UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA
Centro de Ciencias Aplicadas y Desarrollo Tecnológico

Modelado de un ambiente virtual para un sistema de simulación de cirugía de próstata

TESIS PROFESIONAL
para obtener el título de
INGENIERO EN COMPUTACIÓN

ÁREA
Sistemas Inteligentes y Computación Gráfica

Modalidad: Seminario de titulación

PRESENTA:
SERGIO TEODORO VITE

DIRECTOR DE TESIS
M. en C. Miguel Ángel Padilla Castañeda

Ciudad Universitaria, México, Octubre 2008.

Dedicatoria

A mis padres, *Julia Vite Franco* y *Emiliano Teodoro Mojica*, quienes me han lo han dado todo en la vida sin condiciones. Esto es sin duda su logro, ustedes lo hicieron posible. Gracias por estar conmigo siempre y ser mi modelo de lucha e inspiración constante, me han preparado la mejor herencia que un padre puede dar a su hijo, la educación.

A mis hermanos, *Elizabeth*, *Marisol*, *Luis* y *Raúl* (†), sin su apoyo de todo tipo nunca lo hubiera logrado. A pesar de la distancia siempre han estado conmigo y durante mucho tiempo han sido ustedes mi objeto de admiración.

Agradecimientos

A mi asesor, el *M. en C. Miguel Ángel Padilla Castañeda*, quien puso su confianza en mí para encargarme del desarrollo de esta parte tan importante del simulador, por su paciencia y por todo el tiempo que dedicó para la revisión de este trabajo. Sin duda eres mi mejor fuente de inspiración profesional.

A mis familiares y amigos, gracias por estar ahí, siempre al pendiente y por su apoyo en todos los sentidos. En especial, un enorme agradecimiento a *Héctor Curiel*, quien durante este tiempo me ha acompañado con sus consejos y buenos deseos. Gracias por todas las lecciones que me has brindado y por demostrarme que ésto vale la pena. A *Kovak Reyes*, *Eduardo Nicanor*, *Marisol Acosta*, *Francisco Ayala*, *Max*, *Sergio* y en general a todos mis compañeros de carrera y de la vida, quienes comparten conmigo ese deseo de ser cada día mejores.

Al equipo desarrollador del simulador, en especial a *Bartolomé*, *David*, *Eric*, *Gabriela*, *Luceli* y *Rommel*, por brindarme su apoyo para que esta parte del proyecto saliera adelante y por acompañarme durante las etapas de desarrollo, son un excelente equipo de trabajo.

A los responsables del Laboratorio de Análisis de Imágenes y Visualización del CCADET, el *Dr. Jorge Márquez* y el *Dr. Fernando Arámbula*, gracias por sus comentarios y sugerencias durante el desarrollo de este trabajo de tesis.

Al Centro de Ciencias Aplicadas y Desarrollo Tecnológico, por ese gran esfuerzo por mantenerse a la vanguardia en investigación, gracias por brindarme la oportunidad de sentirme parte de esa filosofía de innovación para el bien de nuestro país.

A la DGAPA, con su programa de Fortalecimiento de la Docencia a través del Observatorio de Visualización de la UNAM, IXTLI, por brindarme los estímulos económicos necesarios para el desarrollo de mis actividades.

A mis profesores de carrera, quienes me brindaron su conocimiento para aplicarlo en la vida profesional. En especial, gracias a la *M. en I. Isabel Patricia Aguilar Juárez*, quien me enseñó que la vida académica es todo un reto pero que vale la pena como proyecto de desarrollo profesional; gracias por todos los buenos deseos y por estar siempre al pendiente de mi evolución profesional.

Finalmente, ***a mi alma máter***, la *Universidad Nacional Autónoma de México*, por ser mi fuente de conocimiento y sabiduría, a ella me debo profesionalmente y a ella espero retribuir en todo ese apoyo que me brindó. Gracias por poseer a tanta gente tan valiosa en todos los ámbitos y darme la oportunidad de ser en adelante, un profesionista egresado de esta casa de estudios de México.

Índice de Contenido

Índice de figuras

Índice de tablas

Resumen

Los sistemas de realidad virtual, aplicados en medicina, han tenido un gran impacto sobre tres actividades fundamentales: 1) la enseñanza y entrenamiento de habilidades médicas, 2) la planeación de procedimientos quirúrgicos y 3) la asistencia médica durante y después de la cirugía. Sus usos y metodologías han sido reportados desde hace ya más de una década en las principales revistas especializadas en biomedicina y el cómputo gráfico.

Específicamente para el entrenamiento de habilidades médicas para cirugías de endoscopía, los sistemas completos de realidad virtual, también llamados ambientes virtuales, permiten a los estudiantes acercarse a la realidad por medio de interfaces que transforman el mundo real en un mundo virtual, y que a su vez retroalimentan dos sentidos principales: la vista y el tacto. La vista por medio de un dispositivo de despliegue y el tacto con ayuda de dispositivos hápticos.

El simulador para el entrenamiento del procedimiento de resección transuretral de la próstata (RTUP), es un desarrollo del grupo de Análisis de Imágenes y Visualización del Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET) de la UNAM. Se trata de un conjunto de bloques funcionales que tienen como objetivo la simulación de condiciones reales entorno al procedimiento quirúrgico, haciendo una abstracción de los elementos del mundo real dentro de un entorno virtual.

Dentro de los trabajos desarrollados, esta tesis describe los detalles de implementación de una parte de la estación de gráficos, enfocada al mejoramiento de efectos visuales y la adaptación de dos modelos anatómicos al simulador: un modelo de vejiga urinaria y un modelo del conducto uretral, que en conjunto con el modelo de la glándula prostática, conforman el sistema gráfico de realidad virtual del simulador de RTUP.

Los resultados obtenidos de estas adaptaciones incluyen: una metodología completa para la reconstrucción de mallas de superficie a partir de isosuperficies, una metodología de creación gráfica para el simulador de RTUP, la definición de los elementos del entorno gráfico virtual (la cámara, los modelos tridimensionales, la iluminación y otros efectos) y la inclusión del formato OBJ de Wavefront para la carga de modelos de superficie. Adicionalmente se presentan las herramientas de cómputo empleadas y las aplicaciones desarrolladas en lenguaje C durante todo el proceso de implementación.

Presentación de la tesis

Dado el amplio campo de estudio que ofrecen los sistemas de realidad virtual en el ámbito médico, se presenta en este trabajo de tesis, la descripción de un sistema de simulación del procedimiento de Resección Transuretral de Próstata (RTUP), proyecto actual y en fase de desarrollo por parte del grupo de Análisis de Imágenes y Visualización del Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET), de la UNAM.

En su estado actual, el simulador cuenta con los elementos necesarios para llevar a cabo la exploración endoscópica, una de las primeras habilidades que un residente en urología debe aprender a dominar durante su primer año de entrenamiento. Para lograr esto, el sistema posee una parte electromecánica que permite la manipulación de la herramienta de corte, y una visual que permite al usuario observar la escena virtual. Sobre este último punto se centra el contenido de esta tesis, que aporta una metodología de reconstrucción de modelos tridimensionales a partir de imágenes anatómicas, algunas técnicas para efectos visuales y una metodología de creación gráfica con fines de aplicación sobre el simulador.

Objetivos de la tesis

Por lo tanto, este trabajo supone el planteamiento de 4 objetivos principales:

1. Ofrecer una descripción documental del estado actual del simulador.
2. La proposición de una metodología genérica de reconstrucción de mallas de superficie a partir de isosuperficies y su aplicación sobre un modelo de vejiga urinaria virtual.
3. La proposición de un proceso de creación gráfica para ambientes virtuales, aplicado sobre la estación gráfica del simulador.
4. La documentación de los trabajos realizados, mediante una descripción general del uso de algunas herramientas de cómputo y la presentación de algunos programas desarrollados.

Organización de la tesis

Basados en los objetivos anteriores, la tesis se compone de cinco capítulos descriptivos, un capítulo de resultados, un capítulo de conclusiones, y dos apéndices, estructurados de la siguiente manera:

En el capítulo 1, se ofrece un panorama general del estado del arte en el ámbito del desarrollo de simuladores de realidad virtual, aplicados a las ciencias médicas. Se hace hincapié en sus aplicaciones

mediante casos de estudio, sus procesos de desarrollo, las complicaciones y los resultados obtenidos. Antes de ello, se presenta un modelo de arquitectura genérica para ambientes virtuales, que ha servido como base para la implementación de muchos simuladores médicos actuales.

En el capítulo 2, se hace una breve descripción del procedimiento de Resección Transuretral de la Próstata (RTUP) desde el punto de vista médico para después introducirnos en los elementos que nos proporcionan información para la planeación de un sistema de realidad virtual completo. También se describen los bloques que componen al sistema actual de simulación para entrenamiento del procedimiento, sus elementos técnicos de implementación y sus perspectivas futuras.

En el capítulo 3, se describe de manera detallada la implementación de un método completo de reconstrucción de mallas triangulares 3D a partir de isosuperficies (imágenes en 2D de cortes transversales de tejido humano). Este método propuesto, consiste básicamente en tres pasos: 1) la extracción de contornos a partir de imágenes anatómicas, 2) la aplicación de un algoritmo de muestreo para obtener los puntos de la malla y 3) la aplicación de un método de reconstrucción triangular. Para fines de aplicación médica, se presenta la metodología para reconstrucción de modelos anatómicos a partir de conjuntos de datos del VHP (*Visible Human Project*); sin embargo, la metodología es extensible para la reconstrucción de otro tipo de mallas triangulares, basadas principalmente en la información que nos proporcionan las isosuperficies.

En el capítulo 4, se ofrece una descripción detallada de los elementos gráficos tomados en cuenta para la implementación de la estación gráfica del simulador. Los principales objetivos del capítulo son: presentar una metodología de creación gráfica, resultado de los trabajos realizados sobre la estación de gráficos, la definición de los elementos que conforman la escena tridimensional (la cámara, los modelos y la iluminación), la implementación de estos elementos usando la librería gráfica *OpenGL* y la incorporación de efectos visuales. Así también, el capítulo ofrece una investigación monográfica acerca de los principales conceptos del cómputo gráfico llevados directamente a un caso de aplicación.

En el capítulo 5, se ofrece una breve reseña de los formatos de definición tridimensional para la carga de modelos virtuales en el software de simulación. Además se hace una descripción detallada de la adaptación del modelo de definición OBJ de Wavefront para la carga de modelos de superficie y su comparativa con el uso del formato NODE-FACE, implementado con anterioridad a este trabajo de tesis.

En el capítulo 6, se hace una recopilación de los resultados más sobresalientes al aplicar las diversas metodologías para el mejoramiento y adaptación del ambiente gráfico virtual. También se ofrece un análisis comparativo y estadístico de los logros, así como las proyecciones a futuro y las posibles mejoras. Finalmente se‘presentan las conclusiones al desarrollo de este trabajo en el capítulo 7.

Adicionalmente, se presentan dos apéndices. El apéndice A da un breve tutorial para la manipulación de mallas de superficie sometidas al proceso de mallado. Se aborda principalmente la carga de modelos OBJ, suavizado de mallas, asignación de materiales, texturizado y exportación de mallas para ser empleada en el simulador de RTUP, todo ello usando la herramienta de modelado: Blender. El apéndice B, presenta en una primera parte, el software desarrollado para ilustrar los contenidos descritos en esta tesis, y en una segunda parte, la descripción del uso actual del simulador en su primera versión desde el punto de vista propio y con fines de documentación. El software desarrollado incluye: Instrucciones para extracción de contornos con ImageJ, muestreo de contornos con MATLAB (empleando la implementación del algoritmo de muestreo descrito en el capítulo 3) y el constructor de mallados a partir de puntos o vértices, desarrollados en lenguaje C y usando la API de Windows.

Capítulo 1

Introducción: Medicina y Ambientes Virtuales

En las últimas dos décadas, la medicina, como parte de su proceso de evolución, ha incorporado en gran medida a los sistemas de cómputo como herramientas de diagnóstico, asistencia y para el tratamiento de enfermedades. Por su parte, las tecnologías de la información han ido creciendo de forma muy rápida, incorporando cada vez más sistemas de hardware y software de menor costo, más rápidos y con grados de eficiencia adecuados para un gran número de aplicaciones. Específicamente en el área del cómputo gráfico, el impacto que ha tenido el desarrollo de sistemas de realidad virtual para aplicaciones médicas ha sido muy grande, a tal grado de llegar a formar parte del instrumental básico en algunos procedimientos quirúrgicos.

Los ambientes virtuales, son sistemas cuyo objetivo es abstraer la mayor cantidad de elementos presentes en el mundo real (objetos, actores, sensaciones, comportamientos físicos, psicológicos, etc.), y transportarlos a un mundo parcial o totalmente virtualizado; en el cual un usuario pueda interaccionar con los elementos virtuales tal como lo haría en el mundo real. La aplicación en medicina tiene tres principales áreas: la enseñanza de las ciencias médicas, la planeación de procedimientos quirúrgicos, y la asistencia durante y después de la cirugía[30]. Para fines de entrenamiento, Stylopoulos[58] dice: *"se cree que un sistema de entrenamiento ideal es aquel que es capaz de reproducir las condiciones operatorias, al mismo tiempo que un aprendiz se sumerge en un mundo virtual que es una representación exacta del mundo real"*.

Lo anterior supone un gran reto para la ciencia, que tiene que aportar mecanismos físicos, metodologías de reconstrucción de modelos anatómicos, algorítmos de software para el procesamiento de variables físicas, técnicas de inteligencia artificial, etcétera, para llevar a cabo dicha tarea. Así, se destacan dos características principales necesarias en cualquier sistema de realidad virtual: la retroalimentación de fuerzas y la retroalimentación visual. El primero permite al usuario sentir mediante dispositivos hápticos, efectos de palpar, cortar o inclusive suturar tejidos. El segundo le permite recrear la situación real por medio de la vista y generalmente por medio de un sistema gráfico de despliegue.

Un sistema genérico para simuladores médicos se muestra en la figura 1.1, Éste se compone de tres elementos principales: un dispositivo de retroalimentación de fuerzas, una estación de trabajo y una estación de gráficos. El primer bloque envía las posiciones de una herramienta física "virtual" a un sistema de detección de colisiones con un modelo físico generado por computadora. El bloque de detección de colisiones manda la información procesada a otros dos bloques: uno de cálculo de fuerzas, que se comunica directamente con el dispositivo de retroalimentación, ofreciendo al usuario una interacción física; y otro bloque que emplea los datos para realizar cálculos de deformaciones (en caso de tejidos blandos) y finalmente mostrar la información en un dispositivo de visualización. Este último provee al usuario, de retroalimentación visual.

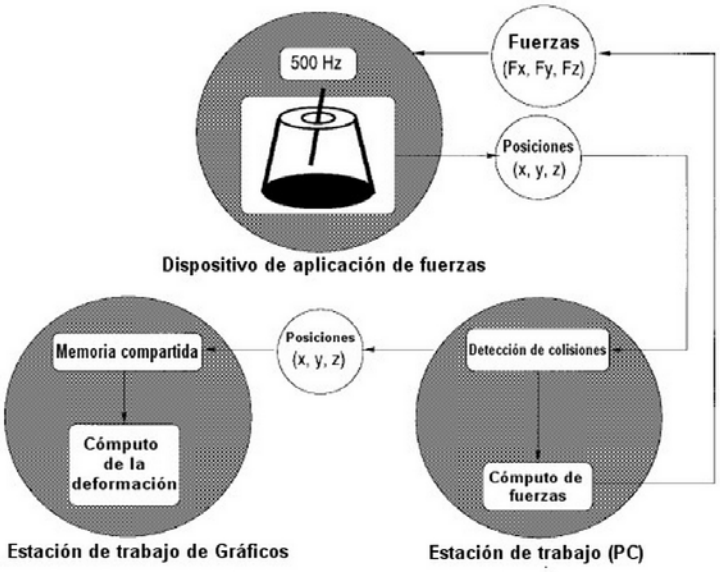


Figura 1.1: *Arquitectura básica de un sistema de simulación en tiempo real (Delingette[22, pág. 515]).*

Actualmente, la mayoría de los sistemas de realidad virtual para simulaciones médicas, poseen los tres bloques anteriores. En algunos casos, la herramienta de retroalimentación de fuerzas se sustituye por dispositivos de interacción, como el ratón o el teclado, que no aportan efectos realistas sobre el sistema, pero que sin embargo ofrecen la interactividad necesaria para que el usuario logre un cierto grado de inmersión. Otros sistemas, más sofisticados, implementan bloques de audio y voz con el fin de alimentar de la mayor cantidad de elementos de información al ambiente virtual.

En las siguientes secciones se exponen algunas implementaciones de entornos virtuales en medicina, con el fin de sustentar la importancia de este tipo de sistemas para los fines de esta tesis.

1.1. Ejemplos de sistemas de Realidad Virtual en Medicina

Como se mencionó anteriormente, las aplicaciones de la realidad virtual en medicina tienen tres principales objetivos: el entrenamiento de habilidades médicas, la planeación de procedimientos quirúrgicos y la asistencia médica. Los siguientes casos de estudio corresponden a reportes de este tipo de sistemas en revistas y congresos internacionales. Estos casos son:

- Ambiente virtual para rehabilitación de deficiencias motoras[46].
- Simulador computarizado para procedimientos de inserción de agujas guiados por ultrasonido[35].
- Sistema de entrenamiento de laparoscopía asistido por computadora[58].
- Sistema de cirugía craneofacial asistida por computadora[29][30].
- Entrenamiento virtual de Cirugía ortopédica[57].
- Otros sistemas: Sistema de entrenamiento de cirugías endoscópicas[32], Sistema de realidad aumentada para marcación craneal[55] y un Sistema de remoción de tumores en neurocirugía[49].

1.1.1. Ambiente virtual para rehabilitación de deficiencias motoras

El sistema de rehabilitación para pacientes con deficiencias motoras[46], es un desarrollo de investigadores de la Universidad de Padovia, Italia en 2005. La terapia consiste en someter al paciente ante un ambiente virtual en sesiones de una hora, en un lapso de un mes, durante el cual, el paciente realiza actividades cotidianas, tales como tomar una pelota, mover un vaso y en general mover cualquier objeto. El estudio consideró a 50 sujetos con lesiones en uno o ambos brazos, relacionadas con algún daño cerebrovascular. Se realizaron mediciones del grado de la lesión antes y después del tratamiento con el ambiente virtual y finalmente se analizaron los resultados con respecto a una escala estándar. Los datos obtenidos de los estudios indicaron una recuperación considerable de los pacientes y en el reporte final, se concluye: *"El presente estudio demuestra que después de la terapia de rehabilitación, empleando un sistema de realidad virtual que refuerza el aprendizaje y supervisa los mecanismos de aprendizaje, se pueden llegar a mejorar las deficiencias motoras de brazos y se provee una evaluación cuantitativa de las habilidades motoras en pacientes postraumáticos"*(Piron[46]).

El sistema de realidad virtual posee las siguientes características de cómputo:

- PC: Pentium IV a 1.2 GHz, 256 MB de RAM y tarjeta gráfica con 32 MB de memoria.
- Proyector LCD de alta resolución (1200 lúmenes).
- Pantalla de proyección.
- Sistema de Tracking para posicionamiento 3D.
- Software especializado desarrollado en el MIT[1] que crea el ambiente virtual que se despliega en la pantalla de proyección.
- Objetos manipulables (pelotas, cubos de poliestireno, vasos de plástico).

La figura 1.2 muestra el sistema de realidad virtual en funcionamiento.

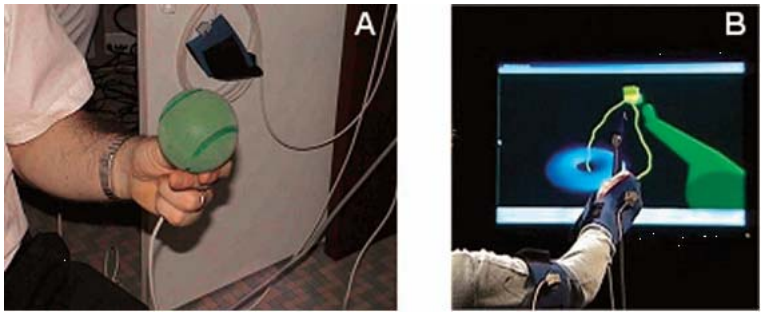


Figura 1.2: *Terapia de rehabilitación usando un ambiente virtual. A) El sensor magnético detecta la posición del efector situado dentro de la pelota. B) Representación virtual del brazo y los efectores usando tres receptores simultáneamente[46, pág. 734].*

[1]MIT, *Massachusetts Institute of Technology, USA.*

1.1.2. Simulador computarizado para entrenamiento de procedimientos de inserción de agujas guiados por ultrasonido

Este sistema es un desarrollo de la Universidad de Leeds del Reino Unido. Se trata de un sistema de entrenamiento del procedimiento de inserción de agujas guiadas por ultrasonido. El simulador está compuesto por un conjunto de datos volumétricos que se alinean a un conjunto de datos obtenidos de un maniquí. Los datos del maniquí son obtenidos por medio de sensores magnéticos que detectan la posición, en el espacio de tres dimensiones, de puntos colocados sobre la superficie. El resultado de la alineación entre espacios, genera imágenes de ultrasonido virtuales que sirven para guiar a los aprendices en el proceso de simulación. Según los reportes, la prueba se realizó sobre 10 sujetos, a los cuales se les realizaron pruebas de las cuatro principales habilidades clínicas en este tipo de procedimiento, que son:

1. Controlar la trayectoria de la aguja entre el punto de inserción y el objetivo, que debe describir una línea recta.
2. Inserción de la aguja y el tiempo que tarda para visualizarse en el ultrasonido.
3. Lograr el mínimo movimiento de la aguja durante el ultrasonido.
4. Lograr visualizar en todo momento la trayectoria de la aguja.

Los resultados obtenidos de las evaluaciones demostraron que el sistema de realidad virtual tenía una participación significativa en el proceso de aprendizaje de aquellos sujetos cuya capacidad para efectuar el procedimiento era limitada. Así, el trabajo concluye: *"El estudio demuestra que es posible la medida de los niveles de aprendizaje de los principiantes. Sin embargo, la evaluación realizada se basa en procedimientos fuera de lo real por lo que queda por determinar si es transferible a procedimientos del mundo real. Se planea la inclusión de algunos efectos como: modelos deformables, retroalimentación háptica y un mejor realismo en las imágenes"* (Magee[35]).

El sistema de realidad virtual posee las siguientes características de cómputo:

- PC Pentium III a 3GHz, 2 GB de RAM sobre sistema operative Microsoft Windows XP.
- Maniquí a escala real de material plástico.
- Un par de sensores de posición y orientación.
- Software desarrollado en C++ usando la librería gráfica OpenGL.

La figura 1.3 muestra el sistema de realidad virtual en funcionamiento.

1.1.3. Sistema de entrenamiento de laparoscopía asistido por computadora

Este sistema fue desarrollado por los Departamentos de Radiología y Cirugía del Hospital de Massachusetts, en conjunto con el Colegio de Boston de los Estados Unidos. El trabajo plantea la necesidad de los sistemas asistidos por computadora para la cuantificación de parámetros clínicos durante el entrenamiento de procedimientos quirúrgicos, tales como: los movimientos del instrumento, las fuerzas aplicadas, la orientación del instrumento y la destreza del alumno.

El principal objetivo se centra en el desarrollo de un sistema de entrenamiento de laparoscopía[58], capaz de evaluar las habilidades del alumno con respecto a medidas estándar, dadas por la

Figura 1.3: *Simulador de realidad virtual para entrenamiento de inserción de agujas guiadas por ultrasonido (Magee[35, pág. 302]).*

medición de parámetros obtenidos de un experto. Los elementos principales de este sistema son: una interfaz mecánica, un conjunto de tareas, una metodología de evaluación estándar y una interfaz de software desarrollada en C++, FLTK[2] y OpenGL. La figura 1.4 muestra el sistema gráfico de realidad virtual empleado para comparar los desempeños de los principiantes con respecto al desempeño de un experto.

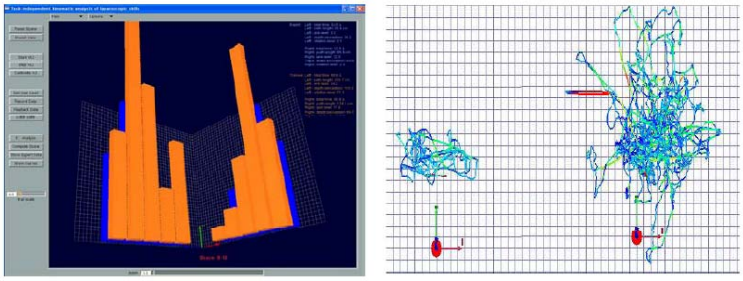


Figura 1.4: *Sistema gráfico de un sistema de entrenamiento de laparoscopía. A la izquierda, interfaz de usuario. A la derecha-izquierda, ruta del experto, a la derecha-derecha, ruta del aprendiz. Una ruta corta es característica de un experto[58, pág. 341].*

[2]FLTK, Fast Light Toolkit, es una herramienta de desarrollo de interfaces gráficas de usuario (GUI). Para mayor información, refiérase a la página del autor: http://www.fltk.org/.

1.1.4. Sistema de cirugía craneofacial asistida por computadora

"La cirugía de deformidades craneofaciales es una tarea muy compleja y requiere de una planeación preoperatoria muy cuidadosa" [29]. De esta premisa parte el desarrollo realizado por investigadores de la Universidad de Erlangen-Nuremberg en Alemania. Se trata de un sistema que permite una visualización tridimensional de la apariencia del paciente después de ser sometido a un procedimiento quirúrgico craneofacial. La cirugía se simula sobre una reconstrucción tridimensional del cráneo y piel del paciente.

Las características de desarrollo del sistema virtual son: programación con lenguaje C++ en estaciones de trabajo Silicon Graphics y la librería orientada a objetos Open Inventor. La reconstrucción de modelos de cráneo se lleva a cabo empleando el algoritmo de *Marching Cubes* (Ver capítulo 3), a partir de imágenes de tomografía computarizada (CT). La planeación de la cirugía se realizó mediante algoritmos avanzados de geometría computacional y Métodos de Elementos Finitos (FEM, por sus siglas en inglés).

La figura 1.5 muestra los resultados obtenidos por el sistema de cirugía asistido por computadora.

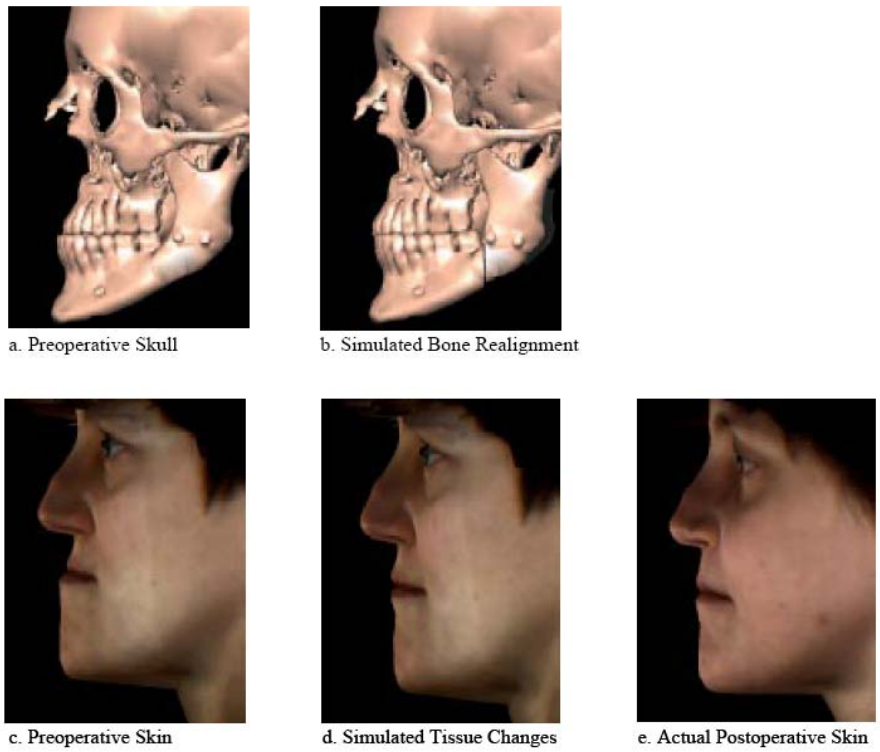


Figura 1.5: *Resultados del sistema de cirugía craneofacial asistido por computadora. a) Reconstrucción del cráneo antes de operar, b) simulación de la alineación de mandíbula, c) piel antes de la cirugía, d) simulación de los cambios en el tejido con la alineación de mandíbula y e) resultado final de la cirugía (Kevee[29][30]).*

1.1.5. Entrenamiento virtual de Cirugía ortopédica

Este trabajo fue realizado por investigadores del Instituto de Ciencias y Tecnología, el Instituto de Computación para Físicos y tecnología, en Rusia, y el Hospital General de Singapur. El objetivo fue el desarrollo de un sistema de entrenamiento ortopédico que permitiera a los alumnos: 1) memorizar las técnicas y herramientas usadas para el tratamiento de fracturas, y a partir de ellas, elija la técnica más apropiada para cada caso, 2) aprender el cómo posicionar correctamente los implantes, con todas las implicaciones técnicas que involucra este tipo de procedimiento y 3) la adquisición de las habilidades necesarias para llevar a cabo implantes que representen el mínimo daño al paciente.

El sistema de realidad virtual se basa en cuatro elementos principales: 1) modelos geométricos 3D, con comportamiento y restricciones, 2) la simulación en tiempo real, que incluye detección de colisiones, sonido y otros efectos, 3) presentación en tiempo real y 4) dispositivos de hardware sincronizados con el sistema de despliegue. La figura 1.6 muestra la interfaz gráfica del sistema de realidad virtual.

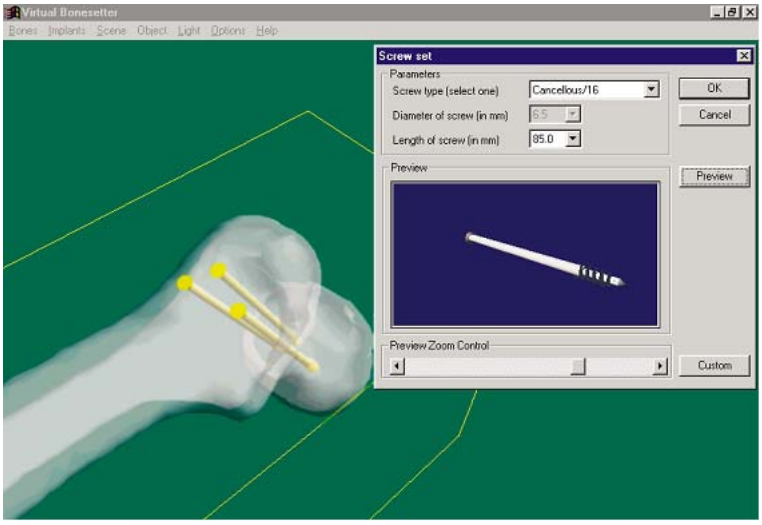


Figura 1.6: *Interfaz de usuario de un sistema de realidad virtual de un simulador de entrenamiento de cirugía ortopédica[57, pág. 9].*

1.1.6. Otras sistemas

Entre otros sistemas de realidad virtual en el campo de la medicina, se destaca los siguientes:

Sistema de entrenamiento de cirugías endoscópicas[32] Sistema desarrollado en el Instituto de Ciencias Aplicadas y Computación de Karlsruhe en Alemania. Se trata de un sistema de realidad virtual de entrenamiento de procedimientos quirúrgicos de mínima invasión (MIS, Minimally invasive surgery), que representan un mínimo riesgo para el paciente. El sistema de entrenamiento permite: coordinación de diferentes instrumentos, manipulación y navegación endoscópica con su correspondiente cámara sintética. Todo ello manipulado por un equipo de cirugía con retroalimentación háptica, un asistente computarizado y un supervisor de cámara (Ver figura 1.7).

Figura 1.7: *Sistema de realidad virtual basado en un escenario de realidad virtual de colecistectomía[32, pág. 672].*

Sistema de realidad aumentada para marcación craneal: La realidad aumentada representa una de las áreas de la realidad virtual con un alto crecimiento en aplicaciones médicas y consiste en la inserción de objetos virtuales en una escena real[55]. Un ejemplo de este tipo de sistemas, es el desarrollado en la Universidad de Tübingen en Alemania, con un sistema de marcación craneal para la planeación quirúrgica (Ver figura 1.8). El sistema utiliza la realidad aumentada para visualizar modelos de cráneos alineados a un sistema de marcaciones, en el cual, el médico o el practicante pueden elegir el tipo de marcación a realizar y a partir de ello tomar decisiones.

Figura 1.8: *Interacción de herramientas con aplicaciones de realidad aumentada en medicina. A) El usuario elige el modo de dibujado, B) el sistema registra los movimientos de la herramienta e inicia el proceso de dibujado sobre el modelo y C) termina el proceso de dibujado[55].*

Sistema de remoción de tumores en neurocirugía: Este trabajo fue desarrollado por el Instituto de Ciencias Aplicadas en Medicina de Salzburg, Austria, el Departamento de Ingeniería Mecánica del Colegio Imperial de Londres, Inglaterra y la Universidad Tecnológica de Braunschweig de Alemania. Consiste en un sistema de realidad virtual que permite al médico la planeación y ejecución de procedimientos de remoción de tumores cerebrales con la ayuda de un robot activo, llamado ROBOSIM. El robot ayuda a las labores del cirujano, que por medio de una interfaz gráfica computarizada simula los efectos de una cirugía real (ver figura 1.9).

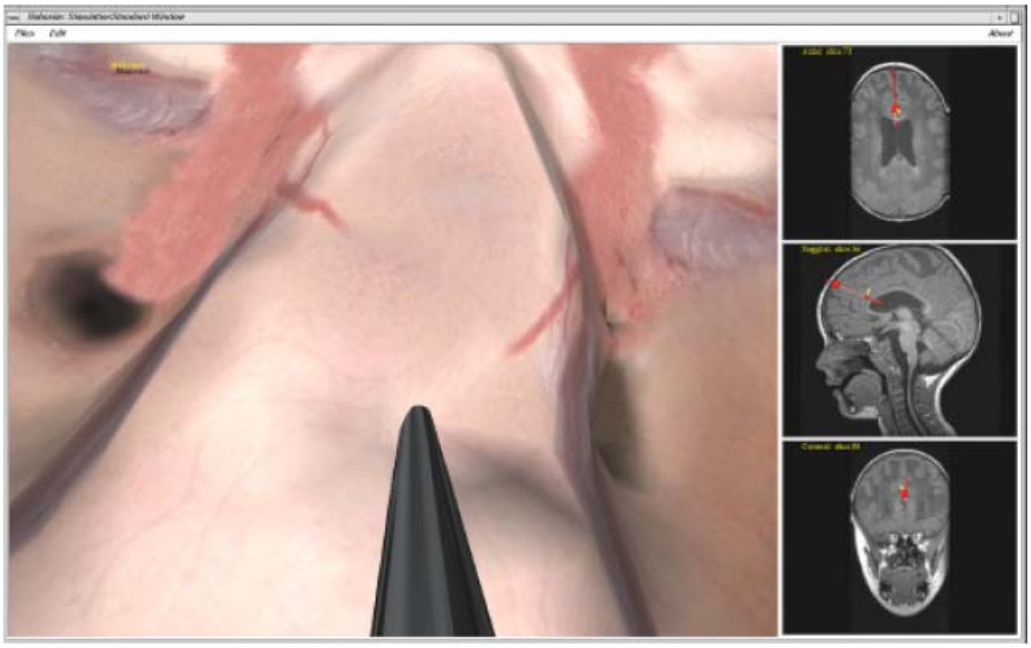


Figura 1.9: *Pantalla de simulación del ROBO-SIM: Proporciona deformación de tejido. La columna de la derecha proporciona información de la posición de la herramienta en imágenes de resonancia magnética (MR)[49, 468pp].*

Capítulo 2

El Simulador de Resección Transuretral de Próstata

El simulador de RTUP para entrenamiento del procedimiento de Resección Transuretral de Próstata, en desarrollo por el grupo de Análisis de Imágenes y Visualización del CCADET de la UNAM, consiste en dos módulos principales; el primero, una interfaz mecatrónica que simula los efectos de la herramienta quirúrgica real y el segundo que simula, mediante un ambiente virtual, el procedimiento de RTU. Este ultimo se basa en un sistema gráfico completo de modelos 3D con OpenGL y algoritmos de deformación de mallas que emulan el comportamiento "real" de una cirugía *in-vivo*.

Uno de los primeros pasos en la planeación de un entorno virtual es la descripción y caracterización del caso a simular. Esto es, con la descripción se define explícitamente el problema físico o teórico y con la caracterización se establecen los elementos que intervienen, así como las causas que lo provocan. Para el simulador de RTUP se ha empleado esta filosofía para su implementación, llevándose a cabo una investigación exploratoria del caso a simular y los elementos que intervienen en una cirugía real de este tipo. Así, a primera vista el problema de la simulación por computadora de una cirugía, se resuelve analizando cada elemento real del caso médico y después esto se transporta a un caso médico simulado o virtual.

Por tanto, en lo siguiente se describe el procedimiento de Resección Transuretral de la Próstata (RTUP) y su representación simulada. En una primera parte se abordan los elementos que intervienen, tales como: las características anatómicas de un paciente real[1] (próstata, vejiga urinaria y uretra principalmente) y las características de las herramientas quirúrgicas (el resectoscopio y sus propiedades) empleadas durante una cirugía real. En una segunda parte se explica a grandes rasgos, los elementos del simulador RTUP, basados en la investigación exploratoria y con base a los fundamentos para la implementación de una estación de simulación completa.

2.1. Descripción del caso médico

La Hiperplasia Prostática Benigna (HPB) es un padecimiento de pacientes hombres, caracterizado por un crecimiento paulatino de la Próstata y en consecuencia la obstrucción del flujo normal de la orina y/o irritación en la zona uretral. La predisposición al padecimiento comienza a partir de los 50 años de edad, presentandose en una tercera parte de la población masculina y la cifra aumenta

[1]Como paciente real se toma a aquel cuyas características físicas lo colocan como caso promedio al caso clínico propuesto y que por tanto está predispuesto al problema de salud. En algunos casos, por ejemplo, los datos a simular se toman de bases de datos anatómicas del cuerpo humano; estas bases de datos tienen como propósito la recolección de información para el uso en investigaciones médicas y propósitos de visualización científica.

considerablemente hasta el 90% a partir de los 90 años. Entre las complicaciones se encuentran:

- Obstrucción parcial o total del flujo normal de la orina.
- Infección de la vejiga urinaria por colonización de bacterias (cistitis).
- Pielonefritis o infección de la pelvis renal[2].
- Daños renales derivados de la pielonefritis: septicemia, atrofia renal, hipertensión sanguínea e insuficiencia renal.

De acuerdo a la evolución del padecimiento se distinguen tres fases:

- **Estadio irritativo:** La vejiga se vacía en su totalidad; sin embargo, el flujo de la orina disminuye su fuerza, hay goteo, la micción es más frecuente (poliuria) y llegan a presentarse evacuaciones de orina durante las noches (nicturia).
- **Incontinencia urinaria compensada:** La vejiga urinaria no se vacía por completo, quedando una pequeña cantidad llamada "orina residual". Los intervalos de micción son más cortos y el flujo de orina es menor. El paciente experimenta una mayor irritación al orinar y comienza la proliferación de bacterias debidas a la retención de la orina en la vejiga.
- **Retención urinaria descompensada:** Se presenta un desorden en el flujo urinario que finaliza con la obstrucción total del conducto urinario. Se presenta goteo continuo a través de la uretra y en su etapa más avanzada provoca insuficiencia renal.

Ante la presencia de la enfermedad, el médico responsable realiza estudios de valoración al paciente y a partir de los datos recabados se realiza la indicación del tratamiento; que puede ser: Tratamiento quirúrgico o tratamiento médico. La Tabla 2.1 muestra los tipos de procedimientos a seguir en cada tipo de tratamiento.

Tratamiento quirúrgico	Resección Transuretral (estándar)
	Prostatectomía abierta
	Electrovaporización transuretral
	Incisión transuretral
	Vaporización transuretral con láser
	Coagulación Transuretral con láser
	Resección enucleación con láser de Holmio
Tratamiento médico	$\alpha-$bloqueantes
	Finasteride
	Fitoterapia

Tabla 2.1: *Hiperplasia prostática benigna. Tratamientos. Extracto[24, p. 12].*

Para fines de este trabajo, el procedimiento que nos interesa es la *resección transuretral de próstata*. Esta cirugía se realiza por vía endoscópica. Se dice que es el tratamiento quirúrgico estándar dado que no supone heridas al paciente y presenta la menor cantidad de efectos secundarios. La única limitación se basa en el grado de crecimiento de la próstata.

Sin embargo, antes de definir los detalles del procedimiento, es necesaria una breve descripción de los órganos (vejiga urinaria y uretra) y la glándula (próstata) que intervienen; su morfología y fisiología.

[2][33] Ver LIPPERT, pág. 336.

2.1.1. Morfología y fisiología de la Vejiga

La vejiga urinaria es el órgano responsable de colectar la orina proveniente de los riñones. Tiene tres funciones fundamentales:

- Almacenar la orina hasta la micción.
- Impedir la reabsorción de la orina por parte del organismo.
- Evacuarse por completo.

La capacidad de una vejiga en estado normal varía desde los 200 ml a los 400 ml en estado de evacuación. Su forma depende de su estado y varía desde la compresión (vejiga vacía) debida a la presión ejercida por el intestino delgado; hasta tomar una forma esférica[3] (vejiga llena).

La vejiga se divide en cuatro partes [33, p. 355](Ver Figura 2.1):

- *Vértice:* Comprende desde su polo superior, aproximadamente a nivel del inicio de la pelvis. Se encuentra unida al ombligo mediante el ligamento umbilical formando una especie de vértice.
- *Cuerpo:* Comprende la mayor superficie de la pared vesical.
- *Fondo vesical:* Comprende el fondo de la vejiga con la desembocadura de los dos uréteres.
- *Cuello:* Es el punto de unión con la uretra prostática.

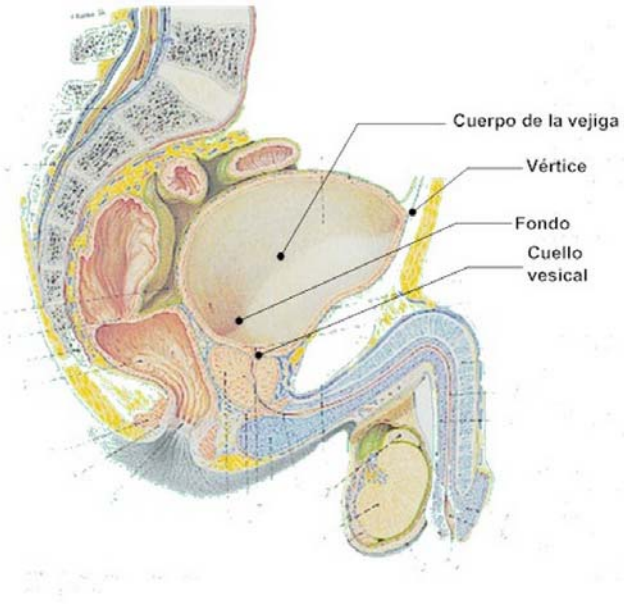


Figura 2.1: *Anatomía de la vejiga urinaria[33, p. 426]. Se distinguen las cuatro regiones que la conforman: vértice, cuerpo, fondo y cuello.*

[3]Lippert[33, p. 355] precisa, la esfera es entre todas las figuras geométricas, la que, a igual volumen, presenta la menor superficie . . . De todos modos, no debemos contar con esferas en el sentido geométrico estricto.

2.1.2. Morfología y fisiología de la Uretra

La uretra es un conducto músculo-membranoso que comunica la vejiga urinaria con el exterior. Su longitud y calibre varia de un individuo a otro. En promedio, el diámetro de las porciones que la conforman varía entre 7 a 15 mm. Los instrumentos de exploración que se emplean en urología, tienen diámetro máximo de 10 mm, lo que les permite introducirse en la mayoría de las uretras [2, p. 19].

La uretra masculina se divide en tres secciones[33, p. 425] (Ver figuras 2.2 y 2.3):

- **Uretra Prostática:** Atraviesa la próstata en unos 3 cm. Es una sección ligeramente dilatada y presenta una ligera curva cóncava. La sección transversal tiene forma de herradura. Dentro de la próstata, la cresta uretral está engrosada en forma de colina, por lo que a esta zona se le conoce como *veru montanum*. A los lados de este último se localizan los orificios que son la desembocadura de la mayoría de los conductos prostáticos[2, p. 20].
- **Uretra membranosa:** Se encuentra rodeada por el esfínter uretral estriado, entre la salida de la uretra de la próstata y su entrada en el cuerpo esponjoso de la uretra.
- **Uretra esponjosa:** Compone el tubo esponjoso del conducto uretral que desemboca en el glande a través del orificio uretral externo. Poco antes de desembocar, el tubo tiene forma dilatada en forma de fosa navicular (lat. *navis=barco*).

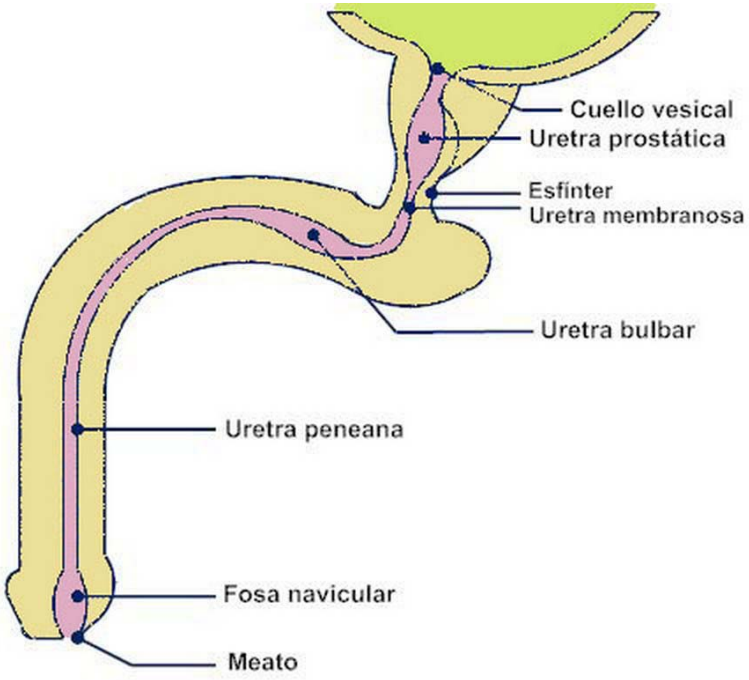


Figura 2.2: *Corte de la uretra[2, p. 19]. Se identifican las principales zonas que la conforman y sus interconexiones con otros órganos.*

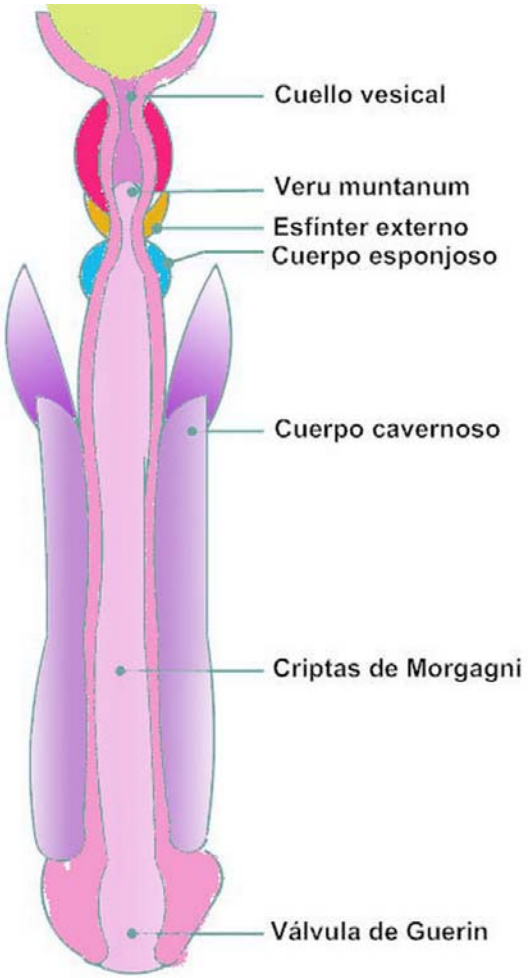


Figura 2.3: *Vista longitudinal de la Uretra masculina[2, p. 21]. Se puede observar el cambio de longitudes transversales del conducto uretral, desde la base hasta la zona del cuello vesical.*

2.1.3. Morfología y fisiología de la Próstata

La próstata es una glándula masculina de secreción externa, responsable del 40% del líquido seminal. Se encuentra situada alrededor de la uretra posterior[67, p. 298] y unida al cuello de la vejiga urinaria. Tiene forma de cono aplastado de delante a atrás y cuya base está dirigida hacia arriba. Su tamaño promedio es de 4 cm de ancho, 3 cm de largo y 2 cm de espesor.

Anatómicamente, a la próstata se le consideran cuatro zonas[2, p. 27] (ver figura 2.4):

- La *zona periférica*: Ocupa la mayor cantidad de volumen y se situa en la parte posterior y parcialmente a los lados del conducto uretral.
- La *zona central*: Se situa por encima de la zona periférica y es la unión casi completa con el cuello vesical; únicamente separada por la zona de transición con el conducto uretral posterior.
- La *zona de transición*: Es una zona pequeña que rodea el conducto uretral y se une con el cuello vesical. A su vez se encuentra rodeada por la zona central. Entre esta zona y la zona periférica se localiza el veru montanum.
- La *zona anterior*: Se localiza en la porción anterior de la próstata. Es la segunda en tamaño después de la periferia y se extiende a los lados en forma de una capa delgada que constituye la cápsula prostática.

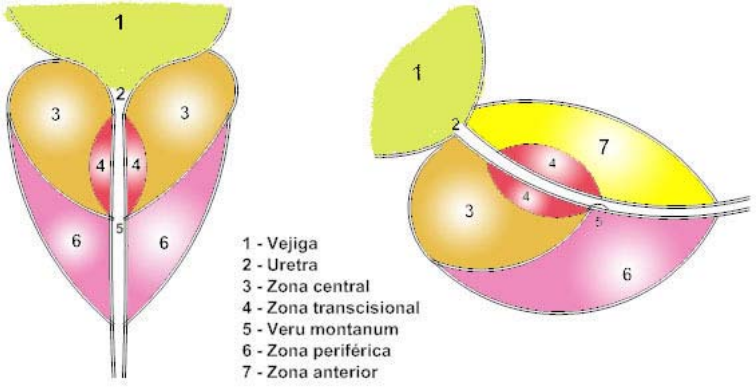


Figura 2.4: *Anatomía de la próstata[2, p. 27]. Se distingue la forma y sus divisiones.*

2.1.4. El procedimiento de resección transuretral de próstata

La evolución de las cirugías vía transuretral ha sido todo un reto a través de la historia para la comunidad médica, debida principalmente a la dificultad que implican los métodos empleados y el desarrollo de instrumentos confiables capaces de asegurar los mínimos efectos postoperatorios en el paciente.

El origen de este tipo de procedimientos se remonta a miles de años atrás en las principales civilizaciones como la china, la egipcia y la hindú. Sin embargo fue en la isla de Cos en el mar Egeo, donde el griego *Erasistos* alrededor del año 300 a.C., empleaba un tubo de plata en forma de S, cuyas curvaturas se adaptaban al sistema urinario del varón y realizaba el procedimiento de cateterización. En la India, 600 años después, *Sushruta Samhita* describe el uso de tubos de oro y plata lubricados

con manteca, así como los primeros procedimientos para el tratamiento de las llamadas "callosidades de la uretra"[4] [12, p. 142]. A partir de entonces los esfuerzos por el desarrollo de procedimientos y herramientas para el tratamiento de transtornos del sistema urinario han sido constantes. La Tabla 2.2 muestra un compendio de estos avances a lo largo de la historia hasta la creación del resectoscopio.

Hoy en día El procedimiento de resección transuretral de próstata (RTUP), es el procedimiento por excelencia entre la comunidad médica pára el tratamiento quirúrgico de la HPB. Consite en una cirugía en la cual el especialista introduce vía transuretral el resectoscopio [5] y remueve parte del tejido blando de los lóbulos hipertrofiados de la próstata, que obstruyen la uretra. La Figura 2.5 muestra el esquema básico del procedimiento de RTUP y la Figura 2.6 esquematiza el protocolo a seguir por el médico especialista[6].

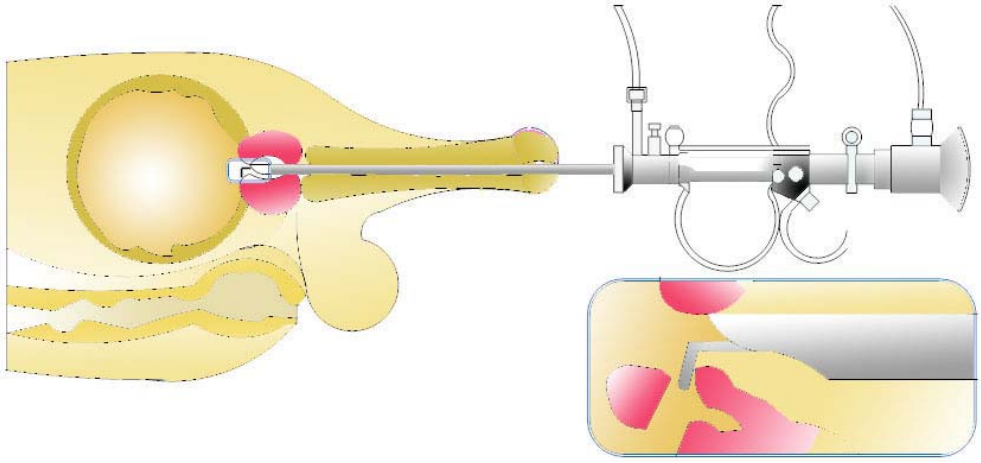


Figura 2.5: *Resección Transuretral de la Próstata[67, p. 305]. En el recuadro se puede observar el corte que se realiza sobre la próstata.*

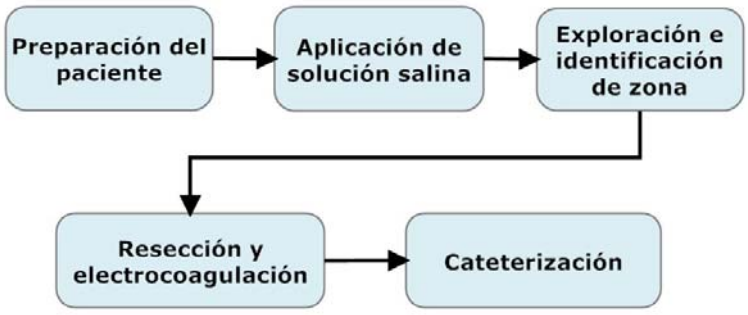


Figura 2.6: *Protocolo médico de una cirugía de RTUP. La aplicación de solución salina contribuye a una mejor visualización durante la cirugía, además de contribuir al buen funcionamiento de la herramienta de corte.*

[4]Término antigüamente empleado para describiur las obstrucciones del flujo normal de la orina. Fue hasta 1564 cuando *Ambrosio Paré*, médico cirujano francés, considerado el "Padre de la Cirugía Moderna" sugirió la implicación de la próstata como órgano responsable de la obstrucción del conducto urinario. [12, p. 142]

[5]El resectoscopio es un dispositivo quirúrgico, indicado para extraer tejido obstructivo del sistema urinario. Su etimología (de acuerdo por el diccionario *Durvan*) surge de la raíz latina "resectio" que significa acción de cortar y del griego "skopio" que significa mirar o examinar.

[6]Para mayor información sobre el protocolo médico consulte la tesis *"Interfaz mecatrónica para un simulador de cirugía de próstata"*[1, Págs. 8,9].

Año	Procedimiento
300 a.C.	El griego *Erasistos* emplea una sonda de plata en forma de S para cateterización de la uretra.
30 d.C.	El romano *Cornelio Celso* describe el uso de catéteres de bronce con curvatura adaptada a hombres y otros para mujeres. Además describe el procedimiento que hoy en día se sigue empleando para la introducción de estos catéteres.
300 d.C.	El hindú *Sushruta Samhita* describe el uso de tubos de oro y plata lubricados con manteca.
980	El médico árabe *Alli Abulkasis* inventa un instrumento metálico al que llamó almul. El instrumento permitía el sondeo, cateterización y diagnóstico de la litiasis vesical (Obstrucción del flujo urinario debido a la presencia de piedras en la vejiga).
1020	El médico árabe *Ali Alhusein* Ibn (mejor conocido como *Avicena*), describe la cateterización con sondas blandas y maleables, lubricadas con leche y con "maniobras suaves".
1564	El médico francés *Ambrosio Paré*, describe el papel de la próstata en la obstrucción del conducto uretral.
1575	*Paré* describe un procedimiento transuretral utilizando una sonda rígida con una cresta afilada en la punta que permitía "roer" tejido obstructivo en el cuello de la vejiga.
1575	El español *Francisco Díaz* describe la práctica de la uretrotomía interna, mediante una fina sonda metálica y una pequeña hoja oculta.
1800	El francés *Georges de la Faye* describe una sonda metálica curva que poseía un estilete triangular.
1836	El francés *Louis Auguste Mercier* desarrolla un instrumento similar al de Faye y un instrumento para el corte de una porción del tejido prostático.
1874	*Erico Bottini* inventa el primer instrumento de corte usando corriente eléctrica sobre una hoja de corte en su extremo curvo.
1897	En *Berlín Freudenberg* mejora la herramienta de Bottini agregando una lente para observación directa durante la cirugía.
1909	*Hugh Hamton Young* realiza los primeros procedimientos endoscópicos con un uretroscopio modificado, al que después agregó un motor rotativo para remover tejido prostático.
1910	*Edwin Beer* anuncia el éxito de una herramienta basada en corriente de alta frecuencia aplicada sobre un tumor de vejiga. El instrumento además tenía la capacidad de trabajar "bajo agua" y con iluminación.
1920	*W. Braasch* y *J. Caulk* desarrollan una herramienta de raspado, corte y coagulación al tiempo que cortaba.
1926	*Maximilian Stern* describe un aparato capaz de trabajar bajo el agua y provisto de un ansa movible de alambre de tungsteno capaz de extirpar fragmentos longitudinales de tejido, al que llamó: ***resectoscopio***.
1997	Investigadores del mundo desarrollan en el Colegio Imperial de Ciencia, Tecnología y Medicina Mecatrónica en Medicina de Londres, el primer robot activo para cirugía de resección de próstata: PROBOT[48].

Tabla 2.2: *Evolución de las herramientas y metodologías para el tratamiento de transtornos urinarios por métodos transuretrales. Extracto, [12, Ver pags. 142-144].*

Una de las razones por las cuales el procedimiento tiene una gran aceptación por los especialistas es su simplicidad y el tiempo que conlleva. Sin embargo, una de las problemáticas no es en sí el procedimiento sino su entrenamiento. Este último consiste en un proceso largo de aprendizaje, en el cual los estudiantes residentes adquieren la habilidad, en primera instancia de controlar, coordinar y orientar el resectoscopio; para después aprender a realizar el procedimiento quirúrgico con todas las implicaciones técnicas y clínicas que conlleva, como las posibles complicaciones y efectos secundarios resultantes de una mala cirugía. De acuerdo al médico urólogo Sergio Durán, del Instituto Nacional de Rehabilitación, los residentes en urología pasan por un proceso de aprendizaje se divide en 4 años. En el primer año, los estudiantes aprenden a realizar la exploración endoscópica y la manipulación de la herramienta con modelos artificiales. En el segundo año se dedican específicamente al tratamiento de patologías y procedimientos médicos. En el tercer año, el alumno recibe la asesoría necesaria para realizar una RTUP; y en el último año aprende a realizar cirugías de próstata de acuerdo con la patología específica del paciente.

La metodología tradicional de entrenamiento consiste en la observación, y eventual participación del alumno, en cirugías *in-vivo* realizadas por el médico especialista. Ello conlleva a un aprendizaje poco práctico y en ocasiones abstracto, al realizarse con materiales didácticos muy poco realistas o bien modelos físicos artificiales que tienen un elevado costo. Es entonces que cobra importancia la necesidad de un simulador que permita a los alumnos una experiencia más próxima a lo que realmente enfrentará ante el paciente. El reto del simulador es enorme y tiene un sinfin de implicaciones desde el punto de vista técnico hasta el punto de vista psicológico y médico, al tratar de hacer creer al aprendiz que se encuentra en una cirugía muy cercana a la realidad y que los resultados que devuelva el sistema correspondan al de un paciente real.

2.2. Los elementos del simulador de RTUP

Como se vió en el capítulo anterior, la arquitectura básica de un simulador de propósito general está compuesto por tres elementos fundamentales: Un conjunto de dispositivos de retroalimentación de fuerzas, una estación de trabajo o de cálculo y una estación gráfica. El simulador TURP se basa en dicha arquitectura y su implementación trata de simular de manera completa el proceso de exploración que lleva a cabo un médico residente en urología en su primer año de aprendizaje.

Así podemos esquematizar al simulador como un conjunto de bloques funcionales, los cuales interaccionan para formar un sistema completo de simulación en tiempo real. La Figura 2.7 muestra un esquema general de funcionamiento del simulador, en donde se resalta la integración del bloque de gráficos con la estación gráfica. Así, aunque el simulador se base en la arquitectura de tres bloques, en general se puede hablar de dos módulos principales: la interfaz mecatrónica y el ambiente virtual.

En las siguientes secciones se describen a grandes rasgos, los bloques funcionales del simulador y sus implementaciones actuales. Para fines de esta tesis, especializada en la descripción del ambiente gráfico virtual (bloque de gráficos y parte de la estación de cálculos), se ofrece un panorama muy superficial para después entrar en detalles en los capítulos posteriores.

2.2.1. La interfaz mecatrónica

La interfaz mecatrónica del simulador de RTUP es un desarrollo del Grupo de Análisis de Imágenes (GAIV) y Visualización del Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la UNAM(Ver Figura 2.8). El proyecto se compone de dos partes principales: un dispositivo mecánico que simula el instrumento de corte (resectoscopio) y un protocolo de comunicación serial con el ambiente gráfico virtual. Estos dos últimos subsistemas relacionados mediante una interfaz electrónica que convierte las

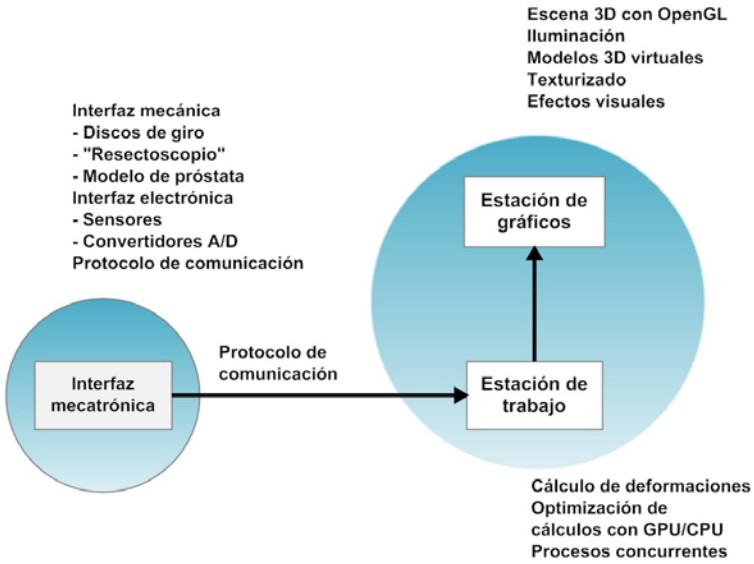


Figura 2.7: *Bloques funcionales del simulador. Se distinguen los dos módulos principales y los procesos que cada bloque involucra.*

señales analógicas y digitales provenientes de los sensores situados en el dispositivo mecánico, en una señal digital contenedora de la información que se transporta a la estación de trabajo.

Figura 2.8: *Interfaz mecatrónica completa[1, p. 14].*

La imagen 2.9 muestra un esquema completo de la interfaz mecatrónica implementada en 2007 por Altamirano del Monte[1]. En su trabajo define los principios básicos de implementación de una interfaz, capaz de simular los movimientos reales que un médico lleva a cabo con el resectoscopio en una cirugía de RTUP, su registro mediante sensores y la trascripción de la información con un arreglo de microcontroladores PIC.

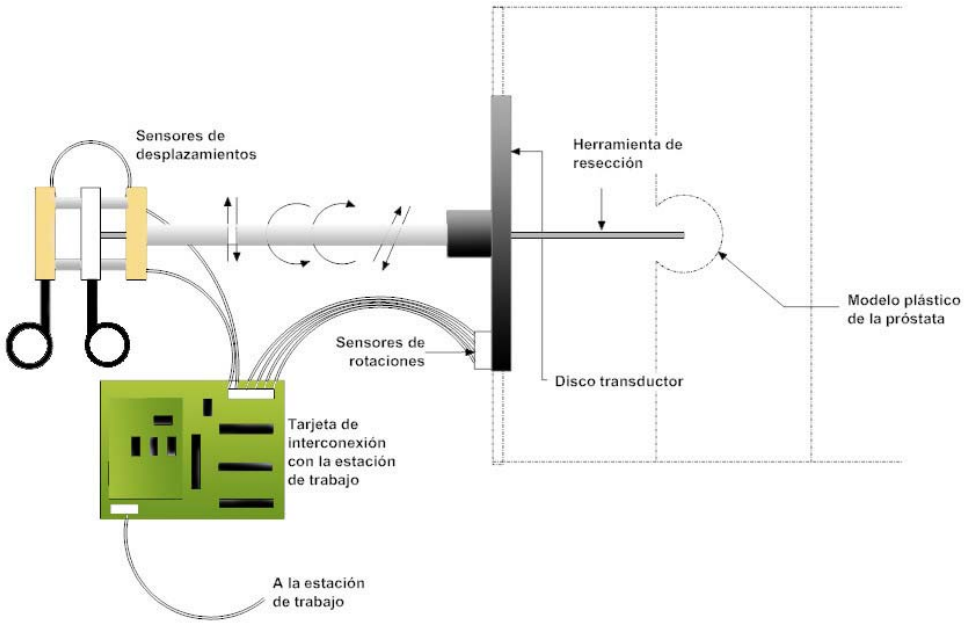


Figura 2.9: *Esquema de la interfaz mecatrónica del simulador RTUP. El resectoscopio virtual, consistente en un tubo rígido, es sensado conforme se mueve sobre su base. Los registros incluyen información de rotaciones y desplazamientos.*

En este mismo año, Reyes[50] describe el protocolo de comunicación y la calibración de la interfaz mecatrónica con la estación gráfica. El protocolo implementado sobre comunicación serial RS-232C en lenguaje C, se basa en el análisis del modelo gráfico 3D y su adaptación con los datos entregados por la interfaz mecatrónica.

Altamirano[1, pág. 55] concluye, "El trabajo de instrumentación de la interfaz mecánica, el diseño electrónico, y su conexión con el modelo en 3D completan el primer prototipo pasivo de simuladores de cirugía y establece una línea de investigación y desarrollo tecnológico del Grupo de Análisis de Imágenes y Visualización del CCADET de la UNAM además de las bases de proyectos futuros.". Actualmente esta interfaz se encuentra en fase 2 de desarrollo con un prototipo con mayor estabilidad, menor grado de errores de sensado, la capacidad de retroalimentación de fuerzas y un protocolo de comunicación USB[7].

Otra de las alternativas, actualmente en fase de desarrollo, para la representación del efecto de retroalimentación de fuerzas y el resectoscopio real, consiste en el uso de un dispositivo háptico[8]. La imagen 2.10 muestra el PHANTOM Omni®[45] de la empresa Sensible Technologies, especializada en dispositivos de interfaz humana. Este dispositivo emplea el protocolo de comunicación IEEE-1394a FireWire® y es ideal para interfaces gráficas al poseer un módulo programable que relaciona la parte gráfica con la parte háptica. En el caso del simulador, el dispositivo proporciona los movimientos con los grados de libertad necesarios para simular el efecto de una herramienta real; así como los efectos de sensación táctil resultantes de las colisiones mostradas visualmente en el ambiente virtual.

[7] Universal Serial Bus.

[8] Los dispositivos hápticos son empleados en aplicaciones de Realidad Virtual y su objetivo es servir como interfaz entre la computadora y el ser humano mediante el sentido del tacto.

Figura 2.10: *PHANTOM Omni®. Sensible Technologies 2007. El dispositivo háptico tiene la capacidad de registrar los movimientos del usuario y representar un efecto de oposición ante ciertas condiciones definidas en el programa de simulación.*

2.2.2. La estación de cálculos

Uno de los retos más grandes en el desarrollo de un simulador, es transportar el mundo "real" en un mundo totalmente "virtualizado", lo que conlleva a crear métodos de transferencia basados en las propiedades físicas reales y en la aplicación de efectos tanto internos como externos que influyen en el comportamiento del sistema. Para ello, el simulador cuenta con el bloque de cálculos optimizado con GPU (Graphics Processing Unit), el cual tiene como propósito la ejecución de algoritmos de deformación de mallas de volumen y el procesamiento de todas las variables físicas que intervienen en el proceso de simulación en tiempo real.

Sin duda, uno de los algoritmos más complejos usados en el simulador de RTUP (que implica tiempo de cómputo y recursos del procesador), es el algoritmo de deformación de mallas de volumen empleando el método de masas y resortes. Este método, que realiza una analogía entre una malla geométrica tridimensional y un conjunto de partículas conectadas mediante resortes, resuelve la asignación de propiedades estáticas (propiedades del movimiento) y dinámicas (fuerzas que causan el movimiento) al modelo 3D de la próstata virtual; las cuales, al estar constantemente sometidos a un algorítmo de detección de colisiones, proporcionan al simulador un grado de realismo adecuado (Ver figura 2.11).

Sin embargo, este bloque va más allá de la complejidad de los cálculos numéricos, al proveer una metodología de ejecución de procesos concurrentes o multihilos, en los cuales se ejecutan paralelamente tareas de lectura del puerto serie (de los datos provenientes de la interfaz mecatrónica) y tareas de cálculo numérico con GPU/CPU(Ver figura 2.12). Lira[34] describe detalladamente las ventajas del uso de programación concurrente y el uso de la GPU como herramienta de cálculo, aplicadas al caso específico del simulador.

Así, este bloque provee un procesamiento y una comunicación optimizada, entre los datos que se reciben de la interfaz mecatrónica y el comportamiento del sistema gráfico de realidad virtual, al resolver el problema de la sincronización entre lo que el usuario manipula (resectoscopio mecatrónico virtual) y el aumento de la frecuencia de refresco del dispositivo de visualización, al pasar de 15 cuadros/segundo a una versión más realista que superó los 30 cuadros/segundo[9].

[9]Se toma como estándar el valor de 30 cuadros/segundo como el suficiente y necesario para lograr el efecto de sensación

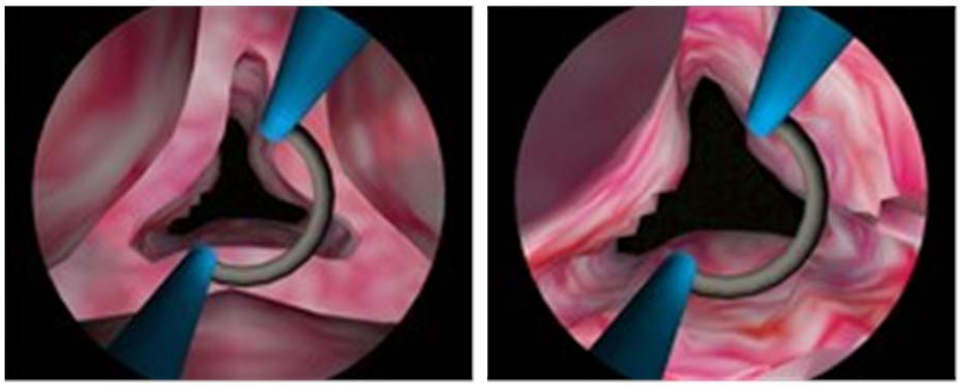


Figura 2.11: *Simulador gráfico de RTUP en ejecución en tiempo real. A la izquierda, modelo de la malla sin deformar. A la derecha, modelo de la malla deformada mediante el método de masas y resortes.*

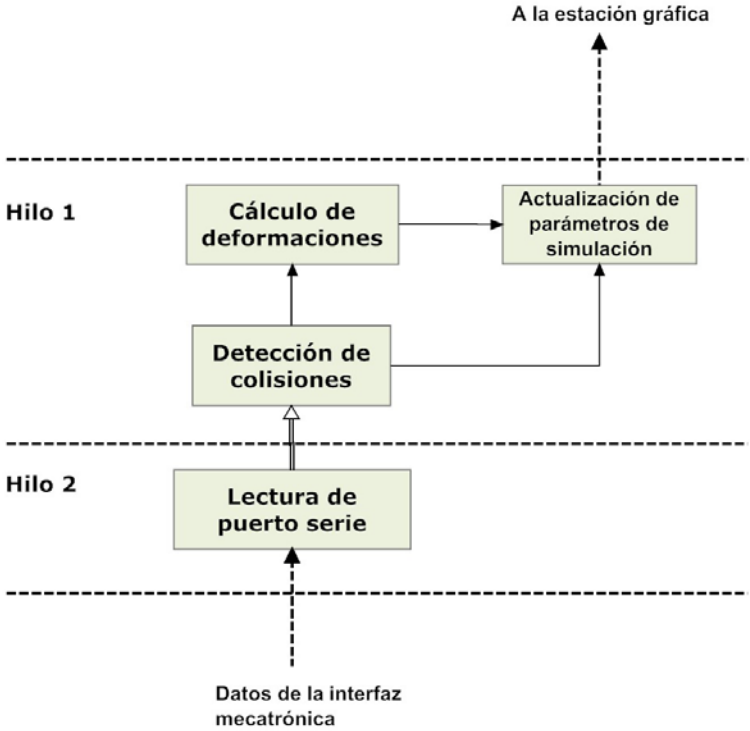


Figura 2.12: *Esquema funcional del bloque de cálculos del simulador RTUP. Los procesos concurrentes ofrecen la ventaja de mayor realismo al simulador completo.*

2.2.3. La estación Gráfica

La vista, es uno de los sentidos más difíciles de engañar en el desarrollo de un ambiente virtual; por ello, la información empleada para simular lo que el usuario visualizará en pantalla, debe corresponder a un caso real en todos los sentidos, desde la perspectiva de visión durante una cirugía real hasta los mínimos detalles de la escena; tales como la iluminación, el texturizado, la forma de los modelos, el control de la herramienta y otros efectos visuales que den la impresión de inmersión con el sistema de realidad virtual..

Así, la parte gráfica del simulador es uno de los puntos más críticos para lograr un realismo adecuado. Su dificultad de implementación, radica en la tarea de simular tejidos con un comportamiento muy parecido al real, en este caso, tejido blando de la próstata, la uretra y la vejiga. No sin antes llevar a cabo el proceso de reconstrucción de los modelos anatómicos tridimensionales, lo cual implica la búsqueda de información de los órganos involucrados, el procesamiento de esta información y la aplicación de algoritmos de reconstrucción eficientes que ofrezcan un resultado aceptable. Por tanto, el simulador de RTUP cuenta con un bloque totalmente funcional, consistente en un sistema gráfico virtual que representa gráficamente la primera etapa que lleva a cabo un residente de urología durante su entrenamiento: la exploración.

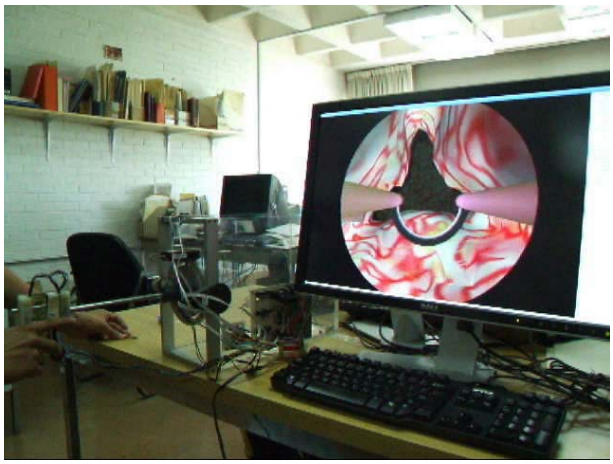


Figura 2.13: *Simulador de Realidad Virtual de RTUP completo del GAIV del CCADET. El bloque de gráficos consiste en el despliegue de la escena 3D con OpenGL en un monitor estándar de computadora.*

El sistema se basa en el despliegue de gráficos 3D en un monitor estándar de computadora (Ver figura 2.13), usando para ello la biblioteca de gráficos abierta *OpenGL* y programación estructurada en lenguaje C. La gran ventaja del uso de *OpenGL* en el desarrollo del ambiente virtual, fue su facilidad de programación y su potencia para generar la escena 3D durante el *rendering*[10] final. El manejo de la cámara virtual, la iluminación (incluyendo sus componentes), los efectos visuales, la optimización del proceso de dibujado, etc., son algunas razones de peso por las cuales este sistema de realidad virtual empleó dicha biblioteca.

Es de destacarse, dentro de la funcionalidad del software de simulación, el uso de una interfaz gráfica controlable sin la necesidad del dispositivo mecatrónico; esto es, mediante ratón y teclado (Ver

de movimiento continuo durante una visualización en pantalla. Un valor menor en la frecuencia de refresco da un efecto de movimientos entrecortados y visualmente poco realistas.

[10]En cómputo gráfico, el *rendering* es el proceso de obtención de una imagen en dos dimensiones, a partir de la información de un espacio de tres dimensiones. Se puede hacer una analogía de este proceso con una cámara fotográfica, en la cual, el mundo representa la información 3D y se plasma en una imagen en 2D.

figura 2.14). Ello, aunque que no proporciona un realismo físico, encuentra una gran aplicación para fines de visualización y enseñanza del modo de exploración para un estudiante, al poder éste, identificar virtualmente la zona en la cual se encuentra y poder explorarla como si de un videojuego se tratara.

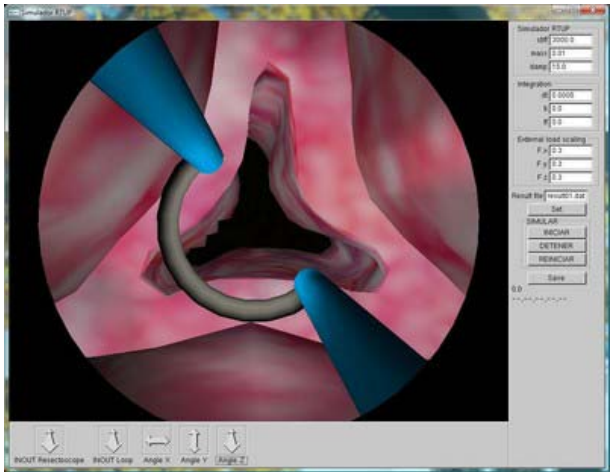


Figura 2.14: *Software de simulación de RTUP. La interfaz gráfica de usuario (GUI), permite el control total de la escena (sin necesidad de la interfaz mecatrónica) con el uso del ratón y el teclado.*

Figura 2.15: *Primera versión del software de simulación de RTUP. El paquete incluye: CD de instalación y manual de usuario. Diseños de portada propios.*

Los modelos tridimensionales que se visualizan en pantalla, corresponden a los dos órganos (vejiga y uretra), la glándula (próstata) y el resectoscopio virtual, que intervienen durante un procedimiento quirúrgico de resección transuretral. Éstos fueron resultado de la reconstrucción a partir de imágenes en dos dimensiones de las estructuras anatómicas reales, tomadas de una base de datos. En el caso de la próstata, se empleó un método de muestreo y la técnica de tetraedrización de Delaunay. En su trabajo de investigación, Soriano[56] explica a detalle el procedimiento empleado y los resultados obtenidos al generar una malla de volumen de la próstata, lista para su inclusión en el simulador. Los otros dos órganos, la vejiga y la uretra, son resultado de la aplicación de un método de reconstrucción de superficies, abordado más adelante en este trabajo de tesis. El resectoscopio virtual se basa en el asa de corte real y se generó a partir de un algoritmo mallador de cilindros y toroides.

Actualmente, el simulador gráfico cuenta con una primera versión, lanzada en Julio de 2008 (Ver

figura 2.15), la cual está disponible por el momento como parte de un proyecto de desarrollo institucional en el CCADET. Próximamente se espera que esta versión del software pueda ser implementada en el observatorio de visualización IXTLI[11] (Ver figura 2.16), donde uno de los objetivos es lograr el mayor grado de inmersión por parte del usuario dentro del ambiente virtual.

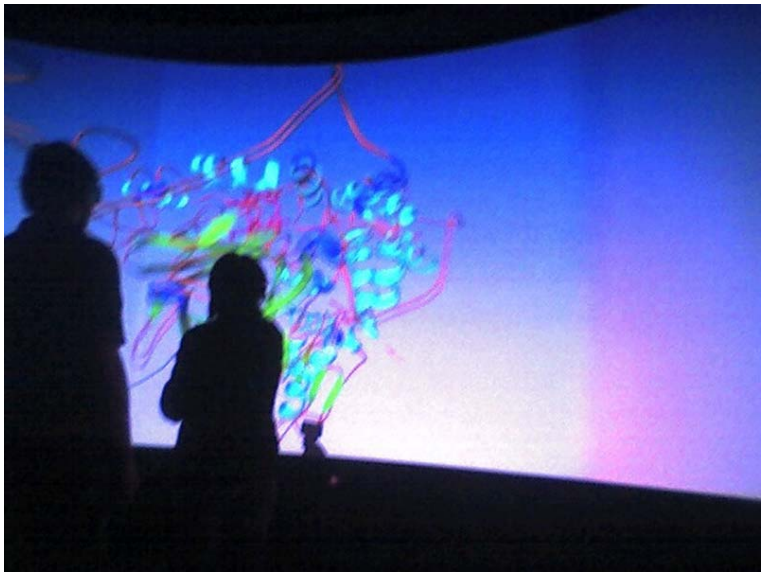


Figura 2.16: *Observatorio de Visualización Ixtli de la UNAM [27]. El observatorio de visualización permite el mayor grado de inmersión del usuario en un ambiente virtual.*

[11]IXtli, el Observatorio de Visualización de la UNAM, es una sala de alta tecnología diseñada para visualizar y simular objetos complejos e imágenes en tercera dimensión (3D), mediante un sistema de realidad virtual inmersiva. Las imágenes son proyectadas en mono o en estéreo en una pantalla semicilíndrica de 140 grados, que mide 8.90 mts. de largo por 2.55 mts. de ancho. La forma cilíndrica permite cubrir gran parte del campo de visión del usuario lo que le ayuda a sentirse dentro del ambiente virtual [27].

Capítulo 3

Modelado de Mallas 3D a partir de Isosuperficies

El desarrollo de las ciencias médicas, en gran medida se ha beneficiado gracias a la inclusión de las nuevas tecnologías de la información. Los nuevos dispositivos médicos computarizados han permitido el estudio, diagnóstico y tratamiento de diversas enfermedades del cuerpo humano, siendo de gran importancia el auge de las nuevas generaciones de simuladores médicos. Estos últimos, requieren de modelos anatómicos totalmente virtuales, cuya representación sea realista al ojo humano y además se comporte de manera similar a como lo haría una estructura anatómica real.

La principal fuente de información para estos casos: las imágenes; tienen la particularidad de ser representaciones en dos dimensiones de las estructuras reales, tomadas a intervalos de distancia suficientes como para definir su forma y posición en el espacio de tres dimensiones. Esta tarea no es fácil, y en gran medida se requiere de una buena dosis del procesamiento de las imágenes, tarea de la imagenología. Thalmann [61] dice, el reto para la imagenología es proporcionar capacidades y técnicas avanzadas para la adquisición, el procesamiento, visualización y análisis de imágenes biomédicas que permitan la extracción confiable de información científica y clínica.

Así, podemos definir el primer paso para la reconstrucción de estructuras anatómicas: la búsqueda de una metodología acorde a la información disponible. En la actualidad, existen diversas técnicas de reconstrucción, siendo una de ellas y con gran auge entre la comunidad científica, el "algoritmo de *Marching Cubes*" o de cubos marchantes. Gastélum[14, p. 31] describe: Este algoritmo se basa principalmente en el recorrido de un volumen, voxel a voxel[1] (De ahí el nombre de "cubos marchantes"), construyendo una superficie, en malla triangular, que delimita los objetos contenidos (componentes conexas).

La funcionalidad y el amplio campo de aplicación del algoritmo de *Marching Cubes* lo hacen la opción ideal para la reconstrucción de superficies 3D a partir de isosuperficies; sin embargo, algunas de sus desventajas son: la gran cantidad de triángulos generados y el tiempo de cómputo necesario para el proceso de reconstrucción cuando se trata de un conjunto muy grande de isosuperficies. Por ello, el objetivo principal de este capítulo se centra en el desarrollo de una metodología simple y alternativa a otros métodos de reconstrucción de superficies triangulares (como el algorítmo de *Marching Cubes*), que implique un menor tiempo de proceso y nos proporcione buenos resultados. En este caso, la importancia de la metodología fue para la reconstrucción de un modelo de la vejiga urinaria para su inclusión en el simulador de cirugía de próstata. La figura 3.1 nos muestra un esquema de la metodología de reconstrucción empleada; partiendo del espécimen humano (estructura real) hasta la obtención de una malla de superficie (estructura virtual) totalmente generada por computadora.

[1] Voxel es el término empleado para definir un elemento mínimo de volumen en el espacio de tres dimensiones. Equivale al píxel (elemento punto) en el espacio de dos dimensiones.

Para la implementación de esta metodología, se empleó un conjunto de herramientas de cómputo, enfocadas al procesamiento de imágenes y edición de mallas triangulares 3D. Las herramientas empleadas fueron: ImageJ, para el procesamiento de las imágenes anatómicas; MATLAB, para implementar el algoritmo de muestreo; y Blender, para la edición final de la malla. Todas estas implementaciones se describen en los apéndices A y B al final de este trabajo de tesis.

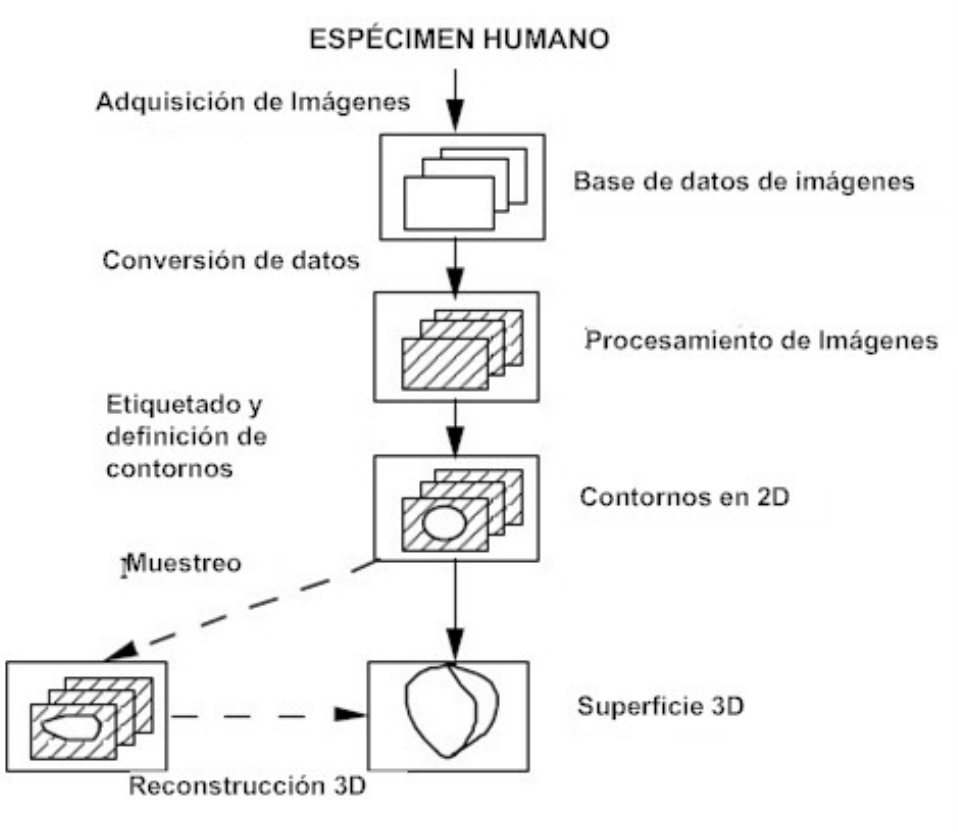


Figura 3.1: *Reconstrucción 3D de modelos anatómicos empleando una técnica de contornos[61].*

3.1. Extracción de contornos

De acuerdo a la figura 3.1, el proceso para la obtención de contornos que representen información de un modelo anatómico, conlleva a una serie de pasos. Inicialmente se parte de un espécimen humano, que representa sólo una muestra de la población; y cuyas características permiten el uso confiable de los datos. Este proceso de selección y adquisición de la información, se lleva a cabo por especialistas en diversas ramas, quienes a partir de estudios estadísticos y morfológicos, determinan la muestra.

Una vez obtenido un conjunto de imágenes a partir de la muestra, el siguiente paso es la conversión de la información a un formato adecuado para su análisis y procesamiento con alguna herramienta de cómputo. Hoy en día, existen una gran variedad de herramientas que nos auxilian en la programación de algoritmos de procesamiento de imágenes y otras que ya poseen módulos implementados para

llevar a cabo dicho proceso; tal es el caso de *ImageJ*[2] y *MATLAB*[3]. La figura 3.2 muestra un esquema completo de la metodología empleada en esta tesis para la extracción de contornos.

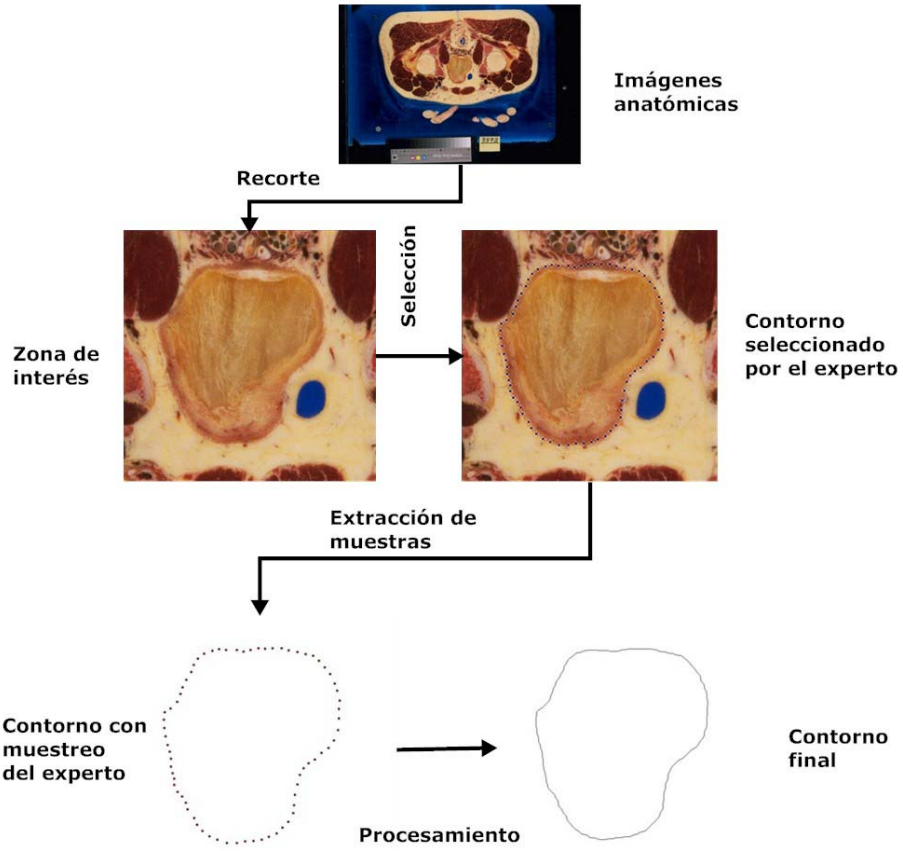


Figura 3.2: *Metodología empleada para la extracción de contornos a partir de imágenes anatómicas.*

3.1.1. El Proyecto del humano visible, The Visible Human Project (VHP)

El Proyecto del Humano Visible, conocido como VHP [62] (*Visible Human Project*®), es un trabajo realizado en la Biblioteca Nacional de Medicina de los Estados Unidos, consistente en la creación de un atlas completo del cuerpo humano, tanto masculino como femenino. Todo ello a partir de imágenes de Resonancia Magnética (MRI, *Magnetic Resonance Imaging*), Tomografía Computarizada (CT, *Computer Tomography*) e imágenes anatómicas tomadas de cadáveres (Ver figura 3.3).

[2] *ImageJ*[25] es un programa para el procesamiento y análisis de imágenes. Contiene algoritmos de procesamiento programados en Java y también ofrece la oportunidad de anexar módulos propios.

[3] *MATLAB*[36] es un lenguaje de programación de alto nivel y un entorno de desarrollo integrado (IDE), que permite el procesamiento de datos de manera rápida y eficiente. La programación y ejecución de algoritmos de procesamiento de imágenes se lleva a cabo mediante su *Image Processing Toolbox*.

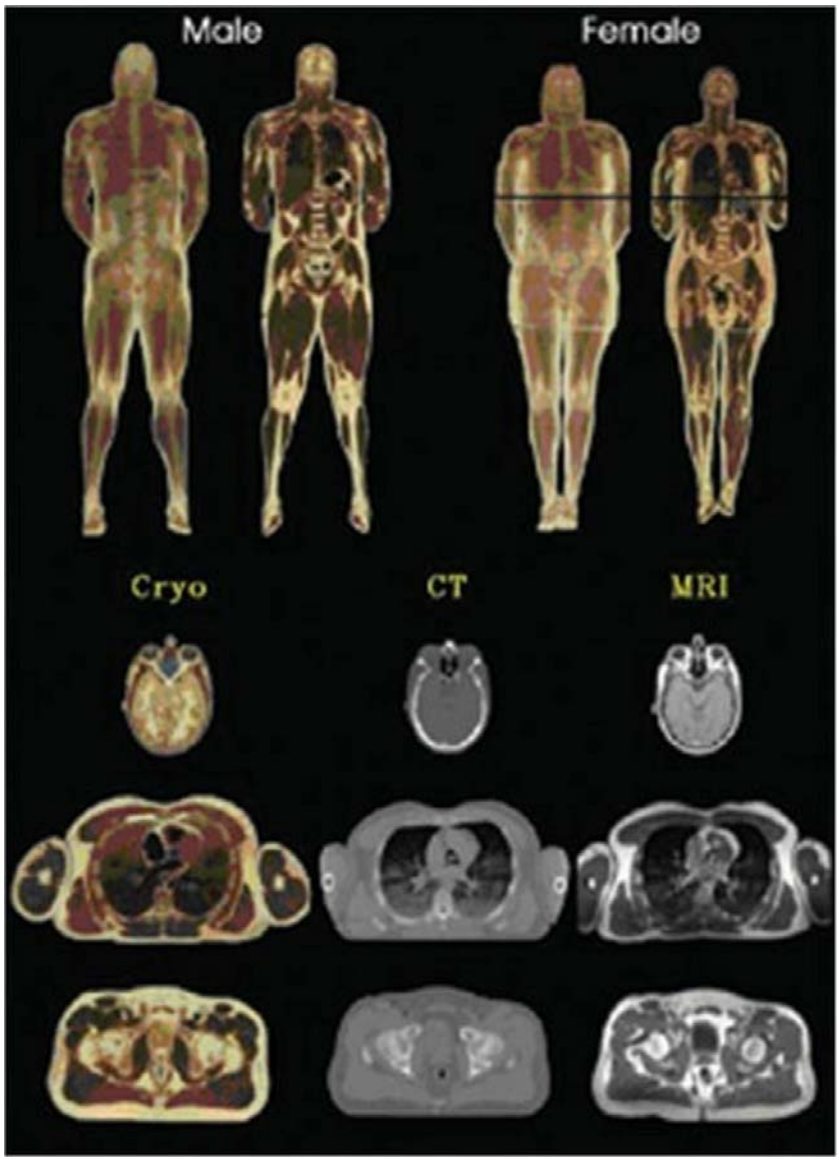


Figura 3.3: *Conjunto de imágenes del Visible Human Project. Arriba: Imágenes computarizadas del cuerpo humano completo del hombre (izquierda) y la mujer (derecha) (VHM:Visible Human Male y VHF:Visible Human Female), a partir de fotografías anatómicas. Abajo: Imágenes del cuerpo del hombre tomadas con diferentes técnicas (Criosección, Tomografía Computarizada y Resonancia Magnética), a intervalos de 1 mm [51, p.242].*

Los cadáveres usados fueron considerados como "normales" y representativos de todo el conjunto poblacional. El primer cuerpo seleccionado provenía del estado de Texas, en los Estados Unidos, tenía una edad de 38 años, 186cm de estatura y pesaba 90kg. A este conjunto de datos se le denominó VHM (*Visible Human Male*). La resolución del píxel en cada imagen era de 0.33mm x 0.33mm, tomadas a intervalos de 1mm (Ver figura 3.4) En el caso de la mujer, su edad era de 59 años, 167cm de estatura y 72kg de peso A este conjunto de datos se le denominó VHF (*Visible Human Female*). La resolución del píxel en cada imagen era de 0.33mm x 0.33mm, tomadas a intervalos de 0.33mm. Actualmente, también se disponen de imágenes tomadas de un hombre de 58 años, con las mismas características del cuerpo original[4].

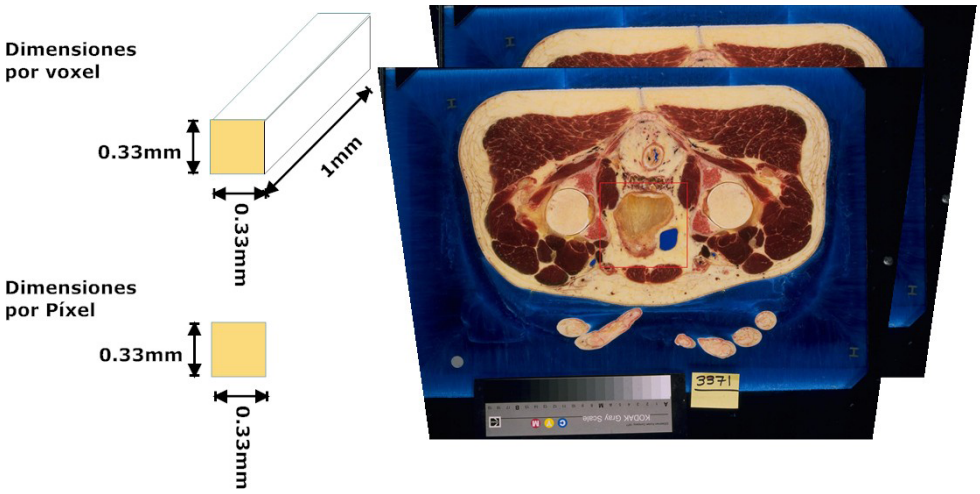


Figura 3.4: *Dimensiones por voxel y por píxel del Visible Human Dataset. El voxel, un elemento de volumen, es ampliamente usado para la reconstrucción de estructuras anatómicas con volumen. En esta tesis, se usa la información del voxel (profundidad) para determinar el tamaño real del órgano a reconstruir.*

La base de datos actual para el cuerpo del hombre (principal interés en esta tesis), está compuesta por:

- Un conjunto de imágenes de Resonancia Magnética Axial y longitudinal de cabeza y cuello tomadas a intervalos de 4 mm. La resolución de estas imágenes es de 256x256 píxeles con una profundidad de 12 bits en escala de grises.
- 1871 imágenes de Tomografía Computarizada de cuerpo entero, tomadas a intervalos de 1mm con una resolución de 512x512 píxeles y 12 bits de profundidad en escala de grises.
- 1871 imágenes anatómicas axiales de cuerpo entero, tomadas a intervalos de 1mm con una resolución de 2048x1216 píxeles y una profundidad de color de 24 bits en escala RGB.
- 1871 imágenes anatómicas axiales de cuerpo entero, tomadas a intervalos de 1mm con una resolución de 4096x2700 píxeles y una profundidad de color de 24 bits en escala RGB.

El campo de aplicación del VHD (*Visible Human Dataset*) del VHP, es muy grande, siendo el principal el campo de estudio biomédico, donde la reconstrucción de modelos anatómicos y su visualización es de gran importancia. La tabla 3.1, muestra los principales usos y aplicaciones del VHD

[4]Para mayor información, consulte XU[65, p. 479] y VHP [62].

dentro del campo biomédico. En el presente trabajo de tesis, la razón de su uso fue el modelado de una vejiga virtual, con el fin de adaptarla en el simulador de entrenamiento de RTUP.

Algoritmos	**Modelado**	**Prototipos**	**Educación**
Rendering	Modelos deformables	Cirugía asistida	Anatomía digital
Segmentación	Propiedades físicas	Intervenciones quirúrgicas	Bibliotecas de referencia
Registros	Propiedades psicológicas	Endoscopia virtual	Bases de conocimiento
Clasificación	Animaciones realistas	Procesos de entrenamiento	Marco de trabajo biológico

Tabla 3.1: *Algunos usos y aplicaciones del conjunto de datos del VHP[51, p.241]. En esta tesis, los datos se usan para modelado y aplicados al desarrollo del prototipo de entrenamiento virtual.*

3.1.2. Reducción del contorno

Los contornos, para fines de análisis de imágenes biomédicas, son curvas contenidas en planos paralelos (isosuperficies), los cuales delimitan una estructura anatómica y facilitan la descripción de su forma. Para llegar a ellos, es importante aplicar una técnica de transformación que nos permita pasar de la imagen original (que contiene un exceso de información no útil), a una imagen que contenga únicamente la información necesaria.

Así, una de las técnicas más simples para la extracción de contornos, es la segmentación manual con ayuda del experto, quien a partir de una imagen, busca la región de interés, la delimita y muestrea con puntos de selección. Este proceso se lleva a cabo por el experto porque representa el conocimiento empírico y en consecuencia hay un menor grado de error (Ver figura 3.5).

No obstante de la valiosa información que le proporciona al experto una imagen anatómica, en ocasiones es necesario realizar, antes del proceso de selección, un análisis de los canales de color que componen la imagen[5]. El procedimiento a seguir es muy sencillo y en parte también depende de la persona especializada que lo lleve a cabo. Básicamente consiste en partir de una imagen anatómica y descomponerla en sus tres canales de color (Rojo, Verde y Azul), cada uno de los cuales contiene información representada en una escala de 0 a 255 niveles. A partir de ellas, se elige una, la que proporciona más información visual, y se realiza sobre ésta la selección manual de puntos.

Los puntos de selección, que por su naturaleza son irregulares en cuanto a la distancia entre ellos, nos proporcionan información muy valiosa de la curva que delimita la estructura anatómica en un plano de corte. Sin embargo, aún son inútiles para aplicar un algoritmo de reconstrucción de superficies de manera simple. Por ello, es necesario procesar el conjunto de puntos para obtener un contorno "continuo" y en definición, represente un contorno mínimo, en el cual, cada píxel de la imagen únicamente posea dos píxeles vecinos (Definición 1).

Definición **1**

Vecinos de un píxel[16, p. 66] (Figura 3.6). Un píxel p de coordenadas (x, y) tiene dos vecinos horizontales y dos vecinos verticales, cuyas coordenadas vienen dadas por:

$$(x+1, y),\ (x-1, y),\ (x, y+1), (x, y-1)$$

[5]Este proceso de selección del canal, se describe a detalle en el trabajo de Gastélum[14, págs. 20-23].

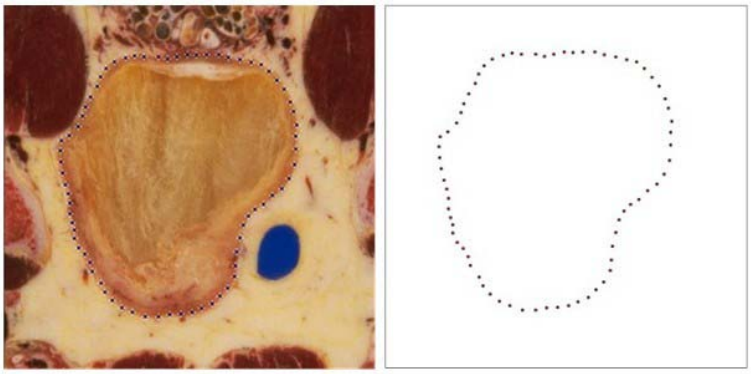


Figura 3.5: *Segmentación manual de una imagen anatómica. A la izquierda, muestreo sobre la imagen anatómica original. A la derecha, imagen binaria con la información de los puntos muestreados. Los puntos nos proporcionan información valiosa de la forma de la curva que delimita la estructura anatómica.*

Este conjunto de píxeles, denominado los *4-vecinos* de p, se representa por $N_4(p)$. Cada píxel está a una unidad de distancia de p. Si p está en el borde de la imagen, algunos de los vecinos caen fuera de la imagen digital.

Los cuatro vecinos en diagonal de p tienen las coordenadas:

$$(x+1, y+1),\ (x+1, y-1),\ (x-1, y+1),\ (x-1, y-1)$$

Y se representan por $N_D(p)$. Estos puntos, junto a los *4-vecinos*, se denominan los *8-vecinos* de p, y se representan por $N_8(p)$. Al igual que antes, algunos puntos de $N_D(p)$ y $N_8(p)$ caen fuera de la imagen si p está en el borde de la misma.

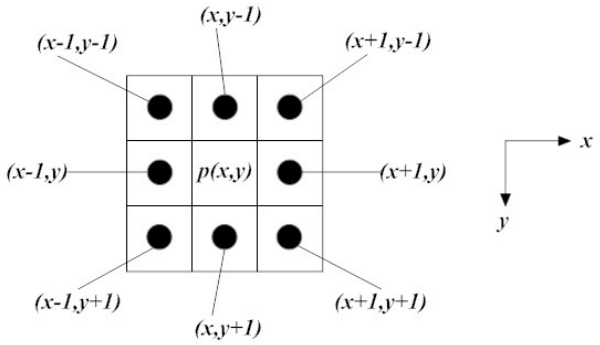


Figura 3.6: *Vecinos de un píxel p de coordenadas (x,y). Los puntos alrededor de p, representan los 8-vecinos de p; conformados por sus 4-vecinos y sus 4 vecinos en diagonal.*

El procesamiento requiere de la conversión de la imagen original (anatómica) a una imagen binaria[6]. Para los propósitos de esta tesis, la convención empleada fue 0-blanco para un punto no seleccionado y 1-negro para un punto seleccionado (Ver figura 3.7).

[6]Las imágenes binarias se caracterizan porque cada uno de sus píxeles únicamente pueden tomar dos valores de color:

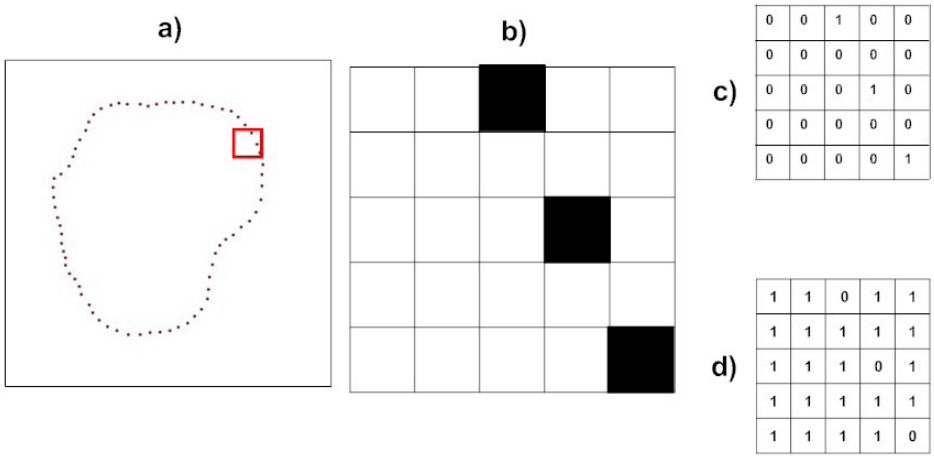


Figura 3.7: *Representación numérica de una imagen binaria. a) Imagen binaria completa, b) Representación en píxeles de una sección, c) Representación en valores binarios por píxel usando convención 0-blanco y 1-negro, d) Representación en valores binarios por píxel usando convención 1-blanco y 0-negro.*

El procedimiento básico para la obtención de la curva del contorno, consiste en la dilatación (Definición 2) de cada uno de los píxeles seleccionados, tantas veces como sea necesario para obtener un contorno engrosado. Posteriormente éste último se somete a un algoritmo de adelgazamiento mínimo[7] y después a un proceso de suavizado.

DEFINICIÓN **2**

Dilatación (Figura 3.8). Sean A y B dos conjuntos de Z^2, la dilatación de A por B, representada por $A \oplus B$, se define como:

$$A \oplus B = \{z|(\widehat{B})_z \cap A \neq \emptyset\} \tag{3.1}$$

La ecuación consiste en obtener la reflexión de B sobre su origen y desplazando la reflexión por z, tal que $\widehat{B}$ y A se sobrepongan en al menos un elemento[16, p. 523-524].

DEFINICIÓN **3**

Distancia entre píxeles[16, p. 68-69] (*Distancia de tablero de ajedrez*). Sean los píxeles p y q, de coordenadas (x_1, y_1) y (x_2, y_2) respectivamente. La distancia entre p y q, denotada por $D_8(p, q)$ se define como:

$$D_8(p, q) = máx\left(|x_1 - x_2|, |y_1 - y_2|\right) \tag{3.2}$$

blanco o negro. Para fines de procesamiento de imágenes, el blanco representa un 1 (valor máximo de color) y el negro un 0 (valor mínimo de color).

[7] A este proceso se le conoce como obtención del esqueleto (*skeletonize*).

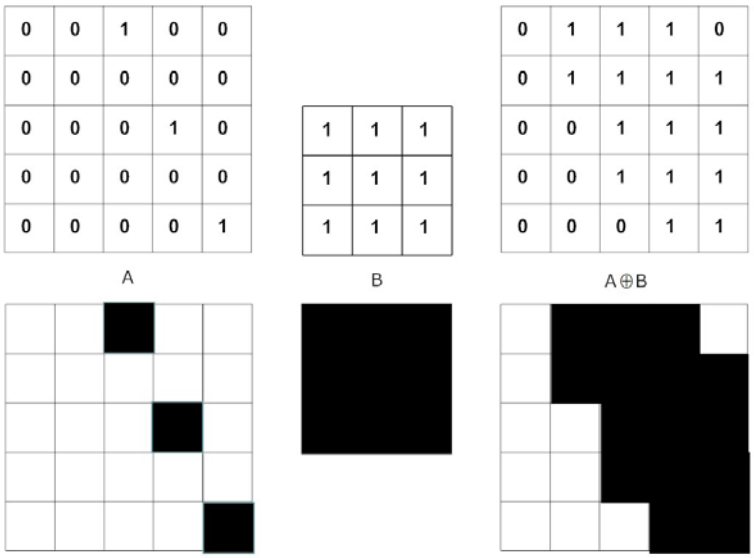


Figura 3.8: *Dilatación de una imagen binaria. A la izquierda, conjunto* $\boldsymbol{A}$ *no dilatado; al centro, conjunto* $\boldsymbol{B}$ *(elemento estructurante); a la derecha, dilatación de* $\boldsymbol{A}$ *por* $\boldsymbol{B}$*.*

El número de iteraciones de dilatación dependerá totalmente de la distancia entre cada uno de los píxeles seleccionados. Si la distancia entre puntos o píxeles es grande, cada iteración aumentará el grado de error al agregar información adicional. Por esta razón, es recomendable que la resolución de puntos de selección sea considerable.

Proposición **1**

Iteraciones de dilatación para unir dos píxeles. Sean p_1 y p_2, puntos del conjunto de selección, tal que, p_2 es el punto más cercano a p_1 en una dirección de recorrido de la curva definida por los puntos de muestreo. El número de iteraciones de dilatación, denotado por $I_{p_1 \to p_2} \in \mathbb{N}$, necesarias para unir los puntos p_1 y p_2 está dado por:

$$I_{p_1 \to p_2} = \frac{D_8(p_1, p_2)}{2} \tag{3.3}$$

A partir de la proposición 1, podemos deducir que el número de iteraciones reales a aplicar a todo el conjunto de selección será igual al valor máximo de iteraciones por cada par de puntos. Esto es, si P representa el conjunto de puntos de selección, formado por $\{p_1, p_2, p_3, ..., p_n\}$; entonces, el número de iteraciones a aplicar para generar una curva engrosada, denotado por I_C, está dado por:

$$I_C = máx\{I_{p_1 \to p_2}, I_{p_2 \to p_3}, ..., I_{p_{n-1} \to p_n}, I_{p_n \to p_1}\} \tag{3.4}$$

El siguiente paso para la obtención de un contorno útil para fines de reconstrucción usando el método descrito en esta tesis, es el adelgazamiento del contorno engrosado. Este método, también llamado "obtención del esqueleto del contorno" consiste en aplicar procesos de erosión (Definición 4) y apertura (Definición 5) a la imagen binaria, tal como lo muestra la definición 6. La apertura tiende a suavizar el contorno, rompiendo istmos estrechos y eliminando protuberancias estrechas; mientras que la erosión tiende a adelgazar el contorno a su máximo.

DEFINICIÓN **4**

Erosión[16, p. 525-527] Sean los conjuntos A y B en Z^2, la erosión de A debida a B, denotada por $A \ominus B$, se define como:

$$A \ominus B = \{z|(B)_z \subseteq A\}. \tag{3.5}$$

Esto es, la erosión de A debida a B es el conjunto de todos los puntos z, tales que B, trasladado por z, están contenidos en A.

DEFINICIÓN **5**

Apertura[16, p. 528-532] Sean A y B en Z^2, la apertura de A debida a B, denotada por $A \circ B$, se define como:

$$A \circ B = (A \ominus B) \oplus B. \tag{3.6}$$

Esto es, la apertura de A debida a B es simplemente la erosión de A por B, seguida por una dilatación del resultado por B.

DEFINICIÓN **6**

Obtención del esqueleto[16, p. 543-545] Sea A un conjunto de puntos que definen un contorno engrosado. El esqueleto de A, denotado por $S(A)$ se define en términos de erosiones y aperturas, que puede ser expresado como:

$$S(A) = \bigcup_{k=0}^{K} S_k(A) \tag{3.7}$$

con,

$$S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B \tag{3.8}$$

donde, B es el elemento estructurante, y $(A \ominus kB)$ indica las k-ésimas erosiones sucesivas sobre A; esto es:

$$(A \ominus kB) = (...((A \ominus B) \ominus B) \ominus ...) \ominus B \tag{3.9}$$

y K es la última iteración antes de que A se convierta en el conjunto vacío. Esto es:

$$K = máx\{k|(A \ominus B) \neq \oslash\} \tag{3.10}$$

Hasta este paso, el contorno obtenido es útil para nuestros fines. Sin embargo, en muchas ocasiones no se logra el resultado deseado debido a la forma de la curva obtenida. Para ello, se aplican algunas técnicas descritas en al apéndice B, sección "Extracción de contornos con *ImageJ*", consistentes en el filtrado de la curva mínima para obtener curvas suavizadas y con el menor efecto causado por la discretización (Ver figura 3.9).

En otros casos, la información que nos proporcionan las imágenes anatómicas es insuficiente. Esto es, los cortes son tomados a intervalos fijos y la información intermedia se pierde. Para ello, se han desarrollado metodologías para la "recuperación" de esa información mediante técnicas de interpolación. En imágenes, una técnica comúnmente empleada es la interpolación morfológica, consistente en la aplicación de *operaciones binarias*[8] sobre una imagen.

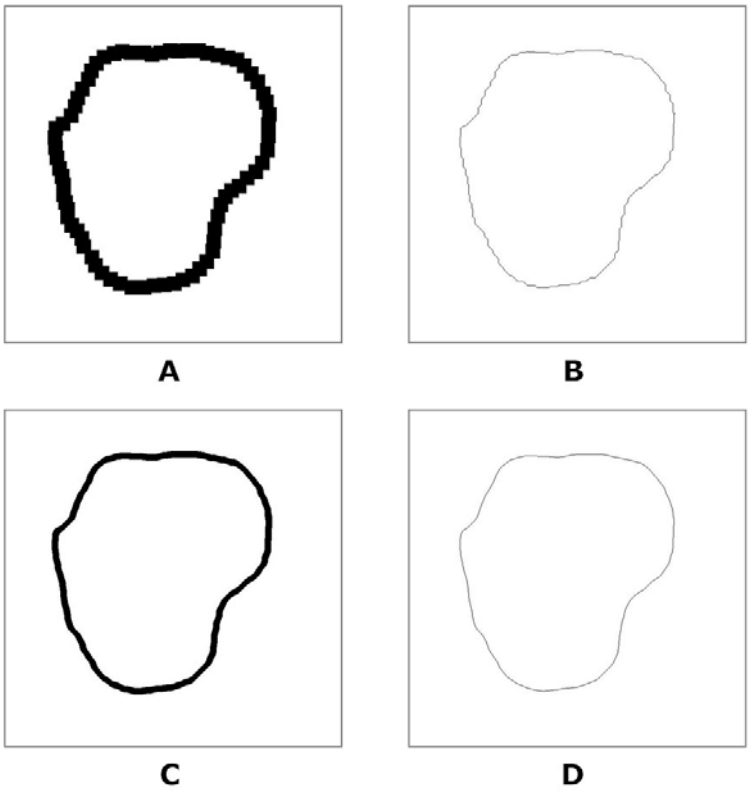


Figura 3.9: *Extracción del contorno al aplicar la técnica de obtención del esqueleto sobre una curva engrosada. A) Contorno engrosado tras 14 dilataciones, B) Esqueleto de A, C) Suavizado de B usando un filtro Gaussiano y D) Esqueleto de C.*

La interpolación morfológica parte de dos conjuntos A y B (imágenes binarias) para generar un tercero, C, cuyo contenido es la interpolación de A y B. Se basa en el concepto de media morfológica, la cual se define por la ecuación 3.11.

$$C = M(A, B) = \bigcup_{\forall \lambda} \{(A \oplus \lambda B) \cap (A \ominus \lambda B)\} \tag{3.11}$$

[8] En imágenes, las operaciones binarias son aquellas que operan sobre representaciones de ceros y unos (blanco y negro). Entre las principales operaciones binarias se encuentran: Dilatación $\oplus$, Erosión $\ominus$, Apertura $\circ$, Cierre $\bullet$, Adelgazamiento $\bigotimes$, entre otros.[16, cap. 9]

Esta ecuación se interpreta como: la interpolación morfológica de los conjuntos A y B es igual a la unión de todas las intersecciones resultantes de dilatar y erosionar el conjunto A por el conjunto B (elemento estructurante) en un factor λ[9]. La figura 3.10 muestra un ejemplo de aplicación de la interpolación morfológica.

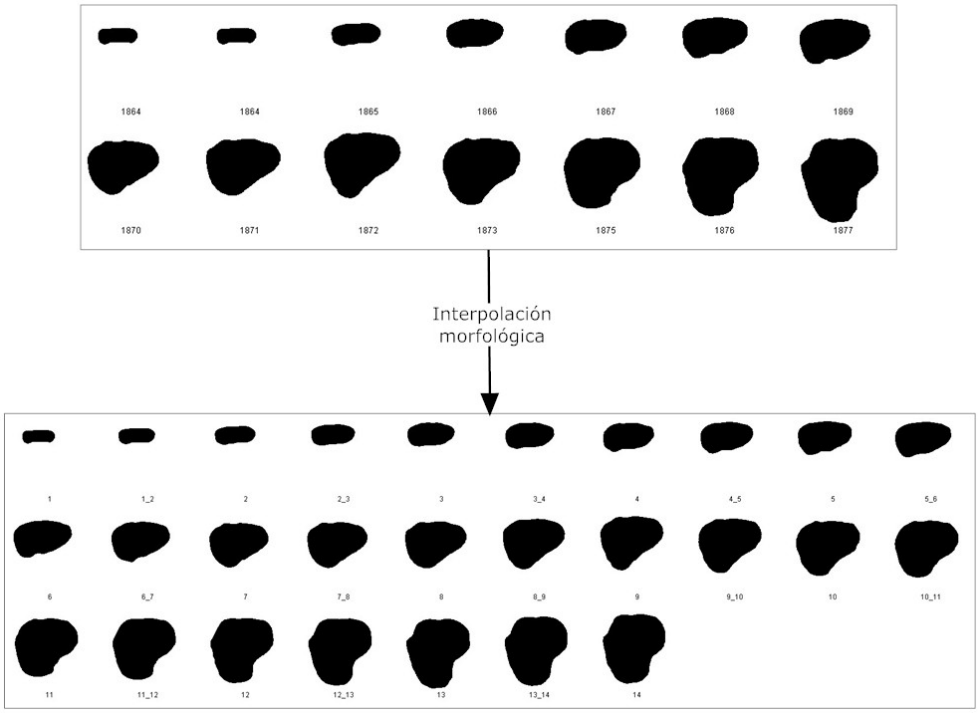


Figura 3.10: *Interpolación morfológica aplicado a un conjunto de cortes binarios de una vejiga. Arriba, conjunto original proveniente de la extracción de contorno de los cortes anatómicos. Abajo, interpolación de los cortes; el número total de imágenes generadas, es igual al número de imágenes originales menos uno.*

3.2. Muestreo de Contornos

En la sección anterior, se describió de manera detallada el proceso de extracción de contornos mínimos o esqueletos a partir de imágenes anatómicas del VHP. El objetivo fue obtener un conjunto de puntos muestreados por el experto y a partir de ellos generar una curva "continua". En esta sección se describe un algoritmo de muestreo automatizado y ordenado, de contornos mínimos para la obtención de puntos que definen los vértices de una malla de superficie 3D.

El algoritmo de muestreo, propuesto en este trabajo de tesis, funciona bajo tres condiciones principales: 1) condición de centrado, 2) condición de muestreo y 3) la condición de curva única. La primera condición se debe a que uno de los primeros pasos llevados a cabo por el algoritmo, es identificar el punto medio horizontal de la imagen y comenzar a barrer verticalmente hasta encontrar el primer punto que pertenece al contorno; ese primer punto se le denomina "punto inicial del recorrido". La segunda condición, nos asegura que el contorno contiene la información suficiente para aplicar el algoritmo de muestreo. La tercera condición se debe a que el algoritmo realiza un recorrido sobre una

[9]Para mayor información, consulte SORIANO[56, cap. 2] y IWANOWSKI[26, p. 50-51]

y sólo una curva a partir de un punto inicial. Así, estas condiciones quedan definidas como se muestra a continuación.

Condición **1**

Condición de centrado. Los puntos del contorno caen dentro de las dos regiones definidas por la línea vertical que divide la imagen en dos partes iguales (ver figura 3.11).

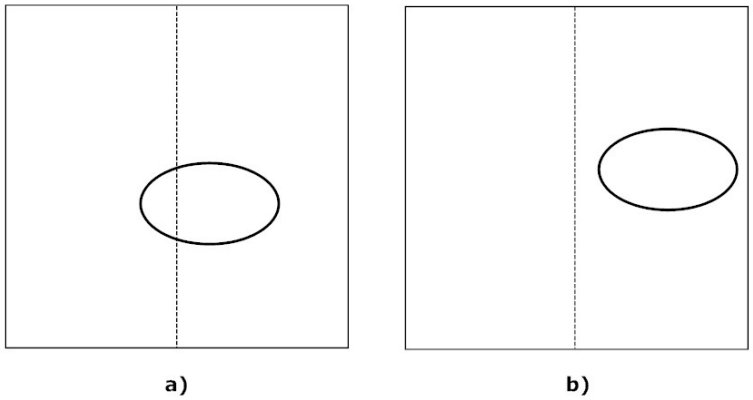


Figura 3.11: *Condición 1 de validez del algorítmo de muestreo. a) Imagen válida y b) Imagen no válida.*

Condición **2**

Condición de muestreo. (Criterio de Nyquist[10]) El número total de puntos en la curva es mayor o igual a dos veces el número de muestras requeridas. Esto es:

$$|P_C| \geq 2m \tag{3.12}$$

Donde P_C representa el conjunto de puntos que forman la curva y m el número de muestras a obtener del contorno.

Condición **3**

Condición de curva única. Los puntos definen una y solo una curva de contorno (ver figura 3.12).

[10]Criterio de Nyquist: "la frecuencia de muestreo debe de ser mayor al doble de la frecuencia más alta presente en la señal". En imágenes, "el espaciamiento mínimo entre muestras debe de ser menor a un medio del tamaño más pequeño de detalles a representar".Gastélum[14, pág. 23].

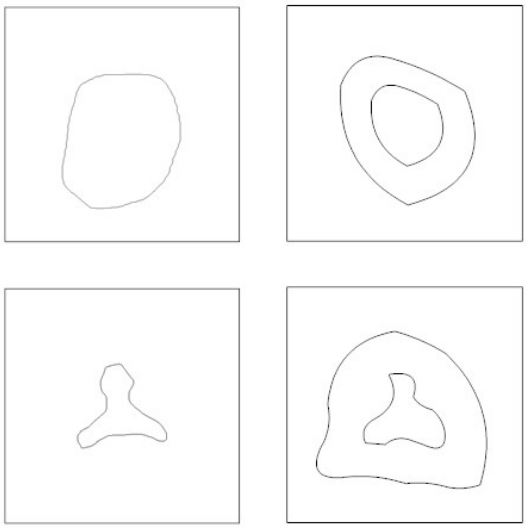


Figura 3.12: *Condición 3 de validez del algoritmo de muestreo. A la izquierda, imágenes válidas. A la derecha, imágenes no válidas; en este caso, el algorítmo de recorrido solo tomaría la curva más externa, ello se debe a que opera únicamente sobre una curva.*

3.2.1. El algoritmo de muestreo

El algoritmo de muestreo tiene por objetivo la obtención de puntos muestra de un contorno mínimo. Para su ejecución, se parte de la identificación de un punto inicial en la curva. A continuación, se inicia un proceso de recorrido, en el cual, se van guardando las posiciones de los puntos que definen la curva, en una lista. Estos puntos por ser resultado de un recorrido continuo, se encuentran ordenados y representan las coordenadas en x y y de cada punto en el plano (en este caso, una imagen) donde se encuentra contenido el contorno. Una vez obtenido el conjunto de puntos ordenados, se aplica un proceso de selección de puntos, en el cual, a partir de un intervalo de muestreo definido, se toman sólo los puntos que satisfacen ese intervalo.

Las variables empleadas para la definición del algoritmo son[11]:

- A: Conjunto binario con la información del contorno a muestrear
- B y C: Conjuntos temporales
- m, n: m-filas y n-columnas de los conjuntos binarios A, B y C
- I: Lista de puntos $\bullet$
- M: Lista de puntos muestreados
- $|P_C|$: Número de puntos en el contorno
- $|M_C|$: Número de muestras del contorno a obtener
- Δp: Intervalo de muestreo

[11] Con el fin de ofrecer una forma simple de representar los puntos de análisis, se emplea el símbolo $\bullet$ para definir un punto de interés y $\circ$ para representar puntos que representan el fondo de la imagen binaria.

- p_0: Punto inicial del recorrido de coordenadas (x,y)
- p_{-1}: Punto anterior de coordenadas (x,y)
- p_i: Punto actual de coordenadas (x,y)
- p_{+1}: Punto siguiente de coordenadas (x,y)

A continuación se presenta el algoritmo de muestreo propuesto.

Algoritmo de muestreo de contornos

1. *Definir conjunto A y sus propiedades (m,n)*
2. *Inicializar dos conjuntos B y C vacíos*
3. *Inicializar puntos de referencia* $p_0 = (0,0)$, $p_i = (0,0)$, $p_{+1} = (0,0)$ *y* $p_{-1} = (0,0)$
4. *Inicializar la lista de puntos vacía* $I = \oslash$
5. *Inicializar la lista de puntos muestreados vacía,* $M = \oslash$.
6. *Determinar el número de puntos en el contorno* $|P_C|$
7. *Encontrar el punto de inicio del recorrido*
   a) *Para* $i = 1 : m$ *en A; al cumplirse por primera vez la condición:* $A(n/2, i) = \bullet$ *entonces* $p_0 = (n/2, i)$
   b) *Agregar punto de inicio* p_0 *en la lista de puntos I*
8. *Listar los puntos que conforman el contorno, tomando como punto de inicio a* p_0
   a) *Hacer* $p_i = p_0$
   b) *Marcar punto* p_i *en B,* $B(p_i) = \bullet$
   c) *Analizar vecinos de* p_i *en A y determinar* p_{-1}
   d) *Marcar punto* p_{-1} *en B,* $B(p_{-1}) = \bullet$
   e) *Analizar vecinos de* p_i *en A y determinar* p_{+1}
   f) *Agregar punto* p_{+1} *en la lista de puntos I*
   g) *Marcar punto* p_{+1} *en B,* $B(p_{+1}) = \bullet$
   h) *Para* $i = 1$ *hasta* $i = |P_C| - 2$ *con incrementos de* 1
      1) $p_{-1} = p_i$
      2) $p_i = p_{+1}$
      3) *Borrar* p_i *de A:* $A(p_i) = \circ$
      4) *Analizar vecinos de* p_i *en A y determinar* p_{+1}
      5) *Agregar punto* p_{+1} *en la lista de puntos I*
      6) *Marcar punto* p_{+1} *en B,* $B(p_{+1}) = \bullet$
9. *Calcular Intervalo de muestreo:* $\Delta p = \frac{|P_C|-1}{|M_C|}$
10. *Para* $i = 1$ *hasta* $i = |P_C| - 1$ *con incrementos de* Δp, $i \in \mathbb{N}$
    a) $C(I[i]) = \bullet$

b) *Agregar $I[i]$ a la lista de puntos muestreados M*

Claramente, el algorítmo tiene una complejidad de $\theta(n)$. Al final de la ejecución del algoritmo se tiene que: B es igual al conjunto A original, C es el conjunto muestreado y M es la lista de puntos muestreados ordenados.

El análisis de los vecinos para determinar el punto anterior y siguiente siguen la lógica de la figura 3.13.

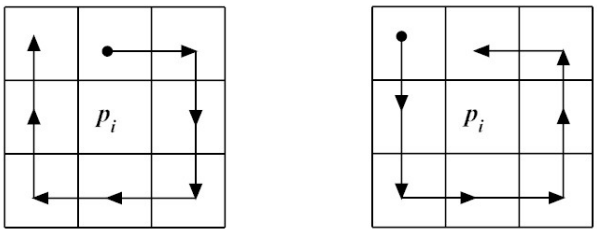


Figura 3.13: *Análisis de vecinos para encontrar puntos anteriores y siguientes en el algoritmo de muestreo. A la izquierda, recorrido para encontrar punto anterior, p_{-1}. A la derecha, recorrido para encontrar punto siguiente, p_{+1}.*

El resultado de aplicar este algoritmo a un conjunto de contornos, es una lista ordenada de puntos en x y y. Sin embargo, para fines de reconstrucción 3D hace falta agregar una tercer componente, la profundidad (z). La tercer componente se calcula a partir del intervalo entre cada corte de las imágenes anatómicas originales. Así, todos los puntos muestreados que pertenecen a un mismo plano de corte deberán tener la misma componente de profundidad.

Hasta este paso se tiene un conjunto ordenado de puntos que definen muestras del modelo anatómico original. Sin embargo, ellos solos son inútiles para fines de visualización y es necesaria la aplicación de una metodología que, a partir del origen de los puntos muestreados, determine los enlaces existentes entre ellos; definiendo así la geometría tridimensional del modelo. En la siguiente sección se define una metodología de reconstrucción de superficies a partir de la nube de puntos muestreados obtenidos al aplicar el algoritmo de muestreo.

3.3. Triangulación a partir de una nube de puntos ordenados

La reconstrucción de una malla de superficie tridimensional implica la definición de sus propiedades geométricas. Básicamente se pueden distinguir dos de ellas: el vértice y la cara triangular. El vértice es un punto geométrico, cuya propiedad principal es su posición, que se define por sus componentes coordenadas (x, y, z) en el espacio de tres dimensiones (Definición 7). La cara triangular, definida como el triángulo formado por la unión de tres vértices, determina la superficie que delimita el espacio geométrico; en nuestro caso: el modelo anatómico.

DEFINICIÓN **7**

Vértice. Sea P el conjunto ordenado de puntos p_i de la superficie, con $i = \{1, 2, 3, ..., m|S|\}$; donde $|S|$ es el número de planos de corte del modelo anatómico y m el número de muestras por plano de corte. Un vértice, denotado por v_i, situado en el espacio de tres dimensiones; se define como:

$$v_i = p_i = [x_i, y_i, z_i], \tag{3.13}$$

donde x_i, y_i y z_i, son las coordenadas en (x, y, z) del vértice.

El resultado de aplicar el algoritmo de muestreo a un conjunto de contornos contenidos en planos paralelos, es un conjunto ordenado de puntos, cuyas coordenadas (x, y, z) determinan la posición y la forma del modelo anatómico en el espacio tridimensional. Al unir este conjunto de puntos ordenados por medio de la definición de caras triangulares, obtenemos una superficie. La figura 3.14 muestra la lista de vértices y su representación tridimensional mediante caras triangulares.

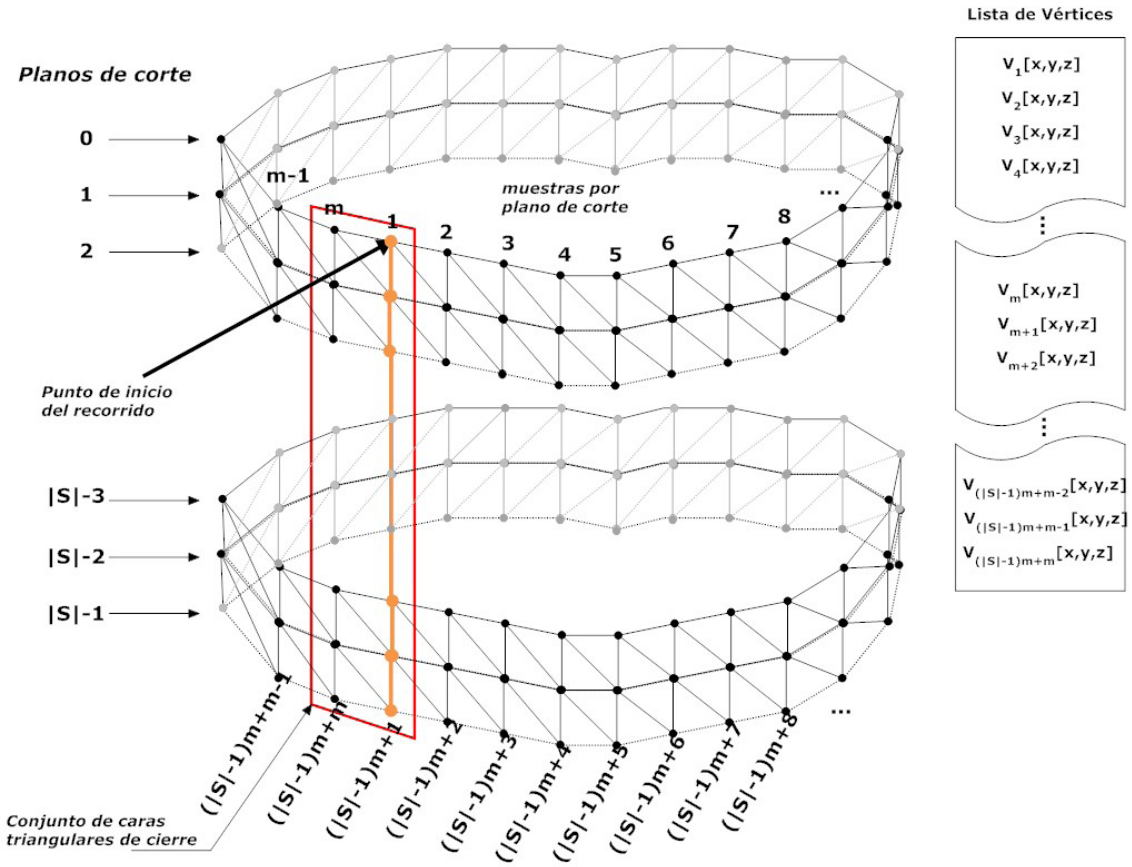


Figura 3.14: *Estructura de vértices y caras triangulares para la reconstrucción de una malla de superficie. A la izquierda, orden de vértices y caras triangulares. A la derecha, lista de vértices ordenados, resultados del algoritmo de muestreo para un conjunto de contornos contenidos en planos paralelos.*

Antes de pasar a las ecuaciones de reconstrucción, definamos dos métricas muy importantes: el número de vértices de una malla y el número de caras a generar. Estos dos valores dependen totalmente de la información proveniente de los puntos muestreados.

Definición **8**

Número de Vértices. Sea $|S|$ el número de planos de corte del modelo anatómico, y m

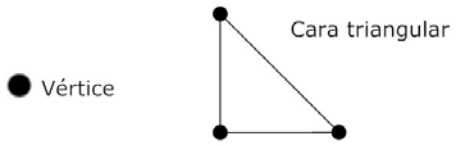


Figura 3.15: *Formas básicas que conforman una malla tridimensional. El vértice, formado por tres componentes (x,y,z). La cara triangular, definida por tres vértices.*

el número de muestras por plano de corte. El número de vértices de la malla de superficie, denotado por $|V|$ está dado por:

$$|V| = m|S| \tag{3.14}$$

Definición **9**

Número de caras en la malla. Sea $|S|$ el número de planos de corte del modelo anatómico, y m el número de muestras por plano de corte. El número total de caras triangulares que conforman la malla, denotado por $|F|$, está dado por:

$$|F| = 2m(|S| - 1) \tag{3.15}$$

Una forma de representar las caras triangulares es por medio de índices. Éstos tienen la ventaja de ser una referencia a una definición de vértice que se declara una y sólo una vez. Por ejemplo, supongamos los vértices v_1, v_2, v_3 y v_4, definidos como: $v_1 = (150, 129, 0)$, $v_2 = (170, 120, 0)$, $v_1 = (184, 118, 1)$, $v_4 = (201, 117, 1)$; y la cara triangular $f_1 = (v_1, v_2, v_3)$. Un formato de archivo para definición de vértices y caras triangulares[12] sería:

```
--------------------------
//definición de vértices
//íd x y z
1 150 129 0
2 170 120 0
3 184 118 1
4 201 117 1
...

//definición de caras
//id v1 v2 v3
1 1 2 3
2 1 2 4
...
--------------------------
```

[12]En el capítulo 5 se da una explicación más a fondo de los formatos gráficos empleados, sus ventajas y desventajas con respecto a otros.

En él, cada cara triangular tiene un valor de referencia a tres vértices. La cara 1 referencia a los vértices 1, 2 y 3; mientras que la cara 2, referencia a los vértices 1, 2 y 4. Este tipo de estructuras nos hace proponer un sistema de indexado; el cual nos permita asignar a cada cara triangular una tercia de vértices referenciados. Para lograr esto, en esta tesis se propone la definición 10. El arreglo indexador nos permite desdoblar la malla a generar y nos da una mejor visión de las conexiones entre vértices.

Definición **10**

Arreglo indexador de vértices. Sean los conjuntos $i = \{0, 1, 2, 3, ..., (|S| - 2)\}$ y $j = \{1, 2, 3, ..., (m - 1)\}$. El arreglo indexador de vértices, denotado por V_S, se define como:

$$V_S = [V_{i,j}] = \begin{bmatrix} V_{0,1} & V_{0,2} & V_{0,3} & V_{0,4} & V_{0,5} & V_{0,6} & ... & V_{0,m} \\ V_{1,1} & V_{1,2} & V_{1,3} & V_{1,4} & V_{1,5} & V_{1,6} & ... & V_{1,m} \\ V_{2,1} & V_{2,2} & V_{2,3} & V_{2,4} & V_{2,5} & V_{2,6} & ... & V_{2,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ V_{|S|-1,1} & V_{|S|-1,3} & V_{|S|-1,3} & V_{|S|-1,4} & V_{|S|-1,5} & V_{|S|-1,6} & ... & V_{|S|-1,m} \end{bmatrix} \tag{3.16}$$

Donde $V_{i,j}$ es el índice del vértice en la posición (i, j) del arreglo. El valor de $V_{i,j}$ se calcula como:

$$V_{i,j} = mi + j. \tag{3.17}$$

La ecuación anterior se interpreta como: "$V_{i,j}$ es el índice del vértice v_{mi+j}".

3.3.1. Ecuaciones de reconstrucción

A partir de las estructuras definidas anteriormente, podemos dividir la reconstrucción de la malla en dos grupos: el primero, formado por el conjunto de triángulos inferiores y sus correspondientes triángulos de cierre (ver figura 3.16); y el segundo, formado por el conjunto de triángulos superiores y sus correspondientes triángulos de cierre (ver figura 3.17). Las definiciones 11 y 12 describen las ecuaciones necesarias para llevar a cabo el proceso de reconstrucción.

Definición **11**

Reconstrucción de triángulos inferiores. Sean $i = \{0, 1, 2, 3, ..., (|S| - 2)\}$ y $j = \{1, 2, 3, ..., (m - 1)\}$

El primer grupo de triángulos que representa la malla está dado por:

$$F_{i,j} = \left[V_{im+1,j}, V_{(i+1)m+1,j}, V_{(i+1)m+1,j+1}\right] \tag{3.18}$$

Por ejemplo, si tomamos los extremos de ambos valores, $i = |S| - 2$ y $j = m - 1$, tenemos que la cara de la esquina inferior derecha es:

$$F_{|S|-2,m-1} = [V_{(|S|-2)m+1,m-1}, V_{(|S|-1)m+1,m-1}, V_{(|S|-1)m+1,m}]$$

Los triángulos de este mismo grupo, que cierran la malla en los extremos derecho e izquierdo están dados por:

$$F_{i,j} = \left[V_{im+1,j}, V_{(i+1)m+1,j}, V_{(i+1)m+1,1}\right] \tag{3.19}$$

Donde $i = \{0, 1, 2, 3, ..., (|S| - 2)\}$ y $j = m$.

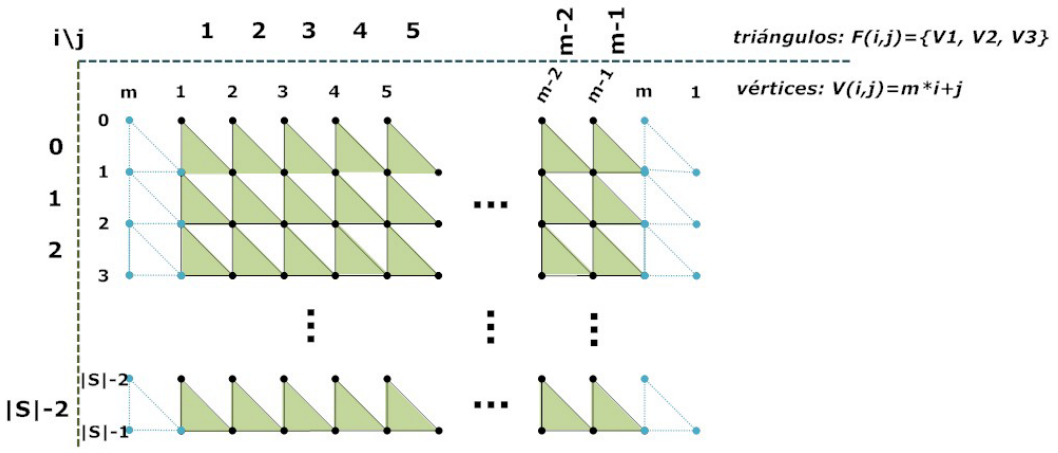


Figura 3.16: *Reconstrucción de triángulos inferiores.*

Definición **12**

Reconstrucción de triángulos superiores. Sean $i = \{0, 1, 2, 3, ..., (|S| - 2)\}$ y $j = \{1, 2, 3, ..., (m - 1)\}$

El segundo grupo de triángulos que representa la malla está dado por:

$$F_{i,j} = \left[V_{im+1,j}, V_{(i+1)m+1,j+1}, V_{im+1,j+1}\right] \tag{3.20}$$

Por ejemplo, si tomamos los extremos de ambos valores, $i = |S| - 2$ y $j = m - 1$, tenemos que la cara de la esquina inferior derecha es:

$$F_{|S|-2,m-1} = [V_{(|S|-2)m+1,m-1}, V_{(|S|-1)m+1,m}, V_{(|S|-2)m+1,m}]$$

Los triángulos de este mismo grupo, que cierran la malla en los extremos derecho e izquierdo están dados por:

$$F_{i,j} = \left[V_{im+1,j}, V_{(i+1)m+1,1}, V_{im+1,1}\right] \tag{3.21}$$

Donde $i = \{0, 1, 2, 3, ..., (|S| - 2)\}$ y $j = m$.

Finalmente, una vez obtenidas las referencias, el paso final es aplicar las ecuaciones de reconstrucción por vértice, como se muestran en la figura 3.18.

Hasta este punto, se tiene una malla triangular con la información suficiente para editarla y aplicarle métodos de suavizado y corrección de artefactos. Como se mencionó al inicio del capítulo, esta

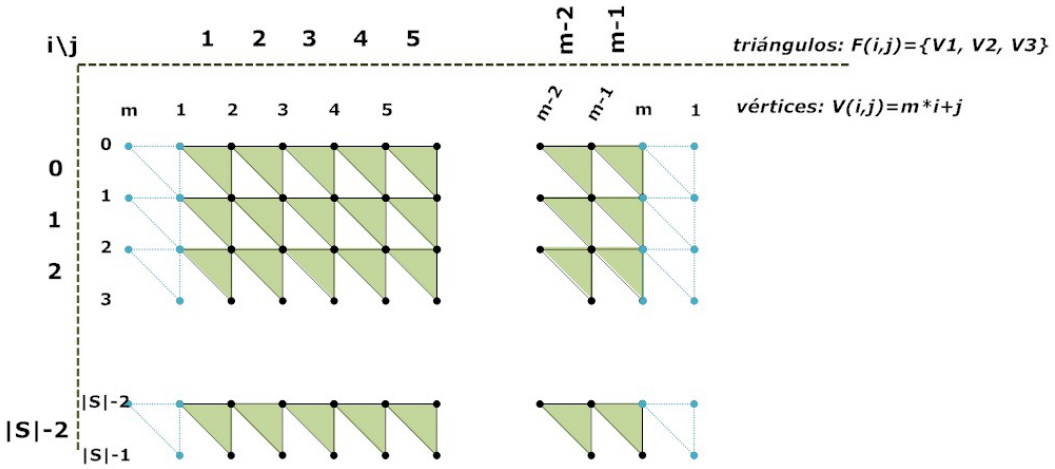


Figura 3.17: *Reconstrucción de triángulos superiores.*

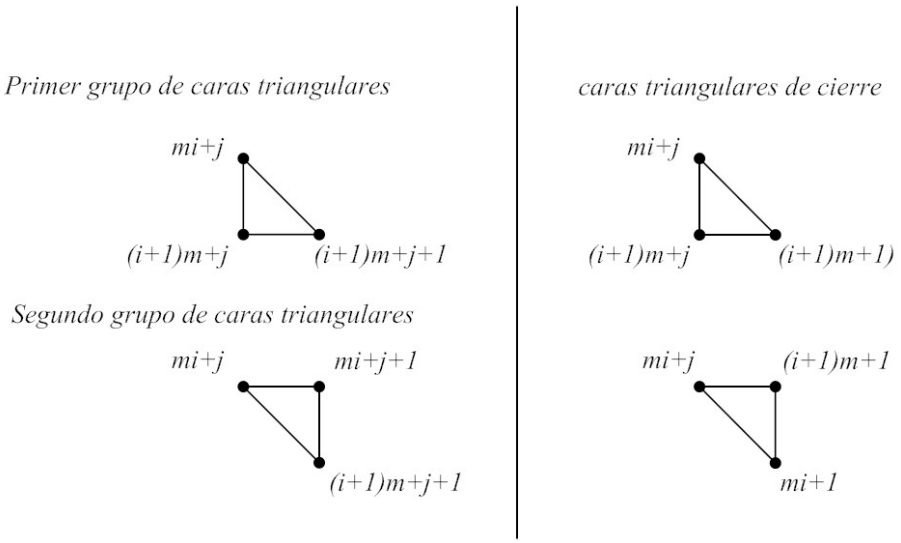


Figura 3.18: *Ecuaciones de reconstrucción con base en las referencias a los vértices.*

metodología se empleó para reconstruir un modelo de la vejiga urinaria, la cual una vez reconstruida, se editó en Blender para corregir algunos artefactos. En el capítulo 6 de resultados, se describen detalladamente los datos empleados para dicha reconstrucción y los resultados obtenidos. La figura 3.19, muestra el modelo de la vejiga resultante de aplicar el método descrito en este capítulo.

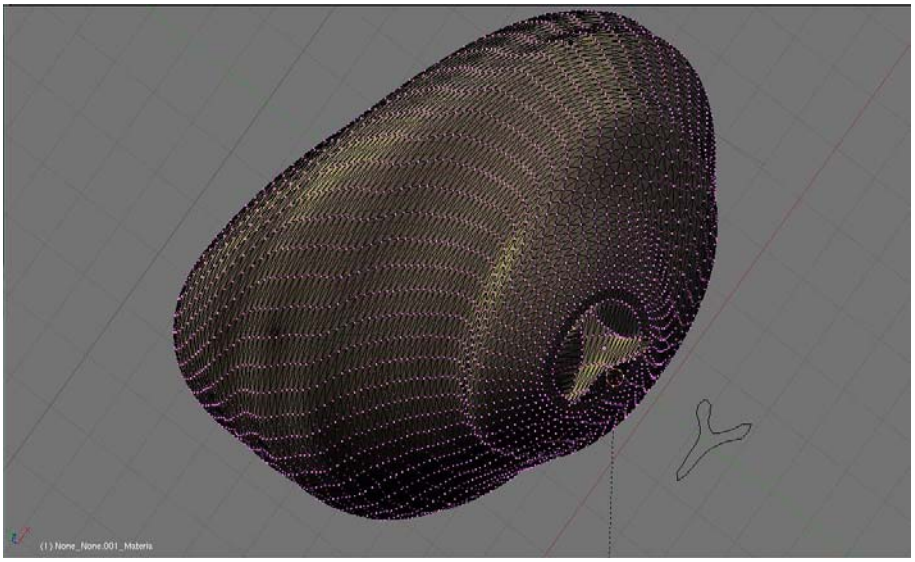


Figura 3.19: *Vejiga urinaria virtual. Malla triangular visualizada en Blender, resultado de aplicar el método de reconstrucción a partir de isosuperficies; método propuesto en este trabajo de tesis.*

3.4. ANEXOS: Programas de Muestreo (MATLAB) y Programa de reconstrucción (C)

Algoritmo de muestreo de contornos en MATLAB

Función en MATLAB que muestrea un contorno. Recibe como parámetros de entrada: nombre de la imagen, tamaño vertical en píxeles, tamaño horizontal en píxeles y el número de muestras a obtener. Regresa un vector con la lista ordenada de puntos muestreados.

```
function vector=muestreaContorno(imageName,filas,columnas,numMuestras);

A=imread(imageName);
N=columnas;
M=filas;

pxInicio=[0 0];
pxAnterior=[0 0];
pxActual=[0 0];
pxSiguiente=[0 0];
indices=[];

B=ones(M,N);
C=ones(M,N);

count_pxs=0;

%%Dtermina el número de puntos en el contorno

for I=1:M
    for J=1:N
        if A(I,J)==0
            count_pxs=count_pxs+1;
        end
    end
end

%%Encuentra punto de inicio del recorrido

for I=1:M
        if A(N/2,I)==0
            pxInicio=[N/2 I];
            indices=[indices;pxInicio];
            break
        end
end


%%Reconstrucció a partir del algorítmo de recorrido

pxActual = pxInicio;
B(pxActual(1,1),pxActual(1,2))=0;
```

```
if A(pxActual(1,1),pxActual(1,2)-1)==0
    pxAnterior = [pxActual(1,1) pxActual(1,2)-1];
elseif A(pxActual(1,1)+1,pxActual(1,2)-1)==0
    pxAnterior = [pxActual(1,1)+1 pxActual(1,2)-1];
elseif A(pxActual(1,1)+1,pxActual(1,2))==0
    pxAnterior = [pxActual(1,1)+1 pxActual(1,2)];
elseif A(pxActual(1,1)+1,pxActual(1,2)+1)==0
    pxAnterior = [pxActual(1,1)+1,pxActual(1,2)+1];
elseif A(pxActual(1,1),pxActual(1,2)+1)==0
    pxAnterior = [pxActual(1,1) pxActual(1,2)+1];
elseif A(pxActual(1,1)-1,pxActual(1,2)+1)==0
    pxAnterior = [pxActual(1,1)-1 pxActual(1,2)+1];
elseif A(pxActual(1,1)-1,pxActual(1,2))==0
    pxAnterior = [pxActual(1,1)-1 pxActual(1,2)];
elseif A(pxActual(1,1)-1,pxActual(1,2)-1)==0
    pxAnterior = [pxActual(1,1)-1 pxActual(1,2)-1];
end

B(pxAnterior(1,1),pxAnterior(1,2))=0;

if A(pxActual(1,1)-1,pxActual(1,2)-1)==0
    pxSiguiente = [pxActual(1,1)-1 pxActual(1,2)-1];
    indices=[indices;pxSiguiente];
elseif A(pxActual(1,1)-1,pxActual(1,2))==0
    pxSiguiente = [pxActual(1,1)-1 pxActual(1,2)];
    indices=[indices;pxSiguiente];
elseif A(pxActual(1,1)-1,pxActual(1,2)+1)==0
    pxSiguiente = [pxActual(1,1)-1 pxActual(1,2)+1];
    indices=[indices;pxSiguiente];
elseif A(pxActual(1,1),pxActual(1,2)+1)==0
    pxSiguiente = [pxActual(1,1) pxActual(1,2)+1];
    indices=[indices;pxSiguiente];
elseif A(pxActual(1,1)+1,pxActual(1,2)+1)==0
    pxSiguiente = [pxActual(1,1)+1,pxActual(1,2)+1];
    indices=[indices;pxSiguiente];
elseif A(pxActual(1,1)+1,pxActual(1,2))==0
    pxSiguiente = [pxActual(1,1)+1 pxActual(1,2)];
    indices=[indices;pxSiguiente];
elseif A(pxActual(1,1)+1,pxActual(1,2)-1)==0
    pxSiguiente = [pxActual(1,1)+1 pxActual(1,2)-1];
    indices=[indices;pxSiguiente];
elseif A(pxActual(1,1),pxActual(1,2)-1)==0
    pxSiguiente = [pxActual(1,1) pxActual(1,2)-1];
    indices=[indices;pxSiguiente];
end

B(pxSiguiente(1,1),pxSiguiente(1,2))=0;

for I=1:count_pxs-2

    pxAnterior = pxActual;
    pxActual = pxSiguiente;
    A(pxActual(1,1),pxActual(1,2))=1;
```

```
    if A(pxActual(1,1)-1,pxActual(1,2)-1)==0
        pxSiguiente = [pxActual(1,1)-1 pxActual(1,2)-1];
        indices=[indices;pxSiguiente];
    elseif A(pxActual(1,1)-1,pxActual(1,2))==0
        pxSiguiente = [pxActual(1,1)-1 pxActual(1,2)];
        indices=[indices;pxSiguiente];
    elseif A(pxActual(1,1)-1,pxActual(1,2)+1)==0
        pxSiguiente = [pxActual(1,1)-1 pxActual(1,2)+1];
        indices=[indices;pxSiguiente];
    elseif A(pxActual(1,1),pxActual(1,2)+1)==0
        pxSiguiente = [pxActual(1,1) pxActual(1,2)+1];
        indices=[indices;pxSiguiente];
    elseif A(pxActual(1,1)+1,pxActual(1,2)+1)==0
        pxSiguiente = [pxActual(1,1)+1,pxActual(1,2)+1];
        indices=[indices;pxSiguiente];
    elseif A(pxActual(1,1)+1,pxActual(1,2))==0
        pxSiguiente = [pxActual(1,1)+1 pxActual(1,2)];
        indices=[indices;pxSiguiente];
    elseif A(pxActual(1,1)+1,pxActual(1,2)-1)==0
        pxSiguiente = [pxActual(1,1)+1 pxActual(1,2)-1];
        indices=[indices;pxSiguiente];
    elseif A(pxActual(1,1),pxActual(1,2)-1)==0
        pxSiguiente = [pxActual(1,1) pxActual(1,2)-1];
        indices=[indices;pxSiguiente];
    end

    B(pxSiguiente(1,1),pxSiguiente(1,2))=0;

end

indicesMuestreados = [];

%% Factor de muestreo y normalización de la curva

factorM=1.0/numMuestras;

step = (count_pxs-1)/numMuestras;

%% Obtención de lista de puntos muestreados

for I=1:step:count_pxs-1
    C(indices(double(int32(I)),1),indices(double(int32(I)),2))=0;
    indicesMuestreados = [indicesMuestreados;indices(double(int32(I)),1)
                          indices(double(int32(I)),2)];
end
path='muestreadas\';
imwrite(C,[path imageName],'tiff','Compression','none');
%figure, imshow(C), title('Imagen C muestreada');

vector = indicesMuestreados;
```

Programa de reconstrucción de mallas en C

Programa que aplica las ecuaciones de reconstrucción y exporta el resultado a un archivo de definición gráfica *.face.

```
int convertToFACE(char *fileOut,float *x, float *y, float *z,int slices, int samples){

     FILE *faceFile;             //Face export file

     char linea[MAX_LENGTH];
     int i,j;                    //Program Counters
     int numComp;                //Coordinates number
     int compCnt=0;              //Init counter

     numComp = slices*samples;

     if (numComp>0){
         if (faceFile=fopen(fileOut,"w"),faceFile){
                    fprintf(faceFile,"%d 1\n", 2*samples*(slices-1));

                    //faces first group
                    for (i=0;i<=slices-2;i++){
                        for (j=1;j<=samples-1;j++){
                            fprintf(faceFile,"%d %d %d %d 1\n",compCnt+1,
                                                               i*samples+j,
                                                               (i+1)*samples+j,
                                                               (i+1)*samples+j+1);
                            compCnt++;
                        }
                    }

                    //close faces first group
                    j=samples;
                    for (i=0;i<=slices-2;i++){
                        fprintf(faceFile,"%d %d %d %d 1\n",compCnt+1,
                                                           i*samples+j,
                                                           (i+1)*samples+j,
                                                           (i+1)*samples+1);
                        compCnt++;
                    }

                    //faces second group
                    for (i=0;i<=slices-2;i++){
                        for (j=1;j<=samples-1;j++){
                            fprintf(faceFile,"%d %d %d %d 1\n",compCnt+1,
                                                               i*samples+j,
                                                               (i+1)*samples+j+1,
                                                               i*samples+j+1);
                            compCnt++;

                        }
                    }
```

```
                    //close faces second group
                    j=samples;
                    for (i=0;i<=slices-2;i++){
                        fprintf(faceFile,"%d %d %d %d 1\n",compCnt+1,
                                                           i*samples+j,
                                                           (i+1)*samples+1,
                                                           i*samples+1);
                        compCnt++;
                    }
                    fclose(faceFile);
          }
          else{
                    return 2; //couldn't write face file
          }
     }
     else
         return 1; //file open error

     return 0;
}
```

Capítulo 4

Los elementos de la estación gráfica del Simulador de RTUP

La estación gráfica del simulador de RTUP, es uno de los bloques más importantes dentro del sistema completo, debido a que es la principal fuente de interacción visual entre el usuario y la computadora. El principal reto consiste en acercarse lo más posible a la realidad en un entorno totalmente virtualizado, lo cual conlleva a la programación de algoritmos de cómputo gráfico para la visualización y emulación de leyes físicas en tiempo real que rigen el comportamiento de los objetos en un espacio llamado escena.

En adelante este trabajo de tesis se centra en los detalles de implementación del simulador desde un punto de vista gráfico, comenzando con los elementos que componen la escena, los efectos visuales y finalmente la presentación en pantalla.

Antes de ello es importante la definición de una metodología general, llamada en esta tesis como el *proceso de creación gráfica del simulador de RTUP*, consistente en la transformación de los espacios real y virtual mediante una serie de pasos, descritos como: 1) investigación, 2) planeación, 3) modelado, 4) simulación, 5) generación de la escena y 6) despliegue; mismos que se detallan a continuación (Ver también figura 4.1).

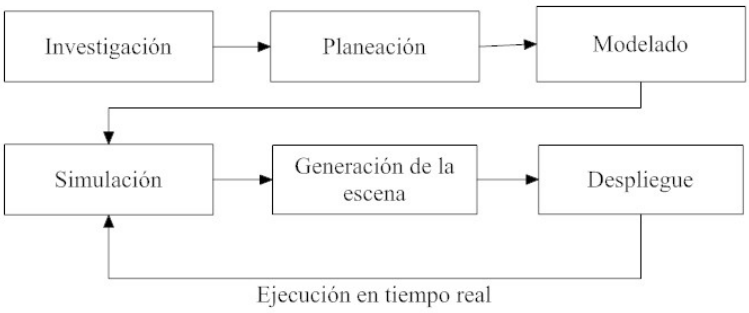


Figura 4.1: *Proceso de creación gráfica del simulador de RTUP. Este diagrama muestra la serie de pasos llevados a cabo para transportar los elementos del mundo real a uno totalmente virtualizado generado por computadora.*

1. **Investigación:** Consiste en la exploración, auxiliados del experto, de los detalles visuales de una cirugía de RTUP. Con base en este enfoque se distinguen algunas características, tales como: restricciones de la vista endoscópica, localización espacial del resectoscopio, movimientos de la

cámara, propiedades de textura, efectos de deformación por la presencia de la lente, iluminación, comportamientos del tejido ante acciones de corte, deformación y sangrado, entre otros. Este trabajo de tesis se auxilió de los videos de cirugías de próstata proporcionados por el urólogo, el Dr. Sergio Durán, del Instituto Nacional de Rehabilitación.

2. **Planeación:** Consiste en el análisis de cada uno de los elementos que intervienen en el procedimiento quirúrgico. Como resultado del análisis, se obtuvieron un conjunto de soluciones, cada una enfocada a resolver características específicas. Por ejemplo, para la restricción de la vista endoscópica se optó por una máscara circula que en todo momento siguiera la cámara. En el caso de la vista espacial del resectoscopio, se analizó el punto donde debería iniciar el recorrido una vez iniciada la simulación. Para los movimientos de cámara, se tomaron en cuenta los grados de libertad que el médico posee durante el procedimiento y en base a ello, se programaron desplazamientos y giros. En la iluminación se fijaron parámetros de visibilidad, con base en los videos y la consistencia de los tejidos. En general, el proceso de análisis consistió en tomar la base real, analizar las alternativas de solución para la transformación al mundo virtual y finalmente la elección de la alternativa más práctica, en términos de implementación.

3. **Modelado:** Implica cuatro tareas principales: 1) la reconstrucción de la malla de volumen de la próstata, 2) la reconstrucción de la malla de superficie del conducto uretral (uretra membranosa y uretra esponjosa), 3) la reconstrucción de la malla de superficie de la vejiga urinaria y 4) la generación del modelo de la herramienta de resección. Este trabajo de tesis plantea dos objetivos principales: la obtención del modelo de la vejiga urinaria a partir de la aplicación del método descrito en el capítulo 3, y la generación de una malla de superficie del conducto uretral (partiendo de su morfología cilíndrica) usando la herramienta de modelado Blender[4]. En el caso del modelo de la próstata, se usó el obtenido por Soriano[56], y para el modelo de la herramienta se tomó el generado por Padilla[40]. Ambos trabajos, realizados sobre las primeras versiones del simulador.

4. **Simulación:** Los modelos en mallas triangulares y de tetraedros, por sí solos no son de gran utilidad mas que para fines de visualización. Por tanto, la asignación de propiedades físicas, haciendo una analogía con la realidad, se convierte en una tarea muy compleja y en gran medida involucra elementos físicos teóricos provenientes del comportamiento del tejido anatómico en el mundo real. Así, en cuestión de simulación, se plantea una transformación entre modelo anatómico físico y modelo anatómico virtual, usando un sistema de nodos con masa (vértices de la malla) conectados por resortes (conectividad entre nodos) que se desarrollan en un medio viscoso. Esto permite obtener, además del modelo gráfico, un modelo físico cuyo comportamiento es muy similar a su análogo en el mundo real. Al respecto, en este trabajo de tesis, el enfoque se basa fundamentalmente en la adaptación de propiedades físicas a los modelos de conducto uretral y vejiga urinaria usando el formato OBJ (Ver capítulo 5). La próstata es una adaptación completa del modelo físico de masas y resortes realizado en trabajos anteriores (Para mayor información ver Padilla[41], Soriano[56, págs. 37-39] y Lira[34, cap. 3]).

5. **Generación de la escena:** Este bloque, también llamado de *rendering*, es uno de los procesos más complejos, ya que consiste en conjuntar todos los elementos de la escena (características de la cámara, modelos anatómicos y luces) y pasar de un mundo abstracto virtual, definido en tres dimensiones, a una imagen en 2D, que se despliega en pantalla. Para tal fin, este trabajo hace uso de las funciones de la biblioteca gráfica *OpenGL*.

6. **Despliegue:** Finalmente, el despliegue permite mediante un dispositivo de visualización, observar el resultado final de todo el proceso de creación gráfica. Como se muestra en la figura 4.1, la etapa de despliegue deriva a una salida que va directamente de regreso al proceso de simulación, ello se debe a que el simulador de RTUP se ejecuta en tiempo real y la interacción con el usuario obliga a la actualización constante de los datos, tales como posición de cámara, cambio de posición de los nodos debido a las deformaciones sufridas por el tejido tras cada colisión con la herramienta,

movimientos de la herramienta, etc., que deben ser reenviados al bloque de simulación para volver a generar una nueva perspectiva de la escena.

Esta metodología de creación gráfica se basa en el proceso encauzado usado en la animación por computadora (PARENT[42, págs. 15-17]), con la diferencia de que no se trata de una animación definida en un tiempo determinado, sino que es una secuencia que cambia en tiempo real y que el usuario puede interactuar con ella. Claramente, este proceso sólo abarca características de la estación de gráficos. Así, en las siguientes secciones se describen los elementos de creación gráfica, los detalles de implementación, las herramientas empleadas y algunos extractos de código escritos en lenguaje C con *OpenGL* para la presentación final del simulador en un dispositivo de visualización.

4.1. Los elementos del simulador de RTUP

Uno de los conceptos más importantes en el desarrollo de un entorno gráfico virtual, es el concepto de escena. En general, podemos definirla como el conjunto de elementos que interactúan para representar una perspectiva de la realidad. En términos de una escena generada por computadora, existen tres elementos básicos: la cámara, los modelos y la iluminación. Todos éstos, que actúan en conjunto para la presentación final, son comunes en cualquier sistema gráfico de realidad virtual.

Haciendo esa analogía con la realidad, el ambiente gráfico virtual del simulador de RTUP, se compone de una escena cuyas características son: un sistema coordenado tridimensional, un modelo de cámara virtual (que representa la perspectiva del observador), un conjunto de modelos (vejiga, próstata, conducto uretral y herramienta de corte) y un modelo de iluminación (Ver figura 4.2).

El sistema coordenado tridimensional transfiere las coordenadas del mundo real al virtual por medio de vectores de coordenadas (x, y, z) representables en la computadora. Los ejes coordenados, comunes en cualquier entorno virtual, tienen una disposición un tanto diferente con respecto al estándar[1], de la tal forma que x corresponde al eje vertical, y al eje horizontal y z al eje de profundidad. Esto último obedece a cuestiones de compatibilidad entre los ajustes realizados por este trabajo de tesis y otros trabajos ya realizados sobre el simulador.

4.1.1. La cámara

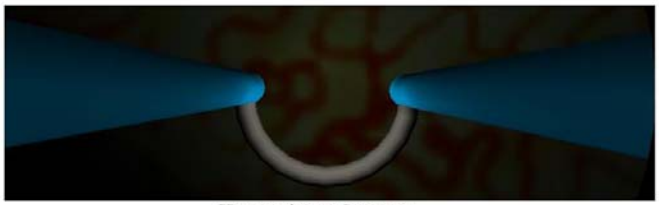
La cámara, dentro de un ambiente virtual, juega el papel del observador dentro de la escena, tal como lo hiciera éste último en el mundo real. En el caso de una endoscopía, la cámara es un elemento físico y sustituye al observador. Así, el proceso de obtención de una imagen a través del mundo virtual puede considerarse como una analogía con la toma de una fotografía o video[63, pág. 96]. Bajo este enfoque, pueden considerarse cuatro pasos, mostrados en la tabla 4.1.

Paso	Cámara real	Cámara virtual
1	Posicionar la cámara	
2	Apuntar al modelo	Direccionar la cámara
3	Ajustar la lente	Definir el volumen de visión
4	Tomar video	Renderizar la escena

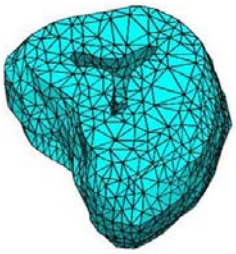
Tabla 4.1: *Analogía entre la toma de video y el proceso de render en tiempo real.*

[1] El sistema de referencia estándar define los tres ejes como: horizontal (x), vertical (y) y de profundidad (z).

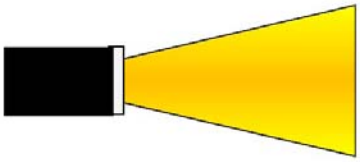
Modelo de vejiga



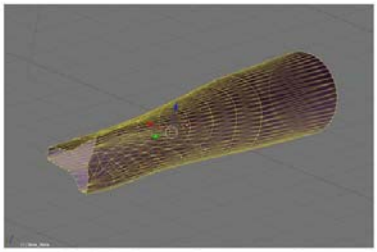
Herramienta de corte



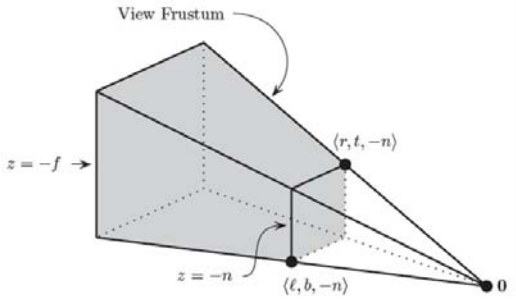
Modelo de la próstata



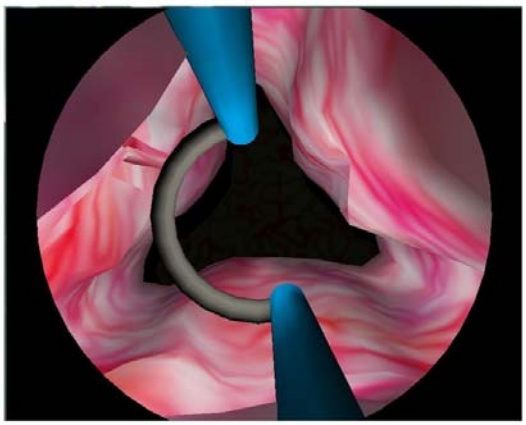
Modelo de iluminación



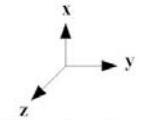
Modelo de conducto uretral



Modelo de cámara



Ambiente gráfico virtual



Sistema de referencia

Figura 4.2: *Los elementos gráficos del entorno virtual del simulador de RTUP. Modelos tridimensionales, sistema de referencia, modelo de cámara y modelo de iluminación virtuales.*

Lo anterior nos hace pensar en la idea de un modelo de cámara, que entre otras cosas contemple: la región de la escena a visualizar, sus transformaciones dentro de la escena (traslaciones y rotaciones principalmente) y el tipo de proyección. El sistema de navegación del simulador tiene estas tres características (Ver figura 4.3), implementadas en lenguaje C y *OpenGL*. Dado que se trata de un sistema que se ejecuta en tiempo real, el proceso se convierte en repetitivo a una tasa de refresco de al menos 30 Hz para causar la sensación de movimiento.

Programación de la cámara con *OpenGL*

La forma de posicionar y direccionar la cámara es mediante la función `gluLookAt()`, la cual tiene como parámetros 9 valores, mismos que corresponden a tres tercias de coordenadas (x, y, z). La primera define la posición del observador (*eye*), la segunda el punto de visión (*center*), tal que, entre esta última y la posición del observador formen un vector de dirección; y la tercera, es un vector normal al plano horizontal de visión (*up*). El formato de instrucción es:

`gluLookAt(`eye_x`,` eye_y`,` eye_z`,` $center_x$`,` $center_y$`,` $center_z$`,` up_x`,` up_y`,` up_z`);`

Para la definición del volumen de visualización, se disponen de dos modos de proyección: ortogonal y en perspectiva. Ésta última ofrece un mayor grado de realismo y su implementación es muy sencilla. Para la proyección en perspectiva, la librería dispone de la función `glFrustrum()`, que sirve para especificar los parámetros del volumen de la escena, compuesta por una pirámide rectangular truncada (ver figura 4.4). El formato de instrucción es:

`glFrustum (float` l`, float` r`, float` b`, float` t`, float` n`, float` f`);`

donde, l (*left*), r (*right*), b (*bottom*) y t (*top*), definen la relación de aspecto del área plana de visualización definida en un plano de corte cercano ($z_n = -n$). Y f (*far*) define el plano de corte lejano ($z_f = -f$). Buss[7, págs. 54-57] define a la región de visualización como el conjunto de todos los puntos que satisfacen las siguientes condiciones:

1. Pertenecen a la región delimitada por los planos cercano y lejano, esto es:
$$n \leq -z \leq f \tag{4.1}$$
2. Pertenecen a la región delimitada por los planos superior, inferior, derecho e izquierdo, esto es:
$$l \leq \frac{nx}{-z} \leq r \tag{4.2}$$
$$b \leq \frac{ny}{-z} \leq t \tag{4.3}$$

Dado que el simulador funciona en tiempo real y tiene movimientos restringidos por los grados de libertad del resectoscopio, estas instrucciones de cámara se implementan dentro de la función de dibujado, la cual, constantemente es actualizada con los valores dados por la interfaz mecatrónica o en su caso por la interfaz de usuario. Esta última, de acuerdo con los modelos de interfaz de Butow[2][8], es de tipo gráfica (GUI[3]), compuesta por un conjunto de botones de control y un conjunto de botones de simulación[4]. Los botones de control manipulan: el movimiento del asa de resección y la cámara virtual;

[2]Butow[8, págs. 27-37] clasifica los modelos de interfaz de usuario como: interfaces por lotes, interfaces de línea de comandos (CLI, *Command-Line Interface*), interfaces de texto (TUI, *Text User Interface*), interfaces gráficas (GUI) e interfaces Web.

[3]GUI, *Graphic User Interface*. Las Interfaces Gráficas de Usuario, desde 1990, han sido el estándar para la comunicación entre el usuario y la computadora desde un punto visual. Fueron introducidas por primera vez en 1984 con la computadora *Macintosh*, de Apple[8, pág. 30-32].

[4]En el apéndice B se tiene una referencia completa sobre el uso de la interfaz de usuario del simulador de RTUP.

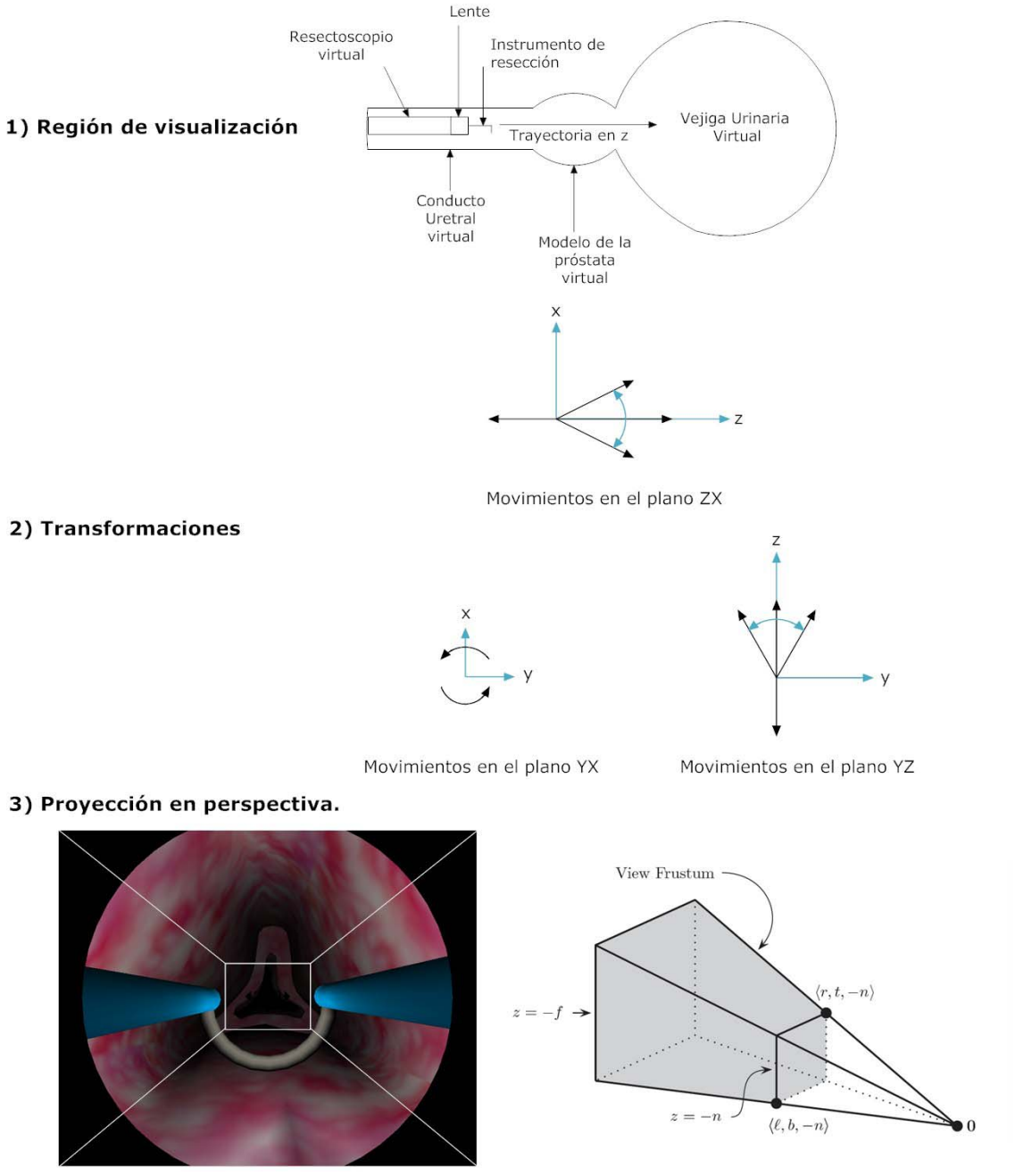


Figura 4.3: *Modelo de cámara del simulador de RTUP. 1) Región de visualización, 2) transformaciones de cámara y 3) proyección en perspectiva[7, pág. 56].*

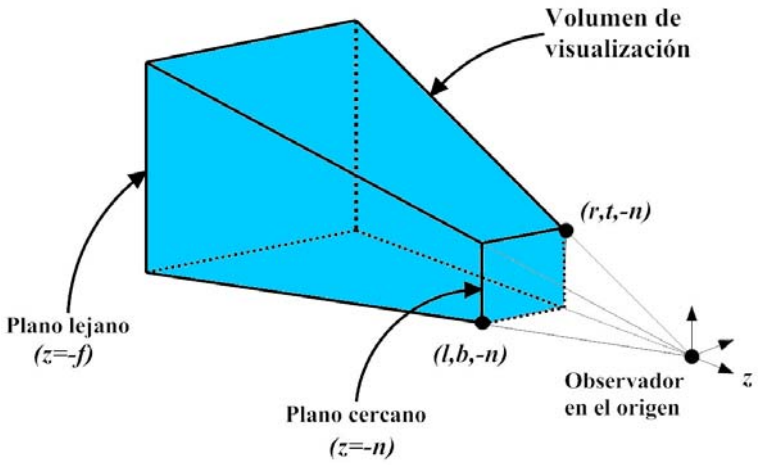


Figura 4.4: *Volumen de visualización de una escena 3D[7, pág. 56].*

logrando sustituir los cinco grados de libertad (tres rotacionales y dos longitudinales) del resectoscopio mecatrónico, como se muestra en la figura 4.5.

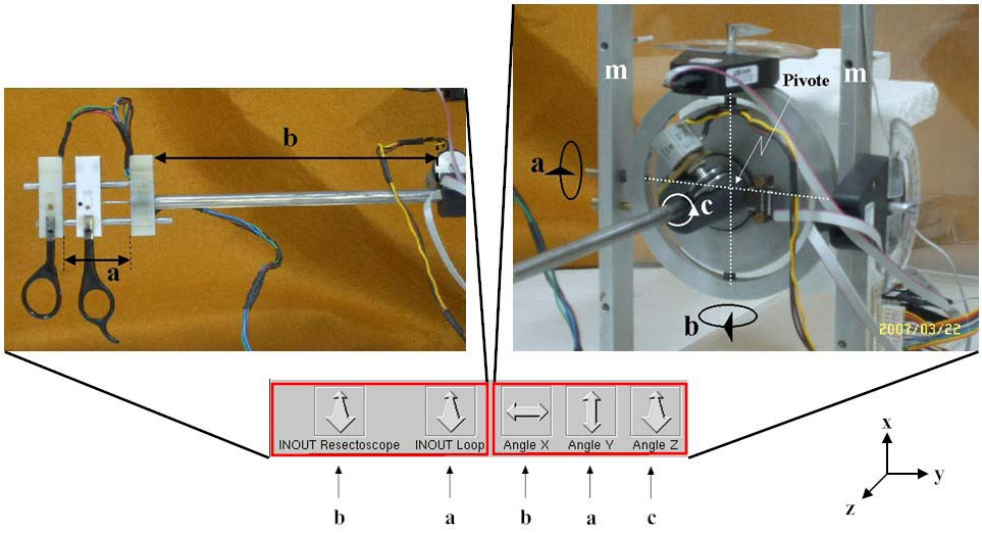


Figura 4.5: *Grados de libertad de la interfaz gráfica del simulador de RTUP. A la izquierda, grados de libertad longitudinales: a) movimientos del asa de resección y b) movimientos de la camisa del resectoscopio. A la derecha, grados de libertad rotacionales: a) rotación en Y, b) rotación en X y c) rotación en Z. Fotos Altamirano[1, Págs. 14,15].*

4.1.2. Los modelos virtuales

Como se vió en el capítulo 2, el sistema de realidad virtual contempla cuatro modelos: tres anatómicos y una herramienta de resección. Los modelos anatómicos son resultado de procesos de reconstrucción y se cargan en el programa principal como archivos de definición 3D (ver capítulo 5)

que contienen información de sus vértices y caras principalmente. La figura 4.6 muestra los cuatro modelos que conforman el software de simulación gráfica.

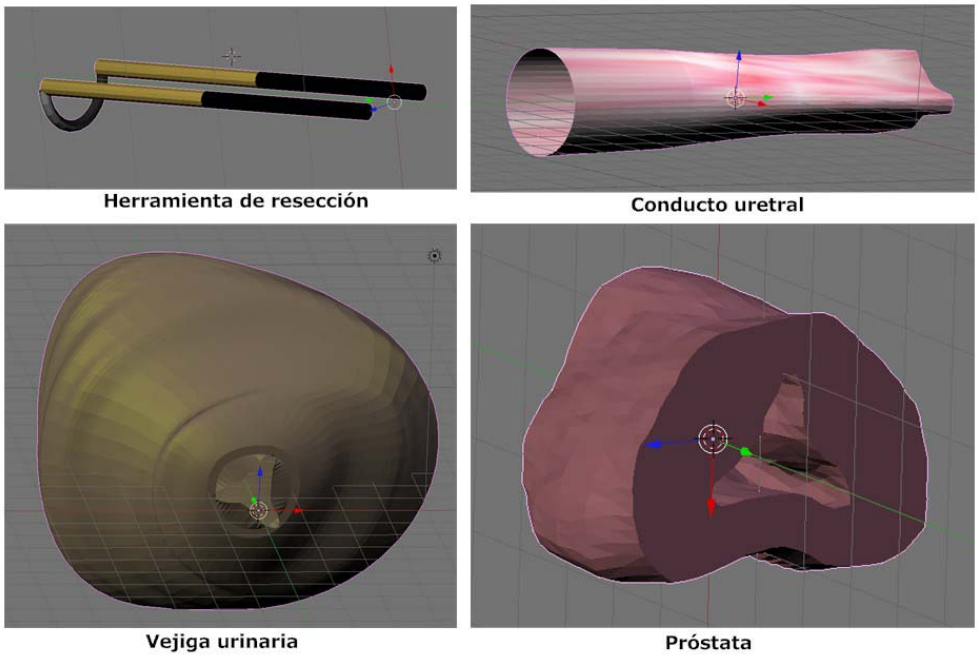


Figura 4.6: *Herramienta de corte y modelos anatómicos del simulador de RTUP.*

Las características principales de los modelos (en formato OBJ y NODE-FACE, según el capítulo 5) son:

- Su forma se basa en modelos físicos reales, en el caso de la próstata a partir de imágenes de ultrasonido, para la vejiga del VHP[62] y la uretra, videos reales de RTU´s como referencia.
- Se definen dos tipos de mallas: de volumen y de superficie. Las primeras están compuestas por tetraedros y las segundas por caras triangulares. Cabe hacer la aclaración que para fines de visualización, las mallas de tetraedros también definen regiones de caras triangulares, lo cual reduce el tiempo de dibujado y por ende el tiempo de procesamiento durante la etapa de *render*. La vejiga, el conducto uretral y el asa de resección son mallas de superficie, mientras que la próstata es una malla de volumen, pues para las deformaciones se considera como un sólido viscoelástico.
- Emplean dos tipos de texturas: 2D y 3D. Las texturas 2D se emplean para los modelos de vejiga, conducto uretral y herramienta de corte. Las texturas 3D se emplean para el modelo de la próstata.
- Las imágenes que sirven de textura fueron generadas a partir de técnicas procedurales (como se explica en la siguiente sección) que dan el efecto de estructuras vasculares.
- Todos los modelos poseen propiedades de material para fines de iluminación.
- Emplean el modelo de iluminación de Phong para la presentación final en pantalla.

- Los modelos de uretra, vejiga y el asa de resección, hasta este punto de desarrollo, son superficies rígidas. El modelo de la próstata posee propiedades de tejido blando, por lo que se clasifica como una malla de volumen deformable.

Texturizado de los modelos

En cómputo gráfico, una textura es un patrón precalculado o calculado en tiempo real que se asocia a la geometría de un objeto. Los elementos de una textura se denominan *texeles* (*texture element*). Idealmente, un texel corresponde a un píxel; sin embargo, realmente un texel corresponde a N píxeles, o bien, N texeles corresponden a un píxel, dependiendo de la posición del observador.

Existen diversas técnicas para la obtención de texturas, entre ellas: la fotografía, el dibujo, los *shaders* (sombreadores) y las técnicas procedurales. La técnica de texturizado a emplear juega un papel decisivo en la visualización final, ya que dependiendo de ésta se determina parte del realismo del sistema gráfico completo.

En este trabajo se usaron texturas procedurales como base para el mapeo de los modelos anatómicos. En el caso del modelo de la herramienta de corte, se usó una textura muy simple, formada por dos áreas de color en degradado. La textura se aplicó usando mapeo 2D, un color para el semi-toroide de corte y otro para los mangos.

Esta tesis no tiene por objetivo la descripción detallada del proceso de creación de texturas procedurales, dado que se trata de otro tema de tesis, actualmente en proceso por parte del equipo desarrollador del simulador. Sin embargo, en lo consecuente se hace una breve introducción a este tipo de texturas y su uso para el mapeo de los modelos gráficos.

Texturas procedurales La generación de texturas procedurales, consiste en la generación de imágenes de textura sintéticas, que se acerquen lo más posible a la textura de un objeto real, y son resultado de la aplicación de alguna técnica procedural. Ebert[9, pág. 1] define, las técnicas procedurales son segmentos de código o algoritmos que especifican alguna característica de un modelo o efecto generado por computadora. Estas técnicas han ido evolucionando, cada vez incorporando nuevas formas de "hacer" texturas a partir de expresiones matemáticas. La tabla 4.2 muestra una breve descripción de la evolución de las técnicas de generación procedural.

Ventajas y desventajas del uso de texturas procedurales En la selección del tipo de texturas a emplear, surge la necesidad de analizar las ventajas y desventajas entre el uso de uno u otro tipo. Así, el simulador toma en cuenta las siguientes afirmaciones[9]:

Ventajas:

- Requieren menor cantidad de memoria dada su representación compacta.
- Ofrecen el mismo nivel de detalle, no importando qué tan cerca o lejos se encuentre la superficie o volumen texturizado.
- Es extensible, arbitrariamente del tamaño del área mapeada, minimizando al máximo el efecto de "costura" debida a la repetición de patrones.

Desventajas:

- Dificultad de implementación y depuración.

- Impredecibles: es necesaria una evaluación del experto para saber si se acerca o no a la realidad.
- Mayor tiempo de cómputo cuando en necesario recalcular en tiempo real.
- Fenómeno de *aliasing*[5].

De acuerdo a lo anterior, una de las grandes ventajas del uso de texturas procedurales y que se aprovecha en la estación de realidad virtual, es la propiedad de "continuidad"; esto es, al "mapear"[6] una textura procedural sobre una superficie no se requiere de una imagen de gran tamaño, ya que al ser una señal estocástica, generalmente estacionaria, es suficiente un patrón base que se repite con el mínimo efecto de costura[7] (Ver figura 4.7d). La figura 4.7 muestra el conjunto de texturas procedurales usadas para los modelos anatómicos de vejiga, próstata y conducto uretral; éstas se basan en un algorítmo de generación de estructuras vasculares que dan la apariencia de venas.

AÑO	DESCRIPCIÓN
1976	Blinn y Nowell emplean la síntesis de Fourier.
1978	Fu y Lu proponen una técnica de generación basada en gramáticas sintácticas
1979-1980	Schacter y Ahuja (1979); y Schacter (1980) usan la síntesis de Fourier y modelos estocásticos de varios tipos para generar texturas aplicadas a simuladores de vuelo.
1981	Investigadores desarrollan modelos de texturas estadísticos, partiendo de las propiedades de textura comunes y su representación con datos estadísticos.
1982-1984	Fourier, Fussell y Carpenter (1982); Haruyama y Barsky (1984), proponen métodos de división estocástica (fractales).
1984	Cook describe los sistemas "shade trees" (árboles sombreadores), uno de los primeros sistemas para generar texturas procedurales durante la fase de *rendering*.
1985	Perlin describe un lenguaje completo para la generación de texturas procedurales y sienta las bases para una de las técnicas más populares, el ruido de Perlin (*Perlin Noise*).
1991	Witkin y Kass describen modelos de texturas sintéticas, inspiradas por procesos bioquímicos que producen patrones de pigmentación en algunas pieles de animales.

Tabla 4.2: *Evolución de las técnicas de generación de texturas procedurales. Extracto Peachey[9, pág. 11]*

Dado que se usaron dos tipos de mapeo (2D y 3D) para los modelos anatómicos, las texturas actualmente disponibles se componen de:

- 1 imagen de textura del tipo de la figura 4.7a de 128x128 píxeles, para mapeo 2D.

[5]El fenómeno de *aliasing* se presenta principalmente en los bordes, donde las líneas que definen al área a mapear se observa dentado. Existen métodos para minimizar este efecto, llamados métodos antialiasing; sin embargo, son generalmente más complejos cuando se aplican a texturas procedurales. Para mayor información acerca de cómo minimizar el efecto de aliasing, ver Peachey[9, pág. 52] y Woo[63, cap. 6]

[6]En este trabajo, se entiende por "mapeo" a la transformación de coordenadas entre espacios del objeto y el de texturas.

[7]En cómputo gráfico, el efecto de costura es muy común y es muy visible en los puntos de unión al usar dos texturas sobre un mismo objeto.

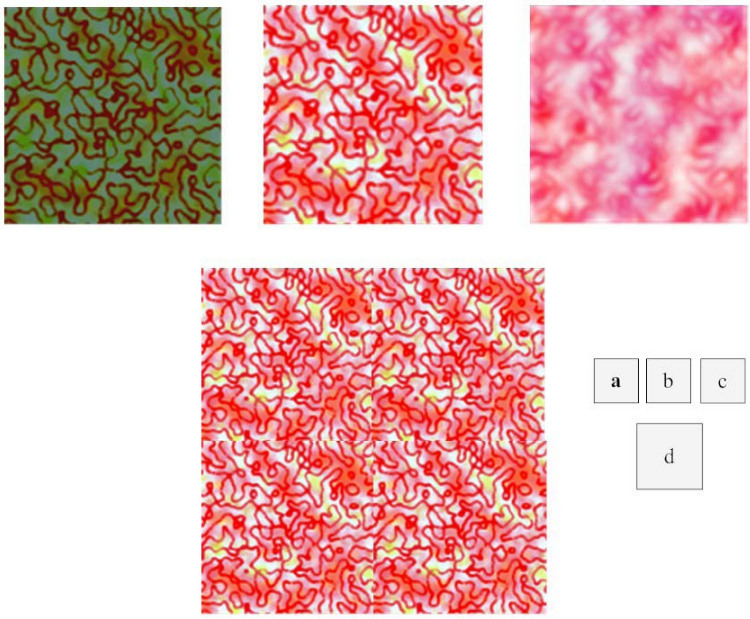


Figura 4.7: *Texturas procedurales empleadas para el mapeo de modelos anatómicos del simulador de RTUP. a) Vejiga (textura 2D), b) próstata (textura 3D), c) próstata (textura 3D) y conducto uretral (textura 2D); d) efecto de continuidad al unir 4 texturas procedurales.*

- 128 imágenes de textura del tipo de la figura 4.7b de 128x128 píxeles, para mapeo 3D.
- 256 imágenes de textura del tipo de la figura 4.7b de 256x256 píxeles, para mapeo 3D.
- 128 imágenes de textura del tipo de la figura 4.7c de 128x128 píxeles, para mapeo 3D.
- 256 imágenes de textura del tipo de la figura 4.7c de 256x256 píxeles, para mapeo 3D.

Mapeo de Texturas 2D Consiste en la parametrización del área de una superficie dentro de un espacio de coordenadas (u, v) unitario y su correspondencia con las coordenadas (s, t) en el espacio de textura. El *render* final usando texturas 2D, es resultado de la proyección del color del píxel mapeado sobre el área de visualización, como se muestra en la figura 4.8.

Mapeo de Texturas 3D El mapeo de texturas 3D es una extensión del mapeo de texturas en dos dimensiones, la diferencia radica en la incorporación de una nueva componente: r. Se define además un espacio del objeto, paramétrico unitario de tres dimensiones, cuyas coordenadas (u, v, w) encierran al objeto en un cubo unitario, transportable a su correspondiente en el espacio de textura. El cubo también se define en el espacio de textura y contiene una pila de imágenes continuas; donde cada componente se mapea entre uno y otro espacio, como se muestra en la figura 4.9.

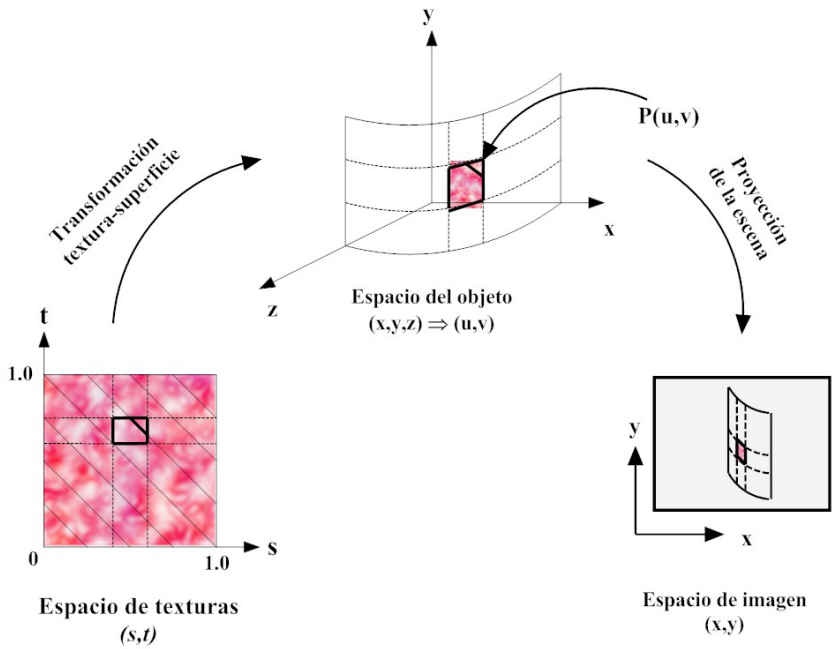


Figura 4.8: *Mapeo de una textura 2D. La imagen de textura, en coordenadas* (s,t) *unitarias, se mapea sobre la superficie, de coordenadas* (u,v) *unitarias. La proyección del color de la superficie sobre el área 2D de visualización determina la vista final.*

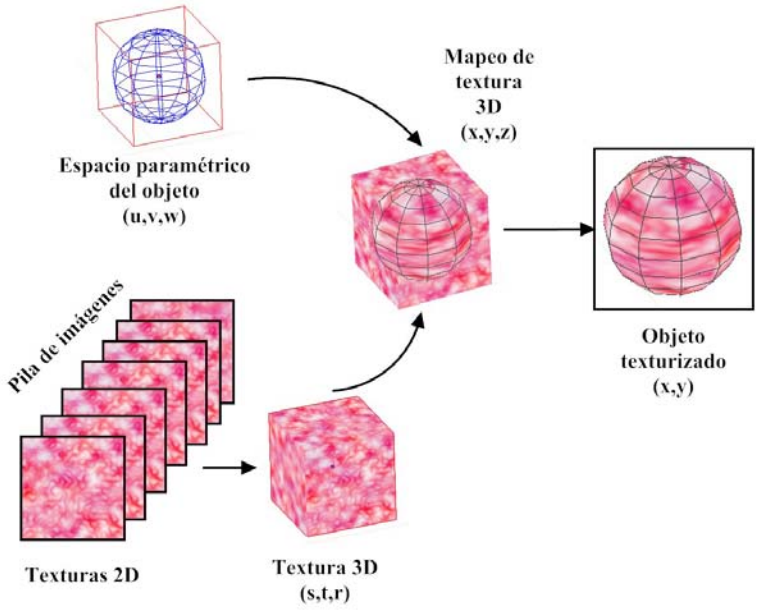


Figura 4.9: *Mapeo de una textura 3D. La pila de imágenes y el modelo de superficie o de volumen, son trasladados al espacio paramétrico unitario definido por un cubo. El modelo texturizado es resultado de las correspondencias entre las coordenadas* (u,v,w) *del modelo (precalculadas o calculadas en tiempo real) y las coordenadas* (s,t,r) *del espacio de textura.*

Mapeo de texturas con OpenGL OpenGL tiene la gran ventaja de poseer un módulo de carga de texturas y mapear en: una, dos o tres dimensiones. El algoritmo de mapeo para dos y tres dimensiones, usado en el simulador, se muestra a continuación (Hart[9, págs. 413-416]).

Algoritmo de mapeo de texturas

- Cargar texturas
- Normalizar textura con respecto al espacio de objetos
- Activar textura
- Para cada polígono p
  - Iniciar polígono
  - Por cada vértice i
    - TexCoord($p[i].uv$) (Texturas 2D) / TexCoord($p[i].uvw$) (Texturas 3D)
    - Vértice($p[i].xyz$)
  - Finalizar polígono

La elección de uno u otro tipo de texturizado (1D, 2D o 3D), se elige por medio de la función `glEnable()`, teniendo como parámetro la opción `GL_TEXTURE_1D`, `GL_TEXTURE_2D` o `GL_TEXTURE_3D`. Para deshabilitar alguno de los tipos se usa la función `glDisable()`. La carga de texturas 2D se realiza mediante la función `glTexImage2D()`, y para texturas 3D se usa `glTexImage3D()`. La activación se realiza por medio de la función `glBindTexture()`; y finalmente, para la asignación de coordenadas de textura se usa la función: `glTexCoord2f(s,t)` para texturas 2D y `glTexCoord3f(s,t,r)` para texturas 3D[8].

4.1.3. Iluminación

Los modelos de iluminación, en cualquier ambiente gráfico, tienen por objetivo la simulación realista de la interacción de la luz con el material de una superficie (Peachey[9, pág. 7]). Existen una gran variedad de modelos propuestos, cada uno con características propias. Se dividen en dos tipos: locales y globales. Los modelos locales son aquellos que analizan la interacción directa de los rayos de luz con el material, mientras que los modelos globales se resuelven a partir de la interacción indirecta; resultado de la reflexión, refracción y dispersión causados por la presencia de otras superficies e inclusive del medio en que se desarrolla la escena. Ejemplos de éste último tipo se pueden mencionar las técnicas de *ray tracing*[9] y radiosidad[10].

Dado que la iluminación tiene una implicación directa en el proceso de *render*, existen marcadas ventajas entre el uso de uno u otro modelo, siendo el tópico más importante, el tiempo de cálculo

[8] Para mayor información acerca de la carga de texturas 2D y 3D con OpenGL, consulte *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*[63, cap. 9]

[9] Las técnicas de *ray tracing* o de trazo de rayos, son empleadas principalmente para la obtención de imágenes 2D a partir de una escena tridimensional (*rendering*). Éstas parten de la idea principal del *ray casting*, donde el rayo sigue una trayectoria hasta colisionar con la superficie, y el color del píxel es calculado a partir del color del material de la superficie en el punto de colisión. La diferencia es que en el *ray tracing*, el rayo sigue una trayectoria acumulativa, donde el color final del píxel en la imagen 2D se calcula a partir de todas las contribuciones de los objetos de la escena (Para mayor información acerca de las técnicas de *ray tracing*, consulte Hearn[20, págs. 527-544]).

[10] Las técnicas de radiosidad consideran las interacciones de energía radiante transferidas entre todas las superficies de la escena (Para mayor información acerca de las técnicas de radiosidad, consulte Hearn[20, págs. 544-552]).

necesario para pasar de la escena 3D a la imagen que se despliega en pantalla. Los modelos locales, sobre los modelos globales, ofrecen un menor tiempo de cálculo; aunque los resultados no son tan realistas como los globales, sin embargo, para fines de simulaciones en tiempo real el realismo es suficiente. Por lo tanto, dentro del proceso de planeación se optó por un modelo de iluminación local, el cual contemple el uso de una fuente de luz (presente físicamente durante una endoscopía) y la asignación de propiedades de material a los modelos. Estas características son típicas del modelo de iluminación de Phong.

El modelo de iluminación de Phong dota a cada superficie con propiedades de material[11] y contempla tres tipos de iluminación: ambiental, difusa y especular. El material de una superficie se define por cuatro componentes, que son: componente de luz emitida, componente ambiental, componente de reflexión difusa y componente de reflexión especular.

Componente de luz emitida Se refiere a la propiedad que define la emisividad de una superficie. La emisividad controla qué tanta luz emite una superficie ante la ausencia de cualquier luz incidente. La luz emitida, no actúa como una fuente de luz que ilumina otras superficies; por lo tanto, ésta sólo afecta el color que ve el observador (Buss[7, pág. 70]). La figura 4.12a muestra un ejemplo de la propiedad de emisividad de una superficie.

Componente ambiental (Ka): Se refiere a la propiedad que permite iluminar de manera uniforme los objetos en la escena. La luz proviene de una fuente situada en el infinito e incide sobre las superficies en todas direcciones por igual, provocando un efecto de color plano. En algunas ocasiones, a esta componente se le asocia con el color de una superficie. Sin embargo, es importante mencionar que ello sólo sucede cuando su valor es tal que el color predominante, durante el cálculo de la iluminación, es la componente de color ambiental del objeto. La figura 4.12b muestra un ejemplo de objeto iluminado únicamente con luz ambiental.

Componente de reflexión difusa (Kd): Se refiere a la propiedad de sombreado de una superficie (Ver figura 4.12c). Éste último se logra al considerar una o varias fuentes de luz, cuyos rayos inciden sobre la superficie, causando los fenómenos de dispersión y reflexión (Ver figura 4.10). Para el cálculo de las contribuciones de esta componente en la iluminación final, es necesaria la definición de los vectores de dirección de la reflexión; usualmente se suelen usar las normales geométricas de la superficie. En el simulador, estos vectores de dirección se calculan en tiempo real para el modelo de la próstata, debido a que la deformación del tejido blando provoca cambios en las caras triangulares. En el caso de los modelos de vejiga y conducto uretral, los vectores de dirección son precalculados en Blender y exportados a un archivo de definición 3D OBJ (Ver capítulo 5).

Componente de reflexión especular (Ks): Se refiere a la componente de iluminación causante de un brillo pronunciado en cierta región de la superficie, debida a un fenómeno de la luz conocido como reflexión especular[12] (Ver figura 4.12d). Como en el caso de la reflexión difusa, es necesario el cálculo de cada vector de dirección que depende del ángulo de incidencia (formado entre el rayo luminoso y la normal a la superficie en el punto de incidencia) y el ángulo de reflexión; éste último dado por la primera ley de la reflexión[13] (Ver figura 4.11).

[11] El material es la propiedad de un objeto reflector que define qué tanto es iluminado por la presencia de una fuente de luz (Buss[7, pág. 70]).

[12] Hecht[21, pág. 833] dice: *"Cuando un rayo incide sobre una superficie reflectora pulida, la luz remitida por millones de átomos se combinará para formar un solo rayo bien definido, en un proceso denominado reflexión especular (en espejo)."*.

[13] La ley de la reflexión, dice: *"El ángulo de incidencia es igual al ángulo de reflexión, esto es* $\theta_i = \theta_r$*."*[21, pág. 832].

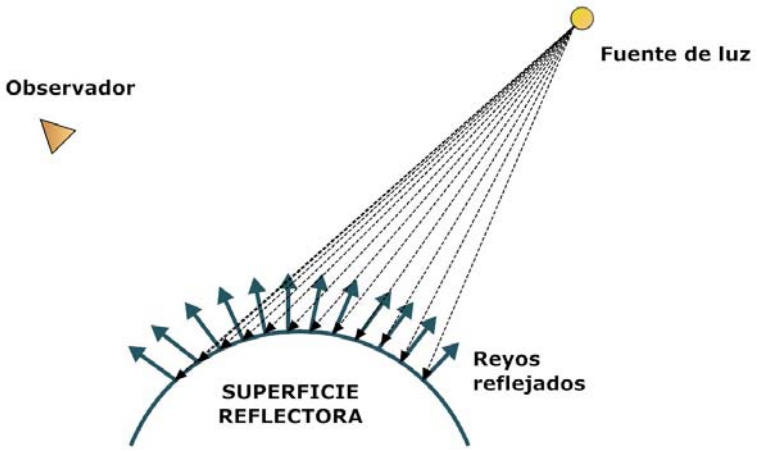


Figura 4.10: *Reflexión difusa. Los rayos provenientes de la fuente de iluminación puntual son reflejados en la superficie en una dirección específica (usualmente la normal geométrica). El observador recibe sólo cierta cantidad de color del objeto dependiendo de su posición, lo que provoca el efecto de sombreado.*

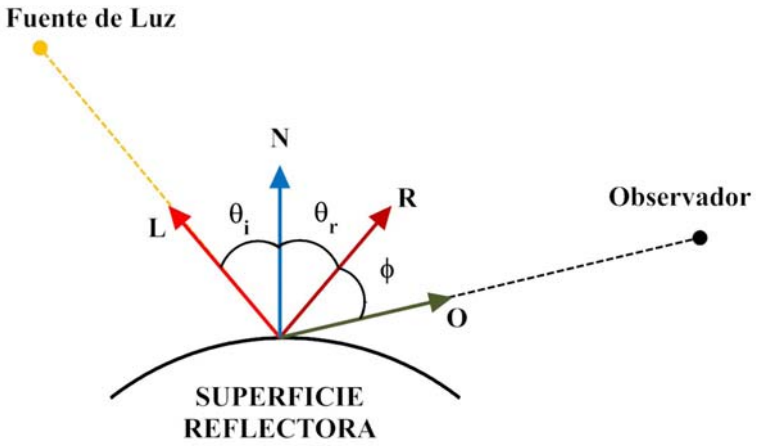


Figura 4.11: *Reflexión especular. El ángulo de incidencia es igual al ángulo de reflexión. En la figura,* $\boldsymbol{L}$ *representa un vector que apunta en dirección a la fuente de luz,* $\boldsymbol{N}$ *es un vector normal a la superficie en el punto de incidencia,* $\boldsymbol{R}$ *es el rayo ideal reflejado y* $\boldsymbol{O}$ *es un vector que apunta en dirección al observador en un ángulo* ϕ*, con respecto al ángulo del rayo reflejado[20, pág. 501].*

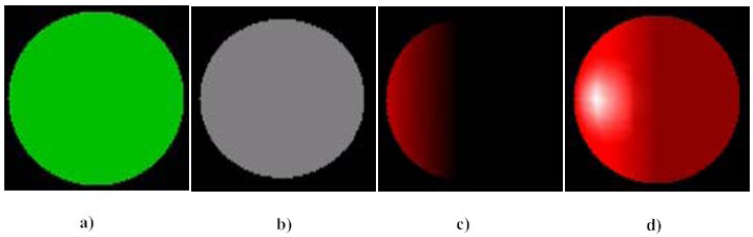


Figura 4.12: *Superficie iluminada usando diferentes componentes de iluminación. a) Usando componente de luz emitida, b) usando sólo componente ambiental, c) usando componentes ambiental y difusa, d) usando componentes ambiental, difusa y especular (Gastélum[14, pág. 45:46]).*

En el simulador de RTUP, la fuente de luz y los modelos tridimensionales son considerados como objetos luminosos[20, pág. 496]. La fuente emisora (direccional) se define como un cono, cuyo vértice tiene coordenadas en (x, y, z), un vector de dirección y un ángulo de apertura. La figura 4.13 muestra un esquema de la fuente de luz empleada y sus características básicas. Los modelos tridimensionales tienen propiedades de material, siendo de mayor interés la asignación de características metálicas al asa de resección virtual, como se muestra en la tabla 4.3.

MATERIAL	R	G	B
Oro	0.93	0.88	0.38
Iridio	0.26	0.28	0.26
Hierro	0.44	0.435	0.43
Níquel	0.50	0.47	0.36
Cobre	0.93	0.80	0.46
Platino	0.63	0.62	0.57
Plata	0.97	0.97	0.96

Tabla 4.3: *Tabla de valores para la asignación de propiedades de iluminación de materiales metálicos (Touloukian and Witt, 1970)[7, pág. 97].*

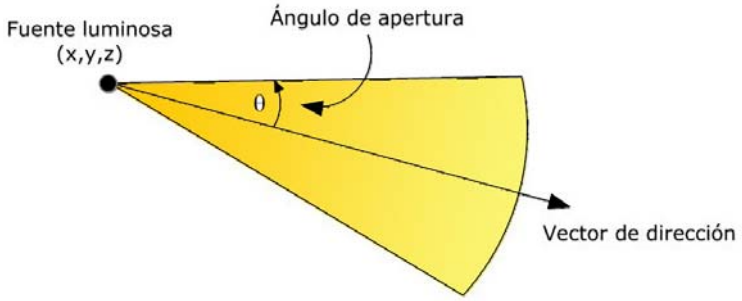


Figura 4.13: *Fuente de luz direccional. En la escena tridimensional, la fuente de luz posee posición, dirección y componentes de iluminación.*

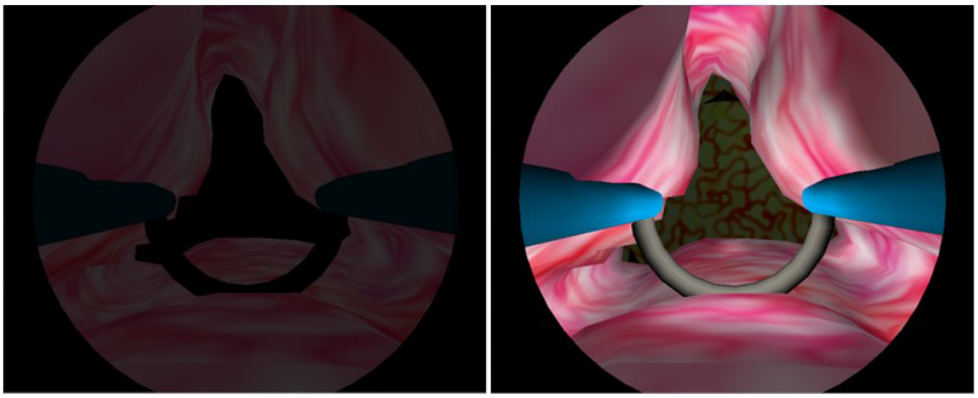


Figura 4.14: *Iluminación del sistema de RTUP. A la izquierda, sin fuentes de iluminación, sólo iluminación ambiental. A la derecha, iluminación con una fuente de luz direccional.*

Implementación del modelo de iluminación con OpenGL

La implementación del modelo de iluminación se llevó a cabo en dos pasos: el primero consiste en la definición de una luz direccional que se mueve en conjunto con el resectoscopio gráfico virtual, y el segundo en la definición de propiedades de material a los objetos reflectores, en este caso los modelos anatómicos y la herramienta de corte.

El siguiente código corresponde a la implementación de la luz direccional que ilumina la escena del simulador.

```
//propiedades de reflexión
GLfloat l_amb[] = {0.2, 0.2, 0.2, 1.0};
GLfloat l_diff[] = {1.0, 1.0, 1.0, 1.0};
GLfloat l_spec[] = {0.5, 0.5, 0.5, 1.0};

glLightfv(GL_LIGHT0, GL_DIFFUSE, l_diff);
glLightfv(GL_LIGHT0, GL_AMBIENT, l_amb);
glLightfv(GL_LIGHT0, GL_SPECULAR,l_spec);

//definición de tipo de luz direccional
glLightf( GL_LIGHT0, GL_SPOT_CUTOFF, 90.0f);
glLightf( GL_LIGHT0, GL_SPOT_EXPONENT, 120.0f);

//propiedades del modelo de iluminación y activación de luz
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_FALSE);
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
....
```

Dentro de la función de dibujado, la función que permite el desplazamiento de la luz direccional junto con la cámara es:

```
glLightfv(GL_LIGHT0,GL_POSITION,lightPosition);,
```

donde `lightPosition` es el vector de posición definido por la posición de la herramienta de corte.

La dirección de la luz está dada por

`glLightfv(GL_LIGHT0,GL_DIRECTION,lightDirection);`,

donde `lightDirection` define el vector de dirección de la luz. En el caso del simulador siempre apunta hacia la trayectoria longitudinal en dirección a la vejiga virtual.

La asignación de materiales depende del tipo de malla. Las mallas anatómicas poseen material plástico para lograr un mejor efecto de reflexión, mientras que la malla del asa de resección posee propiedades metálicas.

Las funciones de *OpenGL* usadas para la asignación de materiales fueron:

```
glMaterialf ( GL_FRONT_AND_BACK, GL_EMISSION, emission_body );
glMaterialfv( GL_FRONT_AND_BACK, GL_AMBIENT,  ambient_body );
glMaterialfv( GL_FRONT_AND_BACK, GL_DIFFUSE,  diffuse_body );
glMaterialfv( GL_FRONT_AND_BACK, GL_SPECULAR, specular_body );
```

donde `emission_body`, `ambient_body`, `diffuse_body` y `specular_body`, corresponden a las propiedades de emisividad, componente ambiental, reflexión difusa y reflexión especular respectivamente, definidas en el modelo de iluminación de Phong.

Los valores usados en el simulador para estas últimas componentes se muestran en la tabla 4.4.

Componente	Loop-cylinder	Loop-torus	Prostate	Urethra	Bladder
Emission	12.00000	25.00000	11.26400	2.000000	2.000000
Ka (R)	0.000000	0.192250	0.250000	0.000000	0.000000
Ka (G)	0.678400	0.192250	0.207250	0.000000	0.000000
Ka (B)	0.960000	0.192250	0.207250	0.000000	0.000000
Ka (A)	1.000000	1.000000	1.000000	1.000000	1.000000
Kd (R)	0.427451	0.507540	1.000000	0.800000	0.660818
Kd (G)	0.470588	0.507540	0.829000	0.800000	0.577718
Kd (B)	0.541176	0.507540	0.829000	0.800000	0.444758
Kd (A)	1.000000	1.000000	1.000000	1.000000	1.000000
Ks (R)	0.333333	0.508273	0.296648	0.800000	0.415173
Ks (G)	0.333333	0.508273	0.296648	0.800000	0.487882
Ks (B)	0.333333	0.508273	0.296648	0.800000	0.000000
Ks (A)	1.000000	1.000000	1.000000	1.000000	1.000000

Tabla 4.4: *Tabla de valores para materiales de los modelos del simulador de RTUP.*

4.2. Otros efectos visuales

4.2.1. Efecto de la vista endoscópica

La vista endoscópica durante una cirugía de RTUP, muestra dos características principales: el efecto de distorsión de la lente y la visión limitada por una máscara circular. Esta última limita el campo de visión a un radio determinado por el tamaño de la lente. Otra característica, que no es objetivo de análisis en esta tesis pero que puede llegar a ser muy importante para el realismo del simulador, es el efecto de reflexión de la luz en la superficie de la lente; un efecto que en su momento causa la sensación de halos luminosos y obstrucción de la visibilidad.

Efecto de la lente

El efecto de la lente (en desarrollo), en un futuro permitirá al simulador proveer al usuario la sensación visual de distorsión, provocada por la misma geometría. Este efecto tiene muchas implicaciones y en gran medida involucra la caracterización de la herramienta endoscópica; la cual, de acuerdo con el médico especialista, tiene dos características: un ángulo de inclinación de 30° con respecto al plano de visualización y una lente gran angular; la cual provoca que la amplificación de la imagen sea mayor en el centro y vaya decayendo conforme se avanza radialmente hacia el borde de la máscara circular, este efecto comúnmente se le denomina *distorsión de barril*[15] (Ver figura 4.15).

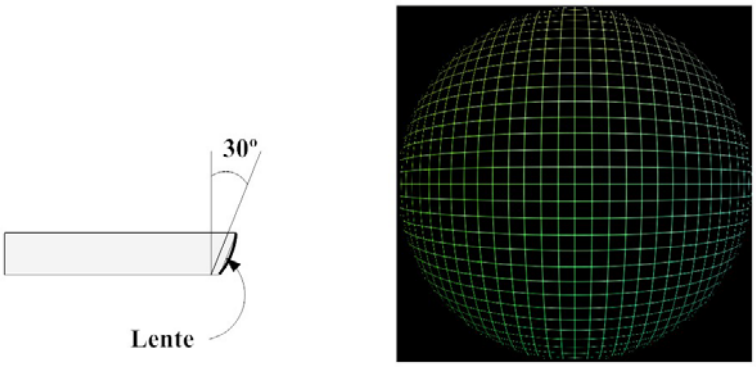


Figura 4.15: *Efecto de la lente. A la izquierda, ángulo de inclinación de la lente. A la derecha, distorsión de barril[15].*

En una primera fase de desarrollo, el principio básico de distorsión parte de una textura resultado del proceso de *rendering*. La información de color de cada texel en la textura (en términos de almacenamiento en memoria, *framebuffer*) es sometido a una función de correspondencia, de tal forma que el efecto de distorsión óptica se adapte al modelo de barril, descrito anteriormente. Bourke[5] describe un método de deformación basado en una función de correspondencia no lineal. El algoritmo que describe este método, trabaja sobre los texeles de una textura inicial (resultado del proceso de *rendering*), mismos que se definen en función de coordenadas (u_i, v_j). Los texeles $t(u_i, v_j)$ definen conjuntos de píxeles encerrados en regiones definidas por vértices. Así, un texel se conforma por nxn píxeles, donde n es la resolución vertical y horizontal del texel. En términos de píxeles, los vértices se definen por el cuadrado formado por las coordenadas de píxel: (i, j), $(i+n, j)$, $(i+n, j+n)$ y $(i, j+n)$ como se muestra en la figura 4.16.

Entonces, sean T_1 y T_2 dos texturas (*framebuffers*) en memoria, el algoritmo de distorsión de barril en T_2 a partir de T_1 se describe como:

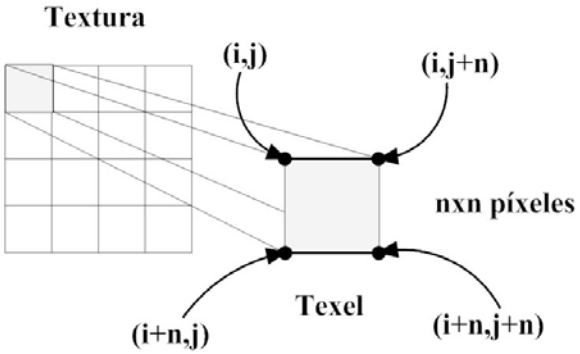


Figura 4.16: *Vértices que definen un texel. El área definida por los cuatro vértices contiene nxn píxeles, donde n es la resolución horizontal y vertical del texel.*

Algoritmo de mapeo de texturas

1: Activar T_1 resultado del rendering

2: Obtener T_2 distorsionada a partir de la información de T_1

- Para cada texel t de coordenadas (u_i, v_i) en T_1
  - Transformar $t(u_i, v_j)$ y obtener $t'(u_i^x, v_j^y)$
  - Pintar texel $t'(u_i^x, v_j^y)$ en T_2

La función de transformación $f : t \rightarrow t'$ trabaja sobre cada vértice que conforma el texel.

Así, para cada vértice (i, j) del texel $t(u_i, v_j)$, con $w = screenwidth$ (tamaño horizontal de la textura) y $h = screenheight$ (tamaño vertical de la textura):

- Transformar i al espacio de [-1 a 1]:

$$x = \frac{i}{w/2} - 1 \tag{4.4}$$

- Transformar j al espacio de [-1 a 1]:

$$y = \frac{j}{h/2} - 1 \tag{4.5}$$

- Obtener radio:

$$r = \sqrt{x^2 + y^2} \tag{4.6}$$

- Obtener ángulo:

$$\theta = atan(y/x) \tag{4.7}$$

- Aplicar función de distorsión al radio:

$$r_n = \frac{2 * asin(r)}{\pi} \tag{4.8}$$

- Obtener coordenada x con distorsión:

$$x_n = r_n * cos(\theta) \tag{4.9}$$

- Obtener coordenada y con distorsión:

$$y_n = r_n * sin(\theta) \tag{4.10}$$

- Antitransformar x_n al espacio de textura:

$$ix = \frac{w * (x_n + 1)}{2} \tag{4.11}$$

- Antitransformar y_n al espacio de textura:

$$jy = \frac{h * (y_n + 1)}{2} \tag{4.12}$$

Efecto de máscara circular

El efecto de máscara circular tiene por objetivo la obstrucción de la visibilidad en los bordes de la escena, tal como sucede en una endoscopia real. La implementación se logró añadiendo un nuevo elemento a la escena, un plano con un orificio circular que sigue los movimientos de la cámara en todo momento, obstruyendo la visibilidad de parte de la escena. La figura 4.17 muestra el esquema de funcionamiento de este efecto.

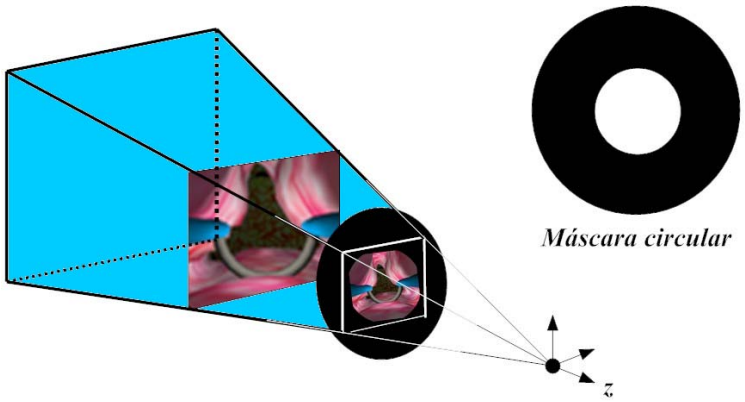


Figura 4.17: *Efecto de máscara circular.*

4.2.2. Efecto de profundidad limitada con aplicación de niebla

En algunas ocasiones, las imágenes en 2D obtenidas del proceso de *rendering* no son del todo realistas, debido a los cambios bruscos en la geometría de los modelos (que se basan en caras triangulares y/o poligonales). Este inconveniente generalmente se resuelve aplicando filtros que suavizan los bordes, y adicionalmente se pueden usar efectos de niebla, sobre todo cuando se requiere que un objeto se desvanezca conforme a la distancia. La niebla es un elemento esencial en simulaciones donde se requiere aproximar al máximo el realismo de una visibilidad limitada.

El efecto de niebla o *fog*, en inglés, es empleado en el simulador para lograr dos efectos: limitación de visibilidad profunda y obstrucción de visibilidad por sobrepaso de límites del tejido por parte del resectoscopio. El primero permite dar la ilusión de visión de profundidad limitada al obstruir la visión lejana mediante nubosidades obscuras, dejando ver sólo una parte del trayecto. Este efecto es muy importante sobre todo en la entrada al conducto uretral, donde únicamente se visualiza una porción

y se logra un efecto de túnel, tal como lo describe el médico especialista (Ver figura 4.18). El segundo efecto ocurre cuando el resectoscopio sobrepasa los límites que encierran los modelos tridimensionales, se presenta un efecto de nubosidad de color rojo que obstruye parcialmente la visión endoscópica virtual. Este último se aplico para simular la obstrucción de visión por sangrado (en proceso de implementación) al cortar tejido durante el procedimiento de resección (Ver figura 4.19).

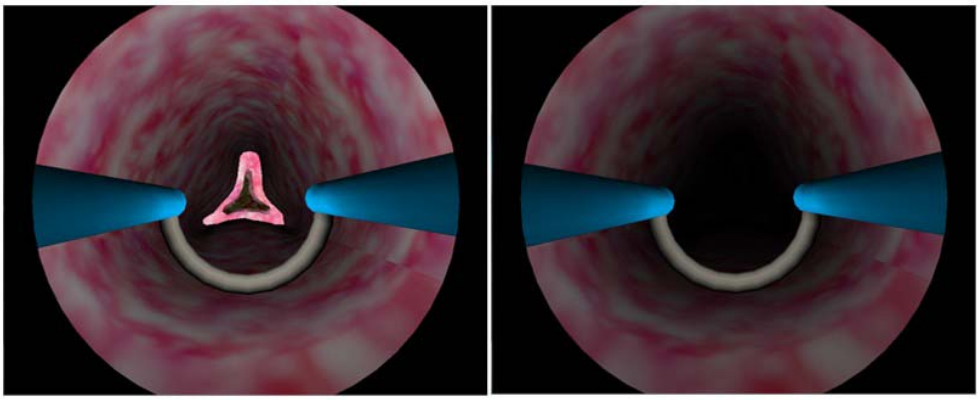


Figura 4.18: *Aplicación del efecto de niebla al simulador de RTUP. A la izquierda, sin efecto de niebla. A la derecha, con efecto de niebla.*

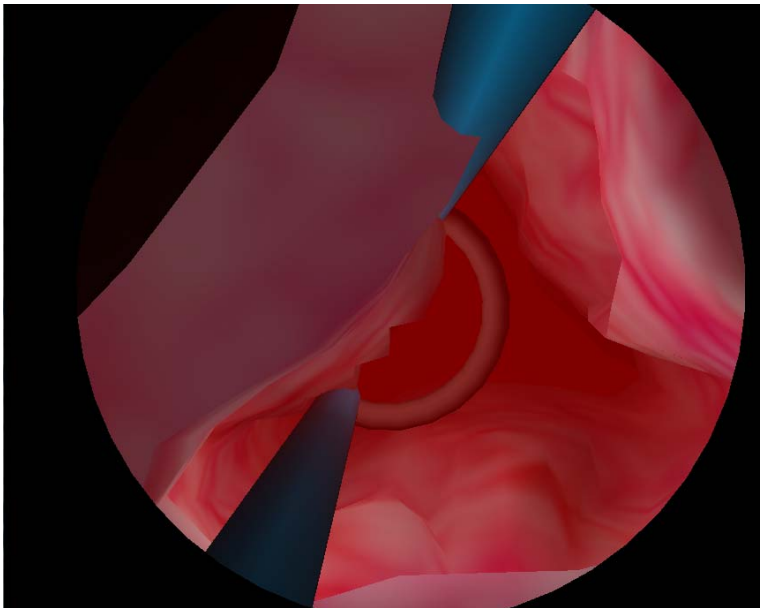


Figura 4.19: *Obstrucción parcial de la visibilidad de la escena por efecto de niebla.*

Agregar niebla con *OpenGL* se vuelve una tarea muy sencilla, dado que esta librería cuenta con un módulo (deshabilitado inicialmente), que se habilita durante la definición de propiedades iniciales de *OpenGL* y posteriormente se le pueden asignar valores a sus parámetros. Para habilitar y deshabilitar el efecto, se usan las instrucciones `glEnable(GL_FOG)` y `glDisable(GL_FOG)`, respectivamente. Para controlar el color y las ecuaciones de densidad a emplear, se usa la función `glFog*()`, donde * representa el tipo de dato que toma el argumento y que pude ser de tipo: flotante (f), entero (i), vector de valores

flotantes (fv) o vector de valores enteros (iv).

La especificación de la instrucción para valores flotantes y enteros se define como:

```
void glFogf  ( GLenum pname , GLfloat param );
void glFogi  ( GLenum pname , GLint param );
```

donde `pname` especifica el identificador de parámetro, que puede ser: `GL_FOG_MODE`, `GL_FOG_DENSITY`, `GL_FOG_START`, `GL_FOG_END`, y `GL_FOG_INDEX`; y `param` especifica el valor del parámetro.

La especificación de la instrucción para valores flotantes y enteros vectorizados se define como:

```
void glFogfv  ( GLenum pname , const GLfloat *params );
void glFogiv  ( GLenum pname , const GLint *params );
```

donde `pname` especifica el identificador de parámetro, que puede ser: `GL_FOG_MODE`, `GL_FOG_DENSITY`, `GL_FOG_START`, `GL_FOG_END`, `GL_FOG_INDEX`, y `GL_FOG_COLOR`; y `param` especifica el apuntador al vector de valores del parámetro.

En la tabla 4.5 se describen los significados de cada identificador de parámetro de la instrucción `glFog*()`.

IDENTIFICADOR	SIGNIFICADO
`GL_FOG_MODE`	Especifica la ecuación de niebla a usar. Loa valores posibles son: `GL_LINEAR` (modo lineal), `GL_EXP` (modo exponencial), y `GL_EXP2` (modo exponencial). El valor inicial es `GL_EXP`.
`GL_FOG_DENSITY`	Especifica la densidad de la niebla a usar en los modos exponenciales. El valor inicial es 1.
`GL_FOG_START`	Especifica la distancia más cercana de la niebla cuando se aplica la ecuación lineal. El valor inicial es 0.
`GL_FOG_END`	Especifica la distancia más lejana de la niebla cuando se aplica la ecuación lineal (modo lineal).
`GL_FOG_INDEX`	Especifica el índice de color i_f. El valor inicial es 0.
`GL_FOG_COLOR`	Especifica el color de la niebla C_f. El color inicial es el negro de componentes RGBA (0,0,0,0).

Tabla 4.5: *Identificadores de parámetro de la función glFog*(). Extracto OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL, Version 1.2[63].*

Las ecuaciones del efecto de niebla

Las ecuaciones que determinan las propiedades de la niebla se presentan a continuación.

$$f = e^{-\rho z} \tag{4.13}$$

$$f = e^{-(\rho z)^2} \tag{4.14}$$

$$f = \frac{end - z}{end - start} \tag{4.15}$$

donde, f es el factor de niebla (factor de atenuación de color), ρ es la densidad, z es la distancia entre el punto de vista y el punto de color, *end* es la distancia más lejana y *start* es la distancia más cercana.

La figura 4.20 muestra las gráficas de desvanecimiento de color conforme la distancia. Puede notarse que en el caso del factor lineal, el decaimiento de color es proporcional y tiene pendiente negativa. Las exponenciales negativas decaen de acuerdo al valor de densidad; entre más grande es este valor, el desvanecimiento de color real es más rápido.

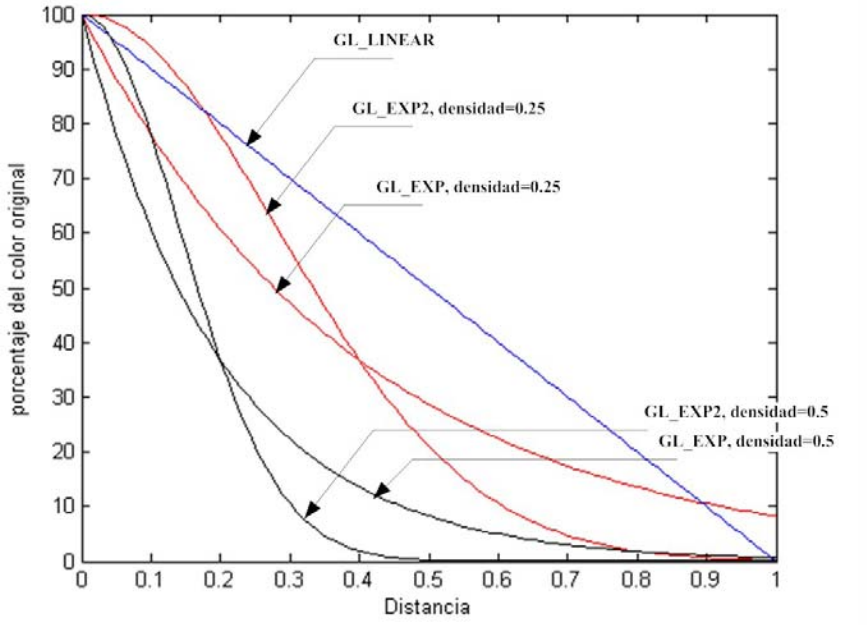


Figura 4.20: *Gráfica de las ecuaciones de densidad de niebla al asignar diferentes valores a sus parámetros.*

4.3. El sistema de despliegue

Finalmente, el sistema de despliegue o *raster*, tiene como principal tarea la conversión de los datos almacenados en memoria, resultado del proceso de *rendering*, a información representable en un dispositivo de visualización. Generalmente éstos últimos consisten en un monitor simple, un proyector sencillo, un casco de realidad virtual, pantallas estéreo de alta definición e inclusive una sala de realidad virtual completa (tales como el observatorio de visualización IXTLI[27]).

La figura 4.21 muestra las arquitecturas de un sistema básico y mejorado de despliegue. El simulador emplea estas dos arquitecturas para lograr la compatibilidad entre plataformas de hardware.

Los dispositivos de entrada y salida corresponden a la interfaz mecatrónica, el teclado y el ratón (aunque estos dos últimos son alternos ante la falta de la interfaz mecatrónica), que intercambian la información del entorno real al entorno gráfico virtual por medio de un sistema de buses. Éstos últimos interconectan tres bloques principales, presentes en cualquier sistema de cómputo: el procesador central (CPU) dedicado al cálculo de deformaciones de tejido (aunque cabe aclarar que el simulador también dispone de la opción de cálculos con GPU[14]); la memoria de video, encargada de almacenar los datos de visualización o *framebuffers* y el controlador de video, encargado de traducir la información en memoria a información visual en el dispositivo de despliegue.

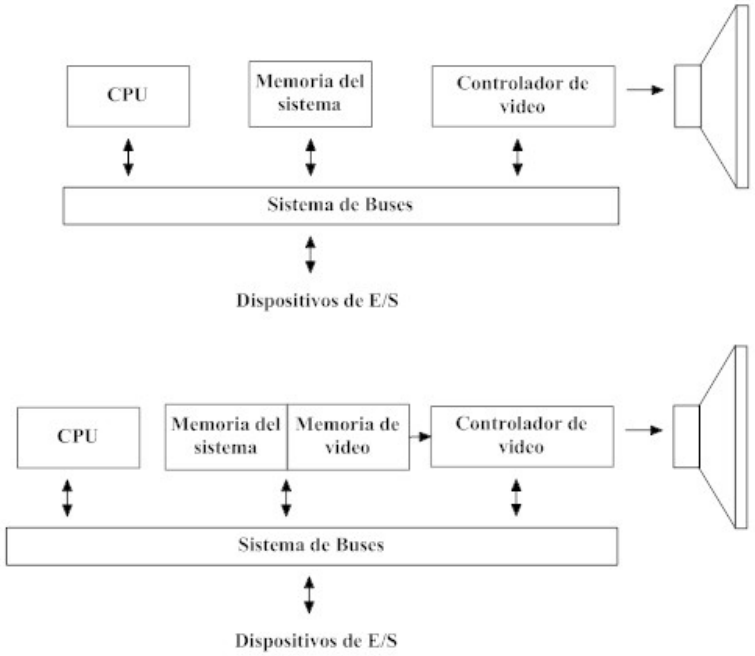


Figura 4.21: *Arquitecturas de un sistema de despliegue. Arriba, Arquitectura de un sistema básico de despliegue de gráficos. Abajo, Arquitectura de un sistema de despliegue mejorado con una porción de memoria del sistema reservada para video. Hearn[20, pág. 53].*

Hasta este punto de desarrollo, el sistema gráfico de realidad virtual del simulador de RTUP, logra un cierto grado de realismo, al incorporar modelos de órganos presentes durante una RTU y efectos visuales que de alguna forma permiten colocar al usuario en una experiencia real, objetivo del ambiente virtual. Otras características, no gráficas pero que tienen un peso sobre lo visual, comprenden la incorporación de efectos audibles y retroalimentación de fuerzas con algún dispositivo háptico. Objetivos que salen de este trabajo de tesis pero que son la pauta para una implementación futura.

[14]Para mayor información acerca de la implementación de cálculo de deformaciones y programación de propósito general con GPU, consulte Lira[34].

Capítulo 5

Carga de modelos tridimensionales virtuales en el simulador de RTUP

Hoy en día, la mayoría de los programadores de entornos virtuales tridimensionales, con el fin de ofrecer una mayor adaptabilidad al momento de visualizar en pantalla, hacen uso de los formatos de definición de propiedades de los elementos de una escena. Estos formatos estándar, tienen como principal ventaja la especificación de valores numéricos iniciales para modelos tridimensionales; tales como: información de posición de vértices, triángulos, tetraedros, propiedades de iluminación, propiedades de textura, condiciones de pertenencia, etcétera. Esta ventaja, permite que un programa de visualización posea una gran cantidad de modelos predefinidos. Así, durante una simulación, la carga de datos referentes a modelos 3D se realiza una sola vez y el resto de la ejecución se basa principalmente en la operación de éstos.

Existen una gran cantidad de formatos gráficos, todos ellos con particularidades específicas; sin embargo, hay características que la mayoría de ellos poseen. Estas características se relacionan principalmente con la geometría del objeto (definición de vértices, polígonos y/o volúmenes) y de propiedades visuales (color, textura, brillo, etc.). Algunos ejemplos de estos archivos son: *VRML*[1] (.wrl), *STL*[2] (.stl), *3D Studio* (.3ds), *DirectX* (.x), *MD2* (.md2), *MD3*[3] (.md3), *Wavefront* (.obj y .mtl), *INRIA*[4] (.node, .face y .ele), entre otros. La estructura básica de un archivo de configuración 3D se muestra en la figura 5.1.

Los elementos del archivo de definición se describen como:

- **Encabezado:** Contiene las propiedades numéricas globales del modelo, tales como: número de vértices, número de caras, número de coordenadas de textura, número de vectores normales, indicadores de tipo de coordenadas (cartesianas u homogéneas), modelo de color a emplear, etc. El objetivo principal de esta parte es facilitar la reserva de memoria al momento de cargar el modelo en el programa principal. Muchos formatos gráficos obvian esta parte, dejando a los programadores la creación de módulos identificadores de todas estas propiedades, tal es el caso del formato *OBJ*. Ejemplos de formatos que usan esta convención son: el formato NODE y FACE, del INRIA.
- **Definición de Vértices:** Contiene la lista de coordenadas de cada vértice del modelo 3D.

[1] *Virtual Reality Modeling Language*, Lenguaje para Modelado de Realidad Virtual.

[2] *Stereolithography*, estereolitografía. Modelado de prototipos 3D impresos en materiales físicos con volumen.

[3] Los formatos *MD2* y *MD3* fueron creados por la empresa *id Software* para el motor de Juegos *Quake*, en sus versiones II en adelante, en 1997.

[4] INRIA, Institut National de Recherche en Informatique et en Automatique, Instituto Nacional de Investigación en Informática y Automatización de Francia.

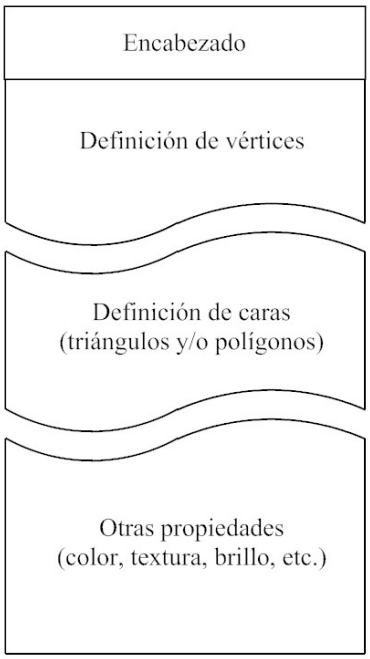


Figura 5.1: *Estructura básica de un archivo de definición 3D. Los archivos de definición juegan un papel muy importante en la ejecución de un sistema de visualización virtual, contienen los parámetros y propiedades iniciales de los modelos 3D.*

Generalmente se emplea la convención en coordenadas cartesianas (x,y,z). En algunos formatos, como el OBJ, los vértices pueden representar tres tipos: coordenadas de posición, normales y coordenadas de textura. Prácticamente todos los formatos gráficos 3D tienen este bloque.

- **Definición de caras:** Contiene la lista de caras, generalmente definidas con vértices referenciados mediante índices. Dependiendo de la geometría del objeto, la cantidad de vértices en cada cara define la forma del polígono. Así, por ejemplo, para tres vértices se tienen caras triangulares y de cuatro en adelante, caras poligonales. Prácticamente todos los formatos gráficos 3D poseen este bloque.
- **Otras propiedades:** En esta parte se definen propiedades de visualización del modelo 3D, tales como: color (dependiendo del modelo a usar, se definen componentes en RGB, RGBA, CMYK, HSV, entre otros)[5], textura (generalmente referencias a archivos externos de imagen), propiedades de brillo, etcétera. Algunos formatos incluyen estas propiedades dentro de la estructura de definición de caras, como el VRML. Otros los definen en archivos separados al principal, como es el caso del OBJ, que define un archivo de configuración de materiales MTL (.mtl).

[5]Componentes de color: RGB-(Rojo, Verde, Azul), RGBA-(Rojo, Verde, Azul, Transparencia-α), CMYK-(Cyan, Magenta, Amarillo, Negro), HSV-(Matiz, Saturación, Contraste)

5.1. Formatos de carga de modelos anatómicos en el simulador de RTUP

El simulador de RTUP, cuenta con un módulo programado para la carga de modelos anatómicos, usando los formatos gráficos 3D del INRIA (.face, .node y .ele). Estos formatos, empleados para inicializar la información básica (vértices, caras o tetraedros de una malla de superficie o de volumen) de los modelos anatómicos, se han ido adecuando a las necesidades del simulador. Así, los últimos formatos estables tienen la estructura mostrada en la figura 5.2.

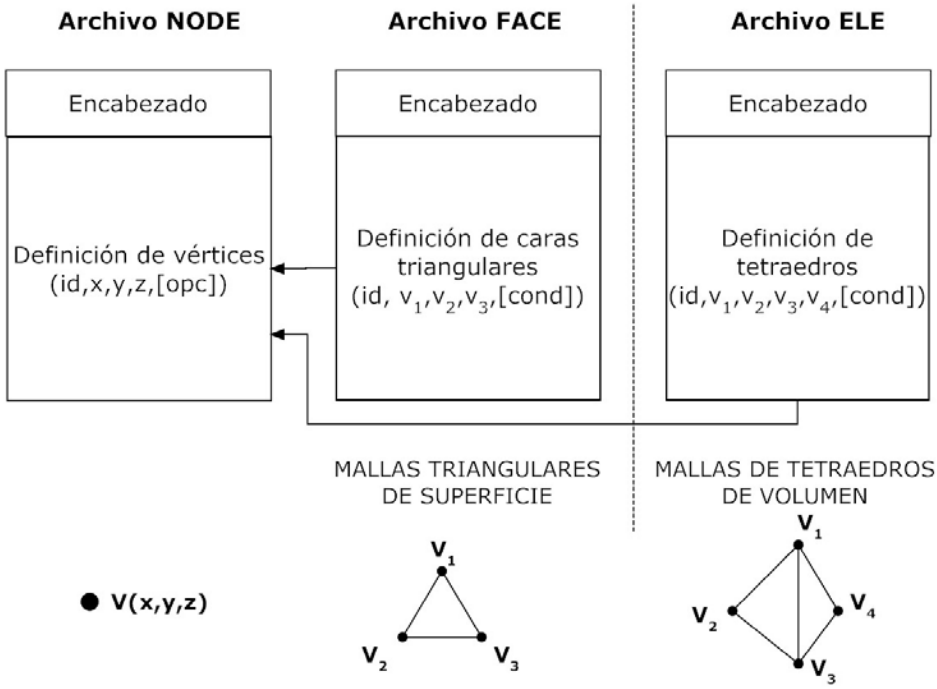


Figura 5.2: *Estructura de los formatos NODE, FACE y ELE. Formatos empleados por el simulador de RTUP para la carga de modelos anatómicos de superficie (NODE y FACE) y de volumen (NODE y ELE). Las opciones ([opc]) y condiciones ([cond]), se emplean para definir regiones de frontera, puntos de colisión, entre otras. Extracto, [40, PADILLA, págs. 1-3]*

Para fines de simulación, el uso de los formatos NODE, FACE y ELE, tienen una gran ventaja sobre otros formatos, principalmente en la definición de volúmenes con la combinación NODE y ELE, ya que la mayoría de los otros se enfocan en la definición de mallas de superficie mediante triángulos y/o polígonos. En el caso específico del simulador, su uso se empleó para cargar el modelo de la próstata virtual como una malla de volumen formada por tetraedros, usados para aplicar el concepto de modelos deformables con el método de masas y resortes o elemento finito (FEM) (Ver figura 5.3).

A pesar de la gran ventaja que ofrecen los formatos anteriores, existen características no definidas y que en tiempo de ejecución restan tiempo de procesamiento, sobre todo para fines de visualización gráfica. Un ejemplo de estas características, es el cálculo de las normales de las caras del modelo deformable, que tienen que calcularse en tiempo real y constantemente están siendo actualizadas. Lo anterior es benéfico cuando se trata de la simulación de tejidos que cambian constantemente de forma debido a la interacción con herramientas quirúrgicas virtuales y que por efectos de iluminación obligan

a su recálculo, como el modelo de la próstata virtual; sin embargo, para modelos de superficie estáticos, donde las normales no cambian, es un costo innecesario su cálculo. En estos casos surge la necesidad de adaptar un nuevo formato gráfico.

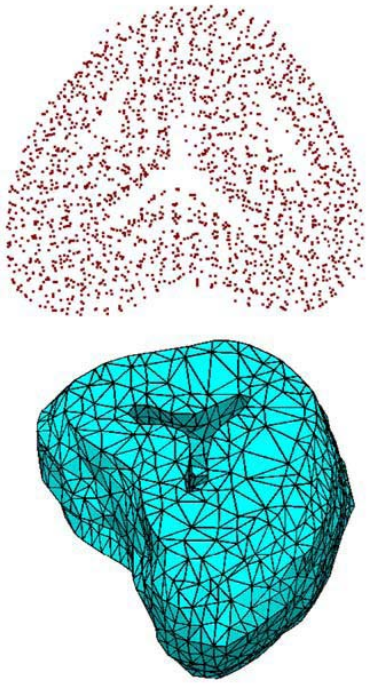


Figura 5.3: *Modelo de la próstata del simulador RTUP. Los formatos gráficos 3D para el volumen(NODE y ELE), se generan a partir de un algoritmo de tetraedrización descrito por Soriano[56, Págs. 31-41]. Arriba, nube de puntos que definen los vértices. Abajo, conjunto de tetraedros que definen el volumen.*

El formato OBJ, desarrollado por la empresa Wavefront Technologies a finales de los 80's[37], ofrece algunas ventajas para fines de visualización. A diferencia de los formatos NODE y ELE, que ofrecen ventajas en la definición de volúmenes, el formato OBJ sólo especifica propiedades de superficies. Sin embargo, al comparar los formatos NODE y FACE con respecto al OBJ, éste último ofrece una mayor ventaja, principalmente por tres razones: la definición inicial de vectores normales, vértices de textura y componentes de color (para fines de iluminación); lo cual resta tiempo de cómputo, al no calcularse en tiempo real. La figura 5.4, define la estructura general del formato OBJ y su archivo asociado MTL, para definición de materiales.

Existen otras adaptaciones del formato OBJ, que definen características más avanzadas, al combinar definiciones de objetos poligonales (puntos, líneas y caras) y objetos de geometría libre (curvas y superficies trascendentes), que para fines de esta tesis sobrepasan lo necesario. En esta tesis se eligió el formato OBJ por las ventajas antes mencionadas y con fines de visualización de dos estructuras anatómicas principales: la vejiga y la uretra; modelos virtuales generados a partir del algoritmo de reconstrucción descrito en el capítulo 3 (Ver figura 5.5). Además se realizaron las adaptaciones necesarias para que el simulador de RTUP cargara el nuevo formato gráfico, en conjunto con el ya implementado, el cargador de archivos NODE, FACE y ELE.

La estructura del formato OBJ, adaptada al simulador de RTUP se realizó mediante la programación de un módulo en lenguaje C de carga de modelos OBJ. Este módulo, en conjunto con las

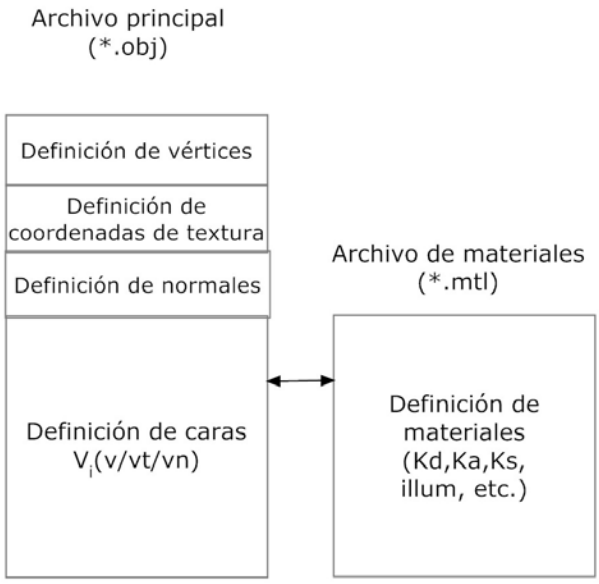


Figura 5.4: *Estructura del formato OBJ y su archivo asociado MTL para definición de materiales. El simulador de RTUP emplea esta estructura para la carga de modelos de superficie estáticos, actualmente los modelos virtuales de la uretra y de la vejiga urinaria.*

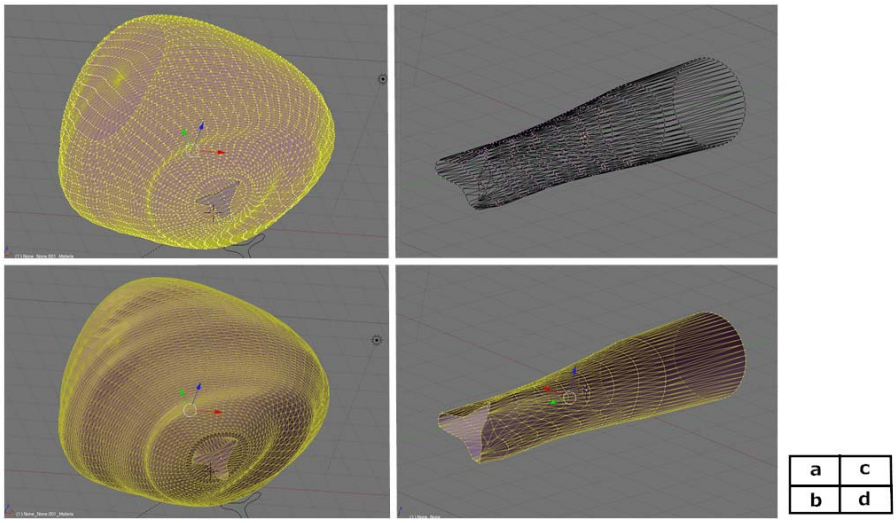


Figura 5.5: *Modelos virtuales de la uretra y la vejiga urinaria usando el formato OBJ (visualizados en Blender[4]). a) Vértices del modelo de la vejiga, b) caras del modelo de la vejiga, c) vértices del modelo de la uretra y d) caras del modelo de la uretra.*

funciones de carga de modelos de volumen, forman la librería de definición de mallas del simulador: *mesh*. Antes de pasar a la descripción de la programación del cargador, es importante abordar de los elementos que nos proporcionan información de un archivo de definición OBJ.

DIRECTIVA	PARÁMETRO	SIGNIFICADO
FORMATO OBJ		
v	x y z w	Define un vértice en coordenadas cartesianas (x,y,z). El parámetro w es opcional y se usa generalmente con la definición de curvas. Si no se usa, el valor por default es 1.0.
vp	u v w	Define un punto en un espacio paramétrico.
vn	i j k	Define un vector normal con componentes i,j y k.
vt	u v w	Define un vértice de coordenadas de textura.
cstype	tipo	Define un tipo de curva o superficie e indica si es una forma racional o no racional.
p	v_1 v_2 v_3 ...	Define un elemento punto y sus vértices.
l	v_1/vt_1 v_2/vt_2 v_3/vt_3 ...	Define una línea y el número de referencia a sus vértices y vértices de textura.
f	$v_1/vt_1/vn_1$ $v_2/vt_2/vn_2$ $v_3/vt_3/vn_3$ ...	Define una cara y sus correspondientes referencias a vértices coordenados, vértices de textura y vértices normales.
curv	u_0 u_1 v_1 v_2 ...	Define una curva, su rango de parámetros y sus puntos de control (como vértices).
curv2	vp_1 vp_2 vp_3 ...	Define una curva con sus vértices paramétricos.
surf	s_0 s_1 t_0 t_1 $v_1/vt_1/vn_1$ $v_2/vt_2/vn_2$ ...	Define una superficie, sus parámetros de control y sus vértices de control (vértices, vértices de textura y vértices normales).
g	$nomG_1$ $nomG_2$ ...	Especifica el nombre para los elementos que se definen después de esta sentencia.
mtllib	Nombre del archivo	Especifica el nombre del archivo *.mtl que contiene características de los materiales
usemtl	Nombre del material	Especifica el nombre del material que se aplicará a los objetos definidos a continuación de esta sentencia.
#	Comentario	Especifica un comentario
FORMATO MTL		
newmtl	Nombre	Define el nombre identificador del material.
Ns	s	Define el brillo del material.
Ka	r g b	Define la componente RGB ambiental de un material.
Kd	r g b	Define la componente RGB difusa de un material.
Ks	r g b	Define la componente RGB especular de un material.
d	*alpha*	Define la componente *alpha* de transparencia del material.
illum	n	Define el modelo de iluminación empleado. illum=1 indica un material plano e illum=2 indica la presencia de fuentes especulares.
map_Kd	Nombre del archivo	Nombre del archivo de textura (en el caso del simulador con formato de textura *.bmp).
#	Comentario	Especifica un comentario

Tabla 5.1: *Identificadores de los formatos de definición 3D, OBJ y MTL. Extracto Wotsit.org [64], FileFormat.info[10].*

Las declaraciones más comunes de un archivo de definición OBJ se muestran en la tabla 5.1. Para los propósitos de esta tesis, únicamente se emplean las directivas de: vértices coordenados (v),

vértices de textura (vt), vectores normales (vn), caras (f) y material (mtllib y usemtl). Para el formato de definición de materiales se emplean principalmente: componentes ambiental (Ka), difusa (Kd) y especular (Ks); brillo (Ns) e imagen de textura (map_Kd).

El cargador básicamente consiste en dos partes: un analizador sintáctico o *parser* y un bloque de visualización con OpenGL. El analizador sintáctico tiene como entrada un archivo de texto plano OBJ, que después convierte en datos operables (vértices, coordenadas de textura, normales, materiales, etc.) y los guarda en estructuras predefinidas. Las estructuras de almacenamiento son listas ligadas controladas por apuntadores. A continuación, el bloque de visualización con OpenGL toma los apuntadores principales a las listas y reconstruye el modelo gráfico tridimensional en pantalla.

De acuerdo a lo anterior, la carga de modelos estáticos OBJ en el simulador de RTUP sigue el siguiente algoritmo básico de carga y dibujado:

Algoritmo básico de carga y dibujado de modelos OBJ

1. *Construir superficie*
   - a) *Crear lista de vértices*
     1) *Obtener número de vértices por escaneo de v´s*
     2) *Guardar cada vértice en la lista de vértices*
   - b) *Crear lista de coordenadas de textura*
     1) *Obtener número de coordenadas de textura por escaneo de vt´s*
     2) *Guardar cada coordenada de textura en la lista de coordenadas de textura*
   - c) *Crear lista de normales*
     1) *Obtener número de vectores normales*
     2) *Guardar cada vector normal en la lista de normales*
   - d) *Crear lista de triángulos*
     1) *Obtener número de caras triangulares por escaneo de f´s*
     2) *Obtener número de materiales por escaneo de usemtl´s*
     3) *Si el número de materiales es mayor a cero*
        Obtener lista de materiales del archivo MTL
     4) *Guardar referencias a los vértices que definen cada triángulo en la lista de triángulos.(En caso de la existencia de materiales, guardar propiedades del material)*
2. *Dibujar modelo con OpenGL*
   - a) *Recorrer la lista de triángulos y dibujar*
     1) *Obtener triángulo actual*
        - a′ *Obtener coordenadas de los tres vértices del triángulo*
        - b′ *Obtener valores normales por vértice*
        - c′ *Obtener valores de coordenadas textura por vértice*
        - d′ *Definir propiedades de materiales (en caso de haberlas)*
        - e′ *Dibujar triángulo*
     2) *Continuar con el siguiente triángulo*

La primera parte (construcción de la superficie), se ejecuta sólo una vez al inicio del programa, durante la carga de parámetros iniciales del simulador. La segunda (dibujado y visualización del modelo), se ejecuta cada vez que se manda a redibujar con OpenGL. En el caso del simulador de RTUP, el proceso anterior se lleva a cabo para cargar y dibujar: la máscara circular del resectoscopio, el modelo virtual de la vejiga urinaria y el modelo virtual de la uretra.

Capítulo 6

Resultados

Los principales resultados obtenidos a raíz de este trabajo de tesis son:

Una descripción completa del estado del arte actual del simulador: El capítulo 2 (*"El Simulador de Resección Transuretral de Próstata"*) representa un compendio de los últimos avances del sistema de realidad virtual desarrollado por el grupo de análisis de imágenes y visualización del CCADET. La idea principal fue describir al simulador como un sistema completo de realidad virtual, con base en los elementos descritos en el capítulo introductorio, en el cual se distinguen tres elementos básicos: una estación de interacción de fuerzas, una estación de trabajo y una estación gráfica.

Así, el sistema completo ha sido probado en estaciones con las siguientes características.

- Características de software:
  - Sistema operativo: Windows XP-SP2/Vista
  - Librería de gráficos: OpenGL 2.1 o superior
  - Para fines de compilación: Dev-C++ 4.9.9.2
- Características de Hardware en condiciones óptimas, modo GPU activado:
  - Estación gráfica: PC Escritorio
  - Procesador: Intel Core Duo 2.4 GHz/2.39GHz
  - Memoria RAM: 2 GB
  - Tarjeta gráfica: nVIDIA GeForce 7900, NVIDIA Quadro FX 550, NVIDIA Quadro FX 1000
- Características de Hardware en condiciones óptimas, modo GPU desactivado:
  - Estación gráfica: PC Escritorio
  - Procesador: Pentium 4 3.0 GHz
  - Memoria RAM: 1 GB
  - Tarjeta gráfica: No necesaria.

El desarrollo e implementación de una metodología genérica de reconstrucción de mallas de superficie a partir de isosuperficies. Esta parte, descrita en el capítulo 3, dio como resultado una metodología de tres pasos: la extracción de contornos a partir de imágenes anatómicas del VHP y segmentación manual, el muestreo de los contornos para definir vértices (incluyendo la proposición

de un algoritmo de muestreo) y la reconstrucción de la malla mediante la aplicación de ecuaciones de triangulación a partir de nubes de puntos ordenados. La metodología se aplicó para la reconstrucción de la vejiga virtual del simulador.

Algunos detalles del proceso de reconstrucción de la vejiga virtual son:

- Se tomaron como referencias las imágenes anatómicas axiales del VHP, a intervalos de 1mm con resolución de 4096x2700 píxeles y una profundidad de color de 24 bits en escala RGB.
- El rango de imágenes que contienen la vejiga urinaria va de la imagen 1863 a la 1903. De las cuales sólo se usaron de la 1964 a la 1902, dado que proporcionaron la mayor cantidad de información. La imagen 1964 fue descartada por estar mal calibrada con respecto a las otras imágenes. En total se usaron 39 imágenes para aplicar la técnica de extracción de contorno.
- Se aplicó interpolación morfológica para obtener un conjunto más grande de contornos, teniéndose como resultado un conjunto de 77 contornos.
- Antes de aplicar el algoritmo de muestreo, de los 77 contornos, se descartó la última por no cumplir con la condición de muestreo.
- Se aplicó el algoritmo de muestreo sobre los 76 contornos (isosuperficies), obteniéndose 50 puntos por contorno y sus correspondientes posiciones en coordenadas (x, y).
- Se aplicaron las ecuaciones de reconstrucción sobre la nube de puntos muestreados y se generó la malla triangular.

Dado que la resolución de la malla, en cuanto a número de triángulos, depende directamente del número de puntos muestreados por contorno, es posible predecir el número de triángulos para diferentes valores de muestreo. La tabla 6.1 muestra las posibles resoluciones de malla y la figura 6.1, una gráfica de las curvas de resolución.

CONTORNOS	MUESTRAS POR CONTORNO	VÉRTICES	CARAS
76	200	15200	30000
76	150	11400	22500
76	100	7600	15000
76	50	3800	7500
76	25	1900	3750
76	20	1520	3000
76	10	760	1500
76	5	380	750

Tabla 6.1: *Resoluciones de mallas variando el número de puntos por contorno para la reconstrucción de una vejiga virtual.*

La proposición de una metodología de creación gráfica y la descripción de los elementos gráficos que componen una escena tridimensional. La metodología de creación gráfica del simulador de RTUP quedó finalmente definida en seis pasos: 1) Investigación, 2) Planeación, 3) Modelado, 4) Simulación, 5) Generación de la escena y 6) Despliegue.

En la descripción de los elementos de la escena, se definieron tres actores: la cámara virtual, los modelos anatómicos y el modelo de iluminación. Los modelos anatómicos en la última versión estable

del simulador tienen las características mostradas en la tabla 6.2[1]. Con estos datos, el modelo de mayor resolución es el de la vejiga, lo cual indica que debe ser sustituida en un futuro por otro modelo a menor resolución. Una alternativa podría ser el modelo de 10 muestras que teóricamente tendría 760 puntos y 1500 caras, lo que representaría una reducción de 6000 triángulos.

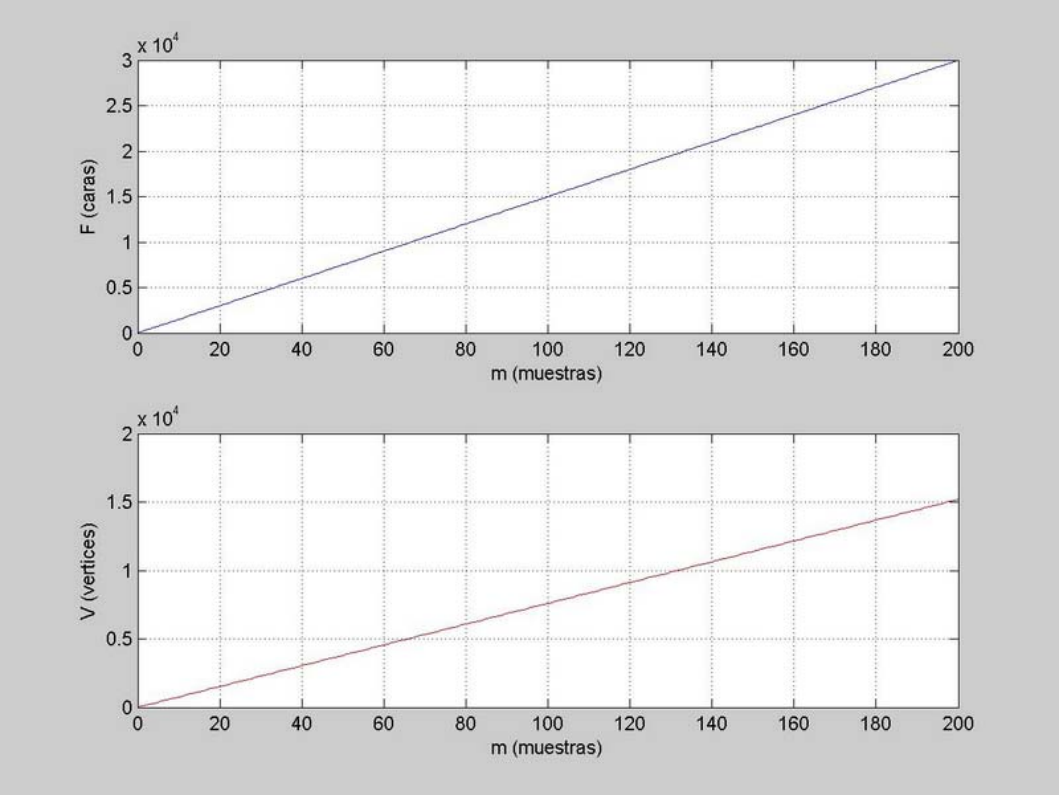


Figura 6.1: *Curvas de resolución del modelo de vejiga urinaria con S=76 contornos y diferentes valores de muestreo m. Arriba, curva de caras. Abajo, curva de vértices.*

Modelo	Tipo de Malla	Vértices	Caras / tetraedros	Textura
Conducto uretral	Superficie	455	820 caras	2D
Próstata	Volumen	1775	7564 tetraedros	3D
Vejiga*	Superficie	3801	7550 caras	2D
Herramienta-Asa	Superficie	662	936 caras	Materiales
Máscara circular	Superficie	96	96 caras	Materiales

Tabla 6.2: *Características generales de los modelos anatómicos del simulador de RTUP.*

En cuanto a visualización, se lograron aplicar los efectos de iluminación sobre los modelos,

[1]*El número de vértices y caras triangulares en el caso del modelo de la vejiga, difieren con respecto a los datos mostrados en la tabla 6.1, debido a que el modelo fue sometido a un proceso de adaptación en las secciones del cuello y vértice vesical. El vértice agregado corresponde al vértice vesical y las 50 caras agregadas, corresponden a las caras formadas al unir el vértice con los puntos del último contorno de la superficie.

suficientes para recrear la exploración endoscópica. La figura 6.2 muestra la visualización final de los modelos dentro de la escena del ambiente gráfico virtual.

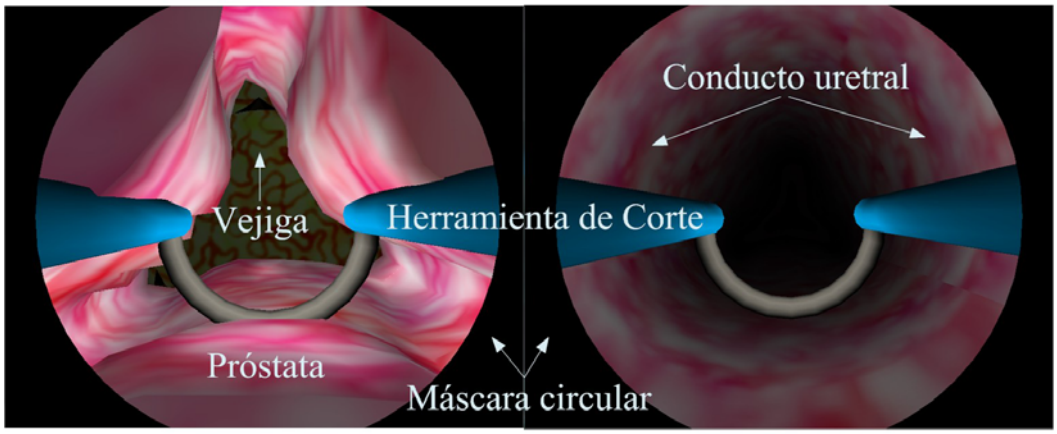


Figura 6.2: *Vistas endoscópicas de los modelos que intervienen durante el proceso de simulación.*

La incorporación de efectos audiovisuales a la estación gráfica de realidad virtual del simulador de RTUP. Los efectos visuales implementados sobre el simulador fueron:

- Efecto de distorsión de la lente. Se aplicó un algoritmo de distorsión basado en la deformación de los texeles de la imagen después de render final. Sin embargo, se determinó que para fines de simulación en tiempo real no era viable el uso de este método, aunque se piensa que si se programa el método en una GPU, podría funcionar en tiempo real.
- Efecto de máscara circular: Para lograr el efecto de la vista endoscópica, se agregó una máscara circular que emula el efecto causado por el endoscopio durante una cirugía real.
- Efecto de profundidad limitada con aplicación de niebla: Este efecto se aplicó para restringir al usuario de la vista profunda a cierta distancia, lo cual agregó un cierto grado de realismo, sobretodo en la entrada por el conducto uretral.
- Efecto de obstrucción parcial de la visibilidad con el uso de niebla: Este efecto se aplicó cambiando el color de la niebla obscura a roja cuando al usuario rebasa ciertos límites que encierran los modelos anatómicos. El efecto puede ser una herramienta potencial para la futura implementación de sangrado durante el corte de tejido.

La figura 6.3 muestra los efectos aplicados sobre el sistema gráfico de realidad virtual.

Finalmente, la figura 6.4 muestra una comparativa entre versiones del entorno gráfico virtual del simulador, se distinguen los estados inicial (como se recibió el simulador) y el estado actual (después de aplicar todas las modificaciones descritas en este trabajo de tesis).

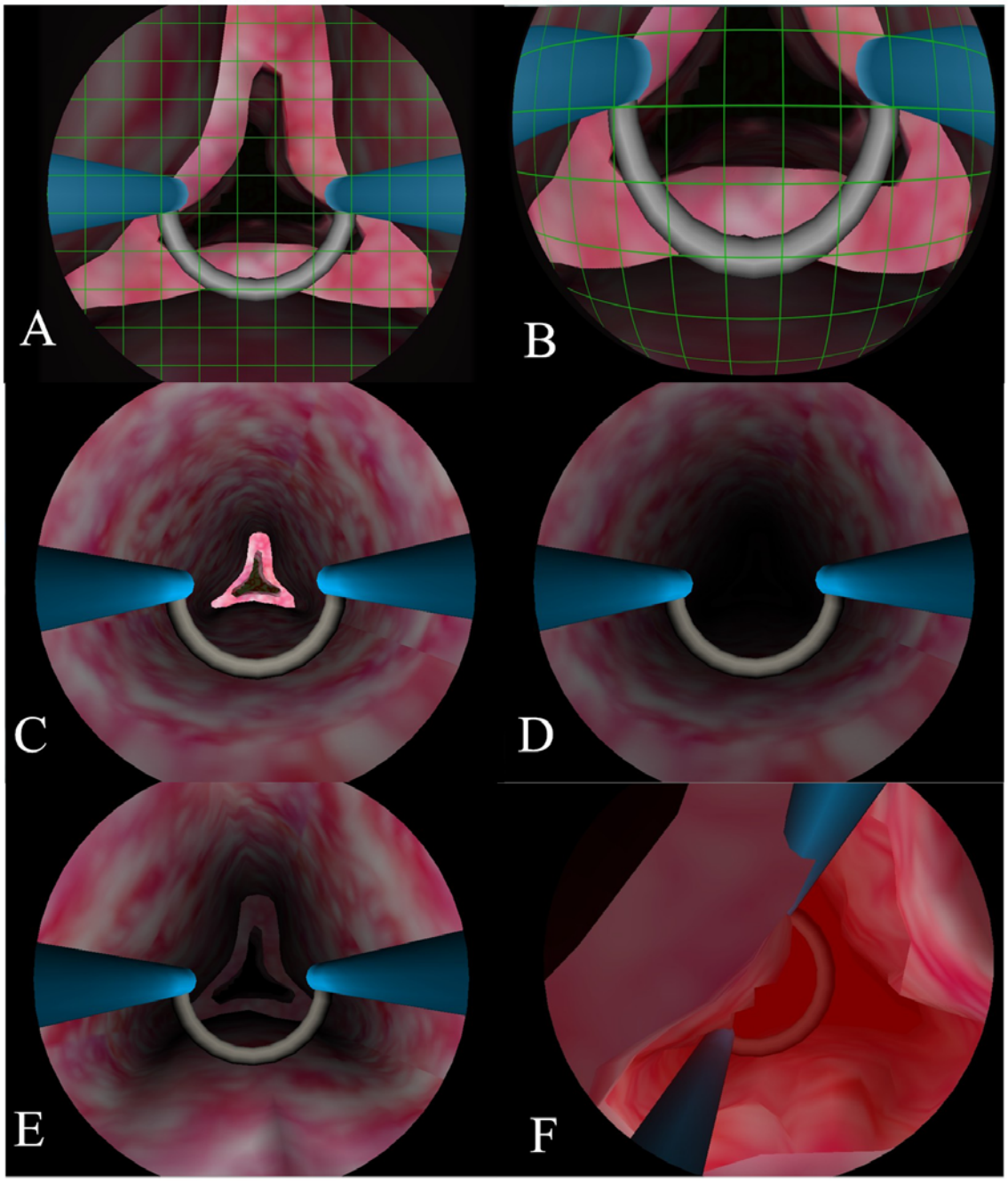


Figura 6.3: *Efectos visuales aplicados sobre el simulador de RTUP. A) Escena sin distorsión de la lente, B) escena con distorsión, C) escena sin efecto de niebla, D) escena con efecto de niebla, E) efecto de máscara circular y F) efecto de obstrucción de la visibilidad por sangrado.*

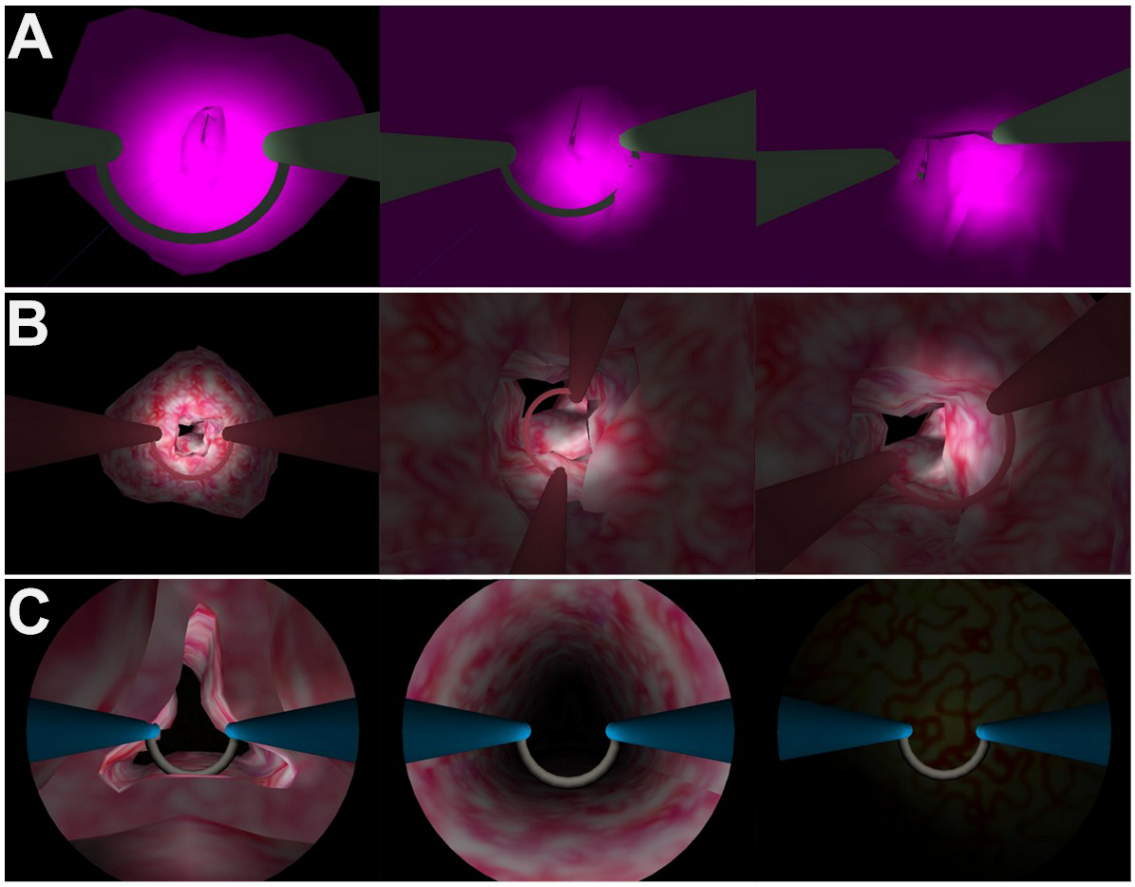


Figura 6.4: *Comparativa entre versiones del simulador de RTUP. A) Primera versión estable, únicamente con modelo de la próstata deformable y resectoscopio virtual. B) Segunda versión (inicial para este trabajo de tesis), con modelo de la próstata deformable texturizada y resectoscopio virtual. C) Versión actual, con modelos de próstata, vejiga y uretra, además de todos los efectos visuales antes descritos.*

Capítulo 7

Conclusiones

- La investigación, desarrollo e implementación de ambientes virtuales aplicados a las ciencias médicas tiene un triple impacto. 1) En el proceso enseñanza-aprendizaje, los sistemas de simulación permiten al alumno la adquisición de habilidades clínicas, al mismo tiempo que refuerzan sus bases teóricas; para ello, algunos simuladores de entrenamiento poseen módulos de evaluación estadística que permiten cuantificar el grado de aprendizaje del alumno. 2) En el proceso de planeación, los sistemas de realidad virtual ayudan al médico a realizar simulaciones en tiempo real de los posibles efectos en la toma de decisiones, que generalmente se llevan a cabo al instante de la cirugía, por lo tanto, durante el procedimiento real se asegura el menor grado de lesión y efectos secundarios sobre el paciente dado que el médico realizó simulaciones previas. 3) En cuanto a la implicación del uso de ambientes virtuales durante y después de una cirugía, se encontró que las aplicaciones son muy limitadas pero cada vez van en aumento. Claramente estas aplicaciones son muy específicas y generalmente se trata de terapias de rehabilitación y cirugías *in-vivo* asistidas por computadora mediante robots.

- En el proceso de planeación de cualquier sistema de realidad virtual y su extensión a un ambiente virtual, el primer paso a dar es la investigación exploratoria de todos los elementos del mundo real que intervienen en el proceso, otorgando una base firme a las razones de su implementación y el impacto sobre el área de aplicación. En el caso del simulador de RTUP, el capítulo 2 representa el marco teórico y metodológico para llevar a la implementación un sistema de realidad virtual completo, resultado del trabajo conjunto entre miembros del grupo de Análisis de Imágenes y Visualización del CCADET y el Instituto Nacional de Rehabilitación. A este respecto, se concluye que la primera versión del simulador puede catalogarse como un sistema de realidad virtual completo, al incorporar los tres elementos básicos: interfaz mecatrónica, estación de trabajo y estación gráfica. Las perspectivas a futuro contemplan: la incorporación de la última versión (con las modificaciones a la parte visual realizadas en este trabajo de tesis), con un sistema háptico de retroalimentación de fuerzas y la implementación del sistema completo en dos modalidades: como estación de entrenamiento individual y como ambiente de enseñanza colectivo en el observatorio de visualización *IXTLI*.

- Este trabajo partió de un sistema de realidad virtual que únicamente contemplaba la visualización de la próstata, totalmente funcional con una interfaz mecatrónica. Sin embargo, para otorgar mayor realismo al sistema, se hizo necesaria la incorporación de otros dos elementos virtuales: la vejiga urinaria y el conducto uretral. El resultado de esta necesidad, fue la creación de una metodología genérica para reconstrucción de mallas de superficie a partir de isosuperficies. El método propuesto, resultó ser muy útil para la reconstrucción de la vejiga virtual dada su forma anatómica y gracias a la información proporcionada por el conjunto de imágenes del VHP. En el caso de la uretra, se concluye que es necesaria la investigación e implementación de otra metodología de reconstrucción tridimensional, más automatizada y en la cual se requiera de la

mínima participación del modelador, tal como sucedió en el modelo de la vejiga.

- El método de reconstrucción de mallas de superficie a través de isosuperficies, resultó ser una buena alternativa con respecto a otros métodos de reconstrucción, como es el caso del algoritmo de *Marching Cubes*, en cuanto a tiempo de procesamiento y la facilidad de implementación. Sin embargo, es muy importante mencionar que no abarca tantas aplicaciones como el algoritmo de *Marching Cubes* y que su efectividad se mide con respecto a la información disponible en las isosuperficies.

- El planteamiento de una metodología de creación gráfica sobre el simulador, dio como resultado la definición del *Proceso de creación gráfica del simulador de RTUP*, el cual contempla 6 pasos: 1) investigación, 2) planeación, 3) modelado, 4) simulación, 5) generación de la escena y 6) despliegue. En virtud del tiempo invertido en cada paso, los primeros tres implicaron la mayor cantidad de trabajo humano. La adaptación de los resultados del modelado al sistema de simulación ya disponible, requirió un menor tiempo de trabajo, debido a la experiencia con la programación en lenguaje C y los antecedentes con el manejo de la librería gráfica OpenGL.

- En cuanto a la definición de los elementos de una escena tridimensional, se concluye que, los elementos gráficos descritos en el capítulo 4 son suficientes para dotar de un mayor grado de realismo al simulador. Los tópicos más sobresalientes en la implementación del ambiente gráfico virtual fueron: el texturizado, el modelo de iluminación y los efectos visuales. De los efectos visuales, quedan por implementar los efectos de distorsión de la lente, que hasta el momento no otorgan resultados satisfactorios, y en un futuro también se espera incorporar algún método de simulación para el efecto de sangrado, una vez teniendo a disposición la metodología de corte de tejido.

- Sobre los modelos de vejiga y uretra, adaptados al sistema de simulación mediante la programación de un módulo de carga de modelos con formato OBJ, se concluye que son suficientes para fines de visualización, sin embargo, dado que el propósito incluye la simulación en tiempo real y con el mayor realismo posible, se hará necesaria la incorporación de propiedades físicas, a fin de que se conviertan en modelos deformables; tal como sucede con el modelo de la próstata.

- Dada la última revisión de la parte médica, el Dr. Sergio Durán comentó que es necesaria la incorporación de varias características más a los modelos, como lo son la visualización del esfínter en el conducto uretral, la visualización del veru-montanum y una mejor visualización de la vejiga urinaria. Con la incorporación de estas últimas características y la incorporación de la interfaz háptica, el simulador estaría listo para su validación médica, la cual provee la participación del experto y después su puesta en marcha como un sistema de entrenamiento clínico, objetivo principal del proyecto.

- En general, los trabajos realizados sobre la parte gráfica del simulador, representaron una aportación muy importante en términos de resultados. Las imágenes comparativas entre el estado anterior y el estado actual, mostradas en el capítulo de resultados comprueban el avance. Sin embargo, aún queda mucho trabajo por realizar y de lograrse, representaría un gran adelanto para el desarrollo de simuladores computarizados en México.

Apéndice A

Modelado Básico con Blender

Blender [4] es un conjunto de herramientas de código abierto, distribuido bajo la Licencia GNU GPL (GNU-General Public License)[17], para la creación y edición de contenido 3D. En su última versión, la 2.46, Blender está disponible para los sistemas operativos: Windows (98/ME/2000/XP/Vista), Linux (x86-32/x86-64), Mac OS X, Irix 6.5, Solaris 6.8 y FreeBSD 4.2.

Este software nació como una idea original de Ton Roosendaal en Holanda a finales de los 80's, como un proyecto de desarrollo de software 3D en sus estudios de animación NeoGeo. Sin embargo, no fue hasta 1995 que el proyecto de la herramienta tomó fuerza y en 2002 fue lanzada al mundo, bajo los términos de la licencia GPL.

Como programa de desarrollo de contenido 3D, a primera vista, tiene una interfaz poco intuitiva, muy diferente a la gran cantidad de paquetes de edición. La forma de trabajar que este software ofrece es a través del teclado (mediante combinaciones de teclas) y el ratón.

En lo consecuente, se ofrece una guía básica para la edición de modelos 3D, partiendo del entorno de Blender, pasando por la carga de modelos anatómicos reconstruidos a partir de la técnica de contornos descrita en esta tesis y finalizamos con la exportación de modelos OBJ, listos para su carga en el simulador de RTUP.

A.1. El entorno básico de Blender

Básicamente, Blender se basa en un sistema de ventanas; cada una de las cuales tiene funciones y propósitos distintos. Los botones y menús (que son una de las principales facilidades de este software), proporcionan a los modelos y objetos de la escena características como: su posición, su tamaño, rotaciones, traslaciones, propiedades de color, textura, propiedades físicas, etcétera. A continuación se muestra el entorno de trabajo básico de Blender (Ver figura A.1) y sus correspondientes funciones.

a) Selección de escena: Consiste en un menú despegable en el cual se puede acceder al conjunto de escenas creadas. Una escena se entiende como el conjunto de objetos en el espacio de tres dimensiones, las luces, las cámaras y las interacciones entre éstos.

b) Selección de modo: Blender dispone de diversos modos para la edición de los objetos involucrados en la escena. Así tenemos diferentes tipos de configuración del entorno, como son: 1- Animación, 2- Modelo, 3-Material, 4-Secuencia y 5- Scripting.

c) Preferencias de usuario: En esta barra podemos localizar el conjunto de menús clásico de cualquier programa de edición. Podemos realizar la apertura, cierre, importado y exportar archivos, de modelos 3D y escenas completas realizadas en Blender u otros programas de edición gráfica.

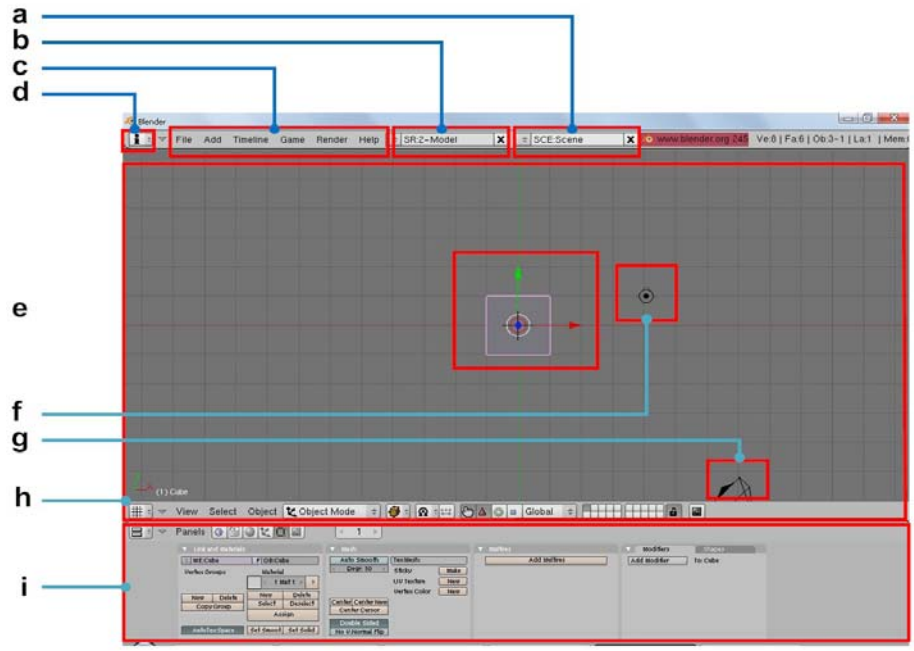


Figura A.1: *Entorno básico de Blender. a)Selección de Escena, b)Selección de modo, c)Preferencias del usuario, d)Selección de tipo de ventana, e)Objeto 3D en vista superior, f)Iluminación por defecto, g)Cámara, h)Ventana de edición, i)Ventana de herramientas.*

d) Selección del tipo de ventana: Blender maneja un sistema de ventanas muy peculiar; ya que cada una dispone de un menú de selección de tipo. Los tipos disponibles son:

- 3D View - Vista de escena 3D.
- Ipo Curve Editor - Edición de curvas de interpolación de animación.
- Action Editor - Editor de acciones de animación.
- NLA Editor - Editor de acciones y curvas de interpolación mezcladas.
- UV/Image Editor - Editor de coordenadas de textura.
- Video Sequence Editor - Editor de secuencias de video.
- Timeline - Línea del tiempo de la escena.
- Audio Window - Reproductor y editor de efectos de audio.
- Text Editor- Editor simple de texto para documentación.
- User preferences- Preferencias estándar del usuario.
- Outliner - Árbol de jerarquías entre objetos de la escena.
- Buttons window - Ventana de paneles de edición.
- Node editor - Editor de nodos de mallas.

- Image Browser - Buscador y previsualizador de imágenes.
- File Browser - Buscador de archivos gráficos.
- Scripts Window - Ventana de scripts (programación con Phyton).

e) Objeto: Área de trabajo con los objetos que conforman la escena en la ventana *3D View*.

f) Iluminación: El sistema de luces en Blender es muy sencillo de implementar y solo basta con agregar un objeto nuevo de tipo "light" con los parámetros definidos en los paneles de propiedades.

g) Cámara: Blender por default coloca una cámara en la vista 3D. Es posible agregar un conjunto de cámaras en proyectos muy grandes, en donde cada cámara capta una perspectiva de la escena.

h) Ventana de edición (3D View): Es la ventana más importante de Blender, ya que es una vista "preeliminar" del modelo final. Contiene los objetos, las luces y las cámaras, con sus correspondientes propiedades, dadas por la ventana de herramientas.

i) Ventana de herramientas: Contiene los paneles de botones que permiten la configuración de todos los parámetros de la escena 3D, incluidas propiedades de objetos, luces y cámaras; así como de la escena global completa.

A.2. Herramientas de edición de mallas

Las herramientas de edición que ofrece Blender son muy útiles para manipular mallas anatómicas y en general para aplicaciones donde se requiera el uso de modelos 3D. Para este propósito, se disponen de tres herramientas básicas: la Vista 3D (3D View), el panel de herramientas (Buttons window) y el editor de coordenadas de textura (Ver figura A.2).

A.2.1. La vista 3D (3D View)

La Vista 3D ofrece una gran cantidad de opciones para la edición de mallas (Ver figura A.3). Primero, porque podemos transformar los elementos de la escena libremente y ver el efecto causado en tiempo real; y segundo, porque cuenta con varios módulos que nos permiten visualizar los modelos en diferentes modos (edición, objeto, peso, textura, malla de alambre, bounding box, etc.).

Para una mejor visualización, el entorno ofrece cuatro vistas distintas: vista superior, vista de lado, vista frontal y vista definida por el usuario. Esta última se logra presionando el botón central del ratón (sin soltar) y moviéndose en la dirección de visualización deseada, lo cual permite observar detalles que con las otras vistas a veces es imposible distinguir. También se ofrecen dos modos de proyección de la escena: en perspectiva y ortográfica.

A.2.2. Los paneles de herramientas (Buttons window)

Los paneles de herramientas (Ver figura A.4) contienen todas las propiedades y métodos aplicables a una malla 3D y en general, a los objetos de la escena. Por ejemplo, para una malla se le pueden asignar propiedades de color, textura, sombreado, material, efectos de iluminación; o métodos como suavizado, cálculo de coordenadas de textura, propiedades de visualización, dibujo de caras, aristas, calcular normales, etc.

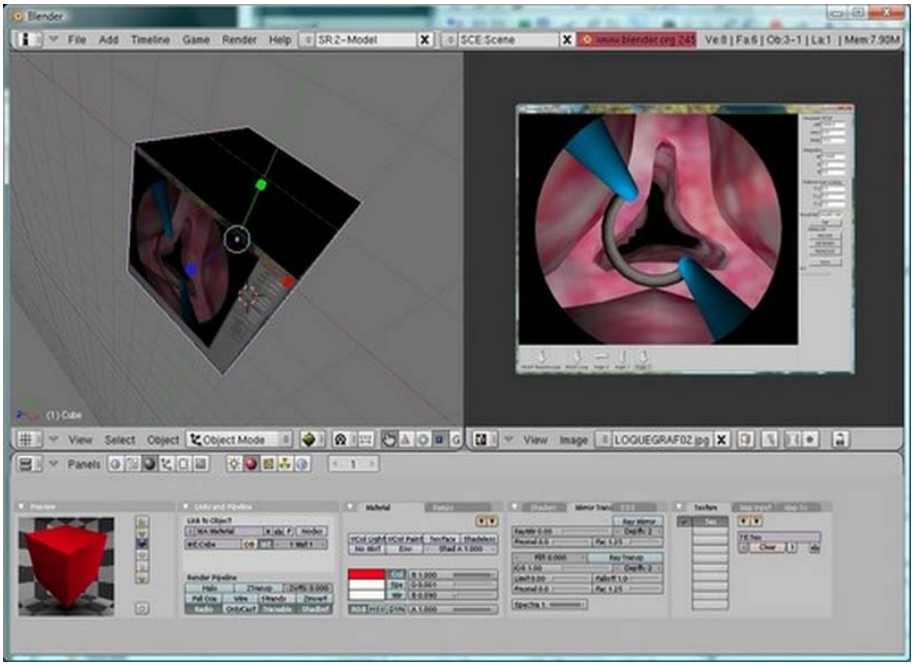


Figura A.2: *Entorno de Blender con las tres ventanas de edición: vista 3D, los paneles de herramientas y la ventana de coordenadas de textura.*

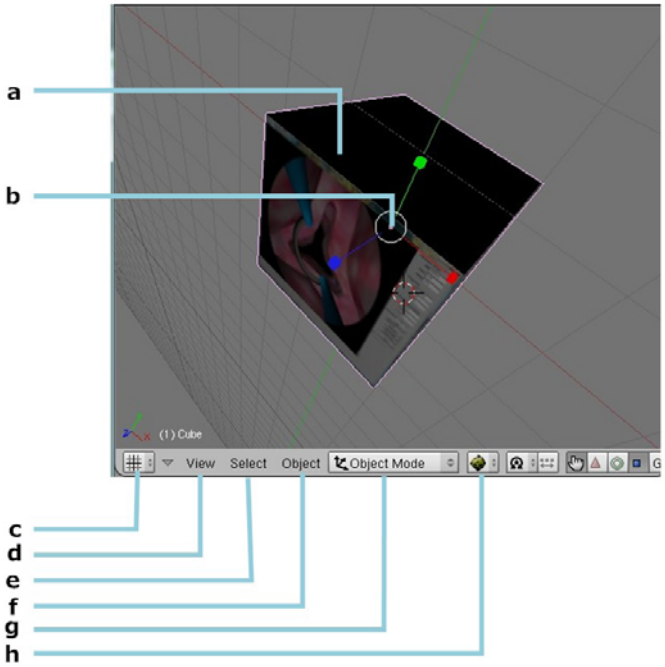


Figura A.3: *Entorno de la ventana de vista 3D. a) Objeto, b) Sistema coordenado de referencia, c) Vista 3D (view 3D), d) Menú de vistas, e) Menú de Selección, f) Menú de transformación del objeto, g) Selección de modo de edición, h) Selección de modo de visualización.*

Los módulos más importantes en esta parte son: el sombreado (*shading*, tecla F5) y el módulo de edición (*Editing*, tecla F9). En el sombreado se definen básicamente propiedades de materiales y de texturas, mientras que en el de edición, se definen propiedades de forma de la malla.

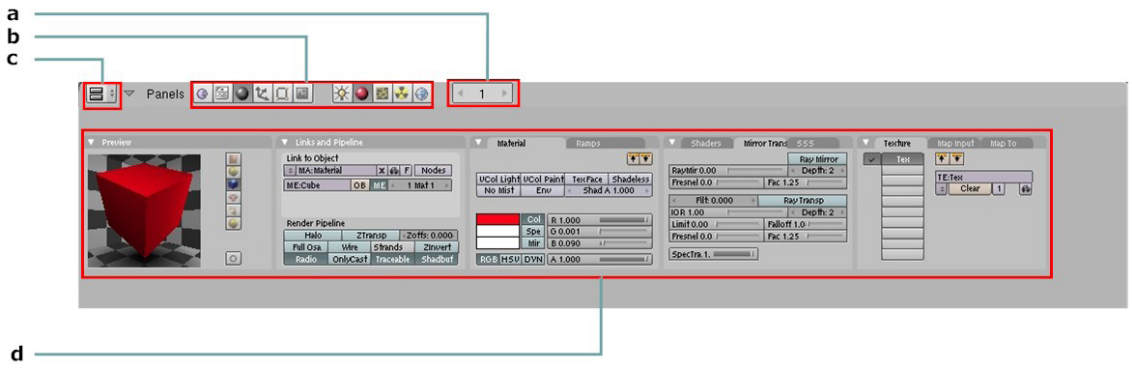


Figura A.4: *Paneles de herramientas. a) Número de cuadro (usado para animaciones), b) Paneles, c) Ventana de botones (Buttons window), d) área de propiedades y métodos.*

A.2.3. El editor de coordenadas de textura (UV Image Editor)

El editor de coordenadas de textura (Ver figura A.5), tiene como objetivo la edición de coordenadas de textura de una malla a partir de la información de sus vértices y caras. El entorno de esta ventana permite la carga de una imagen de textura y asociar, a cada cara del modelo, sus correspondientes coordenadas UV. Además, ofrece la facilidad de poder generar coordenadas de textura de acuerdo a la geometría del objeto, inclusive el poder realizar cortes del objeto y asociar las coordenadas a una sola imagen.

A.3. Edición de una malla 3D en formato OBJ

En su versión 2.46, Blender soporta los principales formatos 3D para la carga de modelos externos, entre éstos se encuentran los siguientes:

- Virtual Reality Modeling Language - VRML 1.0 (.wrl)
- DXF (.dxf)
- STL (.stl)
- 3D Studio (.3ds)
- AC3D (.ac)
- Autodesk DXF (.dxf)
- COLLADA 1.3.1 y 1.4(.dae)
- DEC Object File Format (.off)

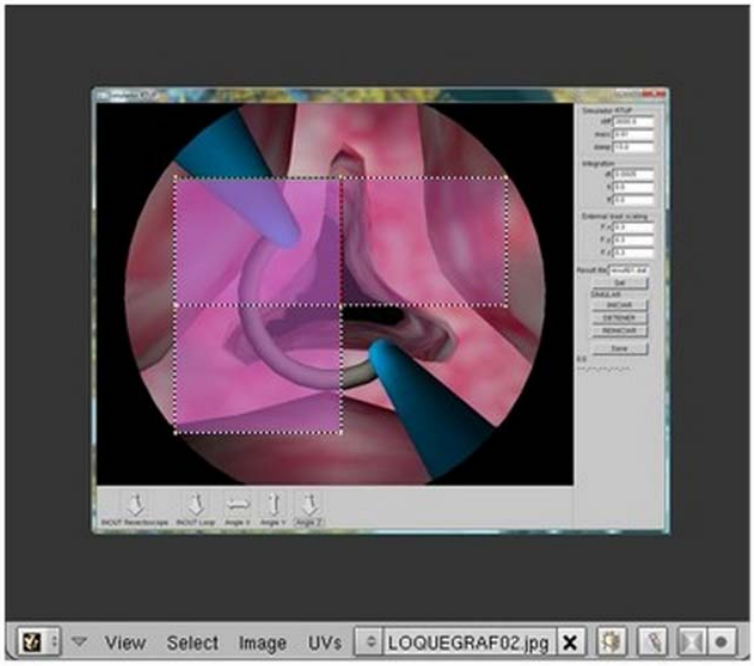


Figura A.5: *Editor de coordenadas de textura.*

- DirectX (.x)
- LightWave (.lwo)
- MD2 (.md2)
- Motion Capture (.bvh)
- OpenFlight (.flt)
- Paths (.svg, .ps, .eps, .ai, Gimp)
- Pro Engineer (.slp)
- Raw Faces (.raw)
- Stanford PLY (.ply)
- Wavefront (.obj)

Para el propósito de este trabajo, el formato estándar para carga de mallas fue el OBJ, el cual es empleado por el simulador RTUP y cuya implementación se describe en el capítulo 5.

A.3.1. Exportado de mallas para usar en el simulador RTUP

El simulador TURP, como se ha venido viendo, emplea el formato OBJ para la carga de mallas estáticas. Sin embargo, hay características propias del formato que el simulador emplea para obtener

la información del modelo. Estas características son: Vértices, Normales, Coordenadas de Textura y Materiales.

Blender, por defecto, exporta los archivos Wavefront OBJ con sus propiedades mínimas como son: Vértices, Normales y Materiales. Lo cual nos lleva a hacer uso de la configuración del modo de exportar un modelo para su correcto funcionamiento con el simulador.

La exportación del modelo se realiza una vez que se hayan realizado todas las modificaciones necesarias y mediante el menú $File >> Export >> Wavefront(.obj)$.

Apéndice B

Herramientas de software para el simulador RTUP

B.1. Extracción de contornos con ImageJ

ImageJ[25] es un programa de procesamiento de imágenes de dominio público, escrito en lenguaje Java[28]. Funciona tanto en una interfaz Web como en una versión de aplicación de escritorio, en computadoras que cuentan con la máquina vitual de Java v1.4 o superior. Actualmente las versiones disponibles son para los sistemas operativos: Windows, MacOS X y Linux.

Esta aplicación se usó principalmente para el procesamiento de las imágenes del VHP y la reconstrucción del modelo de la vejiga virtual. Las instrucciones empleadas corresponden a la definición en lenguaje de macros para ImageJ, que al ser ejecutados sobre una imagen producen el efecto deseado. Una observación importante, es que para ejecutar estas instrucciones, la imagen que representa el contorno debe ser necesariamente de tipo binaria y el fondo de la imagen debe ser blanco. Para usar la opción de fondo blanco, se debe desactivar la casilla *"Black background"* del cuadro de diálogo: *Binary Options*, del menú *Process>>Binary>>Binary Options*.

Convertir una imagen a binaria

```
run("Make Binary");
```

Dilatación de puntos de selección

```
run("Dilate");
```

Suavizado de la curva con filtro gaussiano

```
run("Gaussian Blur...", "radius=5");
```

Obtención del esqueleto o contorno mínimo

```
run("Skeletonize");
```

B.2. Muestreo de contornos con MATLAB

Para el muestreo de contornos se implementó un programa en MATLAB[36] que manda a llamar a la función de muestreo, que a su vez ejecuta el algoritmo descrito en el capítulo 3. La sintaxis del programa se muestra a continuación:

```
clear all;
close all;

numeroMuestras = 50;    %%Número de muestras por contorno

x=[];      %%Lista de coordenadas x
y=[];      %%Lista de coordenadas y
z=[];      %%Lista de coordenadas z
deltaZ=1;  %%Distancia entre plano de corte (profundidad en z)

muestrasVector=muestreaContorno('cext0.tif',512,512,numeroMuestras);
n=0;longitud=length(muestrasVector);
x=[x;muestrasVector(1:longitud(1,1),1)];
y=[y;muestrasVector(1:longitud(1,1),2)];
z=[z;zeros(longitud(1,1),1)+n*deltaZ];

muestrasVector=muestreaContorno('cext1.tif',512,512,numeroMuestras);
n=1;longitud=length(muestrasVector);
x=[x;muestrasVector(1:longitud(1,1),1)];
y=[y;muestrasVector(1:longitud(1,1),2)];
z=[z;zeros(longitud(1,1),1)+n*deltaZ];

...

muestrasVector=muestreaContorno('cext31.tif',512,512,numeroMuestras);
n=31;longitud=length(muestrasVector);
x=[x;muestrasVector(1:longitud(1,1),1)];
y=[y;muestrasVector(1:longitud(1,1),2)];
z=[z;zeros(longitud(1,1),1)+n*deltaZ];

muestrasVector=muestreaContorno('cext32.tif',512,512,numeroMuestras);
n=32;longitud=length(muestrasVector);
x=[x;muestrasVector(1:longitud(1,1),1)];
y=[y;muestrasVector(1:longitud(1,1),2)];
z=[z;zeros(longitud(1,1),1)+n*deltaZ];

save coord\x.txt x -ASCII
save coord\y.txt y -ASCII
save coord\z.txt z -ASCII
```

Genéricamente la función de muestreo se define como:

```
muestreaContorno(imagen,filas,columnas,numMuestras);
```

donde: *imagen* es el nombre de la imagen que contiene el contorno, *filas* y *columnas* son el tamaño vertical y horizontal de la imagen respectivamente; y *numMuestras* es el número de muestras a obtener del contorno.

B.3. El mallador de superficies

El mallador de superficies es un programa que tiene como objetivo llevar a cabo el proceso de reconstrucción de una malla de superficie a partir de un conjunto de coordenadas X,Y y Z; resultado de la aplicación del algoritmo de muestreo de contornos.

El programa tiene una interfaz muy simple e intuitiva (Ver figura B.1). El proceso de reconstrucción se limita a importar los archivos de coordenadas ubicados en archivos de texto. El resultado es un archivo en formato OBJ, listo para su visualización con cualquier programa de edición 3D como Blender.

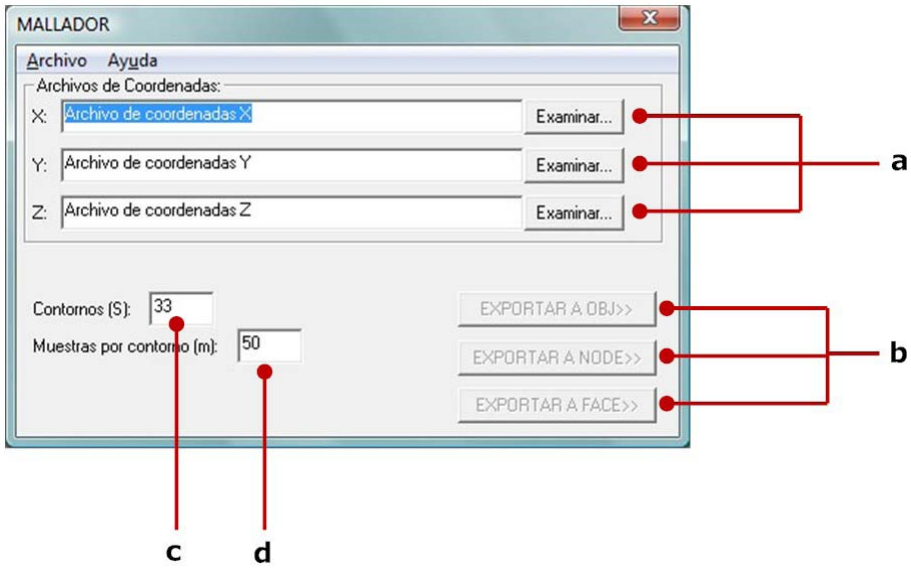


Figura B.1: *Interfaz gráfica del mallador de superficies. a) Carga de archivos de coordenadas; b) Exportar a formato: *.node, *.face, *.obj; c) Número de contornos muestreados y d) Muestras por contorno.*

Las funciones disponibles del programa son:

- a) Carga de archivos de coordenadas: Conformado por tres botones; los cuales sirven para seleccionar la fuente de cada una de las coordenadas (conjunto X, conjunto Y y conjunto Z). El formato de archivo por defecto es el *.txt, de texto plano. Es posible la especificación de otros formatos de archivos; siempre y cuando tengan la estructura de la definición de coordenadas, definida por el programa de muestreo.
- b) Exportar: Estos botones sirven para ejecutar el método de triangulación a partir de las coordenadas definidas. Mientras no se hayan definido las fuentes de los archivos, el programa no

activará estos botones. Los formatos de archivo a exportar son: *.node, *.face (para mallas usando la especificación del IMRIA) y *.obj (para mallas con la especificación de Wavefront OBJ). Si la malla se exportó con éxito, el programa mostrará un mensaje de confirmación.

- c) Contornos: En este campo se especifica el número de contornos procesados durante el algoritmo de muestreo.
- d) Muestras por contorno: En este campo de especifica el número de muestras por contorno empleadas durante el algoritmo de muestreo.

Es importante mencionar que el programa emplea los campos c) y d) para llevar a cabo un proceso de validación entre la información procesada y los archivos de entrada. Si el resultado de esa validación no es correcta, el programa muestra un mensaje de error.

B.4. El Simulador de RTUP v1.0

La primera versión del entorno gráfico del simulador de RTUP, para fines académicos y para fines de ilustración, se lanzó en Julio de 2008. La versión incluye CD de instalación y manual de usuario. A continuación se describe el manual de usuario elaborado durante este trabajo de tesis.

B.4.1. Entorno

El entorno del simulador tiene una interfaz gráfica muy simple e intuitiva para el usuario, cuenta con un panel de visualización 3D y dos áreas de control. La primera, ubicada en la parte inferior izquierda de la ventana, contiene botones que permiten el desplazamiento del resectoscopio virtual. La segunda, ubicada en la parte derecha del área de visualización, contiene campos y botones de configuración en tiempo de ejecución. La figura B.2 muestra la pantalla inicial del simulador, identificando las tres áreas principales.

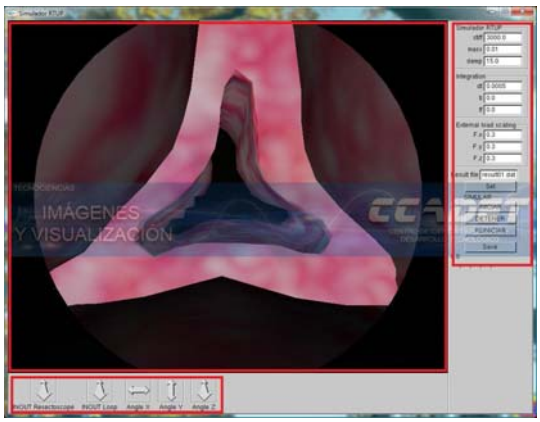


Figura B.2: *Entorno del Simulador de RTUP.*

B.4.2. Traslaciones

El área principal de visualización 3D (escena), puede ser manipulable mediante los botones de traslación. Estos botones, ubicados en la parte inferior izquierda, tienen las siguientes funciones:

- Entrar y salir del por el conducto uretral, próstata y exploración de vejiga. Ver figura B.4.2. Para activar esta función es necesario dar clic izquierdo del ratón en el control y arrastrar en la dirección deseada, arriba o abajo de la pantalla. Cuando el resectoscopio virtual haya alcanzado su límite permitido no seguirá avanzando o retrocediendo.

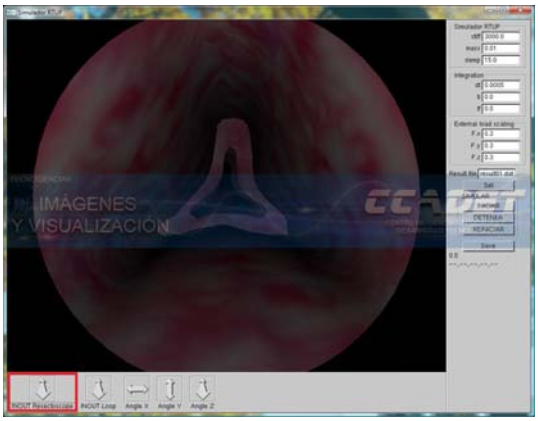


Figura B.3: Entrar y salir.

- Enfundar y desenfundar herramienta (resectoscopio virtual). Ver figura B.4. Para activar esta función es necesario dar clic izquierdo del ratón en el control y arrastrar en la dirección deseada, arriba o abajo de la pantalla, el resectoscopio virtual deberá mostrarse. Cuando la herramienta de corte no siga avanzando, quiere decir que ha alcanzado su límite "físico" permitido.

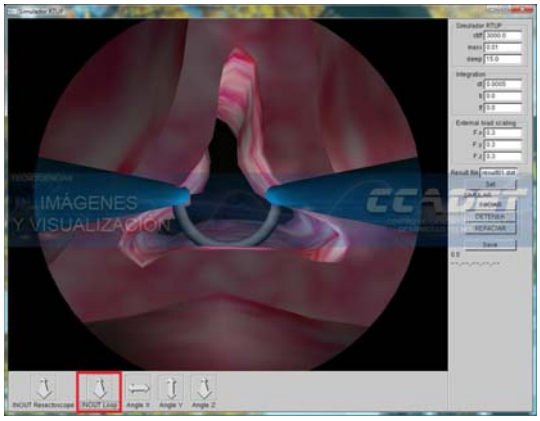


Figura B.4: Desfundar resectoscopio.

B.4.3. Rotaciones

El simulador permite además controlar los giros que un resectoscopio real llevaría a cabo durante un procedimiento in-vivo. estos giros o rotaciones sobre la escena 3D se llevan a cabo mediante los botones de ángulo en X, Y y Z, como se muestra a continuación.

- Rotar a la izquierda y a la derecha de la escena. Ver figura B.5. Para activar esta función es

necesario dar clic izquierdo del ratón en el control y arrastrar en la dirección deseada, izquierda o derecha de la pantalla.

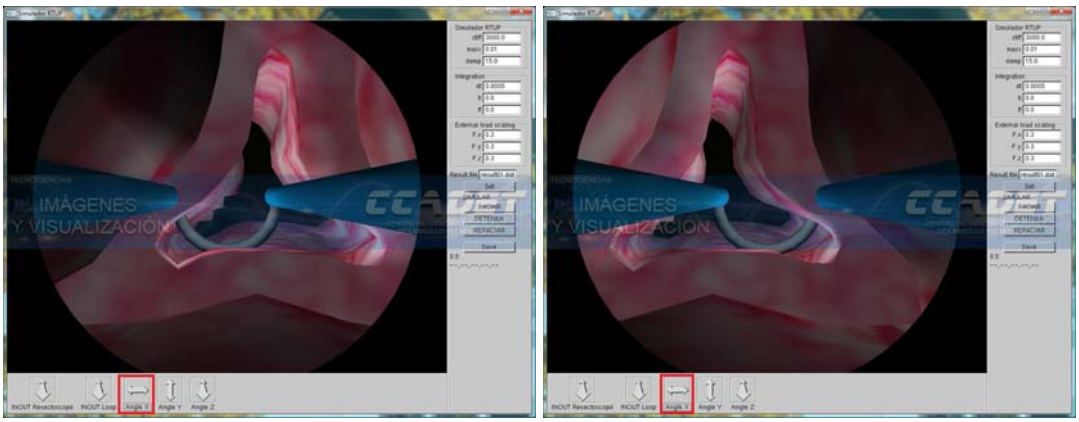


Figura B.5: Rotación izquierda y derecha.

- Rotar hacia arriba y hacia abajo de la escena. Ver figura B.6. Para activar esta función es necesario dar clic izquierdo del ratón en el control y arrastrar en la dirección deseada, arriba o abajo de la pantalla.

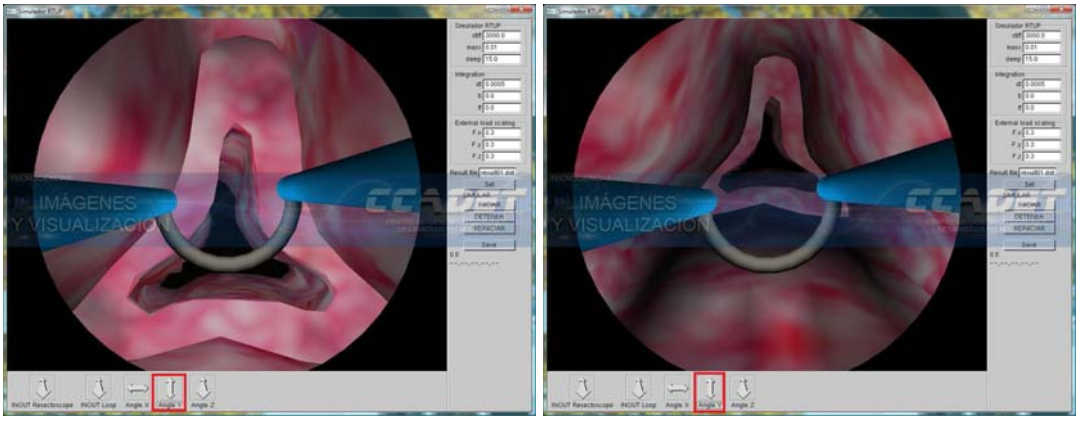


Figura B.6: Rotación arriba y abajo

- Rotaciones con respecto al eje de vista, en sentido horario y sentido antihorario. Ver figura B.7. Para activar esta función es necesario dar clic izquierdo del ratón en el control y arrastrar en la dirección deseada, arriba o abajo de la pantalla. PRECAUCIÓN: Un exceso en los giros con respecto al eje de visión puede causar una distorsión en la visualización, llegando a perderse la orientación del observador.

B.4.4. Parámetros de simulación

Los parámetros de la simulación son variables que pueden modificarse en tiempo de ejecución. Dichos controles se encuentran en la parte derecha de la pantalla principal y consisten en un conjunto de

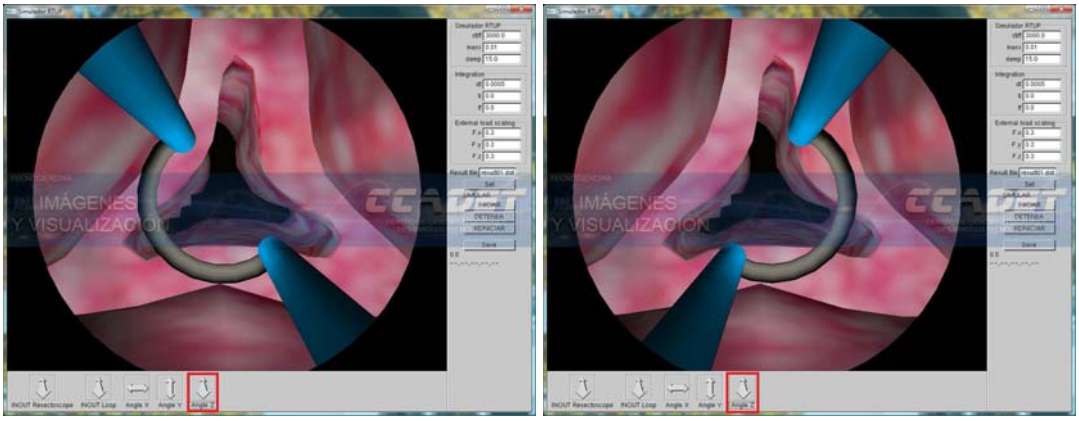


Figura B.7: Rotaciones con respecto al eje de vista.

campos editables con valores predefinidos. Una vez editados estos valores, los cambios en el programa se llevarán a cabo cuando se pulse el botón "Set". Ver figura B.8.

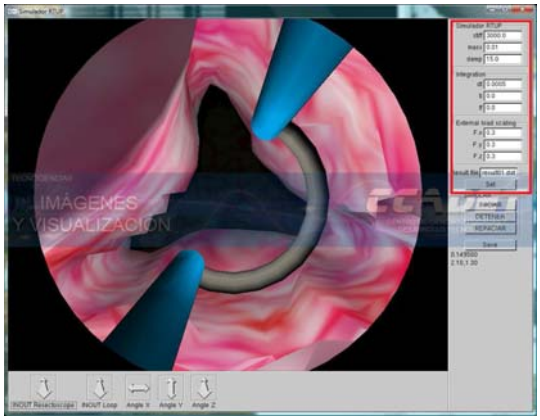


Figura B.8: *Parámetros de simulación.*

B.4.5. Controles de acción

Los controles de acción se sitúan en la parte derecha de la ventana del simulador, justo debajo del control "Set" y son usados para efectuar eventos propios del sistema. Ver figura B.9. Estos controles son:

- `Iniciar:` Inicia simulación, activando el algoritmo de deformación de la próstata.
- `Detener:` Pausa el proceso de simulación.
- `Reiniciar:` Reinicia las variables de simulación y restablece la forma original del tejido de la próstata virtual.

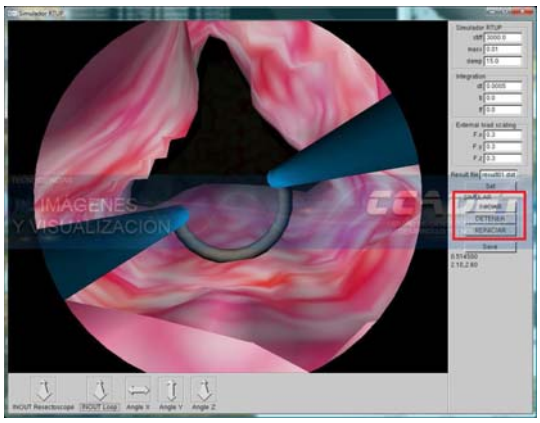


Figura B.9: *Controles de acción.*

B.4.6. Efectos y Teclas especiales

El simulador cuenta con teclas especiales para aplicar diversos efectos y mostrar cierta información gráfica o audible. Estas son

- Tecla `|A,a|`: Activa o desactiva el efecto de audio.
- Tecla `|m|`: Ocultar malla del modelo de la próstata.
- Tecla `|M|`: Mostrar malla del modelo de la próstata.
- Tecla `|n|`: Ocultar normales del modelo de la próstata.
- Tecla `|N|`: Mostrar normales del modelo de la próstata.
- Tecla `|F,f|`: Activa o desactiva modo de pantalla completa
- Tecla `|P,p|`: Activa o desactiva efecto de niebla
- Tecla `|L,l|`: Activa o desactiva efectos de iluminación
- Tecla `|D,d|`: Activa o desactiva efecto de deformación de la lente
- Tecla `|ESC|`: Salir del programa.

B.4.7. Configuración del simulador

El simulador se integra con un archivo de configuración (sim.bat) que emplea el intérprete de comandos para ejecutar el programa. Este archivo tiene la siguiente estructura:

```
prostetra -model models/modelo6bis/pros15_13Adap -tool tools/looptoroid-1
-maskDir models/lens/ -mask mask.obj -bladderDir models/bladder/ -bladder
bladder3.obj -bladderTex tissue128-111.bmp -urethraDir models/urethra/
-urethra urethra5.obj -urethraTex tissueb17.bmp -stiff 3000.0 -mass 0.01
-damp 15.0 -dt 0.0005 -ti 0.0 -tf 0.0 -fs 0.3 -modelposition 0.0 0.0 -4.6
```

```
-loopposition 4.2 -collisiontime 10 -collisiontolerance 0.3 -responsetime
10 -cuttolerance 0.6 -texture on -texturedb texturedb/tissue3D-B-128x128x
128/tissue -texturesize 128 -GPUdeformation on  -interface off -audio on
```

Las opciones configurables son:

- `-model`: Archivo del Modelo de la próstata.
- `-tool`: Archivo del Modelo de la Herramienta.
- `-maskDir`: Directorio del archivo de la máscara que simula la lente.
- `-mask`: Archivo del modelo de la máscara.
- `-bladderDir`: Directorio del archivo del modelo de la Vejiga.
- `-bladder`: Archivo del modelo de la Vejiga.
- `-bladderTex`: Archivo de textura para el modelo de la Vejiga.
- `-urethraDir`: Directorio del archivo del modelo de la Uretra.
- `-urethra`: Archivo del modelo de la Uretra.
- `-urethraTex`: Archivo de textura para el modelo de la Uretra.
- `-stiff`: Rigidez elástica de la próstata.
- `-mass`: Masa de la próstata.
- `-damp`: Coeficiente de Amortiguamiento de la próstata.
- `-dt`: Constante de tiempo para cálculo numérico.
- `-ti`: Tiempo inicial de la simulación.
- `-tf`: Tiempo final de la simulación.
- `-fs`: Factor de escala de las fuerzas de reacción del tejido.
- `-modelposition`: Posición inicial en coordenadas XYZ del modelo de la próstata.
- `-loopposition`: Posición inicial en coordenadas XYZ de la herramienta.
- `-collisiontime`: Tiempo máximo crítico para el algoritmo de detección de colisiones.
- `-collisiontolerance`: Máximo crítico para el algoritmo de cálculo de respuesta.
- `-responsetime`: Tiempo de respuesta.
- `-cuttolerance`: Tolerancia para definir cortes (o zona de corte).
- `-texture`: Activa o desactiva texturizado (on/off).
- `-texturedb`: Fuente de texturas 3D para modelo de próstata.
- `-texturesize`: Número de imágenes en la pila de texturas 3D.
- `-GPUdeformation`: Activa o desactiva cálculo con GPU (on/off).
- `-interface`: Activa o desactiva interfaz mecatrónica (on/off).
- `-audio`: Activa o desactiva audio al inicio del programa (on/off).
- `-stereo`: Activa o desactiva la función de estéreo.

Bibliografía

[1] ALTAMIRANO, Felipe, *Interfaz Mecatrónica para un simulador de Cirugía de Próstata*, Tesis de Maestría en Ingeniería e Intrumentación, Director: Dr. Fernando Arámbula Cosio, CCADET-UNAM, México, 2007. 8:9.

[2] AZCÁRRAGA G., *UROLOGÍA*, Méndez Editores, 7a. Edición, México, 1997. 9-28 pp. 256-264.

[3] Cagatay Basdogan, Chih-Hao Ho, and Mandayam A. Srinivasan, *Virtual Environments for Medical Training: Graphical and Haptic Simulation of Laparoscopic Common Bile Duct Exploration*, IEEE/ASME TRANSACTIONS ON MECHATRONICS, VOL. 6, NO. 3, SEPTEMBER 2001, 269-285.

[4] BLENDER WEB SITE - http://www.blender.org/

[5] BOURKE, Paul: *Nonlinear Lens Distortion with an example using OpenGL*, The University of Western Australia, Australia 2000. http://local.wasp.uwa.edu.au/~pbourke/projection/lenscorrection/.

[6] Morten BRO-NIELSEN, Ph.D.; David HELFRICK M.Sc.; Bill GLASS, M.Sc.; Xiaolan ZENG, M.Phil., Hugh CONNACHER, M.Sc., *VR Simulation of Abdominal Trauma Surgery*, Medicine Meets Virtual Reality 6 (January 1998), pp. 117-123.

[7] BUSS, Samuel R., *3-D Computer Graphics: A Mathematical Introduction with OpenGL* Cambridge University Press, Estados Unidos, 2003.

[8] BUTOW, Eric, *User Interface Design for Mere Mortals: A Hands-On Guide to User Interface Design.*, Segunda Edición, Addison-Wesley - Pearson Education, Estados Unidos, 2007.

[9] EBERT David S., MUSGRAVE Kenton F., PEACHEY Darwyn, PERLIN Ken, WORLEY Steven, *Texturing and modeling : A Procedural Approach.*, 3rd. Edition, Morgan Kaufmann Publishers, Estados Unidos, 2003.

[10] FileFormat.Info; *Alias/WaveFront Material (.mtl) File Format*, http://www.fileformat.info/format/material/.

[11] Jan Fischer, Dirk Bartz, Wolfgang Straβ er, *Intuitive and Lightweight User Interaction for Medical Augmented Reality*, Visual Computing for Medicine 2005, Erlangen, Germany, November 16-18, 2005.

[12] FREDOTOVICH, Norberto, *Historia de la Urología: El auge de la cirugía transuretral*, Revista Argentina de Urología Vol. 69 (3), Sociedad Argentina de Urología, Argentina 2004, 142-145.

[13] GARTNER, Leslie P., HIATT, James L., *Histología: Texto y Atlas*, McGraw Hill Interamericana, 1a. Edición, México, 1997. pp. 426-444.

[14] GASTÉLUM, Alfonso; *Construcción de un modelo del sistema gastrointestinal alto para simuladores de endoscopía*, Tesis de maestría en Física Médica, Tutor: Dr. Jorge Alberto Márquez Flores, CCADET-UNAM, México, 2005.

[15] GERGELY Vass and Tamás PerlakI: *Applying and removing lens distortion in post production*, Second Hungarian Conference on Computer Graphics and geometry 2003, Budapest, speech. http://www.vassg.hu/pdf/vass_gg_2003_lo.pdf.

[16] GONZÁLEZ, Rafael, WOODS, Richard, *Digital Image Processing*, Segunda Edición, Prentice Hall - Pearson Education International, USA, 2002. p. 66.

[17] BLENDER GPL License - http://download.blender.org/release/GPL-license.txt

[18] Blender Foundation History - http://www.blender.org/blenderorg/blender-foundation/history/.

[19] Kim V. Hansen, Lars Brix, Christian F. Pedersen, Jens P. Haase, Ole V. Larsen, *Modelling of interaction between a spatula and a human brain*, Medical Image Analysis 8 (2004) 23-33.

[20] Hearn, D., Baker, M.P.; *Computer Graphics (2nd Edition C Version)*, Prentice-Hall, Estados Unidos, 1997.

[21] HECHT, Eugene: *Física 2: Álgebra y Trigonometría*, Segunda Edición, International Thomson Editores, México, 1999.

[22] HERVÉ DELINGETTE, *Toward Realistic Soft-Tissue Modeling in Medical Simulation*, Proceeding of the IEEE, Vol. 86, No. 3, March 1998.

[23] Joseph Hidler, Timea Hodics, Benjamin Xu, Bruce Dobkin, Leonardo G Cohen, *MR compatible force sensing system for real-time monitoring of wrist moments during fMRI testing*, Journal of Neuroscience Methods 155 (2006) 300-307.

[24] HPBACTUAL-2005, Colegio Mexicano de Urología,*Guía Nacional para el tratamiento de la Hiperplasia Prostática Benigna*. El Colegio Mexicano de Urología A.C., Actualización 2005.

[25] *ImageJ*, http://rsbweb.nih.gov/ij/.

[26] IWANOWSKI, Marcin, SERRA, Jean; *Morphological Interpolation and Color Images*, 10th International Conference on Image Analysis and Processing (ICIAP'99), pp. 50-55.

[27] IXTLI, Observatorio de Visualización IXTLI, Dirección General de Servicios de Cómputo Académico (DGSCA), Universidad Nacional Autónoma de México. http://www.ixtli.unam.mx/.

[28] *JAVA*, Sun Microsystems, http://www.sun.com/java/.

[29] Erwin Keeve, Sabine Girod, Bernd Girod, *Computer-Aided Craniofacial Surgery*, Computer Assisted Radiology, CAR '96 Paris, France, June 26-29, 1996.

[30] Erwin Keeve, Ph.D., Sabine Girod, M.D. D.M.D., Ron Kikinis M.D., Bernd Girod, Ph.D., *DEFORMABLE MODELING OF FACIAL TISSUE FOR CRANIOFACIAL SURGERY SIMULATION*, Computer Aided Surgery, Volume 3 No. 5, 1999, 228 - 238.

[31] R. M. Koch, M.H.Gross, F. R. Carlsy, D. F. von Büren, G. Fankhauser, Y. I.H. Parish, *Simulating Facial Surgery Using Finite Element Models*, International Conference on Computer Graphics and Interactive Techniques 1996, 421 - 428.

[32] U. Kühnapfel, H.K. Cakmak, H. Maaβ, *Dynamic Medical Visualization: Endoscopic surgery training using virtual reality and deformable tissue simulation*, Computers & Graphics 24 (2000) 671-682.

[33] LIPPERT, Herbert, *Anatomía: Estructura y Morfología del Cuerpo Humano*, MARBAN LIBROS, 4a. Edición, España, 2003. 336, 355-361, 422-429.

[34] LIRA, Berra, *Implementación de un modelo deformable de masas y resortes en una GPU para un simulador de cirugía de próstata.* Tesis de Licenciatura en Ingeniería en Computación, Tutor: M. en C. Miguel Ángel Padilla Castañeda, FI-CCADET-UNAM, México, Junio 2008.

[35] Magee D. and Kessel D., *A Computer Based Simulator for Ultrasound Guided Needle Insertion Procedure*, Proc. IEE International Conference on Visual Information Engineering, pp301-308, 2005.

[36] *MATLAB*, http://www.mathworks.com/products/matlab/.

[37] REDDY, Martin; *The Graphics File Formats Page-OBJ Files*, http://www.martinreddy.net/gfx/3d/OBJ.spec.

[38] PADILLA Castañeda, M.A., Sevilla Martínez, P., Arámbula Cosío, F., 2004. *Avances recientes en entrenamiento de cirugía de próstata asistido por computadora*, XIX Congreso de la Sociedad Mexicana de Instrumentación SOMI. SOMI, Pachuca, México., octubre 2004, pp. FAC1936: 1-8.

[39] Miguel A. Padilla Castañeda, Fernando Arámbula Cosio, *Deformable model of the prostate for TURP surgery simulation*, Computers & Graphics 28 (2004) 767-777.

[40] PADILLA, Miguel, *Manual de referencia para las funciones de biblioteca en C para manejo de mallas en 3D.* Reporte técnico, CCADET, UNAM. 2006.

[41] Padilla Castañeda, M.A., Arámbula Cosío. F., 2004, *Three-dimensional deformable model of the prostate for TURP surgery simulation.* Computers & Graphics, 28, pp. 767-777.

[42] Parent Rick, *Computer Animation: Algorithms and Techniques.*, Morgan Kaufmann Publishers, Estados Unidos, 2002.

[43] M. Paula S. F. Gomes, Adrian R. W. Barrett, Anthony G. Timoney, and Brian L. Davies, *A Computer-Assisted Training/Monitoring System for TURP Structure and Design*, IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, VOL. 3, NO. 4, DECEMBER 1999, 242-251.

[44] A. Petersik, B. Pflesser, U. Tiede, K. H. Hoehne, R. Leuwer, *Haptic volume interaction with anatomic models at sub-voxel resolution*, 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, p. 66.

[45] PHANTOM, SensAble, *PHANTOM OMNI® USER´S GUIDE*, SensAble Technologies Inc., U.S.A. 2007. http://www.sensable.com/haptic-phantom-omni.htm.

[46] Lamberto PIRON, Paolo Tonin, Francesco Piccione, Vincenzo Iaia, Elena Trivello, Mauro Dam, *Virtual Environment Training Therapy for Arm Motor Rehabilitation*, Presence, Vol. 14, No. 6, December 2005, 732-740.

[47] PRATT, William K., *Digital Image Processing*, Fourth Edition, Wiley-Interscience, Estados Unidos, 2007.

[48] Mei, Q., Harris, S.J., ArambulaCosio, F., Nathan, M.S., Hibberd, R.D., Wickham, J.E., Davies, B.L.: *PROBOT - A Computer Integrated Prostatectomy System*, Visualization in Biomedical Computing, 1311 (1996) 581-590.

[49] Radetzky A., Rudolph M., *Simulating tumour removal in neurosurgery*, International Journal of Medical Informatics, Vol. 64, No. 2, Dic. 2001, 461-472.

[50] REYES, Bartolomé, *Comunicación y registro de un modelo en 3D con un maniquí para un sistema de simulación de Cirugía de Próstata*, Tesis de Licenciatura en Ingeniería Eléctrica, Tutor: M. en C. Miguel Ángel Padilla Castañeda, CCADET-UNAM, México, 2007.

[51] RICHARD, Robb, DENNIS, Hanson, *Biomedical Image Visualization Research Using the Visible Human Datasets*, Clinical Anatomy, © 2008 Wiley-Liss, Inc., A Wiley Company, Vol. 19 No. 3, Págs. 240 - 253.

[52] Richard S. Wright Jr., Benjamin Lipchak, *OpenGL® SuperBible.*, Third Edition, Sams Publishing, Estados Unidos, 2004.

[53] ROBBINS, Cotran, Kumar, Collins, *Patología Estructural y Funcional*, McGraw Hill Interamericana, 6a. Edición, Colombia, 2000. 1071-1073.

[54] M. A. F. Rodrigues, D. F. Gillies, P. Charters, *Realistic Deformable Models for Simulating the Tongue during Laryngoscopy*, Medical Imaging and Augmented Reality, 2001. Proceedings, 2001,125-130.

[55] Tobias Sielhorst, Tobias Obst, Rainer Burgkart, Robert Riener, Nassir Navab, *An Augmented Reality Delivery Simulator for Medical Training*, AMI-ARCS 2004, Augmented Environments for Medical Imaging September 30th, 2004, Rennes (France).

[56] SORIANO, David, *Generación de modelos de mallas de tetraedros para simuladores quirúrgicos.*, Tesis de licenciatura en Ingeniería en Computación, Tutor: M. en C. Miguel Ángel Padilla Castañeda, CCADET-UNAM, México, 2008.

[57] Sourin, A.; Sourina, O.; Howe Tet Sen, *Virtual Orthopedic Surgery Training*, Computer Graphics and Applications, IEEE, Volume 20, Issue 3, May/Jun 2000 Page(s):6 - 9.

[58] Stylopoulos N, Cotin S, Dawson S, Ottensmeyer M, Neumann P, Bardsley R, Russell M, Jackson P, Rattner D., *CELTS: A clinically-based Computer Enhanced Laparoscopic Training System*, Studies in health technology and informatics, 2003;94:336-42.

[59] Sandeep Subramanian, Luiz A Knaut, Christian Beaudoin, Bradford J McFadyen, Anatol G Feldman, Mindy F Levin, *Virtual reality environments for post-stroke arm rehabilitation*, Journal of NeuroEngineering and Rehabilitation 2007, 4:20.

[60] G. Székely, Bajka, *Virtual Reality Based Surgery Simulation for Endoscopic Gynaecology.*

[61] Magnenat-Thalmann, N.; Cordier, F., *Construction of a Human Topological Model From Medical Data*, Information Technology in Biomedicine, Vol. 4, No. 2, Jun 2000;137 - 143.

[62] The Visible Human Project®, National Library of Medicine - http://www.nlm.nih.gov/research/visible/visible_human.html.

[63] Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner, OpenGL Architecture Review Board; *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*, 3rd. Edition, Addison-Wesley - Sillicon Graphics, Estados Unidos, 2000.

[64] WOTSIT: 3D Graphics Files - www.wotsit.org / Wavefront Object files (Acrobat): http://www.wotsit.org/getfile.asp?file=w_obj&sc=274213343.

[65] XU, G., Chao C., Bozkurt A., *VIP-MAN: An image-based whole-body adult model constructed from color photographs of the Visible Human Project for multi-particle monte carlo calculations*, Health Physics, 0 2000 Health Physics Society, Mayo 2000, Vol. 78, Num. 5, Págs. 476-486.

[66] Zhu, Y., Magee D., Ratnalingam, R., Kessel, D., *A Virtual Ultrasound Imaging System for the Simulation of Ultrasound-Guided Needle Insertion Procedures*, Proc. Medical Image Understanding and Analysis (MIUA), pp61-65, vol 1., 2006. http://www.comp.leeds.ac.uk/drm/us.html.

[67] ZUDAIRE, Juan Javier, *Manual de Urología*, Ariel Ciencias Médicas, España, 2002. 297-325.

Índice alfabético