activos o pasivos. Los primeros están formados por arreglos de LEDs (diodos emisores de luz) y tienen el problema de que necesitan una fuente de energía. Los marcadores pasivos son más cómodos, ya que sólo reflejan luz, que puede ser infrarroja. De esta manera, los marcadores son inalámbricos y el sistema en general funciona de manera transparente en la sala de operaciones.

[1] Ascension. http://www.ascension-tech.com/products/, 2007

El sistema de coordenadas global que usamos está dado por un sistema de rastreo óptico. Este sistema proporciona la posición y orientación de una o más configuraciones de marcadores. Cada marcador es una esfera que refleja luz infrarroja. El sistema de rastreo, mediante al menos dos cámaras, detecta las esferas y calcula la posición y orientación de la configuración de marcadores. Es importante notar que los marcadores fijos a cada herramienta están en una configuración espacial rígida y única. El sistema debe conocer previamente la configuración de marcadores para cada herramienta. Es necesario que no haya dos configuraciones iguales para que el sistema pueda diferenciarlas.

Figura 12.Sistema de rastreo Polaris, de NDI, con herramienta y marcadores.[1]

Usando una configuración de marcadores fija a la aguja de RF podemos saber la posición de ésta en el sistema de coordenadas dado por el rastreador óptico. A pesar de que la aguja de RF es algo flexible, se asume que es rígida. Esto se justifica debido a que durante el tratamiento sólo se aplica fuerza a la aguja en la dirección de su eje longitudinal, que es el más rígido. Esta fuerza no causa deformaciones representativas.

[1] Northern Digital Inc. http://www.ndigital.com/medical/polarisoriginal.php, 2007

Así, teniendo la aguja calibrada con los marcadores, podemos calcular la posición de la punta.

2.4.1. Exactitud del sistema de rastreo

La principal ventaja de usar un sistema de rastreo óptico es la exactitud que se obtiene en las mediciones de posición de los marcadores. El error de localización se propaga a través del sistema de rastreo. Los errores de detección en las imágenes obtenidas por las cámaras (debidos a ruido, falta de nitidez o falta de resolución), se propagan a errores en la localización de los marcadores, que se propaga a errores en la localización 6D (posición y orientación) de la herramienta a la que están fijos. Sin embargo, es importante notar que en esta aplicación nuestro punto de interés no son los marcadores, sino la punta de la aguja de RF. Debido a esto, es necesario tomar en cuenta otra propagación del error. Como la aguja es rígida y los marcadores están fijos, un error de posición en la medición se propaga a la punta de la aguja sin modificación. Sin embargo, un pequeño error de rotación en la configuración de marcadores puede convertirse en un error importante en la punta de la aguja, ya que ésta se encuentra a una distancia significativa. **Error! Reference source not found.**

El error de posición e proveniente del error angular j puede expresarse en función de éste y la longitud de la aguja, L, como

$$e(j, L) \approx L \tan j \ , \qquad (2.1)$$

Por simplicidad, esta fórmula asume que el ángulo j es muy pequeño. Así, para un error angular poco significativo de 1° y una longitud típica para una aguja de RF de 20 cm, el error en la posición sería de cerca de 3.5 mm, como se muestra en la Figura 13. Esta desviación es causada solamente por el error en la medición de un ángulo, para calcular el error total sería necesario tomar en cuenta los errores en los tres ángulos de rotación (a, b y g) y el error en las traslaciones (d). El error RMS se puede definir como:

$$e = \sqrt{e^2(a) + e^2(b) + e^2(g) + e^2(d)}, \qquad (2.2)$$

Así, el error en la posición del punto de interés es igual al error en la posición de la configuración de marcadores más un error en la posición proveniente del error angular propagado, que se incrementa proporcionalmente a la distancia desde los marcadores **Error! Reference source not found.**. La Figura 14 muestra gráficamente la propagación del error a un punto de interés en la herramienta rastreada debido a traslación y a rotación.

Figura 13.Propagación del error angular en 2D.

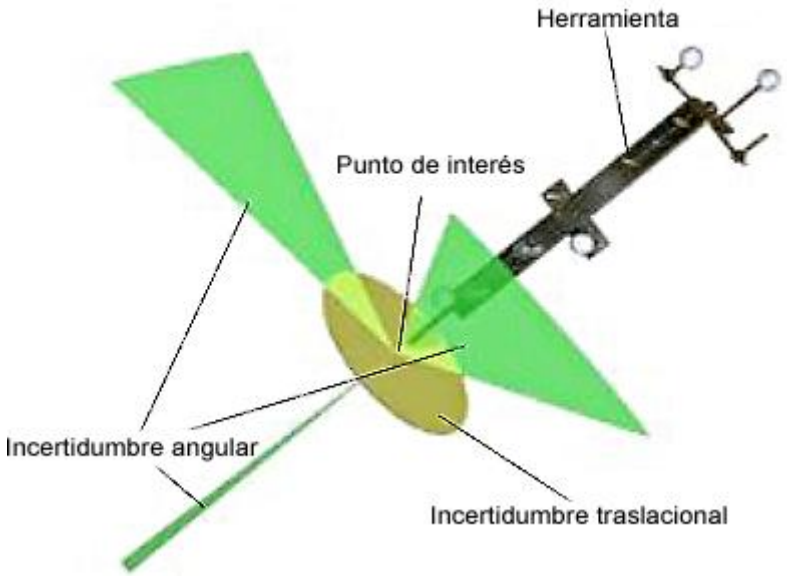

Figura 14.Propagación del error de medición en 3D. Error! Reference source not found.

La forma de minimizar este tipo de errores es construir configuraciones de marcadores adecuadas al problema. En el caso de agujas de RF, para que la configuración sea más confiable, es recomendable que la mayor distancia entre marcadores se encuentre en la dirección longitudinal de la aguja, de forma que este eje presente los errores de menos magnitud. Esto se debe a que el error angular se

disminuye al aumentar la distancia entre marcadores. Un ejemplo de distribución adecuada de marcadores para la aguja de RF se muestra gráficamente en la Figura 15. Nótese que los marcadores deben estar alejados de la punta para permitir el manejo de la aguja. Al mismo tiempo, la distancia máxima entre marcadores se encuentra en la dirección del eje longitudinal de la aguja.

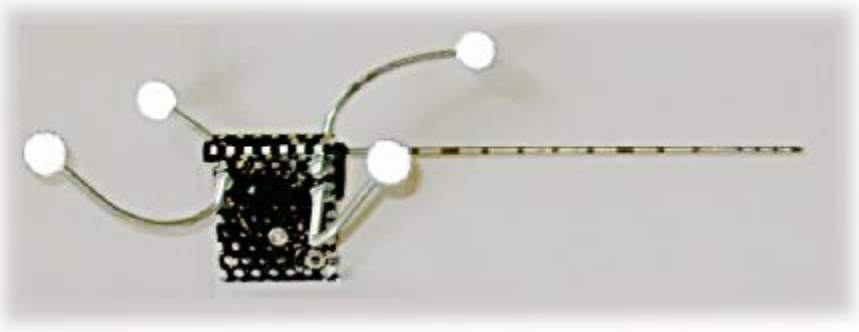

Figura 15.Aguja de RF con configuración de marcadores.

Tomando en cuenta estos lineamientos, y haciendo una calibración adecuada de la aguja, como se describe en la sección **Error! Reference source not found.**, es posible reducir el error en la punta a menos de un milímetro.

2.5 Imágenes de ultrasonido (transoperatorias)

El US es una modalidad de imágenes médicas que se utiliza mucho transoperatoriamente, ya que no requiere de condiciones especiales en la sala de operaciones, ni tiempo significativo de procesamiento, ni altos costos. El sistema funciona con un generador de ondas de US, que son inducidas en el cuerpo por medio de un transductor o sonda, que escucha el eco generado en el cuerpo y genera una imagen en escala de grises. Así, cuando hay un cambio en el medio (el borde de un tejido), las ondas se reflejan y son registradas por el sistema, que despliega píxeles claros en la posición correspondiente al lugar donde se reflejó la onda. Así se forma una imagen con píxeles claros en donde se reflejan las ondas y píxeles oscuros en donde no se detectan ecos.

El problema de esta modalidad de imágenes es que es muy susceptible a ruido multiplicativo, por lo que las imágenes obtenidas son de baja calidad. Otro problema es que las ondas de US no pueden atravesar tejidos duros, como el hueso, por lo que en ocasiones las imágenes muestran zonas ocluidas como vacías u homogéneas. En el caso particular de tumores en el sistema musculoesquelético, es posible detectar el tumor en las imágenes de US, pero no es fácil detectar los bordes y, por tanto, las dimensiones. Es por esto que esta modalidad se utiliza durante la intervención para guiar, pero no previamente para diagnosticar y planear.

En nuestro modelo, el transductor de US tiene fijada una configuración de marcadores y la aguja de RF, otra. Después de un proceso de calibración de US, podemos conocer con bastante exactitud la localización en el espacio del plano que muestra el sistema de US. El método de calibración utilizado es conocido como Calibración con Muro Simple (*Single Wall Calibration)* **Error! Reference source not found.**,**Error! Reference source not found.**. En éste, se toman imágenes del fondo plano de un contenedor de agua desde diversos ángulos y posiciones. Esta superficie es fácil de distinguir en las imágenes de US, por lo que es posible segmentarla automáticamente usando algoritmos de procesamiento de imágenes.

Después de la adquisición de una serie de imágenes, de las que conocemos su posición en el sistema de coordenadas del rastreador, podemos reconstruir un volumen, conocido como *ultrasonido tridimensional* (US-3D) **Error! Reference source not found.**. El volumen obtenido se segmenta manualmente para obtener un nuevo modelo 3D parcial, menos confiable y de menor resolución que el obtenido preoperatoriamente.

El siguiente paso es realizar el registro del modelo basado en imágenes de MRI con este nuevo modelo basado en imágenes de US. Con este registro obtenemos la transformación geométrica necesaria para incorporar el modelo de alta resolución en el sistema de coordenadas dado por el rastreador. Ahora, con todos los modelos en el mismo sistema de coordenadas, es posible desplegar en tiempo real la posición de la

aguja de RF con respecto a la anatomía del paciente y al tumor que se trata de eliminar.

2.6 Algoritmo de registro: Iterative Closest Point (ICP)

Un aspecto clave del navegador quirúrgico es un registro preciso y confiable del modelo 3D de la anatomía del paciente, basado en imágenes de MRI adquiridas preoperatoriamente. La idea principal del registro es encontrar una transformación geométrica adecuada para transformar un modelo en un sistema de coordenadas al sistema de coordenadas de otro modelo, suponiendo que ambos modelos representan el mismo objeto. Al primer modelo se le conoce como modelo de datos. En el proceso de registro, éste será el modelo móvil. El segundo modelo es el modelo de forma o de referencia. Así, lo que se busca es transformar el modelo de datos para que sea lo más parecido posible al modelo de forma. La transformación que logra esto se conoce como transformación de registro. En nuestro navegador el modelo de forma es el obtenido preoperatoriamente (MRI) y el modelo de datos, el basado en imágenes de US obtenidas durante la intervención.

Este sistema está pensado para tratar tumores en el sistema musculoesquelético únicamente. Gracias a esto, podemos asumir que los modelos obtenidos son rígidos, por lo que no son susceptibles a deformaciones debido a movimientos del cuerpo, como la respiración o los latidos del corazón. En el caso general, para órganos del cuerpo en la zona abdominal, es importante tomar en cuenta estas deformaciones, por lo que es necesario utilizar modelos deformables y algoritmos de registro más complejos.

Otra característica que debemos tomar en cuenta es que tanto el sistema de MRI como el de US están adecuadamente calibrados, por lo que conocemos las dimensiones reales de los modelos obtenidos a partir de cada una de estas modalidades de imágenes médicas.

Estas consideraciones son importantes para elegir un algoritmo de registro adecuado al problema. En nuestro caso, el algoritmo que necesitamos debe obtener

una transformación rígida (únicamente traslación y rotación). Además, no conocemos puntos con características específicas en común, por lo que no podemos utilizar un algoritmo que tome como entrada las características comunes de los dos modelos. Otro punto importante es que el registro tiene que calcularse durante la intervención, por lo que el algoritmo debe ser rápido y robusto.

El algoritmo más adecuado para estas condiciones es el método Iterativo de Punto más Cercano **Error! Reference source not found.** (*Iterative Closest Point: ICP*). El algoritmo se resume en los siguientes pasos:

a. *Calcular los puntos más cercanos*. El algoritmo supone que una buena aproximación a los puntos del modelo de forma correspondientes a cada punto en el modelo de datos son los puntos más cercanos, al inicio de cada iteración.

b. *Calcular el registro o transformación*. Éste es el paso más importante. Generalmente, la transformación se representa con un vector que contiene los valores de rotaciones y traslaciones. Para esto son necesarios al menos 6 valores: tres rotaciones y tres traslaciones. En nuestros experimentos utilizamos un vector con 7 valores: 3 valores de traslación sobre los ejes coordenados globales, y un cuaternión para representar una rotación al rededor de un eje arbitrario.

c. *Aplicar el registro*. Para esto, se forma una matriz de rotación R y un vector de traslación T. Así, para cada punto P, se calcula $P' = RP + T$.

d. *Repetir* hasta que el cambio en el error medio cuadrático sea menor que un valor predefinido. Para terminar el algoritmo también se pueden utilizar otras medidas de convergencia o un número máximo de ciclos.

Se ha demostrado que este algoritmo converge al mínimo local **Error! Reference source not found.** con una complejidad computacional adecuada al problema de $O(N_p N_x)$ por iteración, en el peor caso, donde N_p es el número de puntos en el modelo de datos (basado en US) y N_x, el número de puntos en el modelo de forma (basado en MRI).

Es importante notar que el mínimo local, en general, no corresponde con el mínimo global. Para que esto se dé, es necesario proporcionar una transformación inicial buena. Ésta puede calcularse a partir de datos integrados con las imágenes en formato DICOM, puede buscarse mediante un sistema de búsqueda exhaustiva (probar con un número determinado de posiciones predeterminadas), o buscarse manual o semiautomáticamente mediante una interfaz de usuario apropiada.

2.7 Discusión

La construcción y el registro de modelos es una de las características que hacen único al sistema que se propone en esta tesis. Mientras la mayor parte de los sistemas realizan el registro basándose en imágenes preoperatorias únicamente, nuestro sistema incluye un paso más para la construcción del modelo 3D, que se aprovecha para obtener el registro. Gracias a esto, se puede aprovechar la alta calidad de las imágenes de MRI y la flexibilidad del US. Además, la construcción del modelo ayuda a tener una mejor visualización durante la intervención.

El algoritmo ICP probó ser adecuado para el problema de registro gracias a su flexibilidad y eficiencia. Dado que los modelos 3D con los que trabajamos están formados por vértices, es muy natural utilizar un algoritmo de registro de puntos en el espacio.

La utilización de sistemas de rastreo comerciales ayuda a que el sistema sea altamente confiable y preciso, además de agregar una capa de abstracción, necesaria para que el esfuerzo se concentre en la lógica y desempeño del sistema CAS, obviando los procesos de procesamiento de imágenes y métodos numéricos necesarios para rastrear un objeto en el espacio.

3. Implementación del Navegador Quirúrgico para Tratamiento de Tumores

Durante el desarrollo de este proyecto se realizó una estancia de 4 meses en el grupo de investigación en *Procedimientos Médicos Asistidos por Computadora y Realidad Aumentada* [1] (CAMPAR), de la *Universidad Tecnológica de Múnich* [2] (TUM, por sus siglas en alemán). Durante la estancia se implementó un prototipo del navegador que utiliza las clases desarrolladas en CAMPAR para calibración de US y seguimiento. También se realizaron diversos experimentos de validación con un seguidor óptico (*ARTTrack*) y un equipo de US (*Picker Computer Sonograph CS 9300*), que no estaban disponibles en el Laboratorio de Análisis de Imágenes y Visualización del CCADET, UNAM. Como resultado de la estancia de investigación se inició la colaboración entre el Laboratorio de Imágenes y el grupo CAMPAR para el desarrollo de sistemas quirúrgicos basados en CAMPAR.

3.1 Herramientas

Para la implementación del sistema de navegación se utilizaron diversas herramientas de *software* que facilitan el desarrollo, permitiéndonos enfocar los esfuerzos en resolver los asuntos específicos del problema. Algunas de las herramientas principales que se utilizaron son el marco de trabajo CAMPAR y la biblioteca ITK, que se describen con mayor detalle a continuación.

3.1.1. CAMPAR

El sistema CAMPAR **Error! Reference source not found.** es un marco de trabajo desarrollado por el grupo del el nombre en la Universidad Tecnológica de Múnich. Este sistema se diseñó como base para el desarrollo de diversos proyectos del

[1] Chair for Computer Aided Medical Procedures, http://wwwnavab.in.tum.de

[2] Technische Universität München, http://www.tum.de

grupo. Principalmente, se trabaja con dispositivos virtuales, de entrada y salida. Los dispositivos de entrada incluyen cámaras, sistemas de sincronización, sistemas de rastreo y archivos de diversos tipos. Los dispositivos de salida son principalmente interfaces de usuario. El sistema se configura mediante un archivo XML **Error! Reference source not found.** en el que se declaran las propiedades de los dispositivos y sus relaciones.

CAMPAR también incluye una biblioteca de clases para representar objetos matemáticos, como vectores y matrices, que es la base de muchas de las herramientas que ofrece. Este marco de trabajo está totalmente desarrollado en C++ y utiliza como apoyo una serie de aplicaciones de terceros, entre las que destaca Qt **Error! Reference source not found.** como base para implementar interfaces gráficas de usuario. El Anexo I describe la implementación de aplicaciones gráficas basadas en CAMPAR.

En nuestro navegador para tratamiento de tumores se utilizaron los siguientes dispositivos y aplicaciones pertenecientes a CAMPAR:

a. *UltrasoundCalibration*. Una aplicación que utiliza el método *Single Wall Calibration* **Error! Reference source not found.**,**Error! Reference source not found.**, descrito en la sección **Error! Reference source not found.**, para calibrar el sistema de US. La calibración del US se realiza antes del tratamiento y el navegador para ablación por RF sólo utiliza los resultados de ésta.

b. *HotSpotCalibration.* Ésta es una de las opciones para calibrar la punta de la aguja. Es una aplicación que toma una serie de medidas del rastreador óptico y regresa la posición de la punta de la aguja en el sistema de coordenadas de los marcadores fijos a ella. Para esto, asume que todas las posiciones registradas están en la superficie de una esfera, cuyo centro es la punta de la aguja.

c. *TrackingDevice.* Es la clase que representa al sistema de rastreo. Mediante este dispositivo se puede obtener la posición y orientación de las herramientas con marcadores declaradas en el sistema en cualquier momento.

d. *GrabberDevice*. Esta clase abstrae un dispositivo de captura de video análogo. Se utiliza para la adquisición de las imágenes de US en tiempo real.

e. *Synchronizer.* Ésta es una de las clases principales del marco de trabajo. Ayuda a la sincronización de los datos, de forma que las imágenes presentadas y la posición de los marcadores en un momento dado representen efectivamente el mismo momento. Para esto, la clase *Synchronizer* se basa en el protocolo NTP: *Network Time Protocol* **Error! Reference source not found.**. Este protocolo sincroniza el reloj de una computadora cliente con el de uno o varios servidores. Para esto, el servidor manda una serie de mensajes que utiliza el cliente para calcular la diferencia entre el tiempo en su reloj y el del servidor. Con algoritmos de selección descarta respuestas inválidas y con algoritmos de combinación obtiene un único resultado a partir de todas las respuestas sobrantes. Actualiza su reloj con esta diferencia y segundos después se repite el proceso. Con este protocolo es posible sincronizar relojes de computadoras conectadas por red con un error inferior a un milisegundo.

f. *ARCamera.* Implementa la funcionalidad de realidad aumentada, que incluye el despliegue de imágenes de al menos una cámara y la configuración necesaria para superponer elementos 3D a la imagen.

CAMPAR se implementó en C++ por su capacidad de generar sistemas de tiempo real, así como de interactuar directamente con los dispositivos de *hardware* manteniendo un diseño orientado a objetos. La mayor parte de las clases centrales de CAMPAR son clases abstractas, que representan diversos dispositivos. Gracias a esto, es posible programar los sistemas sin tomar en cuenta los dispositivos específicos que se utilicen. Por ejemplo, el mismo sistema puede trabajar de la misma manera con un rastreador ART **Error! Reference source not found.** que con un NDI **Error! Reference source not found.**. Las clases específicas que se usan se declaran y

configuran en el archivo XML que se da como entrada al sistema, de forma que no es necesario recompilar el código para cambiar de proveedor.

3.1.2. *Insight Registration and Segmentation Toolkit* (ITK)

Esta extensiva serie de herramientas se desarrolló inicialmente como parte del proyecto *Visible Human Project* **Error! Reference source not found.** y **Error! Reference source not found.**, de la Biblioteca Nacional de Medicina de Estados Unidos (NLM). El objetivo de ITK[1] es proporcionar las herramientas de segmentación y registro necesarias para procesar imágenes médicas. Está desarrollado completamente en C++, utilizando programación genérica o basada en plantillas. En general, se utiliza para procesamiento de imágenes, de manera que tiene clases para leer, procesar y escribir imágenes, entre muchas otras funciones. ITK tiene un sistema de *tubería* (pipeline), en el que los datos son leídos, procesados por una serie de elementos y pasados a la siguiente etapa. En la última etapa, los datos se mandan a una salida que puede ser un archivo **Error! Reference source not found.**. Además del procesamiento de imágenes, este *toolkit* incluye clases para facilitar el registro de modelos. En este proyecto se utilizaron especialmente las siguientes clases:

a. *itkQuaternionRigidTransform*. Proporciona la funcionalidad necesaria para realizar transformaciones rígidas basadas en cuaterniones. El resultado del registro se expresa en un objeto de esta clase.

b. *itkLevenbergMarquardtOptimizer*. Utiliza el algoritmo de Levenberg-Marquardt de optimización. Se utilizó para encontrar la transformación que minimizara la distancia euclidiana entre dos conjuntos de puntos.

c. *itkPointSet*. Representa un conjunto de puntos. En el sistema desarrollado, un conjunto de puntos es un modelo 3D.

[1] National Library of Medicine Insight Segmentation and Registration Toolkit (ITK). http://www.itk.org

d. *itkPointSetToPointSetRegistrationMethod*. Esta clase utiliza, entre otros recursos, las clases antes descritas para calcular el registro entre dos conjuntos de puntos o modelos.

3.2 Modelado

El modelo 3D de tumor y hueso se crea a partir de una serie de imágenes. En este caso, solamente se realizó el modelo basado en imágenes de MRI. El modelo basado en US no se generó debido a que en el caso de estudio, las imágenes de US no incluyen datos de localización, por lo que no se cuenta con el volumen, que es necesario para el proceso de mallado.

La primera parte del modelado consiste en segmentar las imágenes. En el sistema final, las imágenes serán segmentadas semiautomáticamente, con ayuda de las herramientas que se desarrollan en un proyecto paralelo **Error! Reference source not found.**. Para efectos de este trabajo, la segmentación de las imágenes se realizó manualmente, utilizando *ImageJ* **Error! Reference source not found.**, un programa de procesamiento de imágenes de propósito general. La **Error! Reference source not found.** muestra un ejemplo de imágenes segmentadas mediante este proceso. Con el mismo programa, se exportaron las imágenes a archivos de texto, que posteriormente se concatenaron en un solo archivo y se alimentaron al programa generador de mallas, previamente desarrollado en el Laboratorio de Visualización del CCADET **Error! Reference source not found.**. El resultado fue un archivo de texto con los vértices y los triángulos. Éste archivo se lee desde el navegador, utilizando la clase *MeshNeedleTarget*. Así, es posible visualizar el modelo 3D del tumor y hueso en el mismo espacio que la aguja de RF, como se muestra en la Figura 16. Dado que el caso de estudio analizado no incluye datos de rastreo para las imágenes de US, es imposible realizar el registro de manera que ambos modelos y la aguja se visualicen en el mismo ambiente virtual. En este sentido, la visualización del modelo obtenido sólo se usó como demostración y no como parte integral del navegador.

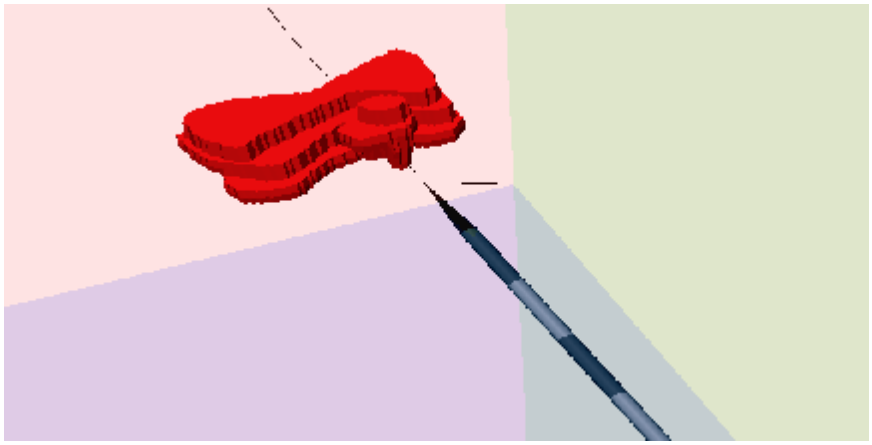

Figura 16. Vista del modelo 3D de tumor y hueso en el navegador quirúrgico.

3.3 Registro

El sistema que se desarrolló para el registro de modelos es una aplicación independiente del sistema de navegación, aunque se pretende agregarlo a éste más tarde, junto con el sistema de segmentación. Como se mencionó en la sección **Error! Reference source not found.**, el algoritmo de registro utilizado es ICP, debido a su buen desempeño y flexibilidad.

También se desarrolló una interfaz gráfica simple para visualizar los resultados del sistema de registro y comprobar cualitativamente estos resultados. Este sistema también es independiente del navegador quirúrgico y sólo se implementó con fines experimentales. Los detalles más relevantes de implementación de estas herramientas se describen a continuación.

3.3.1. Implementación de ICP

El sistema de registro que se desarrolló en este trabajo de tesis consta de una clase principal: *Registration*. Esta clase incluye métodos para cargar los modelos de puntos, actualizar la transformación inicial y realizar el registro mediante el algoritmo ICP. El sistema está desarrollado en C++, por lo que además contiene una función *main*, que utiliza la clase *Registration* para registrar los modelos, como se muestra en la Figura 17.

```
Registration reg;
if (!reg.load(argv[1], argv[2]))
{
    for(int i=0; i<5; i++)
    {
        reg.setInitialTransform(t[i]);
        reg.ICP();
    }
}
else
    std::cerr << "Falló la lectura";
```

Figura 17. Segmento de código fuente que utiliza a la clase *Registration.*

El algoritmo de registro, ICP, se implementó basándose en las clases de ITK. La Figura 18 muestra un segmento de código del sistema de registro que utiliza las clases de ITK para efectuar el registro. En estas líneas se asignan los parámetros del objeto de registro y se ejecuta el mismo. Los parámetros incluyen la transformación inicial, la métrica, el optimizador y los conjuntos de puntos. Después de ejecutar este código, el objeto *transform* tiene los valores con la transformación obtenida por el algoritmo ICP.

```
TransformType::VnlQuaternionType rot(trans0[0], trans0[1], trans0[2], trans0[3]);
transform->SetRotation(rot);
transform->SetOffset(off);
std::cout << "Inicial = " << transform->GetParameters() << std::endl;

registration->SetInitialTransformParameters( transform->GetParameters() );

registration->SetMetric(        metric        );
registration->SetOptimizer(     optimizer     );
registration->SetTransform(     transform     );
registration->SetFixedPointSet( fixedPointSet );
registration->SetMovingPointSet(   movingPointSet   );

try
{  registration->StartRegistration();  }
```

Figura 18. Segmento de código fuente de registro.

3.3.2. Interfaz gráfica para revisar resultados

Con el fin de revisar los resultados del proceso de registro, se desarrolló una interfaz gráfica basada en OpenGL[1]. La interfaz permite visualizar los puntos del modelo de forma y del modelo de datos después de aplicar el registro. Con este pequeño sistema se pueden comprobar cualitativamente los resultados del algoritmo. La Figura 19 muestra una vista de dos ejemplos de registro en este programa. Para este ejemplo se utilizó como modelo de datos (verde) el mismo modelo de forma (azul) con ruido aleatorio con distribución lineal en los 3 ejes. El modelo de forma es un conjunto artificial de puntos que representa una aproximación a un modelo de hueso y tumor adyacente.

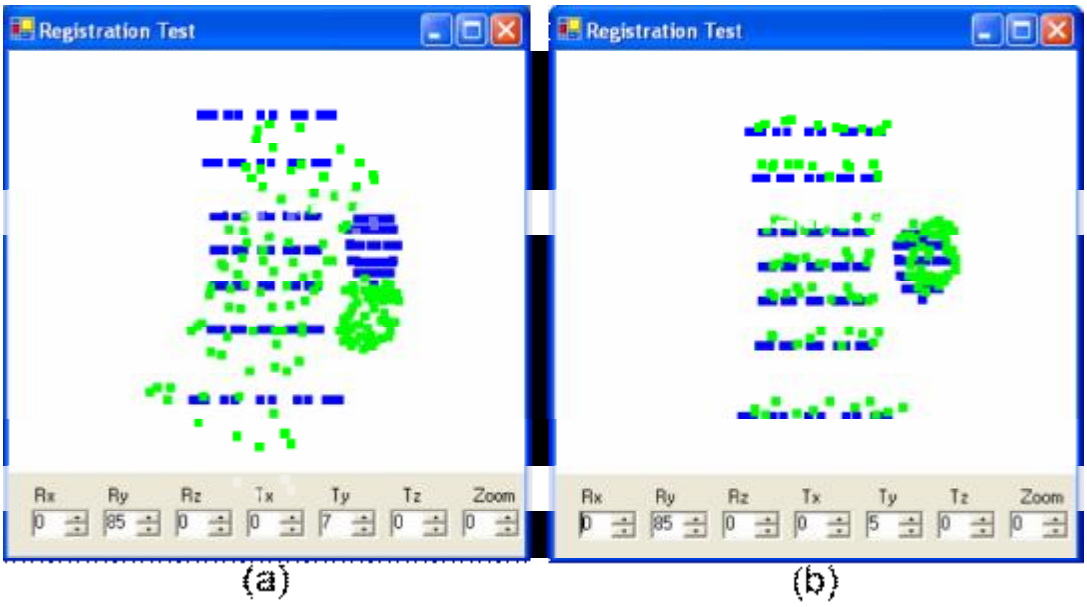

Figura 19. Interfaz gráfica mostrando (a) un mal registro y (b) un buen registro.

El sistema se desarrolló en el lenguaje C#[2], utilizando el marco de trabajo Tao[3] para la visualización con OpenGL. Se implementaron dos clases:

[1] OpenGL, http://www.opengl.org

[2] Microsoft. C#, http://msdn.microsoft.com/csharp

[3] Mono Project. Tao Framework, http://www.taoframework.com

a. *QuatPointSet.* Clase responsable de la lógica del programa, se encarga de leer los archivos de datos con los modelos, realizar las transformaciones definidas en el registro y desplegarlas en pantalla.

b. *MainFrame.* Clase que representa la interfaz gráfica de usuario. Se encarga de administrar la ventana principal, la vista y la interacción con el usuario. Incluye controles para mover el punto de vista, de manera que se pueda analizar los resultados desde distintos ángulos, aprovechando la naturaleza 3D de los datos.

3.4 Prototipo del navegador quirúrgico

El sistema más importante que se desarrolló durante este proyecto es un prototipo del navegador quirúrgico para ayudar a la colocación, guiada por imágenes de US, de una aguja de RF. Este sistema aprovecha diversas aplicaciones y herramientas del marco de trabajo CAMPAR **Error! Reference source not found.**, como se detalla en la sección 3.1.1. La forma de utilizar el sistema es primero definir un objetivo quirúrgico (tumor) y después guiar la aguja de RF hacia él. Para esto es necesario conocer la posición de este objetivo y de la aguja en un sistema de coordenadas común, que, por simplicidad, es el definido por el sistema de rastreo.

La posición de la aguja se define a partir de la posición, en coordenadas del mundo, de su punta, ${}^{W}\boldsymbol{P}$, y de su base, ${}^{W}\boldsymbol{B}$. El sistema de rastreo nos da la transformación ${}^{W}\boldsymbol{T}_{N}$, de la configuración de marcadores fijados a la aguja $\boldsymbol{F}_{N}$, al mundo $\boldsymbol{F}_{W}$. Así, lo que falta conocer es la posición ${}^{N}\boldsymbol{P}$ y ${}^{N}\boldsymbol{B}$, de la punta y de la base, respectivamente, en coordenadas de la configuración de marcadores. En la Figura 20 se muestra gráficamente el sistema de coordenadas de la aguja. Es importante notar que el origen de este sistema no siempre se conoce ni coincide necesariamente con el centro de un marcador. La calibración de la aguja, descrita en la sección 3.4.1, es el proceso para obtener estos valores. La Figura 23 muestra los mismos puntos de interés, tanto en el sistema de coordenadas de la configuración de marcadores en la aguja como en el sistema dado por el sistema de rastreo.

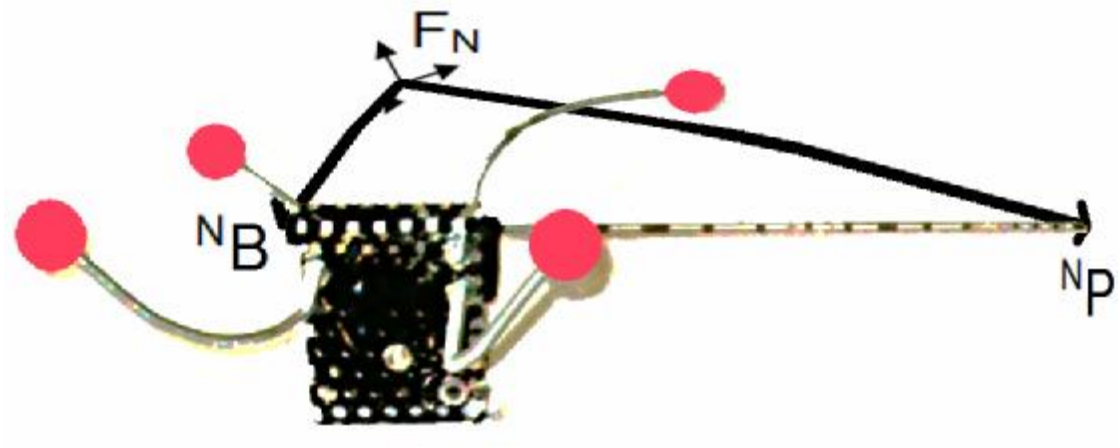

Figura 20. Sistema de coordenadas (F_N) de la aguja de RF con la punta (^NP) y la base (^NB).

Para definir el objetivo quirúrgico se utilizan imágenes de US. La idea es definir un objetivo esférico con un *clic*. Para tener en coordenadas del mundo el objetivo definido es necesario rastrear la sonda de US y tener la transformación del sistema de coordenadas de la imagen (F_U) de US al sistema de coordenadas de la configuración de marcadores de la sonda (F_P), ${}^PT_U$. Esta transformación se define en el proceso de calibración de US **Error! Reference source not found.**, que debe ser realizado previamente. En la Figura 21 se muestra un diagrama de la sonda de US con una configuración de marcadores. También se muestra el origen de los dos sistemas de coordenadas relevantes: F_U (imagen de US) y F_P (configuración de marcadores fija a la sonda). La ecuación 3.1 muestra la concatenación de transformaciones necesaria para calcular la posición de un punto en la imagen de US en coordenadas globales (F_W) definido por el sistema de rastreo.

$$^WT_U = {}^WT_P \, {}^PT_U \tag{3.1}$$

donde ${}^WT_U$ es la matriz de transformación del sistema de coordenadas de la imagen de US F_U al sistema de coordenadas global F_W; ${}^WT_P$, la matriz de transformación de la sonda al sistema global, y ${}^PT_U$, de la imagen a la sonda.

El navegador incluye un modo de visualización con las imágenes de US en tiempo real, que ayuda a la definición del objetivo quirúrgico.

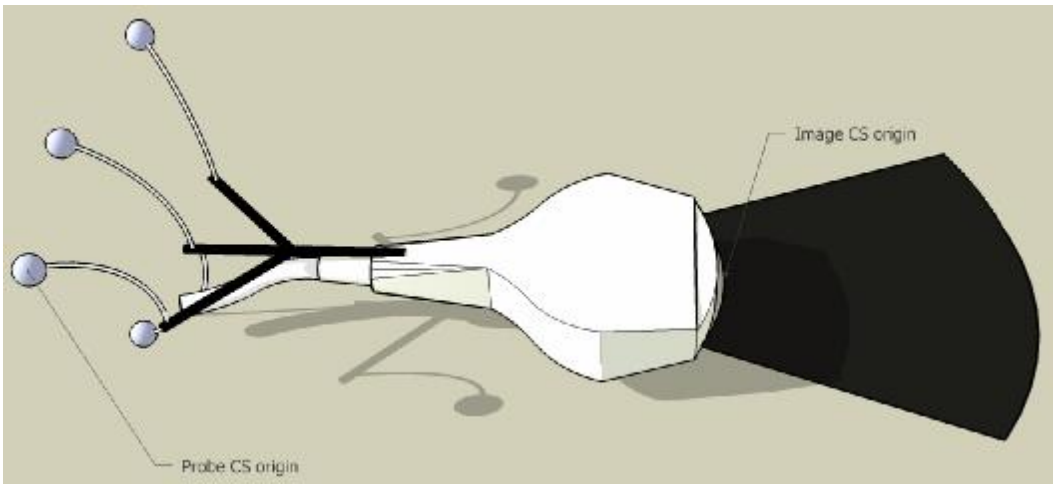

Figura 21. Sonda de US rastreada con puntos de interés. Error! Reference source not found.

Finalmente, para la colocación de la aguja, se desarrollaron 3 modos de visualización del ambiente virtual creado, que incluye el modelo 3D de la aguja, la imagen de US y el objetivo. Estos elementos comparten el sistema de coordenadas definido por el sistema de rastreo, de forma que pueden interactuar entre sí.

Este sistema se desarrolló en C++, ya que está basado en el marco de trabajo CAMPAR. El navegador quirúrgico consta de 3 componentes: Modelo, Vista y Controlador (MVC). Se desarrollaron 8 clases:

a. *NeedlePlacementOutput*. Clase que hereda la funcionalidad de la clase *Device*, de CAMPAR, por lo que es el punto de entrada al sistema. Esta clase se encarga de inicializar los elementos de la interfaz gráfica de usuario, basada en Qt. Como esta clase pertenece a la Vista, no realiza gran parte de la lógica del sistema, por lo que la mayor parte de los mensajes que le llegan pasan directamente al Controlador.

b. *GraphicViewGLWidget*. Esta clase se encarga de desplegar el mundo virtual usando OpenGL. Hereda funcionalidad de *QGLWidget*, por lo que tiene la capacidad de integrarse en la interfaz gráfica basada en Qt y de desplegar objetos 3D con OpenGL. Tiene 3 modos de visualización: WORLD_VIEW, NEEDLE_VIEW y CAMERA_VIEW, que permiten cambiar al modo interactivo, de vista de la aguja y de cámara aumentada, respectivamente.

c. *NeedlePlacementVideo*. Clase responsable de desplegar el video de US. Hereda de *QGLWidget*, por lo que se incorpora en la interfaz de usuario y despliega contenido de OpenGL. Esta clase también se encarga de definir el objetivo quirúrgico cuando el usuario hace *clic* en algún punto de la imagen de US.

d. *NeedlePlacementController*. En esta clase se encuentra la mayor parte de la lógica del sistema. Además, funciona como medio de comunicación entre los demás componentes. Este controlador le da servicios a las demás clases, tales como la obtención de datos de rastreo o de imágenes de video.

e. *NeedleModel*. Esta clase define el modelo de la aguja, que está representada únicamente por un punto en la base y otro en la punta. Además, está relacionada con un objetivo quirúrgico.

f. *NeedleTarget*. Representa el objetivo quirúrgico y es parte del modelo de datos del sistema. Es interesante notar que es una clase abstracta, por lo que fácilmente se puede extender para incluir objetivos más interesantes. Actualmente están implementados dos tipos de objetivos: *MeshNeedleTarget*, que lee una serie de triángulos de un archivo, y *SphereNeedleTarget*, que representa un objetivo simple esférico y es instanciada a partir del algoritmo de segmentación implementado en *NeedlePlacementVideo*.

El diagrama en la Figura 22 muestra la organización de estas clases.

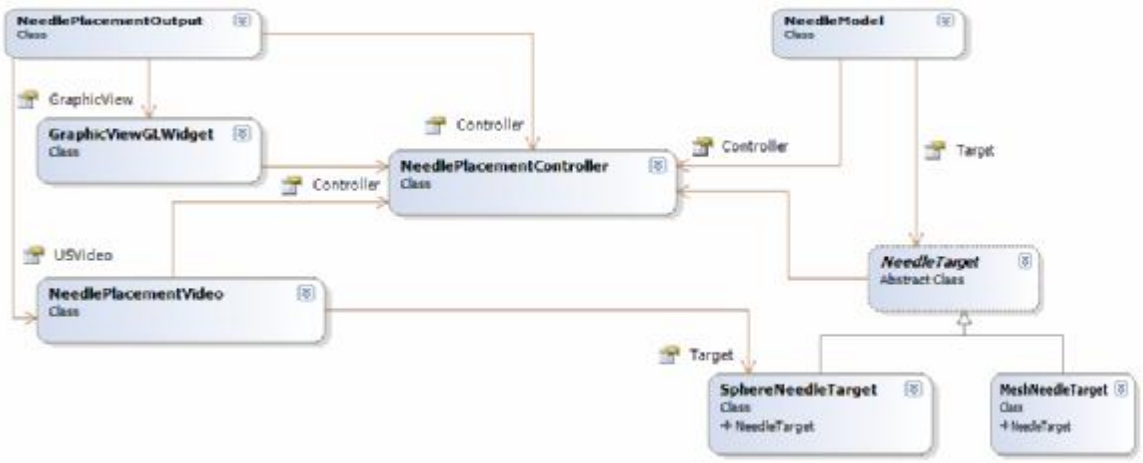

Figura 22. Diagrama de clases del navegador quirúrgico.

3.4.1. Calibración de la aguja

El objetivo de la calibración es encontrar las coordenadas de la punta y la base de la aguja en el sistema definido por la configuración de marcadores fija a la aguja. Así, lo que se busca son dos vectores de tres valores. Estos vectores pueden ser ingresados en el archivo de configuración del sistema cuando se tiene la calibración antes de comenzar a usar el sistema. Como la aguja debe estar esterilizada, es deseable que la calibración se realice durante la intervención o poco tiempo antes. Por lo tanto, es necesario que el proceso sea rápido. Una alternativa es utilizar agujas precalibradas, que incluyan los marcadores de rastreo desde su manufactura.

La posición de la base de la aguja se necesita para poder calcular la longitud y la dirección de ésta. Sin embargo, lo que nos interesa conocer con exactitud es la punta, ya que es la parte que contiene el electrodo con el que se aplica la terapia de RF. Esto se tomó en cuenta al elegir el método de calibración para cada parte. En el navegador se implementó un método sencillo de calibración por *punto conocido*. Esto es, se coloca la parte de interés de la aguja en un punto conocido y se resuelve el sistema de ecuaciones

$$^{W}P = {}^{W}T_{N}\,{}^{N}P\,, \tag{3.2}$$

donde ^{W}P es un punto conocido, que coincide con la punta de la aguja, en coordenadas globales, $^{W}T_{N}$ es la transformación del sistema de coordenadas de la configuración de

marcadores al sistema del mundo, dada por el sistema de rastreo, y ${}^{N}P$ es el punto en coordenadas de la configuración, que es el que se busca con la calibración. Estos puntos se ilustran en la Figura 23. Al despejar ${}^{N}P$, se obtiene la ecuación

$$ {}^{N}P = {}^{W}T_{N}^{-1}\,{}^{W}P\,, \tag{3.3}$$

que obtiene el vector de calibración para un punto, donde ${}^{W}T_{N}^{-1}$ es la inversa de la matriz de transformación del sistema de coordenadas de la aguja al global. Este método se implementó en la clase *NeedlePlacementController*, como se muestra en la Figura 24 (sólo se muestran las líneas de código de interés). En estas líneas de código se obtiene la posición de la aguja (${}^{W}F_{N}$) y el punto conocido en coordenadas globales (${}^{W}P$), después se calcula la diferencia para obtener el vector del origen del sistema de coordenadas de la configuración de marcadores de la aguja a la punta de ésta. Por último, se invierte la matriz de rotación dada por el sistema de rastreo y se premultiplica por el vector de diferencia. El resultado es la posición de la punta de la aguja en coordenadas de su objetivo de rastreo. La Figura 25 muestra la posición de la aguja durante la ejecución de este método. La calibración por *punto conocido* ha probado ser confiable en los experimentos realizados, pero no tan exacta como requiere la punta para nuestra aplicación (±1 mm), que es el punto de interés de la aguja. Esta falta de precisión se debe a que no se conoce ningún punto con la exactitud necesaria en el sistema de coordenadas del seguidor. Además, un pequeño error en la posición o en el rastreo se propaga y crece al mover la aguja. Los experimentos se realizaron con un marcador plano de 15 mm de diámetro. Sabemos que el sistema toma como origen el centro del marcador, pero en principio no conocemos este centro con la exactitud adecuada para colocar la punta de la aguja en él. En un sistema comercial sería suficiente con tener algún elemento mecánico que guíe a la aguja a la posición correcta en los últimos milímetros, de forma que se asegure que la punta de ésta está en el centro del marcador.

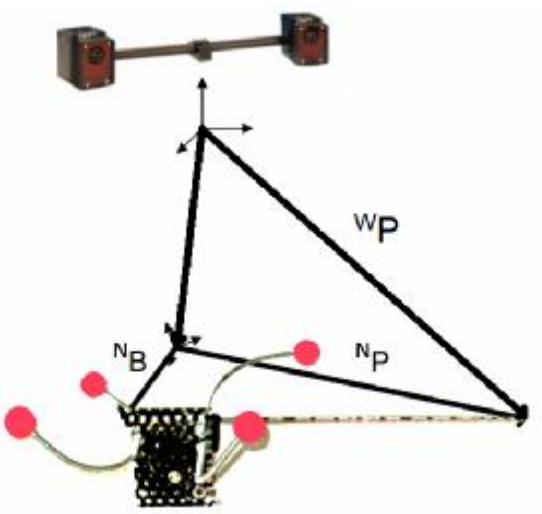

Figura 23. Puntos de interés (*P* y *B*) en la aguja en su sistema de coordenadas (*N*) y el del mundo (*W*).

```
TrackingData spotData = m_needleTrackerDevice->getTrackingData(m_trackingIDs[NP_SPOT_TRACKING]);
TrackingData needleData = m_needleTrackerDevice->getTrackingData(m_trackingIDs[NP_NEEDLE_TRACKING]);
CAMP::Vector3<double> dif1 = spotData.getTranslation() - needleData.getTranslation();
CAMP::Matrix3<double> mat1 = needleData.getRotationMatrix().invert();
CAMP::Vector3<double> res = mat1*dif1;
m_model->tip(res);
```

Figura 24. Código fuente para la calibración de la punta de la aguja de RF.

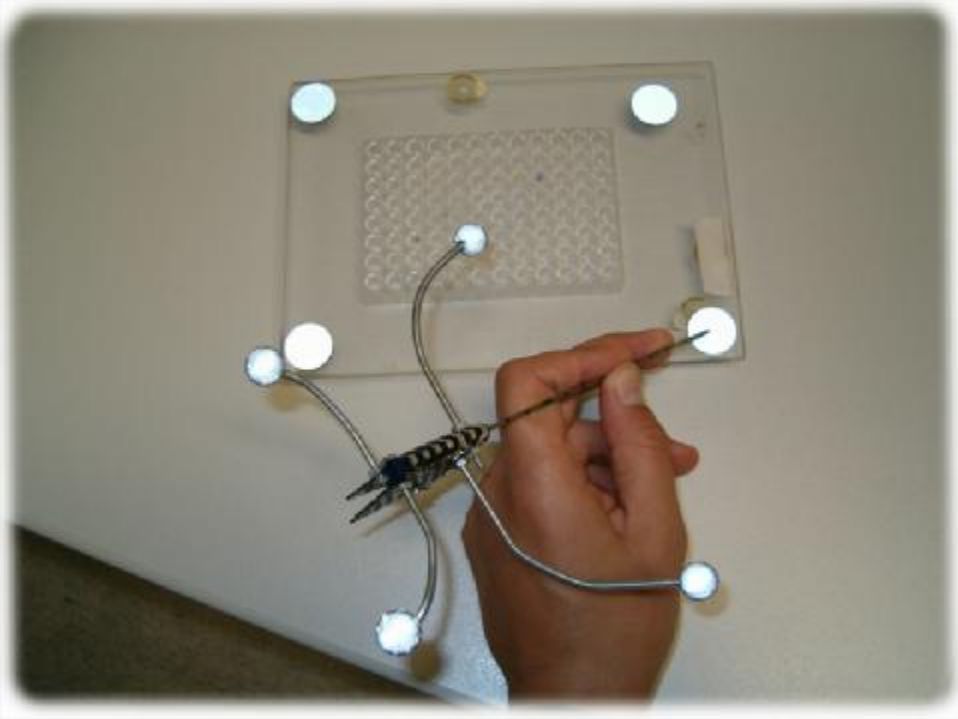

Figura 25. Sistema de calibración por *punto conocido*.

Para obtener la precisión deseada, se eligió la aplicación *HotSpotCalibration*, incluida en CAMPAR. Esta aplicación requiere que se mueva el objeto rastreado alrededor de un punto inmóvil, por lo que es ideal para calibrar la punta de una aguja. Sin embargo, su base (o extremo posterior) tiene un área relativamente grande (5 mm de diámetro), por lo que es difícil mover la aguja sin mover un punto en el centro de ésta. Como en este caso se requiere menor precisión, el método de *punto conocido* funciona suficientemente bien.

3.4.2. Visualización del ultrasonido y definición del objetivo quirúrgico

La función principal del US es localizar el tumor, que será el objetivo quirúrgico en el que se pretende colocar la punta de la aguja de RF. Para esto se desarrolló un protocolo que permite congelar la imagen de US y definir un objetivo esférico. El video se captura con una tarjeta de entrada de video analógico y se despliega en la pantalla principal de la interfaz gráfica como se muestra en la Figura 26. La figura además muestra los demás elementos de la interfaz, que incluyen los controles para seleccionar la vista y ejecutar acciones básicas, del lado izquierdo, y dos paneles informativos en la parte inferior.

Figura 26. Interfaz gráfica del prototipo de navegador quirúrgico con vista de US.

La forma de definir el objetivo quirúrgico es marcar una imagen de US. Para esto se usa el *mouse*. Al hacer *clic* sobre la imagen se congela el video, se almacena la posición de la sonda de US y se ejecuta un algoritmo simple para obtener un círculo. Este algoritmo se usa de manera temporal para efectos experimentales. En etapas posteriores del proyecto se incorporarán los métodos diseñados para esta tarea en el proyecto paralelo **Error! Reference source not found.**. Actualmente se usa una versión modificada del algoritmo de *crecimiento de regiones* **Error! Reference source not found.**. Recibe una posición de la imagen como semilla y un valor de tolerancia, y regresa un centro y un radio para un círculo. El centro es sencillamente la posición semilla. Para determinar el radio, primero se aplica un filtro gaussiano que reduce el ruido y después se ejecutan una serie de iteraciones, comenzando con radio de un píxel. En cada iteración se agregan los píxeles cuya distancia al centro sea menor que el radio, después se calcula la diferencia entre el promedio de intensidad de todos los píxeles en el círculo y el promedio de intensidad de los píxeles recién

agregados. Si este valor es mayor al umbral de tolerancia, definido por el usuario, se termina el algoritmo. En caso contrario, se incrementa el radio y se repite el proceso. El centro y radio devueltos por el algoritmo se usan para definir una esfera. La principal ventaja de este algoritmo es que es muy tolerante al ruido. La desventaja es que depende completamente de la semilla que se ingrese manualmente. Este algoritmo podría aplicarse directamente sobre US-3D, de manera que en lugar de buscar un círculo centrado en la posición marcada, buscara una esfera. Utilizando la transformación definida por la calibración del US y la transformación dada por el sistema de rastreo en el momento del *clic*, se establece la posición 3D de la esfera, que representa el objetivo quirúrgico. La Figura 27 muestra el segmento de código que se ejecuta después del algoritmo de segmentación para definir el objetivo quirúrgico.

```
// Create a SphereTarget at the center of the click.
SphereNeedleTarget *target = new SphereNeedleTarget(m_controller, r*m_scale[0]);
Matrix4<double> calibration;
CAMP::CroppingArea crop = m_controller->getUSCroppingArea();
if (m_cameraDevice->getType().compare("AviForOSDevice") == 0)
    y = ((this->height() * h) / hh) - y;
x -= crop.left(); y -= crop.top();
CAMP::Vector4<double> point(x*m_scale[0], y*m_scale[1], 0.0, 1.0);
CAMP::Vector4<double> newPoint = m_trackingTransform * m_transformation * point;
CAMP::Vector3<double> finalPoint(newPoint[0], newPoint[1], newPoint[2]);
target->setPosition(finalPoint);
m_controller->setTarget((NeedlePlacementTarget*)target);
```

Figura 27. Segmento de código fuente que define el objetivo quirúrgico.

3.4.3. Visualización del mundo

El navegador incluye tres vistas del ambiente virtual, que ayudan a dirigir al médico en la colocación de la aguja. En los tres modos de visualización se ve una aguja virtual y el objetivo quirúrgico, representado por una esfera o una malla, según sea el caso. La aguja virtual, como la real, tiene marcas cada centímetro y cada 5 cm. Además, hay la opción de mostrar la imagen de US en su posición con respecto a los demás elementos, los ejes del sistema de coordenadas global y planos traslúcidos que representan las coordenadas *X=0, Y=0, y Z=0*, respectivamente. A continuación se describen los modos de visualización:

a. *Vista de mundo interactiva*. En este modo de visualización se puede ver el mundo desde cualquier ángulo deseado. La vista se puede manipular usando el *mouse* y sus 3 botones para hacer las transformaciones básicas de rotación, traslación y cambio de escala. La Figura 28 muestra una imagen del mundo virtual desde un punto de vista arbitrario que muestra la aguja de RF, el plano de US, el objetivo quirúrgico esférico, los ejes del sistema de coordenadas y los planos translúcidos.

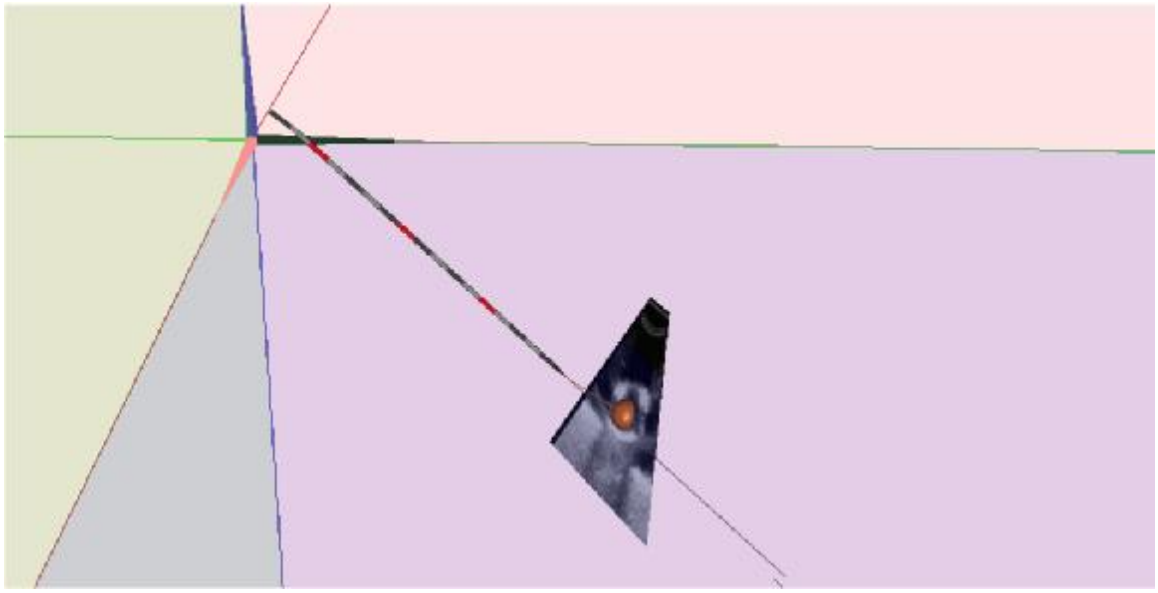

Figura 28. Vista de mundo interactiva.

b. *Vista de aguja*. Este modo mueve la vista de manera que siempre se ve el mundo desde un punto sobre la aguja y en dirección a la punta de ésta. Este es el modo que mejor permite la aproximación al objetivo quirúrgico. La Figura 29 muestra la vista de la aguja antes de la inserción.

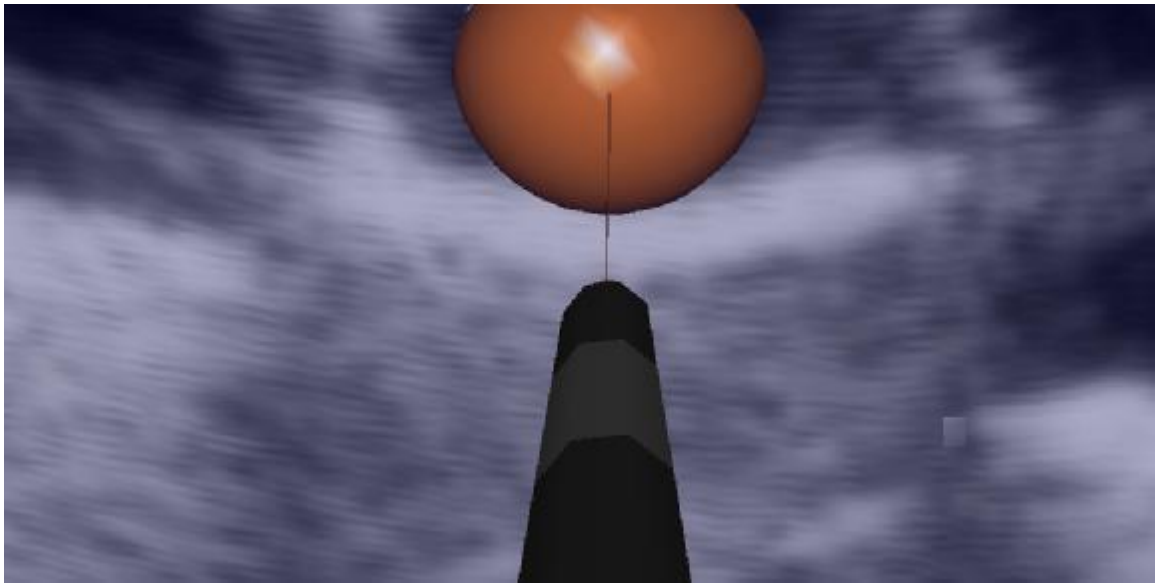

Figura 29. Vista de aguja.

c. *Cámara aumentada*. En este modo de visualización, la vista se acomoda a la posición de una cámara de video, que se encuentra calibrada y rastreada. La imagen de video se despliega como fondo y los elementos virtuales se despliegan sobre éste. Así, los elementos del mundo virtual se funden con los objetos reales en la vista de la cámara. La Figura 30 muestra una imagen de la cámara con elementos virtuales sobrepuestos, incluyendo la aguja de RF, la imagen de US y el objetivo quirúrgico. En esta figura se nota un error en la calibración del US, ya que la imagen de US muestra la aguja en una posición incorrecta.

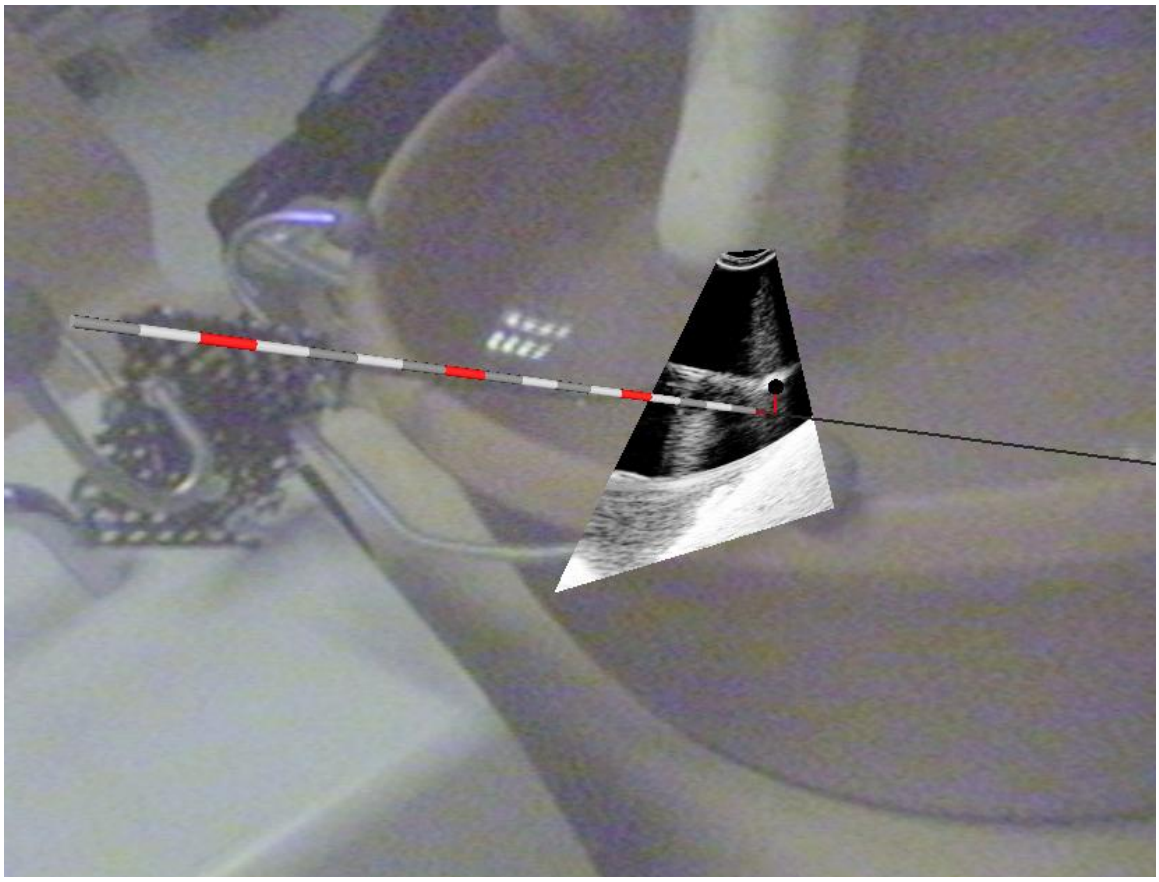

Figura 30. Vista de cámara aumentada.

Para guiar al cirujano durante la intervención se agregaron dos líneas: una va desde la punta de la aguja hasta el centro del objetivo quirúrgico y la otra sale de la punta y sigue en la dirección de la aguja. Así, cuando las dos líneas coinciden, la aguja está apuntando al centro del tumor, como se muestra en la Figura 29. Además, para ayudar a visualizar la profundidad en la pantalla, se define el color del objetivo en función de la distancia de éste a la punta de la aguja. El color verde significa que la aguja está a más de 100 mm. Mientras la aguja se acerca, el color se va volviendo rojo, de manera que cuando los puntos coinciden el objetivo es completamente rojo. La Figura 31 muestra el segmento de código en la clase *NeedleModel*, que define el color del objetivo quirúrgico.

```
CAMP::Vector3<double> targetPos = m_target->getPosition();
CAMP::Vector3<double> tipPos = tip();
float norm = (float)(tipPos - targetPos).norm();
float red = (norm > 100.0)? 0.0f : 1.0f - 0.01f * norm;
float green = (norm > 100.0)? 1.0f : 0.01f * norm;
glColor3f(red, green, 0.1f);
```

Figura 31. Segmento de código fuente que define el color del objetivo quirúrgico.

3.5 Integración del seguidor óptico Polaris a CAMPAR

El marco de trabajo CAMPAR **Error! Reference source not found.** fue diseñado para trabajar de la misma manera con diferentes dispositivos. En particular, el sistema de seguimiento *Polaris*, de *Northern Digital Inc.* (NDI) **Error! Reference source not found.** nunca fue probado por los desarrolladores originales. Una parte importante de este proyecto fue adaptar CAMPAR para trabajar con este sistema en el Laboratorio de Análisis de Imágenes y Visualización del CCADET, UNAM.

CAMPAR incluye las clases *NDITrackerDevice* y *NDICommandHandling*, con las que maneja todos los sistemas de seguimiento de NDI. Estas clases habían sido probadas y adaptadas para sistemas *Aurora* y *Vicra*, que funcionan de manera muy similar al sistema *Polaris*. De esta manera, para lograr que el prototipo de navegador quirúrgico funcione con el seguidor *Polaris* sólo hizo falta modificar ligeramente estas dos clases para manejar las particularidades del seguidor. Así, después de agregar el código correspondiente y configurar la aplicación mediante el archivo XML de entrada, el navegador funcionó perfectamente, tanto con marcadores pasivos inalámbricos como activos conectados a la Unidad de Interfaz de Herramientas de *Polaris.*

3.6 Discusión

El prototipo de navegador que se implementó utiliza una variedad de tecnologías que, trabajando juntas, tienen un gran potencial. El sistema CAMPAR funcionó como base para el desarrollo principal y ayudó en gran medida a la interacción con los diversos dispositivos de *hardware* que se utilizaron. ITK mostró

que tiene una gran cantidad de herramientas de análisis de imágenes y registro, pero requiere de mucha experiencia para aprovechar muchas de estas herramientas.

El mayor esfuerzo de este trabajo de tesis se concentró en el desarrollo del navegador quirúrgico, que proporciona una mejora clara al proceso de ablación de tumores con RF al dividir la tarea en dos etapas: localización del tumor e inserción de la aguja. Los 4 modos de visualización están diseñados para ayudar en las diferentes etapas y le dan al sistema una mayor flexibilidad.

El prototipo de navegador aún necesita ser afinado y probado antes de poder ser utilizado en la práctica. También es una buena base para implementar funciones más complejas de segmentación y registro.

4. Experimentos y Resultados

En este proyecto se realizaron dos tipos de experimentos diferentes. Por un lado, se estudió un caso real, del que se cuenta con un estudio completo de MRI y de US. Estas imágenes se utilizaron para la construcción del modelo y el registro. Por el otro lado, se realizaron experimentos con el navegador quirúrgico que se implementó. El navegador no se probó con pacientes por falta de desarrollo y pruebas. En particular, falta integrar la funcionalidad de segmentación, que está siendo desarrollada en un proyecto paralelo **Error! Reference source not found.** y la parte de registro, que sin ésta no tiene sentido. Así, los experimentos con el navegador se realizaron con maniquíes y tienen el propósito de probar usabilidad y exactitud.

4.1 Caso de estudio

El estudio de MRI y US que utilizamos se tomó de la mano de una paciente de 10 años de edad. Las imágenes muestran un tumor adyacente al hueso en el dedo pulgar de la mano derecha, como se muestra en la Figura 32.

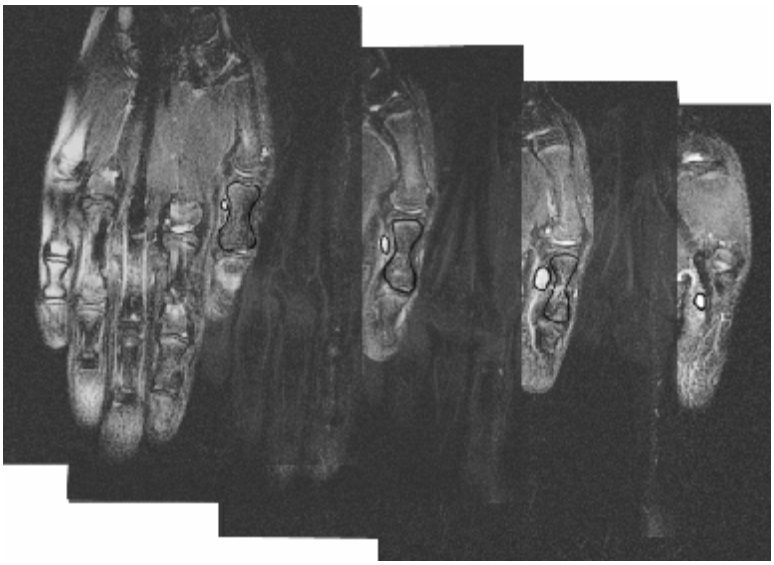

Figura 32.Imágenes anotadas de MRI que muestran un tumor y el hueso adyacente.

La Figura 33 muestra una serie de imágenes longitudinales adquirida con US que muestra el mismo tumor y la superficie del hueso. En las imágenes de US no es posible ver el contorno completo del hueso, ya que las ondas se reflejan casi completamente, de forma que nunca llegan al otro lado y la sonda no percibe ningún eco. En esta imagen también se puede apreciar la gran cantidad de ruido y la mala definición de los bordes. Sin embargo, este tipo de imágenes sí puede ser suficiente para ubicar la posición del tumor, como muestra la Figura 34 por lo que se puede utilizar como base para guiar al cirujano en el posicionamiento de la aguja de RF.

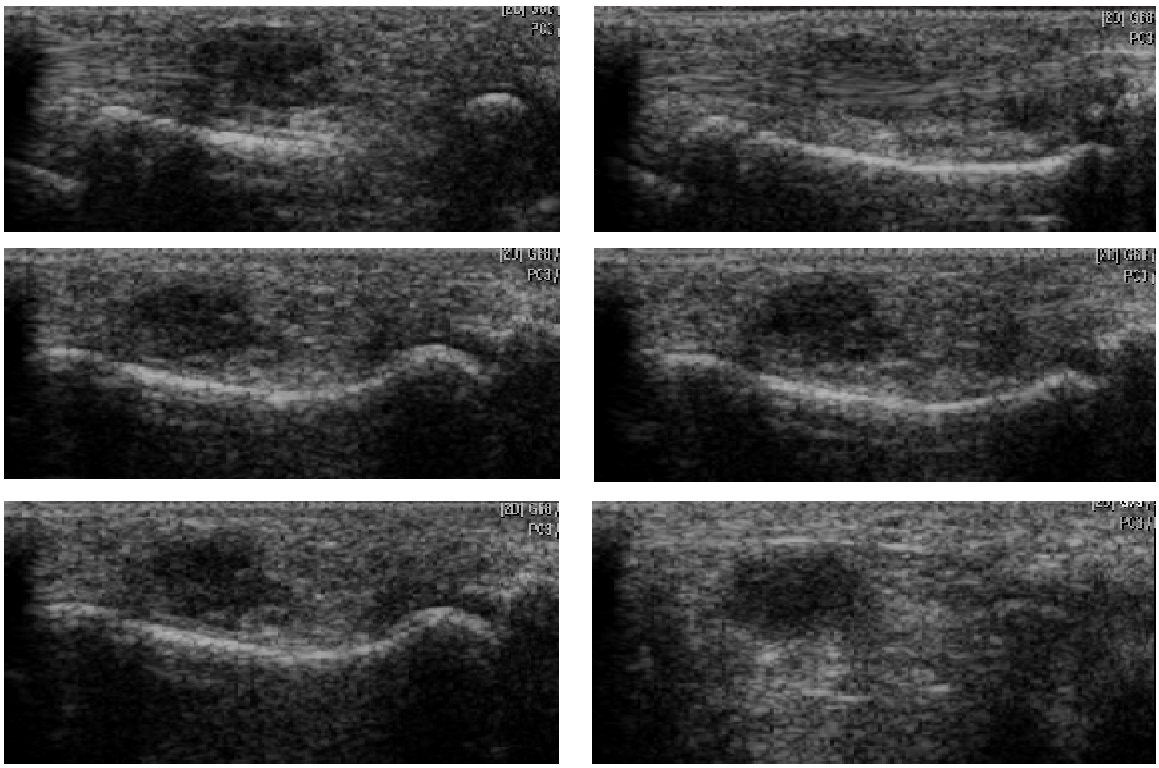

Figura 33.Serie de imágenes de US que muestra el mismo tumor y hueso del estudio de MRI.

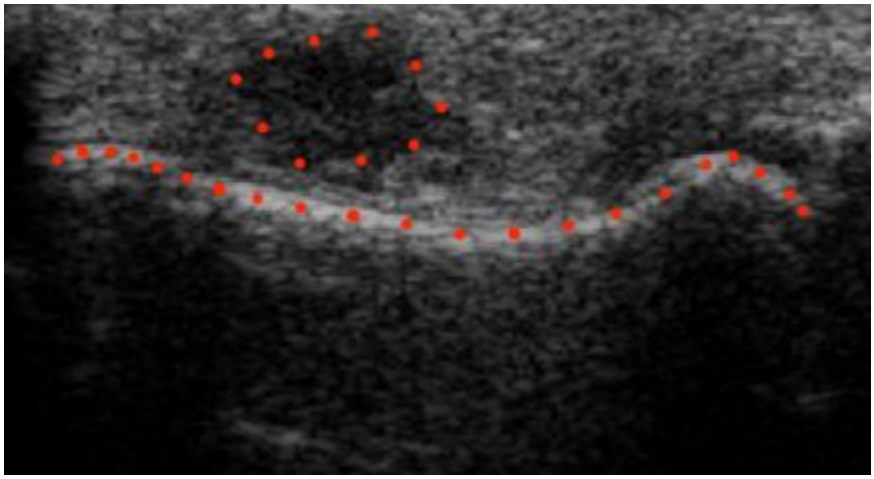

Figura 34.Imagen de US con perfil de hueso de pulgar y tumor marcados.

4.2 Experimentos

4.2.1. Construcción del modelo usando las imágenes de MRI

El objetivo de este experimento es probar el sistema de mallado para nuestro caso de estudio, así como la visualización del modelo en el navegador quirúrgico.

La metodología usada para la construcción del modelo se resume en los siguientes pasos.

a. *Segmentación manual*. Se segmentaron el tumor y el hueso más próximo, presentes en las imágenes de MRI estudiadas. Para esto se utilizó *ImageJ* **Error! Reference source not found.**. Las imágenes se guardaron en archivos de texto y posteriormente, usando un editor de texto común, se concatenaron, de manera que todas las imágenes quedaron en un mismo archivo.

b. *Generación de la malla.* Se alimentó el archivo obtenido en la etapa anterior al programa *mcubes 3.5* **Error! Reference source not found.**. El programa fue configurado con las dimensiones de voxel del volumen, que para el caso de estudio son de 0.375x0.375x3.0 mm^3. Se obtuvieron dos archivos de texto con los vértices y los triángulos respectivamente.

c. *Despliegue del modelo*. Utilizando la clase *MeshNeedleTarget*, desarrollada como parte del navegador quirúrgico, se leyeron los archivos con la malla y se guardaron en la memoria con estructuras de datos adecuadas. La misma clase se encarga de desplegar la malla en el navegador, de manera que es posible interactuar con los diferentes modos de visualización. Este experimento no incluyó registro, por lo que el modelo se ubica en una posición arbitraria y no representa un objeto real en el ambiente virtual.

Se obtuvo un modelo 3D como se muestra en la **Error! Reference source not found.**. Por las características del caso de estudio, el modelo obtenido no tiene la calidad esperada. Primero, es importante notar la gran diferencia que existe entre la resolución espacial en el plano XY y en el eje Z, que es 8 veces menor. Debido a esto, el tumor estudiado sólo aparece en 4 imágenes y el hueso adyacente, sólo en 3 de estas imágenes. Esto se refleja en el modelo como una falta importante de calidad.

4.2.2. Construcción del modelo con imágenes interpoladas

Con el objetivo de obtener un modelo más suave, se realizó el mismo experimento, agregando un paso de sobremuestreo después de la segmentación. Con esto se busca tener una resolución similar en los 3 ejes cartesianos, lo que puede ayudar al algoritmo *marching cubes* a generar una malla que represente mejor al objeto de estudio. Así, se probó aumentar el número de cortes, de 4 a 7, agregando cortes intermedios entre cada par de imágenes.

La interpolación se realizó con un algoritmo simple basado en campos de distancia y posteriormente se realizó el mallado. Los siguientes pasos resumen la metodología utilizada en este experimento:

a. *Calcular el campo de distancia*. En este paso se utilizó *ImageJ* **Error! Reference source not found.** para calcular el campo de distancia en las 4 imágenes segmentadas por separado.

b. *Promediar campos de distancia*. Se promediaron los campos de distancia de imágenes adyacentes, de manera que se obtuvieron 3 muevas imágenes, cada una con el promedio de dos campos de distancia.

c. *Aplicar umbral*. Se aplicó un umbral de valor 1.0 a las 3 imágenes obtenidas en el paso anterior, de manera que todos los píxeles de intensidad menor a este valor se quedaron con valor 255 (objeto) y los demás píxeles, con valor 0 (fondo).

d. *Mallado*. Se aplicó la metodología del experimento 4.2.1 para obtener una malla con el nuevo conjunto de imágenes, formado por las originales y las obtenidas mediante interpolación.

El modelo obtenido es más suave que el modelo original, aunque aún se nota la gran separación entre los cortes. El número de triángulos en la malla obtenida se incrementó de 12,328 a 13,616. Este incremento afecta el desempeño del sistema, pero es relativamente bajo, ya que el sistema de mallado realiza una etapa de optimización en la que elimina triángulos innecesarios. La Figura 35 muestra el modelo 3D obtenido en el experimento anterior y el obtenido después de agregar los cortes interpolados.

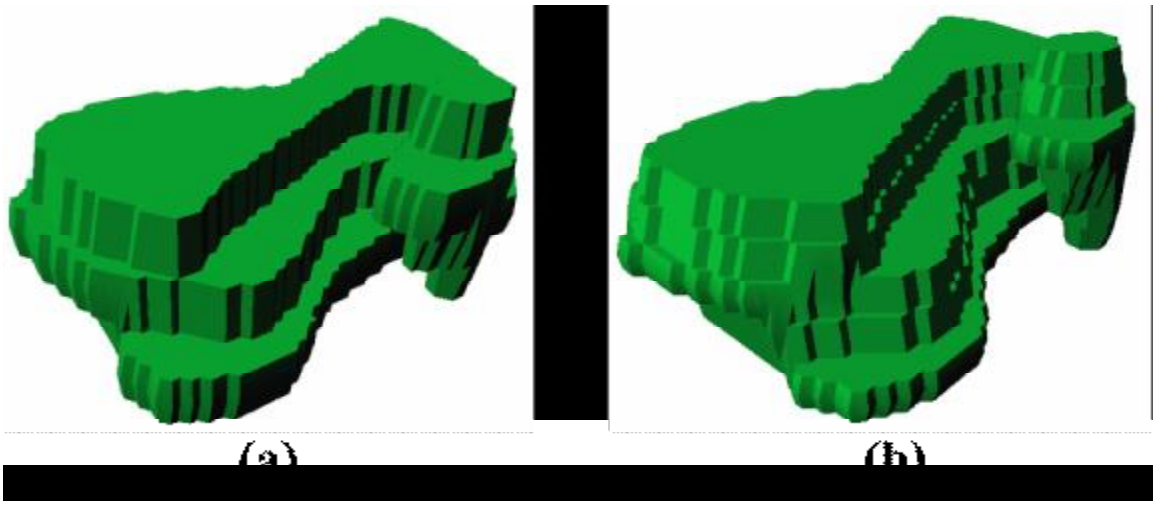

Figura 35.Modelos 3D (a) sin y (b) con cortes interpolados.

Este experimento probó cualitativamente que utilizar un método de sobremuestreo puede ayudar a generar mallas 3D más suaves. Sin embargo, el método utilizado requiere más atención. Una opción que podría mejorar los resultados de este método es hacer

variable el valor del umbral, que se mantuvo constante para cada corte. Sin embargo, la forma de asignar este valor no es un problema trivial y supera los alcances de este trabajo. Otra alternativa es interpolar nuevos puntos sobre el modelo en lugar de hacerlo sobre las imágenes segmentadas.

4.2.3. Registro de modelos artificiales

Para probar el sistema de registro, se utilizó un modelo de hueso y tumor adyacente artificial. Como se mencionó, el caso de estudio no incluye datos de localización espacial de las imágenes de US, por lo que no es posible reconstruir un volumen, como en el caso de las imágenes de MRI. Así, para realizar el registro, se usó un modelo generado artificialmente, que se registró con otro modelo derivado de éste, construido agregando ruido aditivo aleatorio al original.

El objetivo de este experimento es probar el algoritmo de registro ICP en modelos de puntos 3D que no coinciden perfectamente.

Para realizar este experimento se siguieron los siguientes pasos:

a. *Construcción del modelo de forma*. Construir un modelo 3D artificial con formas bien definidas. Para esto, se usó un programa comercial de modelado 3D.

b. *Construcción del modelo de datos.* Se copiaron los puntos del modelo de forma en una hoja de cálculo y se agregó un valor aleatorio en los 3 ejes a cada punto.

c. *Elección de la transformación inicial.* Por construcción, se sabe que la transformación óptima para este modelo es cercana a la identidad, representada por el vector (0, 0, 0, 1, 0, 0, 0). La transformación inicial, por tanto debe ser diferente a ésta, por lo que se eligió una que deforme levemente al modelo en sus 6 grados de libertad. Así, se utilizó la transformación inicial (0.05, 0.05, 0.05, 0.925, 0.1, 0.1, 0.1).

d. *Cálculo del registro.* Se usó el sistema de registro, que implementa ICP **Error! Reference source not found.**, para calcular el vector de 7 valores que define la transformación rígida necesaria para mover el modelo de datos al sistema de

coordenadas del modelo de forma. El sistema devuelve el vector de registro y un valor de error medio cuadrático.

e. *Comprobación de resultados.* Se usaron los modelos construidos y el vector de registro obtenido como entrada para el programa de visualización para observar los resultados. Con esto, se pretende evaluar cualitativamente la calidad del registro obtenido.

Después de ejecutar el algoritmo ICP se obtuvo el siguiente vector de resultados: (-0.001426, 0.005906, -0.003179, 0.9954, -0.10016, -0.09575, -0.09598), con error medio cuadrático de 0.1033. Este vector es claramente cercano al vector de transformación identidad, que se buscaba como meta, con un error medio cuadrático de 0.16875. Las pequeñas diferencias encontradas pueden ser causadas por el ruido inducido al modelo de datos, errores numéricos de la computadora o por la tolerancia del algoritmo.

La Figura 36 muestra gráficamente los resultados del registro. En ésta, los puntos azules representan el modelo de forma, los puntos rojos, el modelo de datos con la transformación inicial, y los verdes, el mismo modelo con la transformación obtenida mediante ICP.

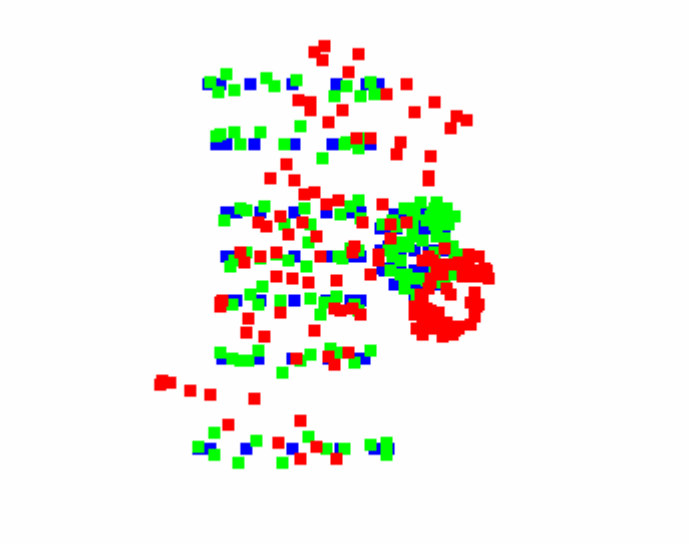

Figura 36.Registro de modelo 3D artificial.

4.2.4. Evaluación de la exactitud del navegador

Para que el tratamiento de ablación con RF sea exitoso es necesario que se eliminen todas las células cancerosas, que es el objetivo principal de la intervención. Por otro lado se tiene que eliminar la mínima cantidad posible de células sanas. Para realizar el tratamiento con estas dos metas es necesario colocar la punta de la aguja con el electrodo en el centro del tumor con la mayor exactitud posible. Un error en la colocación significaría la necesidad matar más células sanas para poder eliminar por completo el tumor, ya que el volumen de ablación es generalmente esférico o elipsoidal. Así, si se conoce la posición de la punta de la aguja con respecto al tumor con cierto error, es posible calcular la esfera de tamaño mínimo, centrada en la punta de la aguja y que cubra el tumor por completo. Si el error es mayor, el radio de la esfera deberá ser mayor para asegurarse de cubrir la lesión.

El objetivo de este experimento es estimar el error en la localización de la punta de la aguja con respecto al objetivo quirúrgico o tumor a tratar.

Se utilizó una aguja de ablación con RF de calibre 17 (1.473 mm de diámetro) y 172 mm de largo, y un sistema de rastreo que consta de 4 cámaras *ARTtrack2* **Error! Reference source not found.**, con una computadora personal con *software* de ART que transmite por red al navegador los datos de rastreo. Las cámaras incluyen un procesador que segmenta las imágenes adquiridas y mandan a la computadora únicamente la posición 2D del centro de cada marcador detectado. El *software* de ART se encarga de combinar los datos de las cámaras para obtener la localización 3D de cada marcador, así como de comparar las configuraciones de marcadores detectadas con las definidas previamente. Así, cuando encuentra una configuración que coincide, registra la posición y orientación de ésta. Cuando termina de procesar un cuadro, manda por red los datos obtenidos a una segunda computadora. Las imágenes de US se adquirieron con un dispositivo *Picker Computer Sonograph CS 9300* con un transductor curvilineal de 3.5 MHz, que se muestra en la Figura 37.

Figura 37.Sistema de adquisición de imágenes de US.

Para medir el error, se realizan los siguientes pasos:

a. *Colocar bola de hule en la punta de la aguja.* Se usó una pequeña bola de hule, de 5 mm de diámetro, que ayudará a ubicar la punta de la aguja en las imágenes de US.

b. *Sumergir en agua.* Se introdujo la punta de la aguja en agua y se dejó en una posición estable, como se muestra en la Figura 38. Los marcadores de rastreo fijos a la aguja se mantuvieron fuera del agua.

c. *Localizar con US.* Usando la sonda de US, se localizó la punta de la aguja, como muestra la Figura 39 que se marcó en el navegador como objetivo quirúrgico. Así, en condiciones ideales, el objetivo debería ser la punta misma de la aguja.

d. *Medir la distancia.* El navegador informa constantemente la distancia entre la punta de la aguja y el centro del objetivo quirúrgico. Esta distancia se consideró como error en la medición.

e. *Repetir hasta obtener una muestra representativa.* Se repitió el experimento con 8 posiciones diferentes de la sonda de US, midiendo el error en cada caso. Se calculó la media de este error.

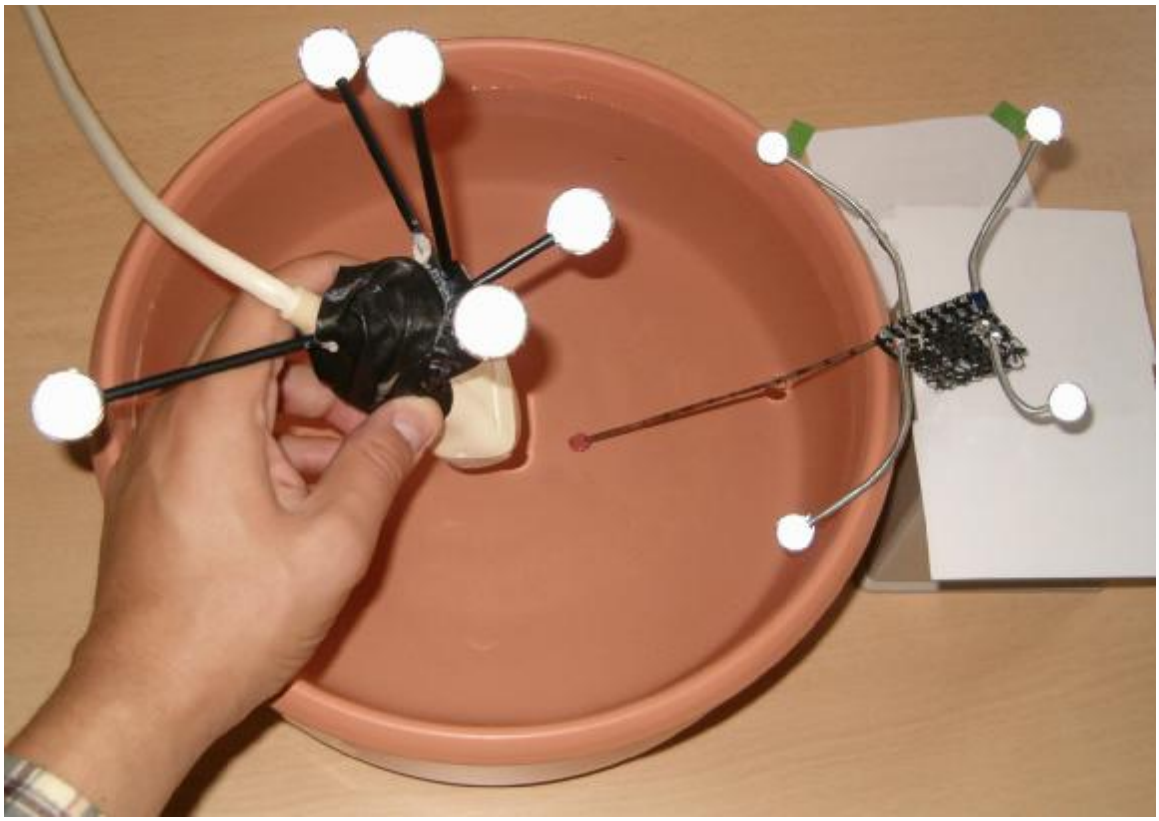

Figura 38.Configuración del experimento para medir la precisión del sistema.

El error obtenido es una combinación del error en el sistema de rastreo, la calibración de la aguja y la calibración del US. La primera depende de la colocación de las cámaras y la calidad de los marcadores y configuraciones de éstos fijos a la aguja y a la sonda de US. Las otras dos dependen de pasos previos a este experimento y representan la mayor fuente de errores, ya que el sistema de rastreo tiene una precisión probada de 0.1 mm.

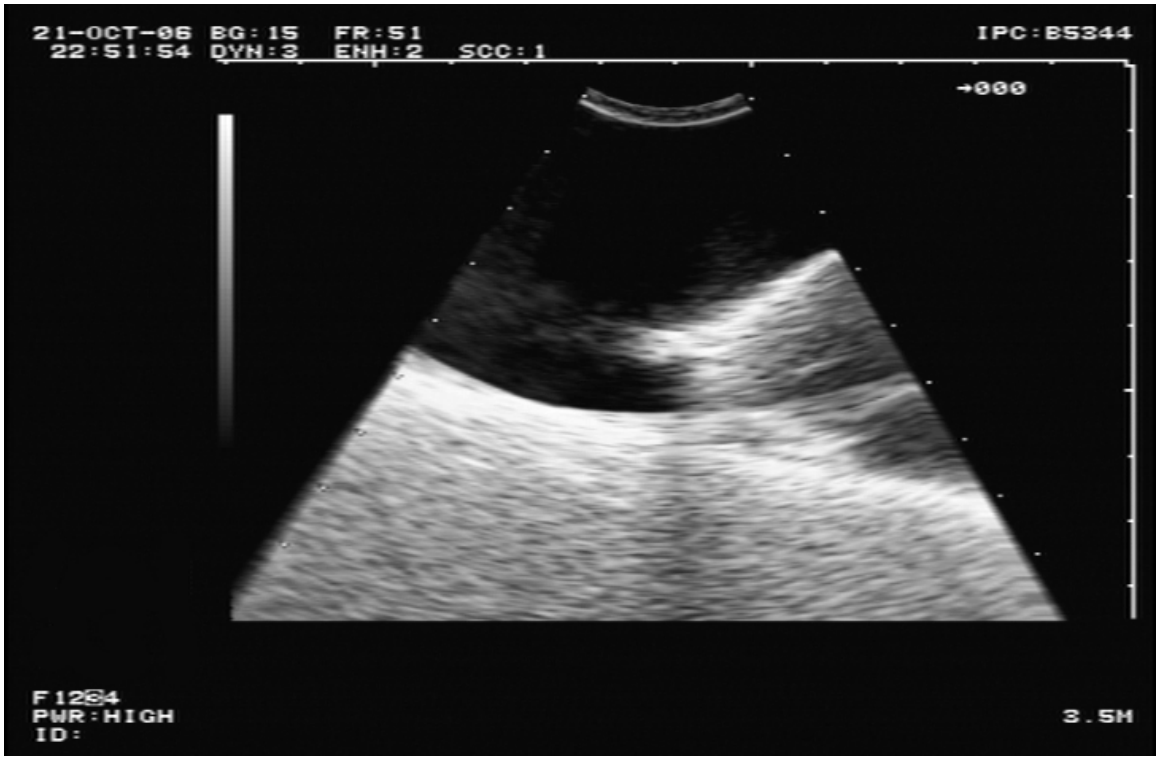

Figura 39.Imagen de US que muestra la punta de la aguja con la bola de hule.

El error medio obtenido con este experimento fue de 1.96 mm, como se reportó en Alcérreca *et al*. **Error! Reference source not found.**.

Con este experimento se probó que la exactitud del sistema de navegación es suficientemente alta para realizar este tipo de intervenciones. La mayor parte del error proviene de la calibración del US, que es una aplicación separada del navegador quirúrgico y por lo tanto está fuera del alcance de este proyecto. Otra fuente importante de error es la necesidad de segmentación manual, que dada la cantidad de ruido en las imágenes, es una tarea difícil. Con sistemas de imágenes de US más modernos sería posible realizar mejores calibraciones y segmentaciones, reduciendo así el error.

4.2.5. Pruebas del navegador con un maniquí de gelatina

Una de las formas más sencillas de probar la funcionalidad de los sistemas CAS es el uso de maniquíes. Con este tipo de pruebas se trata de mostrar el uso del sistema en situaciones que simulan de alguna forma la realidad. Así, es necesario diseñar un modelo

del objeto de estudio que muestre las características relevantes al problema. En el caso del navegador quirúrgico para tratamiento de tumores con RF, el modelo mínimo que se necesita para probar es un tumor, que sea visible en imágenes de US y que sea posible atravesarlo con una aguja. Este experimento también se probó la función de cámara aumentada del sistema, que utiliza imágenes de una cámara de video rastreada y calibrada, de modo que el sistema muestre los modelos 3D virtuales alineados sobre la imagen del mundo real. Esta funcionalidad ayuda al médico a mostrar el modelo desde un punto de vista específico, además de permitir ver objetos rastreados y no rastreados en la misma imagen, lo que ayuda a orientarse mejor en un ambiente virtual.

El maniquí utilizado, que se muestra en la Figura 40, se fabricó con gelatina y una bola de masa de papa de 15 mm de diámetro. Se usaron estos materiales porque son fáciles de conseguir y de manipular, además de cumplir con los requisitos del problema. La gelatina transmite bien las ondas de US y es suficientemente rígida para darle forma al maniquí, por lo que se usó para simular el tejido suave. La masa también transmite el US, pero con diferente velocidad, por lo que las ondas se reflejan en su superficie, haciéndola visible en las imágenes. Así, se usó para simular el tumor. La aguja de RF, el sistema de rastreo y el de imágenes de US que se utilizaron fueron los mismos que en el experimento presentado en la sección 4.2.4.

El experimento consistió en usar el sistema como se usaría en un caso real:

a. *Localizar el tumor en las imágenes de US*. El primer paso es usar la sonda de US para encontrar la bola de masa en la gelatina. Es importante encontrar el centro de la bola. Para esto, cuando se ve un círculo en la imagen, se mueve la sonda transversalmente al plano de adquisición, de un lado al otro hasta que el círculo muestra el diámetro máximo.

b. *Marcar el tumor en el navegador*. Sabemos que el centro del círculo de diámetro máximo que cruza la esfera es el centro de ésta. Así, cuando se selecciona el centro del círculo como objetivo del tratamiento, podemos esperar que la esfera obtenida represente al tumor.

c. *Usar la vista de la aguja para la inserción*. Ya que se definió un objetivo quirúrgico, se cambia a la vista de aguja. Aquí se mueve la aguja para sobreponer la línea de extensión con la línea de dirección y después se inserta presionando la base en la dirección de la aguja hasta que la distancia reportada por el sistema sea menor a 1 mm.

d. *Confirmar resultados en la vista de cámara aumentada*. Se utilizó la vista de cámara aumentada para confirmar que lo que muestra es correcto. Para esto, se comparó visualmente la posición de la aguja y la sonda de US con la posición del modelo de aguja virtual y el plano de US mostrados en la imagen.

Figura 40.Maniquí de gelatina.

Este sólo produjo resultados cualitativos, pero se comprobó que el sistema funciona correctamente y que es muy fácil de usar.

La vista de cámara aumentada probó ser muy útil para seleccionar un punto de vista arbitrario sin usar los controles de la computadora directamente. En la Figura 41 se muestran dos imágenes obtenidas con la cámara y desplegadas en esta vista. La primera muestra la sonda y el plano de US después de marcar el objetivo quirúrgico. La segunda muestra la aguja dentro del objetivo. Es interesante notar que el plano se mantiene en el mismo lugar a pesar de que la sonda fue movida. Esto se debe a que la imagen fue congelada, junto con la posición, al momento de definir el objetivo. El objetivo quirúrgico es de diferente color en las imágenes, ya que depende de la distancia a la que se encuentre la punta de la aguja. La aguja virtual y el plano de US se encuentran en la posición esperada, lo que confirma que la calibración de los tres elementos: la aguja, el US y la cámara, no tienen errores grandes.

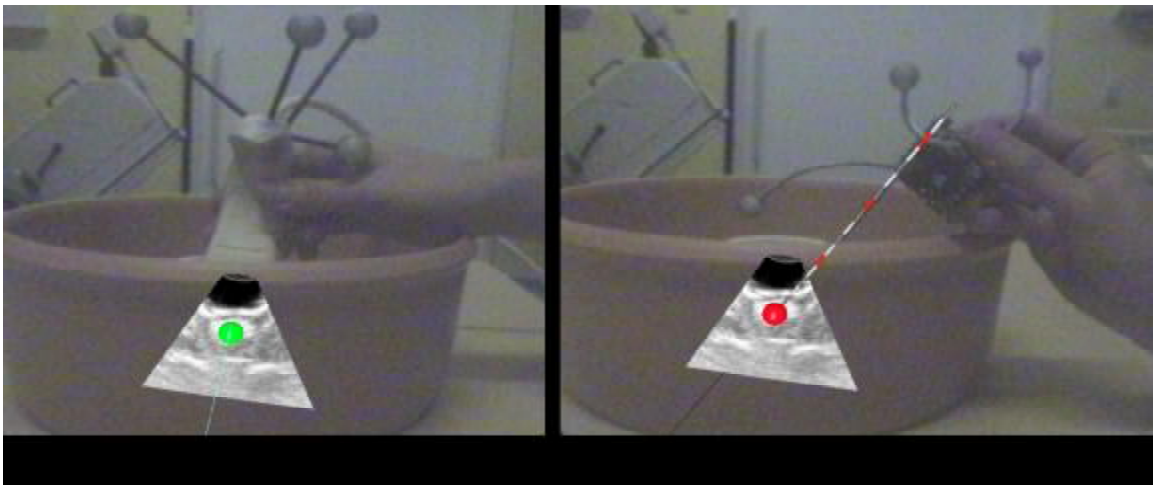

Figura 41.Vista de cámara aumentada mostrando (a) la sonda y el plano de US y (b) la aguja de RF.

4.3 Discusión

Los experimentos presentados en esta sección probaron que el prototipo que se desarrolló puede funcionar como una base sólida para el desarrollo de un navegador quirúrgico seguro y confiable, además de ser, por sí mismo, una mejora considerable al proceso actual. Una de las principales ventajas sobre la inserción de la aguja usando sólo las imágenes de US es que con el navegador se puede dividir la tarea en dos etapas: localización e inserción. Con esta separación de tareas no es necesario manipular al mismo tiempo la aguja y la sonda de US. Actualmente se requiere ver la aguja en las

imágenes de US para conocer su posición con respecto al tumor, por lo que es necesario que el plano del US contenga a la aguja. Debido a esto, la trayectoria de la aguja está limitada a un plano. Con nuestro sistema es posible insertar la aguja desde cualquier ángulo, ya que no se usa como guía la imagen, sino el modelo 3D generado.

5. Conclusiones

En este trabajo se desarrolló un prototipo de navegador quirúrgico para el tratamiento de tumores con RF. El navegador se basa únicamente en imágenes de US para localizar la lesión e insertar la aguja. Mediante un sistema de rastreo, el sistema conoce la posición y orientación de los elementos de interés y los despliega en la pantalla mediante una interfaz gráfica sencilla.

También se desarrollaron herramientas básicas para construcción y registro de modelos, que forman parte del proyecto del que este trabajo forma parte. Estas herramientas están aún en etapa de pruebas, pero con algo más de desarrollo será posible incorporarlas al navegador, de manera que los modelos que éste maneje estén basados en imágenes preoperatorias de MRI. Con esto, el modelo tendrá mayor resolución y más información relevante.

Los experimentos realizados mostraron que el navegador funciona correctamente. Además, es muy fácil incluirlo en el flujo de trabajo médico que se utiliza actualmente, ya que la modificación es mínima. La única diferencia es la división de la inserción de la aguja en dos tareas consecutivas. Antes se localizaba la lesión y se insertaba la aguja en un solo paso, mientras que con el navegador es posible localizar primero e insertar después. Así, cada una de las tareas es más sencilla, ya que no es necesario alinear el plano de US con la aguja durante la inserción, lo que requiere experiencia y habilidad considerables. Gracias a esto, el navegador debe producir tratamientos más exactos y menos traumáticos.

5.1 Trabajo futuro

De acuerdo a los experimentos reportados en este trabajo, el navegador quirúrgico es funcional. Sin embargo, antes de utilizarlo con pacientes es necesario realizar una evaluación exhaustiva de la exactitud del sistema, por ejemplo, utilizando una máquina de medición por coordenadas para localizar el tumor y así poder calcular el error respecto

a la posición indicada por el navegador. Adicionalmente se deben realizar evaluaciones hechas por cirujanos expertos en el tratamiento de tumores con RF.

Con respecto al sistema de construcción de modelos, es necesario agregar la funcionalidad de segmentación, que se ha desarrollado como un trabajo paralelo a éste. Así, para construir un modelo será necesario leer los archivos DICOM entregados por el sistema de imágenes de MRI. Posteriormente, segmentar las imágenes presentes en estos archivos, de manera que se diferencien las estructuras de interés, incluyendo el tumor y huesos adyacentes. Finalmente, usando las imágenes segmentadas y los datos incluidos en el encabezado de los archivos DICOM, se generará el modelo 3D aplicando el algoritmo *marching cubes*. Para el modelo transoperatorio, será necesario usar datos de US-3D y algoritmos de segmentación de volúmenes.

El sistema de registro también deberá ser incorporado al navegador quirúrgico, de forma que cuando se tengan los dos modelos 3D se calcule el registro y se aplique, todo dentro del mismo sistema. Así, será posible transformar el modelo preoperatorio de manera que comparta el sistema de coordenadas con la aguja de RF y con la cámara aumentada.

Por último, se puede continuar el proyecto generalizando el navegador quirúrgico de forma que no sólo ayude en el tratamiento de tumores en el sistema musculoesquelético, sino en cualquier parte del cuerpo. Para esto será necesario agregar la capacidad de manejar modelos deformables, que cambien con los movimientos del cuerpo y con la presencia de las herramientas quirúrgicas, de forma que representen con mayor precisión la anatomía y lesión del paciente en tiempo real.

Anexo I. CAMPAR

El marco de trabajo CAMPAR (*Computer Aided Medical Procedures and Augmented Reality*) es una biblioteca de clases que funciona como base para desarrollar sistemas CAS. La idea principal de CAMPAR es proveer acceso a los datos que requiere un sistema de realidad aumentada médica. Incluye las herramientas para acceder a los datos provenientes de diferentes dispositivos de adquisición, sincronizarlos y desplegarlos en tiempo real. Actualmente CAMPAR sólo se desarrolla sobre *Windows*[1], ya que es la única plataforma para la que los controladores de todos los dispositivos utilizados están disponibles. El desarrollo se realiza sobre *Visual Studio 2005*[2].

Instalación

Antes de instalar CAMPAR es necesario adquirir el código fuente. Para esto se requiere una cuenta en el servidor de *Subversion* (SVN)[3]. El grupo CAMPAR usa SVN para compartir el código fuente de sus sistemas, de forma que cada integrante puede descargar la última versión (que se actualiza todos los días), modificar el sistema y subir los cambios. SVN permite regresar a versiones anteriores del código cuando se detectan errores y permite rastrear todos los cambios realizados por cada usuario. Utilizando un cliente de SVN, como *TortoiseSVN*[4], se accede al servidor con la siguiente dirección:

`svn+ssh://<usr>@svnnavab.informatik.tu-muenchen.de:/svn/svn/trunk/`

Donde `<usr>` es el nombre de usuario en el servidor SVN. Al menos se necesitan los directorios `src`, con el código fuente de CAMPAR y `3rdparty`, con las bibliotecas externas que utiliza el marco de trabajo.

---

[1] Microsoft Windows. http://www.microsoft.com/windows/. 2007.

[2] Microsoft. Visual Studio 2005. http://msdn2.microsoft.com/en-us/vstudio. 2007.

[3] CollabNet. Subversion. http://subversion.tigris.org/. 2006.

[4] TortoiseSVN. http://tortoisesvn.net/. 2007.

Ya que se descargó el código fuente y las bibliotecas externas, es necesario ejecutar `3RDPARTY_PATH.bat`, que se encuentra en el directorio `3rdparty`. Con esto, se agregan los directorios necesarios para la compilación y ejecución de CAMPAR a la variable de estado del sistema `PATH`. También se crea un directorio virtual `C:\Qt`, que es necesario para ejecutar aplicaciones que utilicen Qt.

El siguiente paso es abrir el proyecto completo `CAMPAR.sln`, en el directorio `src\CAMPAR\vs` con *Visual Studio 2005*. Desde ahí se puede compilar el proyecto con el comando *Build*.

Para probar el sistema se puede ejecutar el archivo generado `CAMPAR.exe` (o `CAMPARd.exe` si se compiló en modo *debug*) pasando como parámetro alguno de los archivos XML en el directorio `bin`. Cada archivo XML representa una aplicación. La mayoría de éstas requieren dispositivos que probablemente no estén disponibles en el sistema en que se ejecutan, por lo que es recomendable probar CAMPAR con una aplicación que sólo utilice archivos como entrada. Un ejemplo que funciona correctamente es `test_marco.xml`.

Desarrollo

Para desarrollar una aplicación nueva primero es necesario saber qué se desea hacer. A partir de esa idea se hace un diseño, que incluya los dispositivos de entrada que se usarán (cámaras, sistemas de rastreo, archivos, etc). Ya que se sabe qué se va a desarrollar, hay que crear un nuevo archivo XML en el directorio `bin`. En este archivo, se declaran los dispositivos de entrada. Los archivos XML de otras aplicaciones pueden servir como ejemplo de cómo declarar dispositivos de entrada específicos.

Lo siguiente es crear una nueva clase que represente un dispositivo de salida. Esta clase representa a la aplicación ante CAMPAR. Para esto, la clase debe heredar de alguna de las clases básicas que representan dispositivos de salida, como *OpenGLDevice* o *QTOutputDevice*.

Por convención, los archivos correspondientes a las clases de una aplicación deben estar en directorios específicos. Todos los archivos de encabezado (con extensión .h) deben estar en:

```
src\CAMPAR\include\CAMPAR\Application\<App>
```

Los archivos de implementación (con extensión .cpp) deben estar en:

```
src\CAMPAR\src\CAMPAR\Application\<App>
```

En ambos casos, el directorio `<App>` corresponde al nombre de la aplicación nueva. La clase nueva debe implementar las funciones `updateOutput`, `Clone`, `create`, `xmlConfigure` y `confiugure` como se indica en el documento *CAMPARHowTo*, incluido en el directorio

```
src\CAMPAR\doc
```

Posteriormente, se incluye esta clase al final del archivo *xml* con una etiqueta de la forma `<device type=″clase″>`. Ahora, al ejecutar CAMPAR pasando como parámetro el archivo *xml* creado se ejecutará la función `xmlConfigure`, después, se ejecutará `configure`. Es importante notar que cuando se ejecuta `xmlConfigure` aún no se han configurado los demás dispositivos, por lo que no se tendrá acceso a ellos hasta el método `configure`. Después de esto, comienza el ciclo de ejecución, que incluye una llamada a `updateOutput`. Este método debe ejecutar la parte más importante del código, ya que será llamado continuamente durante la ejecución del sistema.

Consejos

Al desarrollar aplicaciones basadas en CAMPAR pueden surgir problemas que, aunque no son inherentes a la aplicación, pueden causar errores de compilación o de ejecución, por lo que es importante tomar en cuenta algunos puntos. A continuación se describen algunos consejos prácticos para evitar problemas comunes:

a. *Excluir de la compilación*. Hay ocasiones en que CAMPAR no compila porque hay problemas en una clase. Esto puede suceder cuando esta clase ocupa dispositivos que no están instalados en el sistema o cuando alguien realizó un cambio, agregando errores. Como CAMPAR es un proyecto que se desarrolla por un grupo de investigación grande y el código fuente se comparte entre todos, es muy común que estos problemas ocurran. Cuando el error de compilación se encuentra en una clase que no es utilizada por la aplicación, la solución más sencilla es excluir esta clase de la compilación (no excluirla del proyecto). Para esto, hay que ir a las propiedades de los archivos de la clase (h y cpp) utilizando el *Explorador de Soluciones* de *Visual Studio 2005*. Hacer *clic* sobre cada archivo el botón derecho del *mouse*, seleccionar *Propiedades* y en la sección *General*, seleccionar la opción de excluir.

b. *Archivos moc en clases que usen Qt*. Para desarrollar interfaces gráficas en CAMPAR se pueden utilizar las bibliotecas de Qt. Estas bibliotecas incluyen clases para representar ventanas, botones, etiquetas, menús y otros elementos de una interfaz gráfica común. Para compilar clases que hereden funcionalidad de Qt, es necesario generar archivos *moc*. Estos archivos se generan automáticamente, por lo que no se incluyen con el código fuente de CAMPAR. Para generarlos es necesario declarar que se necesitan en las propiedades del archivo de encabezado de la clase que utilice Qt. Para esto, en la sección *Custom Build Step* de las propiedades del archivo, es necesario agregar la configuración necesaria para crear los archivos *moc*. Un ejemplo de cómo deben quedar los valores se encuentra en las propiedades del archivo *NeedlePlacementOutput.h*. Este archivo incluye la configuración para crear un archivo *moc* en el directorio de la aplicación *NeedlePlacement*. Para otra aplicación hay que cambiar estos valores de forma que actúen sobre el directorio correspondiente.

c. *Leer detenidamente la guía CAMPARHowTo*. Junto con el código fuente de CAMPAR se encuentra la guía *CAMPARHowTo* en formato *TeX*. Esta guía indica con mayor detalle que el presente anexo muchos de los pasos que hay

que seguir para desarrollar aplicaciones con CAMPAR. Es muy recomendable leer y entender esta guía antes de comenzar a desarrollar sistemas con este marco de trabajo. También se incluye una guía de lineamientos, *CodingGuidelines,* que se deben seguir para que el código fuente sea consistente.

NeedlePlacement

Durante este trabajo de tesis se desarrollo un prototipo de navegador quirúrgico para tratamiento de tumores con RF. Lo que hace este prototipo es guiar a un cirujano en durante la inserción de una aguja de RF. La idea es colocar la punta de la aguja en el centro de un tumor. Para localizar el tumor, se utilizan imágenes de ultrasonido (US) y un sistema de rastreo. Este sistema provee al navegador con la localización y orientación de la aguja, la sonda de US y una cámara de video. Las imágenes de US se muestran en el sistema. En ellas, se marca un objetivo quirúrgico, que debe corresponder al tumor. Después, se puede utilizar la cámara aumentada u otro sistema de visualización para dirigir la aguja al objetivo.

Este sistema es un prototipo, por lo que requiere más trabajo antes de ser evaluado por cirujanos y utilizado en la práctica. A continuación se describe cómo se pueden agregar algunas de las funciones que requerirá:

a. ***Segmentación de imágenes de US.*** Actualmente, el prototipo de navegador incluye un algoritmo básico de segmentación con el que se obtiene una esfera. Este algoritmo se implementó en la clase *NeedlePlacementVideo* y se ejecuta cuando el usuario hace *clic* sobre la imagen de US. Un algoritmo de segmentación más complejo debería estar implementado en otra clase, especialmente diseñada para esto. Al hacer *clic,* se debería llamar un método en esta clase que realice la segmentación y que genere un objetivo quirúrgico adecuado (modelo 3D de la superficie segmentada o modelo basado en imágenes de MRI registrado con éste modelo). Hay que notar que el objetivo

quirúrgico requiere un punto específico, que debe ser el centro del tumor. El proceso de segmentación deberá proveer este punto.

b. *Objetivos quirúrgicos complejos.* El prototipo de navegador actual incluye tres clases para definir objetivos quirúrgicos: *NeedlePlacementTarget*, *MeshNeedleTarget* y *SphereNeedleTarget*. La primera es la superclase de las dos siguientes. *MeshNeedleTarget* representa un objetivo formado por una malla, que se lee de un archivo de texto al momento de inicializar el modelo. Para que este objetivo sea útil al sistema es necesario agregarle la función de registro, que le dará una transformación geométrica para pasar el modelo leído de su sistema de coordenadas al que usa el navegador. La clase *SphereNeedleTarget* define un objetivo quirúrgico esférico y, por simplicidad, es la que se usa actualmente. Para agregar objetivos formados por modelos deformables u otras representaciones, se recomienda agregar una nueva clase que herede de *NeedlePlacementTarget* y que incluya esta funcionalidad específica. También será necesario agregar el código que cree objetos de esta nueva clase que, de preferencia, sean llamados al segmentar imágenes de US.

El prototipo de navegador, *NeedlePlacement*, está diseñado a partir del patrón MVC (Modelo – Vista – Controlador), por lo que sus clases se dividen en tres componentes: modelo, vista y controlador. La parte central del sistema es el controlador, que está formado por una sola clase: *NeedlePlacementController*. Esta clase proporciona servicios a las demás para que se comuniquen entre ellas sin estar altamente acopladas. Todas las demás clases tienen acceso a este controlador y pueden usar sus servicios, entre los que destacan:

a. *Log.* Este método permite mandar un mensaje al usuario en la ventana de bitácora (esquina inferior izquierda). Con él, cualquier clase tiene el poder de comunicarse directamente con el usuario a través de la interfaz gráfica sin preocuparse de los detalles de despliegue.

b. *Obtener una imagen.* El controlador tiene métodos para obtener las imágenes adquiridas por la cámara o por la tarjeta de captura de video.

c. *Obtener datos de rastreo.* El método *getTrackingData* permite a todas las clases del sistema obtener la posición y orientación de cualquiera de las herramientas rastreadas. Para esto, se declaran constantes que representan cada herramienta en el archivo de encabezado de *NeedlePlacementController*. En el caso del seguidor Polaris, el archivo de configuración debe declarar las herramientas conectadas con números a partir de `01` y las herramientas inalámbricas con letras a partir de `0A`. Además, para las herramientas inalámbricas es necesario declarar el archivo ROM de definición de la herramienta. Por detalles de implementación es necesario conectar una herramienta en el puerto `01` de Polaris, a pesar de no ser usada.

El controlador, a su vez, usa los servicios de la clase *NeedlePlacementOutput* para desplegar información. Un método importante para esto es *registerDataElement*, que registra un apuntador a una cadena de caracteres. Cada que se despliega un cuadro, la ventana de despliegue de datos en la interfaz gráfica se actualiza con el valor actual de todas las cadenas registradas. Para que esto funcione, las cadenas registradas tienen que estar declaradas como miembros de un objeto que permanezca durante toda la ejecución del programa (como *NeedlePlacementController*) o como valores estáticos (*static*).

Anexo II. NeedlePlacement.xml

El archivo XML de configuración es la base de cualquier aplicación de CAMPAR. En él se declaran las fuentes de datos y la aplicación que los va a usar. Para el prototipo de navegador que se desarrolló en este trabajo, las fuentes de datos son un sistema de rastreo y dos dispositivos de adquisición de video. A continuación se muestra el archivo XML en el que se declaran todos los parámetros necesarios para ejecutar el sistema con un sistema de rastreo *A.R.T.*, una tarjeta de adquisición de video *Falcon* y una cámara *Web* calibrada.

```
<?xml version="1.0" encoding="utf-8" ?>
<CAMPAR>
  <Devices>
    <Device type="ArtTrackerDevice">
      <ID>RFNeedleTracker</ID>
      <BufferSize>40</BufferSize>
      <NetConnection>
        <BufferSize>20</BufferSize>
        <IsUDP>true</IsUDP>
        <IP_TimeOut>3</IP_TimeOut>
        <TimeOut>3</TimeOut>
        <Port>5000</Port>
        <RemoteAddress>192.168.0.2</RemoteAddress>
      </NetConnection>
      <Probe6D>
        <ID>1</ID>
        <ApplicationID>USProbe</ApplicationID>
      </Probe6D>
      <Probe6D>
        <ID>2</ID>
        <ApplicationID>RFNeedle</ApplicationID>
      </Probe6D>
      <Probe6D>
        <ID>3</ID>
        <ApplicationID>Spot</ApplicationID>
      </Probe6D>
      <Probe6D>
        <ID>4</ID>
        <ApplicationID>Webcam</ApplicationID>
      </Probe6D>
      <Probe6D>
        <ID>8</ID>
```

```
      <ApplicationID>EndoCam</ApplicationID>
    </Probe6D>
    <Probe6D>
      <ID>9</ID>
      <ApplicationID>EndoRot</ApplicationID>
    </Probe6D>
  </Device>

  <Device type="FalconGrabberDevice">
    <ID>UltrasoundVideo</ID>
    <BufferSize>20</BufferSize>
    <TimeOut>-1</TimeOut>
    <ResX>768</ResX>
    <ResY>576</ResY>
    <ColorFormat>GRAY8</ColorFormat>
    <SystemNumber>2</SystemNumber>
    <DigitizerNumber>0</DigitizerNumber>
  </Device>

  <Device type = "FirewireGrabberDevice">
    <ID>USBCAM</ID>
    <BufferSize>12</BufferSize>
    <TimeOut>-1</TimeOut>
    <ColorFormat>I420</ColorFormat>
    <SystemNumber>0</SystemNumber>
    <CameraNumber>0</CameraNumber>
    <ResX>640</ResX>
    <ResY>480</ResY>
    <CameraModel>
      <ID>CameraModelID</ID>
      <CameraParameters>
        <AlphaX>1.060258e+003</AlphaX>
        <AlphaY>1.061447e+003</AlphaY>
        <PrincipalX>4.044638e+002</PrincipalX>
        <PrincipalY>2.870324e+002</PrincipalY>
        <Rotation>
          1.782495e-001   7.374500e-002   9.812180e-
001
          -9.526212e-001  2.626948e-001   1.533113e-
001
          -2.464549e-001  -9.620567e-001  1.170763e-
001
        </Rotation>
        <Translation>
          5.399390e+001
          -1.189708e+001
          -1.445052e+000
```

```
              </Translation>
              <DistortionCoefficients>
                -2.4794e-001   1.67703e-001    -4.12588e-003
9.677111e-003
              </DistortionCoefficients>
            </CameraParameters>
            <TrackerParameters>

<TrackerDevice>RFNeedleTracker</TrackerDevice>
              <CameraAppID>Webcam</CameraAppID>
            </TrackerParameters>
          </CameraModel>
        </Device>

        <Device type="Synchronizer">
          <ID>Fairness</ID>

<WaitForThisDevice>UltrasoundVideo</WaitForThisDevice>
          <NTPServer>192.168.0.2</NTPServer>
          <NTPPollInterval>2</NTPPollInterval>
          <NTPPrintDetails>false</NTPPrintDetails>
        </Device>

        <Device type="GraphicQt">
          <ID>GL Windows</ID>
          <WindowTitle>Single View Window</WindowTitle>
          <Width>640</Width>
          <Height>480</Height>
          <Stereo>false</Stereo>
          <Fullscreen>false</Fullscreen>
        </Device>

        <Device type="NeedlePlacementOutput">
          <ID>theWindow</ID>
          <Width>900</Width>
          <Height>600</Height>
          <Fullscreen>false</Fullscreen>
          <VideoDevice>UltrasoundVideo</VideoDevice>
          <WebcamDevice>USBCAM</WebcamDevice>
          <NeedleTracker>RFNeedleTracker</NeedleTracker>
          <NeedleID>RFNeedle</NeedleID>
          <ProbeID>USProbe</ProbeID>
          <CalibrationSpotID>Spot</CalibrationSpotID>

          <Tip>2.048199e+002 5.412698e+001
    1.019002e+001</Tip>
          <Base>32.57 47.05 9.95</Base>
```

```
            <USScale>0.20689 0.20689 0.200618</USScale>
            <USCalibrationMatrix>
              -0.878474           -0.477160           0.024525
12.285344
              0.364324           -0.702178           -0.611730
-90.663422
              0.309114           -0.528454           0.790686
-51.647640
              0.000000           0.000000            0.000000
1.000000
            </USCalibrationMatrix>

            <CroppingArea>
              <Top>72.2028</Top>
              <Left>439.12</Left>
              <UpperExtent>46.64</UpperExtent>
              <LowerExtent>257.84</LowerExtent>
              <UpperHeight>393.465</UpperHeight>
              <LowerHeight>1.05616</LowerHeight>
              <LineDistance>10</LineDistance>
            </CroppingArea>
          </Device>

        </Devices>
      </CAMPAR>
```

6. Bibliografía

[1] Ackerman M. J. (1998) *The Visible Human Project*. Proceedings of the IEEE, Vol. 86, Issue 3, pp. 504 – 511.

[2] Adams R., Bischof L. (1994) *Seeded Region Growing*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 4, Junio, pp. 641-647.

[3] Advanced Realtime Tracking, GmbH. (2006) *Camera ARTtrack2*. http://www.ar-tracking.de/

[4] Alcérreca Alcocer C., Arámbula Cosío F., Hazan E. (2006) *Construction and Registration of 3D Models for Computer Assisted Tumor Treatment.* Mexican Symposium on Medical Physics, Guadalajara, Mex.

[5] Alcérreca Alcocer C., Arámbula Cosío F., Hazan E. (2006) *Navigation system for computer assisted tumor ablation using radio frequency*. IEEE – EMBC, NY, EUA.

[6] Alcérreca Alcocer C., Vogel J., Feuerstein M., Navab N. (2007) *A New Approach to Ultrasound Guided Radio-Frequency Needle Placement*. Aceptado en Bildverarbeitung für die Medizin (BVM), Múnich, Alemania.

[7] Arámbula Cosío F., Davies B. L. (1999) *Automated prostate recognition: a key process for clinically effective robotic prostatectomy*, Medical & Biological Engineering & Computing, Vol. 37, pp 236-243, IFMBE.

[8] Bauer M., Schlegel M., Pustka D., Navab N., Klinker G. (2006) *Predicting and Estimating the Accuracy of n-occular Optical Tracking Systems*. The Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality, Santa Barbara, CA, USA, Oct. 22 - 25.

[9] Best P. J., McKay N. D. A (1992) *Method for Registration of 3-D Shapes*, Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, IEEE, pp 239-256.

[10] Birkfellner W. (2000) *Tracking Systems in Surgical Navigation*. Ph. D. Thesis. Universität Wien.

[11] BrainLAB AG. (2006) *VectorVision CAS Platforms.* http://www.brainlab.com.

[12] Cox J., Kirkpatrick R. C. (1896) *The new photography with report of a case in which a bullet was photographed in the leg*. Montreal Medical Journal, 24, Canadá, pp. 661-665.

[13] Davis K., Choi J., Blankenbaker D. (2004) *Radiofrequency ablation in the musculoskeletal system*. Seminars in Roentgenology, Vol 39:1, pp. 129-144.

[14] Gastelum Strozzi A. (2005) *Construcción de un modelo del sistema gastrointestinal alto para simulaciones de endoscopía*. Tesis de Maestría. UNAM, México.

[15] Gazelle G. S., Goldberg S. N., Solviati L., Livraghi T. (2000) *Tumor Ablation with Radio-frequency Energy*, en Radiology; 217, pp 633-646.

[16] Gutiérrez Medina L. R., Arámbula Cosío F. (2006) *Segmentation of ultrasound images for tumor surgery*, IX Mexican Symposium on Medical Physics, Guadalajara, México.

[17] Howard M. H., Nelson R. C., Paulson E. K., Kliewer M. A., Sheafor D. H. (2001) *An Electronic Device for Needle Placement during Sonographically Guided Percutaneous Intervention*. Radiology, 218, pp. 905-911.

[18] Hsu P-W., Prager R. W., Gee A. H., Treece G. M. (2006) *Rapid, Easy and Reliable Calibration for Freehand 3D Ultrasound*, in Ultrasound Med. Biol., 32, pp 823-835.

[19] Ibañez L., Schroeder W., Ng L., Cates C. (2003) *The ITK Software Guide: The Insight Segmentation and Registration Toolkit (version 1.4)*, Kitware Inc. EUA.

[20] Jamaraz B., Hafez M. A., DiGioia A. M., (2006) *Computer Assisted Orthopaedic Surgery*. Proceedings of the IEEE, Vol. 94, No. 9, pp 1689 - 1695

[21] Khamene A., Vogt S., Azar F. S., Sielhorst T., Sauer F., Niemann H. (2003) *Local 3D Reconstruction and Augmented Reality Visualization of Free-Hand Ultrasound for Needle Biopsy Procedures.* MICCAI (2): 344-355.

[22] Lorensen W. E. y Cline H. E. (1987) *Marching Cubes: A high resolution 3D surface construction algorithm*. Proceedings of the 14th annual conference on

Computer graphics and interactive techniques table of contents, ACM press, NY, pp. 163 - 169.

[23] McGahan J. P., Browing P. D., Brock J. M., Teslik H. (1990) *Hepatic ablation using radiofrequency electrocautery*. Invest Radiol; 25:267-270.

[24] National Electrical Manufacturers Association. (2004) *DICOM Standard*, http://medical.nema.org/dicom/2004.html.

[25] NDI: Northern Digital Inc. (2007). http://www.ndigital.com.

[26] NTP: The Network Time Protocol. (2006) http://www.ntp.org/

[27] Peters T. (2000) *Image-guided surgery: From X-rays to Virtual Reality*. Computer Methods in Biomechanics and Biomedical Engineering, Vol. 4, pp 27-57.

[28] Prager R. W., Rohling R. N., Gee A. H. Berman L. (1998) *Rapid Calibration for 3-D Freehand Ultrasound*. Ultrasound Med. Biol., Vol 24, No 6, pp 885-869.

[29] Rasband W. (2006) *ImageJ*. http://rsb.info.nih.gov/ij/, Research Services Branch, National Institute of Mental Health, Bethesda, Maryland, EUA.

[30] Sielhorst T., Feuerstein M., Traub J., Kutter O., Navab N. (2006) *CAMPAR: A software framework guaranteeing quality for medical augmented reality*. International Journal of Computer Assisted Radiology and Surgery, Vol. 1 Suppl. 1, page 29 - 30.

[31] Taylor R. H., Stoianovici D. (2003) *Medical Robotics in Computer-Integrated Surgery*. IEEE Transactions on Robotics and Automation, Vol. 19, No. 5, pp. 765-781.

[32] Treece G. M., Gee A. H., Prager R. W., C. Cash C. J., Berman L. (2003) *High-definition freehand 3-D ultrasound*, Ultrasound Med. Biol. 29(4), pp 529-546.

[33] Troccaz J., Peshkin M., Davies B. (1998) *Guiding systems for computer-assisted surgery: introducing synergistic devices and discussing the different approaches.* Medical Image Analysis, vol. 2, No. 2, Oxford University Press, pp. 101-119

[34] Trolltech. (2006) *Qt*. http://www.trolltech.com/products/qt.

[35] U. S. National Library of Medicine. (2006) *The National Library of Medicine's Visible Human Project*. http://www.nlm.nih.gov/research/visible/visible_human.html, Maryland, EUA.

[36] World Wide Web Consortium. (2006) *Extensible MarkupLanguage (XML)*. http://www.w3.org/XML/.

Zagoria R. F. (2004) *Imaging-guided Radio-frequency Ablation of Renal Masses*. RadioGraphics, Vol 24, pp. 59-71