Universidad de las Ciencias Informáticas
Facultad 5

Estudio de Algoritmos para la detección de colisiones entre objetos complejos en Simuladores Quirúrgicos

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor(es): Julio Alain Delgado Suárez
Julio César Vicente Téllez

Tutor: Ing. Idelkys Ramírez Quintana
Co-tutor: Lic. Juan Manuel Mederos Martínez

Ciudad Habana
Julio, 2007

DECLARACIÓN DE AUTORÍA

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

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

_____ _____

<Nombre autor> **<Nombre autor>**

_____ _____

Firma del Autor Firma del Autor

_____

<Nombre tutor>

_____

Firma del Tutor

Datos de Contacto

Ing. Idelkys Quintana Ramírez
Ingeniera en Telecomunicaciones y Electrónica, UCLV
E-mail: idelkysq@gmail.com

Agradecimientos

Quisiera agradecerle a todos mis amigos, familiares, a la tutora Idelkys, a todo aquel que aportó un granito de arena, para lo que en un pasado quizás pareció un sueño hoy en día podamos decir que ya es una realidad. Quisiera agradecer también a mi novia, que estuvo siempre a mi lado no importa cuan difícil fuera el momento y me dio la fuerzas necesaria para salir adelante. Igualmente a mis padres que siempre confiaron en mí.

Julio Alain Delgado Suárez

Agradecerles a muchas personas, primeramente a mi familia que me han ayudado tanto a lo largo de mi vida, pese a todos los problemas que se puedan presentar, resaltar dos nombres, Abel y Nelly quienes cuando era un crío me dieron un cariño que jamás será igualado. Agradecer también a trabajadores de la universidad que me vieran como un hijo más lo que siempre retribuiré, a mi tutora Idelkys, a un gran amigo Mario, a mi novia Lisset y otros amigos que me han ayudado en la realización de esta tesis, a mi compañero de tesis Julio Alain Delgado Suárez.

Julio César Vicente Téllez

Dedicatoria

Dedicar esta tesis a todos los que de una forma u otra han contribuido en mi desarrollo, desde todas las etapas hasta los que me ayudaron a realizarme como estudiante universitario, a mi familia que debe de estar orgullosa por verme transformar en un profesional, dedicarle también a mis amigos de escuela y los de otras partes, en fin se la dedico a todos incluyendo a mi equipo de béisbol preferido.

Julio César Vicente Téllez

Dedico este presente trabajo a todos los que me apoyaron para que poco a poco me fuera forjando como un hombre de bien en esta vida. A mis padres que siempre estuvieron a mi lado y a mi hermana que siempre fue un motivo de inspiración y de mi desarrollo en la esfera intelectual. A mi novia que llena mi vida de felicidad y a todo aquel que me dio un instante de su tiempo y de su corazón.

Julio Alain Delgado Suárez

Resumen

Los ambientes interactivos para los objetos dinámicamente deformables juegan un papel importante en la simulación de la cirugía. Estos ambientes requieren de modelos deformables rápidos, y técnicas de manejo de colisiones eficaces. Mientras el descubrimiento de la colisión para los cuerpos rígidos es bien investigado, el de los objetos deformables introduce los problemas desafiantes adicionales, entre los cuales se pueden mencionar el del aumento de la complejidad de los algoritmos de detección y el de encontrar la forma más óptima para contener el objeto. Esta tesis está enfocada a los aspectos anteriormente mencionados y resume las recientes investigaciones en el área del descubrimiento de colisiones de objetos deformables. Se discuten varios acercamientos basados en limitar jerarquías de volumen. Se hace una propuesta del algoritmo que se considera el más óptimo basado en los estudios que en la misma se realizan y se presentan técnicas de avanzada para la solución del problema de la detección de colisiones en el entorno de la simulación quirúrgica.

PALABRAS CLAVES

Quirúrgico, colisiones, jerarquías, simulación

Índice

INTRODUCCIÓN 1

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA 5

1.1 INTRODUCCIÓN 5
1.2 SIMULADORES QUIRÚRGICOS VIRTUALES 5
1.3 APLICACIONES 8
1.4 INTRODUCCIÓN DE LOS SSQ EN CUBA 11
1.5 DETECCIÓN DE COLISIONES 12
1.5.1 Detección de colisiones en modelos rígidos *12*
1.5.2 Algunas características de las envolventes *16*
1.6 COMENTARIOS PARCIALES 16

CAPÍTULO II: ESTUDIO DE LOS ALGORITMOS DE DETECCIÓN DE COLISIONES 18

2.1 INTRODUCCIÓN 18
2.2 FUNCIONAMIENTO DE UN SIMULADOR QUIRÚRGICO 18
2.3 DETECCIÓN DE COLISIONES PARA OBJETOS DEFORMABLES 20
2.3.1 Estructuras de datos utilizadas en la detección de colisiones *22*
2.3.2 Construcción de BVHs *25*
2.4 ALGUNAS TÉCNICAS BÁSICAS DE DETECCIÓN DE COLISIONES 27
2.4.1 Fuerza Bruta *27*
2.4.3 Esferas de Inclusión *28*
2.5 MÉTODOS UTILIZADOS EN LA DETECCIÓN DE COLISIONES 28
2.5.1 Algoritmos de Lin-Canny y V-Clip *28*
2.5.2 Diagramas de Voronoi *28*
2.5.4 Árboles OBB *29*
2.6 ALGORITMOS ENCONTRADOS PARA EL TRATAMIENTO DE LAS COLISIONES EN OBJETOS COMPLEJOS 30
2.6.1 Volúmenes de la visión *30*
2.6.2 Detección de colisiones tipo estática *31*
2.6.3 AABB árboles y modelos deformables *33*
2.6.4 V-clip *36*
2.6.5 GJK *52*
2.7 LIBRERÍAS MAYORMENTE UTILIZADAS POR ESTOS ALGORITMOS 52

2.7.1 I-Collide: Tipo de modelo (Modelos o conjunto de modelos convexos) 52
2.7.2 V-COLLIDE: Tipo de modelo (Modelos triangulares) 53
2.7.3 RAPID: Tipo de modelo (Modelos triangulares) 53
2.7.4 SWIFT: Tipo de modelo (Geometrías cerradas y convexas) 54
2.7.5 SOLID: Tipo de modelo (Objetos convexos) 54
2.7.7 V-CLIP: Tipo de modelo (Modelos convexo) 54
CAPÍTULO III: PROPUESTA DEL ALGORITMO 56
3.1 INTRODUCCIÓN 56
3.2 DESARROLLO DE LA PROPUESTA 56
3.2.1 ALGORITMO AABB TREE 57
3.2.2 ESFERAS DE INCLUSIÓN 57
3.2.2.1 Medial-Axis Surfaces 59
3.2.3 OBB TREE 60
3.3 PROPUESTA FINAL DE ALGORITMO 61
CONCLUSIONES 63
RECOMENDACIONES I
REFERENCIAS BIBLIOGRÁFICAS II
BIBLIOGRAFÍA VI
GLOSARIO VII

Introducción

Todas las simulaciones, ya sean softwares de computadora o modelos plásticos, son modelos virtuales de realidad. Pero el término *'realidad virtual*' (RV) generalmente se entiende en referencia a un software interactivo avanzado con gráficos excepcionales en 3D y de una naturaleza inmersiva que permite practicar las habilidades y los procedimientos psicomotores con un alto grado de experiencia sensorial. La RV incrementa de manera espectacular las oportunidades para la interacción en tiempo real con un modelo dinámico de realidad, a través de la interfase computadora-ser humano.

Muchos han sido los avances de la medicina a lo largo de los últimos años y no solo la cantidad sino también la calidad. Se han inventado y mejorado técnicas para el tratamiento de afecciones y males de toda índole, así como han sido desarrollados nuevos y más precisos métodos de medida y recolección de variables corporales, como el peso, la talla, signos vitales entre otras. Todos estos avances han sido logrados gracias a la disposición de un protocolo de pruebas y ensayos, cuyos primeros beneficiarios son los animales. No obstante, en última instancia deben usarse voluntarios humanos para poder certificar la validez del producto en fase de pruebas.

Este es un proceso necesario que no puede sustituirse conforme avanza la tecnología, al menos hasta la actualidad. Pero sí existen otras áreas de la medicina en las que se pueden realizar cambios en la metodología con el fin de mejorar el progreso y la calidad; es el caso del aprendizaje y entrenamiento de los futuros médicos. Como puede resultar obvio, un cirujano no puede entrar por primera vez a un quirófano con un paciente grave sin haber tenido experiencias anteriores en este sentido.

Por tal motivo, la tradición médica ha utilizado diversos tipos de “entrenadores” basados en reproducciones de zonas del cuerpo y en cadáveres, para poder aportar una base realista al aprendiz, y permitirle practicar al principiante que asiste al profesor.

Pues bien, como quiera que la experiencia adquirida por el médico con respecto al trato con pacientes, sea muy importante, también resulta necesario entrenarse lo máximo posible. No sólo eso, sino que el doctor deben tener confianza en sí mismo a la hora de abordar un problema pues la vida y el bienestar de seres humanos dependen de él. El problema radica en que no siempre hay voluntarios humanos para poder realizar prácticas, las que no siempre son satisfactorias, pues no debe olvidarse que hablamos de estudiantes sin experiencia.

Actualmente se estudia una vía de entrenamiento basada en simuladores virtuales, también llamados ***simuladores quirúrgicos***. Estos dispositivos, básicamente, son simuladores de realidad virtual a los que se les programan modelos de órganos y zonas del cuerpo humano. Mediante dispositivos de realimentación, como las gafas de realidad virtual, el médico puede observar las partes deseadas del cuerpo en un entorno tridimensional.

Por supuesto, esto no es suficiente para poder llevar a cabo un entrenamiento de los futuros médicos. Por ello se utilizan otros dispositivos de realimentación: joysticks o herramientas tipo phantom, que brindan al residente una interacción más real con el entorno quirúrgico. Lo que se persigue con la utilización de estos dispositivos es aportar al entorno 3D una dimensión sensorial más al permitir usar el sentido del tacto. Principalmente, el médico es capaz de interactuar con el órgano, pues el dispositivo de realimentación transmite las oposiciones de fuerza que el médico se encontraría si estuviera trabajando con un cuerpo real. [1]

En la Universidad de Ciencias Informáticas (UCI) se encuentra en desarrollo un proyecto para simuladores quirúrgicos, el mismo se encuentra en su estado inicial, lo que hace que se tenga muy poca información referente al tema y que no se abarquen las principales temáticas que son necesarias para cumplir con los objetivos propuestos para dicho proyecto.

Luego de realizar una amplia búsqueda de información, aprovechando todos los grandes beneficios que nos brindan las Tecnologías de la Informática y las Comunicaciones (TICs) y después de una investigación basada en los datos encontrados en la bibliografía consultada, se concluye que el problema a resolver radica en cómo agrupar y sintetizar la bibliografía encontrada sobre los algoritmos de detección de colisiones en Simuladores Quirúrgicos, para de esta manera tener localizada la información necesaria y hacer más comprensible dicha temática dentro del desarrollo del proyecto.

Planteando lo anteriormente expuesto como **situación problémica,** el **objeto de estudio** de esta investigación se centra en la detección de colisiones en objetos complejos y el **campo de acción** está enfocado en el estudio de los algoritmos para la detección de colisiones en objetos complejos dentro de la simulación quirúrgica

Se tiene como **objetivo general** sistematizar en los elementos relacionados con los algoritmos para la detección de colisiones entre objetos complejos en el marco de la cirugía, de manera que ello permita una

mayor eficacia en la simulación de operaciones y de esta forma convertir el entorno virtual del simulador en algo cada vez más real.

Dentro de los **objetivos específicos** se encuentran:

- Determinar las técnicas más importantes para la detección de colisiones
- Determinar los algoritmos, tecnologías y tendencias actuales utilizadas en la detección de colisiones

Se plantean, entonces, un grupo de **tareas de investigación** que permiten satisfacer los objetivos planteados anteriormente y que se pueden resumir en las siguientes:

- Profundizar en las técnicas para la detección de colisiones que puedan proporcionar un mayor grado de realismo al simulador
- Sistematizar en la bibliografía encontrada sobre Simuladores Quirúrgicos
- Buscar algoritmos, tecnologías y tendencias actuales utilizadas en la detección de colisiones.

Para obtener toda la información necesaria que permita llegar al resultado esperado se utilizarán los siguientes métodos teóricos:

Método Teórico:

- Analítico - Sintético: Con el objetivo de analizar documentos para la extracción de los elementos más importantes que se relacionan con el objeto de estudio.
- Histórico – Lógico: Permite estudiar la trayectoria histórica real de los fenómenos, su evolución y desarrollo permitiendo que la investigación pueda constatar teóricamente como ha evolucionado el fenómeno que se estudia en un período de tiempo.

El presente trabajo de investigación está compuesto por tres capítulos. A continuación se le ofrece una breve reseña de cada uno de ellos.

- Capítulo I: Se adentrará en el mundo de los simuladores quirúrgicos, citando definiciones, aplicaciones de los mismos, desarrollo, actualidad, problemas que se han presentado en cuanto a la simulación y entre estos últimos, el que hoy ha llevado a realizar esta investigación *el problema de la detección de colisiones*. Además, se explica de forma breve los algoritmos de detección de colisiones en objetos donde la simetría puede ser definida por un objeto geométrico ya conocido.

- Capítulo II: Se muestra un estudio más profundo de los distintos algoritmos que fueron encontrados. Además, se pondrá a su consideración numerosas técnicas de avanzada a nivel mundial en estos momentos que permiten el desarrollo de las técnicas de detección en Simuladores Quirúrgicos.
- Capítulo III: Se trata de un estudio basado en las ventajas y desventajas de dichos algoritmos y técnicas de detección de colisiones, y se proponen algunos de ellos con el apoyo de los estudios realizados en el capítulo anterior para que sean implementados en el naciente simulador quirúrgico cubano.

Capítulo I: Fundamentación teórica

1.1 Introducción

Este capítulo tiene como objetivo ubicar los principales conceptos sobre el simulador quirúrgico, además de mostrar algunas aplicaciones de estos en diversos campos de la cirugía. Se hará alusión a los avances que se han experimentado en esta tecnología en el mundo y los primeros pasos que ha dado nuestro país en este tipo de simulación para afianzar cada vez más el status de nuestra medicina a nivel mundial. Además, se dará una breve panorámica acerca de la detección de colisiones.

En el mismo se pretende ubicar al lector en el mundo de la simulación quirúrgica y sobre todo en el problema que representa la detección de colisiones para los mismos.

Se aporta, en lo fundamental, un conjunto de ideas que permiten tener una conciencia de las muchas aplicaciones de la simulación quirúrgica, así cómo adentrarlo en el mundo de la detección de colisiones, desde las figuras rígidas hasta el entorno de las complejas.

1.2 Simuladores Quirúrgicos Virtuales

En los últimos años el desarrollo de la informática gráfica ha permitido desarrollar entornos visuales que van a ir permitiendo el entrenamiento de determinadas técnicas quirúrgicas en un ambiente virtual. Un buen simulador quirúrgico debe cumplir una serie de características: mostrar los órganos internos de manera realista, que estos órganos respondan a las interacciones con el cirujano (por ejemplo, deformándose) y que respondan mediante modificaciones estructurales a acciones quirúrgicas habituales como cortes, cauterización y/o sutura. (**Fig. 1 y 2**)

Fig. 1 Ejercicios básicos de coordinación con un simulador virtual.

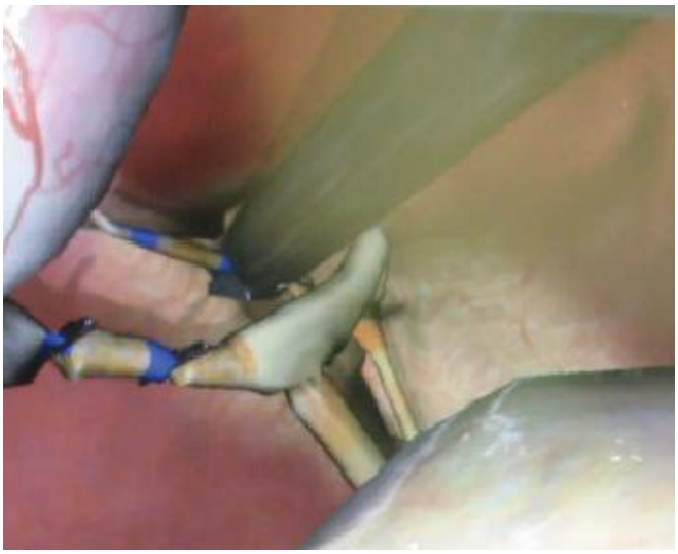


Fig. 2 Ejercicios de corte y disección con simulador virtual

En función de que se consigan estos objetivos, los simuladores quirúrgicos virtuales se agrupan en tres generaciones.

- ***Primera generación****:* Únicamente consideran la naturaleza geométrica de las estructuras anatómicas, pero no nos permiten interaccionar con ellos.
- ***Segunda generación****:* Permiten una cierta interacción con las estructuras anatómicas.
- ***Tercera generación****:* Permiten una interacción con las estructuras y además tienen en cuenta la naturaleza funcional de los órganos.

Actualmente la mayoría de los simuladores virtuales están incluidos en los de ***segunda generación*** y dentro de ellos existen cuatro niveles de complejidad:

- Simuladores de aguja que disponen de objetos visuales simples con un mínimo grado de libertad.
- Simuladores que permiten una exploración endoscópica o instalación de catéteres.
- Simuladores para el entrenamiento en una tarea, que disponen de uno o dos instrumentos que interaccionen con el entorno virtual.

- Simuladores completos que permiten el entrenamiento en determinados procedimientos quirúrgicos laparoscópicos, artroscopia, ginecología o cirugía intraocular.

Fig. 3 Simulador Virtual de Medicina

Además, el simulador quirúrgico incluye dos haptics (interfaces) con realimentación de fuerza, uno por cada mano, proporcionando al cirujano la sensación de tacto durante el entrenamiento quirúrgico. Los instrumentales presentes en la intervención quirúrgica virtual son intercambiables con el objetivo de permitir el reemplazo de un instrumento, como es necesario que ocurra en las intervenciones quirúrgicas normalmente.

Finalmente, una vez terminado el entrenamiento, el cirujano puede obtener un informe en el que se evalúa el ejercicio realizado.

Un sistema de simulación quirúrgica (**SSQ**) estructuralmente se divide en dos subsistemas: el subsistema de interfaz del usuario y el subsistema de sensores.

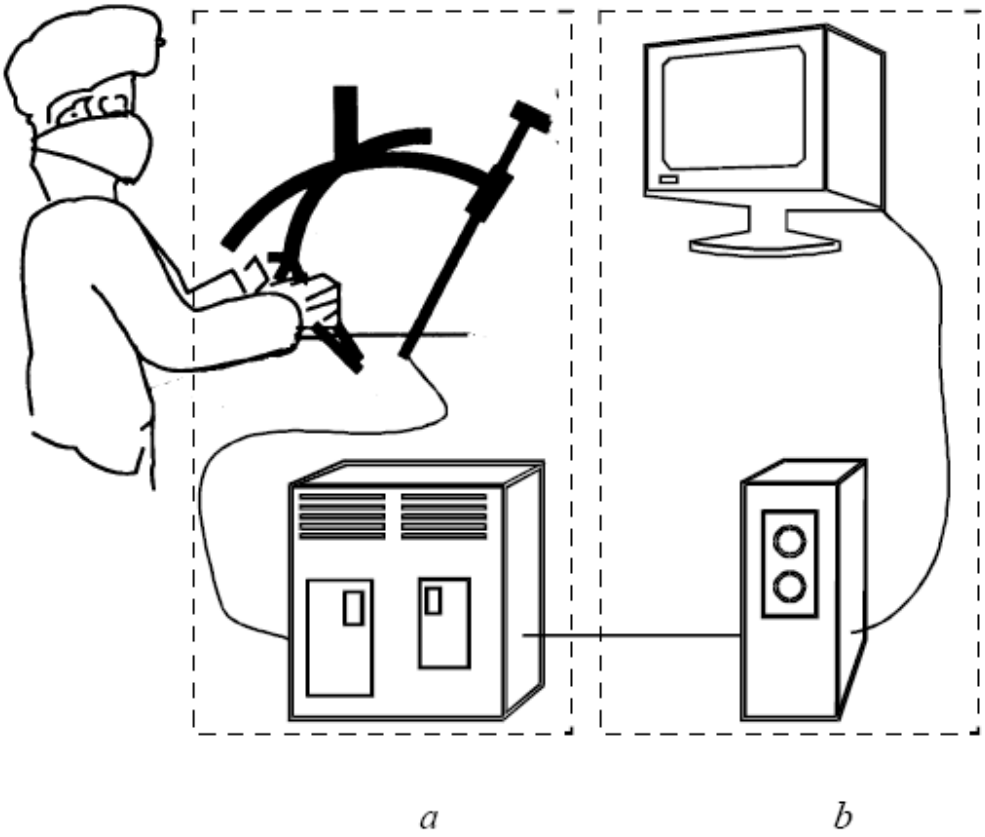


Fig. 4 Esquema de un SSQ: (a) subsistemas de sensores y (b) subsistema de visualización.

1.3 Aplicaciones

“La simulación quirúrgica en Realidad Virtual permite a los cirujanos realizar un entrenamiento con diferentes técnicas de cirugía mínimamente invasiva y repetir una técnica tantas veces como sean necesarias hasta alcanzar un aprendizaje correcto de la misma”. [4]

Un simulador quirúrgico que se usa para la capacitación de la cirugía mínimamente invasiva puede abarcar órganos de animales obtenidos de fuentes éticas, sobre los cuales se realice la perfusión (introducción de líquidos por vía intravenosa) y la práctica. [3]

Las aplicaciones de la realidad virtual prometen ser una revolución de la educación médica. Los cirujanos, para adquirir una destreza gradual en endoscopía, pueden practicar en un quirófano "virtual”, aliviando la

presión ejercida de otros colegas más experimentados que han de supervisarles y entrenarles en pacientes reales. Esto último supone un significativo ahorro de costos al permitir el entrenamiento en el propio centro en lugar de desplazarse a centros endoscópicos acreditados para la formación en endoscopía. [2]

Otra de las aplicaciones de un **SSQ** en la actualidad es la de realizar punciones lumbares donde con una alta tecnología logran que este, tan utilizado método de la medicina, se practique con un menor margen de error.

Los sistemas llamados *"Realidad Aumentada"* superponen a una imagen real otra generada por un ordenador. Esto permite que el cirujano vea sobre la zona de intervención una imagen sintética tridimensional, con indicaciones precisas en un punto de interés especial. Puede ser de gran ayuda en operaciones delicadas, como por ejemplo en el cerebro.

Centrándose el tema en la cirugía, existen pacientes virtuales que adolecen de diversas enfermedades y presentan los síntomas característicos para poner en práctica las habilidades terapéuticas del futuro medico. Una aplicación muy interesante de medicina es la amputación virtual.

Un examen abdominal realizado con un guante de realidad virtual puede ofrecer información vital para un médico especialista localizado lejos del paciente. Una herramienta de realidad virtual permite cuantificar la "medicina" que hay en el tacto de un médico.

Las manos de un médico son dos de los mejores instrumentos de diagnóstico que se pueden tener, y permiten al médico detectar signos sutiles de enfermedad o lesiones solo con tocar al paciente. Hasta ahora, para poder ejercitar esta práctica era necesario que se juntaran el paciente y el médico. Los investigadores de la ***Universidad de Buffalo***, cita en New York, están desarrollando un sistema que permita a los médicos utilizar una nueva forma de realidad virtual, para almacenar información sobre lo que sienten durante el examen de un paciente. Esta información permitirá que cualquier médico pueda repetir el mismo examen sintiendo como si lo estuvieran realizando ellos mismos.

Con este *"Modelo humano virtual para aplicaciones médicas"*, los médicos se pondrían un guante de realidad virtual durante el examen del paciente, que almacenará las sensaciones que tiene el médico mediante unos sensores colocados en las puntas de los dedos del guante.

Fig. 5 Modelo humano virtual para aplicaciones médicas

Otra aplicación encontrada en los **SSQ** es la desarrollada por **LABEIN** (centro de Investigación y Desarrollo bajo contrato), donde se está trabajando en el P*royecto Lahystotrain* en el que participan varias empresas y hospitales de Alemania, Portugal y España. El objetivo del proyecto es desarrollar un sistema de entrenamiento avanzado que combina técnicas de Realidad Virtual, Multimedia y Tutores para entrenamiento de cirujanos que realizan operaciones de Laparoscopia e Hysteroscopia. La Laparoscopia e Hysterscopia son dos tipos de operaciones que utilizan técnicas de cirugía mínimamente invasiva, una práctica quirúrgica que reduce la agresión que supone la cirugía convencional y el tiempo de recuperación del paciente.

El desarrollo de un sistema de entrenamiento altamente realista permitirá adquirir y practicar las habilidades necesarias para los distintos métodos de operación, en especial aquellos de mayor responsabilidad y los que se refieren al tratamiento de complicaciones. En casos frecuentes los sistemas de realidad virtual se convierten en el perfecto aliado del cirujano, puesto que le permitirá entrenarse en un corto periodo de tiempo en un campo que, por la experiencia, en un hospital tardaría varios años.

A diferencia de los sistemas clásicos, el entrenamiento médico simulado basado en *Realidad Virtual* proporciona una interacción intuitiva y muy realista, y permite controlar el procedimiento, introducir dificultades especiales y evaluar el progreso. La información multimedia, con video, audio y tacto mejorará el proceso de entrenamiento y su realismo y permitirá al usuario repetir un procedimiento muchas veces antes de enfrentarse a un paciente real.

La realidad virtual contra el cáncer es otra de las aplicaciones de los **SSQ**. La máquina de realidad virtual crea mediante la computadora el entorno de realidad virtual que simula la fatiga sentida por muchos

pacientes del cáncer. De esta forma se está acortando la distancia entre las percepciones de los médicos y los pacientes del cáncer. La máquina dispone de una silla especial y el usuario es equipado con los guantes y el casco especiales que están conectados electrónicamente con una computadora.

Otro estudio donde se encuentran los **SSQ** es en la endoscopia virtual del sistema ventricular del cerebro humano. En el futuro, se planea combinar el sistema virtual de la endoscopia con un sistema de navegación 3D para proporcionar una conducción perfecta durante el procedimiento real.

1.4 Introducción de los SSQ en Cuba

En Cuba se están dando ya los primeros pasos en tan revolucionaria tecnología. Una forma más de demostrarle al mundo que somos una potencia en la medicina y que cada día estamos en camino al desarrollo. La Telemedicina y la globalización de la información tendrán un impacto sustancial en la docencia y en la formación continuada.

En las próximas décadas se desarrollarán simuladores quirúrgicos en los laboratorios de entrenamiento; la robótica y los endonavegadores permitirán ser extremadamente precisos en las técnicas de la cirugía y las necesidades de formación en procesos repetitivos se harán innecesarias gracias al "telementoring" (método basado en el uso de las telecomunicaciones e informática ya sea por audio o video para dar la dirección o instrucción individual).

Se han informatizado los hospitales de la Misión Milagro, lugares donde son asistidos cientos de Latinoamericanos con diferentes afectaciones visuales, esto es posible debido a un amplio programa de colaboración entre los gobiernos de Cuba y Venezuela en el marco de los programas de integración Latinoamericana conocido como ALBA.

Están en funcionamiento 76 laboratorios y aulas de computación para el desarrollo de un majestuoso proyecto en la formación de nuevos galenos en nuestro país con el uso de las nuevas tecnologías de la información y las comunicaciones, conocido como ***Nuevo Proyecto Policlínico Universitario*** y que está en marcha desde el curso 2004/2005. Los estudiantes que se están formando con este programa gozan de múltiples ventajas, dentro de las cuales podemos mencionar el poseer laboratorios de computación donde la relación computadora-estudiante, es de uno a uno, o sea, tienen su propia máquina disponible durante las 24 horas del día para su uso, tienen acceso a la Intranet nacional del Ministerio de Salud Pública (MINSAP), así como a Internet. Disponen de aulas para teleconferencias equipadas con modernos

equipos. El programa de estudio que se desarrolla con estos grupos de todas las especialidades que lo integran está sobre la base del uso eficiente y continuo de las TICs.

Las clases teóricas se desarrollan a través de teleconferencias haciendo uso del equipamiento disponible y las mayorías de las clases teórico-práctica o prácticas la realizan en los laboratorios de computación haciendo uso de aplicaciones multimedia, tutoriales, simuladores, etc., que permiten la simulación de experimentos de laboratorios, el adiestramiento clínico, además de utilizar eficientemente estos medios para la enseñanza del Inglés, la Filosofía, la Informática Médica, etc.

1.5 Detección de colisiones

Dentro de la simulación quirúrgica hay muchos problemas que se plantean, debido a que no resulta fácil imitar a los órganos, tejidos y otras partes del cuerpo humano, de manera tal que se logre la mayor exactitud.

Uno de esos problemas a los que se hace alusión anteriormente es, sin dudas, la simulación de las colisiones, pues es de vital importancia tener conciencia de la exactitud de la incisión, por solo citar un caso.

La detección de colisiones es un conjunto de algoritmos matemáticos que permiten simular el comportamiento de los objetos en la realidad.

1.5.1 Detección de colisiones en modelos rígidos

El algoritmo básico para detectar las colisiones entre elementos consiste en comprobar las posibles intersecciones entre cada par de polígonos de estos elementos. Esta solución resulta sencilla e intuitiva, sin embargo, supone un elevado coste computacional.

Controlar el tiempo del paso de simulación (tiempo de ejecución) es fundamental para alcanzar el deseado funcionamiento en tiempo real. Puesto que la detección de colisiones es una de las etapas del bucle de simulación, que se ejecuta justo después que comienza el mismo y que se hacen las lecturas de los instrumentos que interactúan en el ambiente virtual (órganos o tejidos y herramienta quirúrgica), debemos buscar estrategias que permitan disminuir el tiempo requerido para determinar las colisiones. Por este motivo, el algoritmo básico mencionado en el párrafo anterior no puede ser utilizado en la práctica.

La detección de colisiones se realiza una vez en cada iteración del bucle de simulación, es decir, cada cierto tiempo (*tsim).* Como consecuencia, si una colisión se produce en un instante intermedio no será detectada, hecho que ocurre en el caso que una acción ocurra muy rápidamente.

Seguidamente se describen algunas estrategias para disminuir el coste computacional del algoritmo basadas en el concepto de envolvente de objeto o volúmenes de inclusión.

> ***Envolvente de objeto o volúmenes de inclusión****: con el objetivo de simplificar los procesos de detección de colisiones entre cada par de objetos es práctica habitual aproximar la forma de los mismos por geometrías sencillas que contengan completamente al objeto.*

La idea es, pues emplear dos modelos de objeto, uno para la detección de colisiones y otro para la representación gráfica del mismo en aras de un realismo visual, el cual deberá ser más cercano a la realidad y por ende más complejo.

De esta forma, en lugar de comprobar las colisiones entre cada par de polígonos de los elementos de la escena, se comprueba sólo si hay intersección entre las envolventes puesto que éstas tienen formas sencillas, esta operación es rápida y poco costosa computacionalmente. Si no hay colisión entre las envolventes podemos asegurar que tampoco la hay entre los objetos. En cambio, el hecho de que se detecte colisión entre las envolventes de objeto no implica que necesariamente la haya entre los objetos.

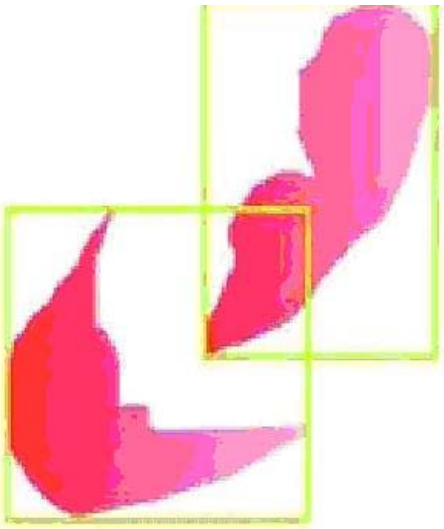


Fig. 6 Colisión entre dos envolventes de objeto y no entre los objetos.

Para solucionar esto se plantean estrategias basadas en jerarquías de volúmenes de inclusión.

Puesto que las colisiones no se comprueban continuamente, sino que sólo se calculan en un instante concreto (en cada paso de simulación), no se detectarán las colisiones que se produzcan en otro momento del bucle. Dependiendo de la aplicación en la que se englobe la detección de colisiones y del tiempo *tsim* este problema tendrá una mayor o menor relevancia.

Una posible solución a este problema son las *envolventes de movimiento*, la filosofía es similar a la de las *envolventes de objeto*. Se trata de construir una superficie o volumen que cubra el espacio ocupado por el objeto durante el intervalo comprendido entre dos invocaciones al módulo de detecciones de colisiones consecutivas en el bucle. En la figura 7 se muestra la envolvente de movimiento de la herramienta. En este caso sí se detectaría la colisión entre dicha envolvente y el órgano.

Fig. 7 Detección de la colisión entre la envolvente y el borde de la esfera.

Es importante construir adecuadamente la envolvente de manera que se obtenga una representación aproximada del movimiento real del objeto con una forma geométrica relativamente simple. Además, es necesario alcanzar un compromiso entre el ajuste preciso de la envolvente a los objetos y la simplicidad de la misma, de la cual depende el coste computacional del algoritmo.

Pero la colisión entre *envolventes de movimiento* no implica necesariamente la colisión entre los objetos. Para evitar que se produzcan colisiones que no sean detectadas, las *envolventes de movimiento* deben englobar, mediante una geometría simple a las posiciones ocupadas por el objeto entre los dos instantes en los que se lleva a cabo el movimiento.

Si la geometría de la envolvente es excesivamente sencilla es probable que la aproximación al verdadero movimiento del objeto sea demasiado grosera y que existan grandes zonas del interior de la misma que no se correspondan con puntos por los que ha pasado realmente el objeto. Por lo tanto, si las envolventes colisionan en estas zonas, se detectarían falsas colisiones entre los objetos.

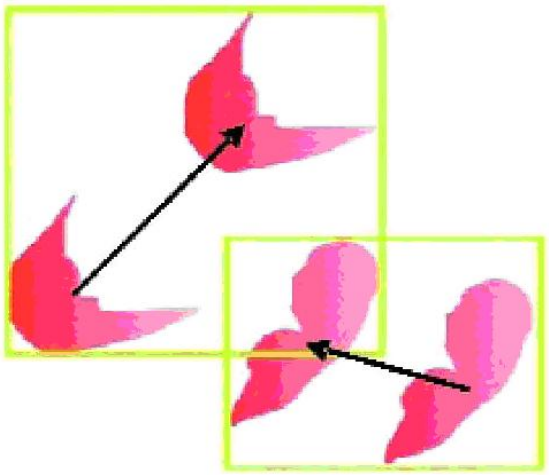


Fig. 8 Envolventes de movimiento de dos objetos; en este caso existe colisión entre las envolventes no sucediendo así entre los objetos.

Otro caso de detección de falsa colisión es el que se muestra a continuación, donde las trayectorias de los objetos se cruzan, pero los objetos no comparten en ningún instante de tiempo la misma posición espacial.

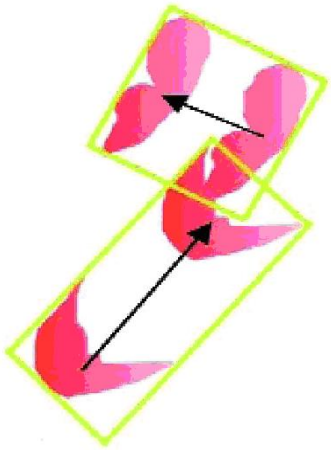


Fig. 9 Detección de falsa colisión.

1.5.2 Algunas características de las envolventes

La geometría elegida influye directamente en el grado de aproximación al objeto que se consiga. Por ejemplo, el ajuste obtenido mediante esferas es generalmente peor que el que se alcanza mediante cubos, por supuesto que esto depende de la forma del objeto al que se esté aproximando. Si los cubos están orientados según el objeto, el ajuste es aún mejor. Para una mejor aproximación a objetos de formas irregulares se suelen emplear los árboles o jerarquías de volúmenes de inclusión.

Se debe añadir que estos árboles jerárquicos de volúmenes de inclusión son fáciles de actualizar para modelos rígidos, contrariamente para el caso de modelos deformables, ya que una deformación puede implicar la necesidad de recalcular toda la estructura jerárquica con la consiguiente pérdida de eficiencia.

1.6 Comentarios parciales

Como se pudo ver en el transcurso del capítulo, se mostraron los adelantos y las ventajas que nos brinda la simulación quirúrgica en nuestros días, cada minuto que pasa nos depara un nuevo adelanto científico-técnico. Se quiere además, hacer una consideración de porqué se piensa que el problema que hoy se muestra, la simulación de colisiones en objetos deformables, no es posible resolverlo con los métodos de detección de colisiones para objetos rígidos.

La detección de colisiones mediante envolventes rígidas, ya sea de objeto o de movimiento, presentan un problema en la detección y es que se pueden producir falsas colisiones entre los objetos, o sea, que

puede que la geometría de la envolvente sea excesivamente sencilla, que la aproximación al verdadero movimiento del objeto sea demasiado grosera y que existan grandes zonas del interior de la misma que no se correspondan con puntos por los que ha pasado realmente el objeto. Por lo que si las envolventes colisionan en estas zonas se detectarían falsas colisiones.

Estos algoritmos no dan solución al problema de la detección de colisiones en objetos deformables en los simuladores quirúrgicos debido a que un pequeño error en el software provocaría grandes problemas para el aprendizaje del cirujano y lo que se desea es que estos **SSQ** sean lo más cercano a la realidad, para que el estudiante o doctor que interactué con el simulador, una vez que se enfrente en la vida real a un problema, logre resolverlo de manera eficaz con las experiencias ya adquiridas con el uso del simulador quirúrgico.

Por tanto, se plantea la necesidad de sistematizar la literatura relativa a los algoritmos de detección de colisiones para objetos deformables en el marco de la cirugía, logrando que con este estudio las bases queden sentadas, para que en un futuro se cuente con un simulador que cada vez se acerque más al fascinante mundo del cuerpo humano.

Capítulo II: Estudio de los Algoritmos de Detección de Colisiones

2.1 Introducción

Este capitulo tiene como objetivo realizar un estudio profundo de algunos algoritmos de detección de colisiones en objetos deformables permitiendo que queden sentadas las bases para futuros proyectos que se le planteen a la universidad referente a la simulación quirúrgica.

Se profundiza en los conocimientos en cuanto a esta rama se refiere, además de proporcionar un detallado seguimiento de todo lo necesario para simplificar los costes computacionales y el tiempo de ejecución en el momento de la simulación de la colisión, para que así sea más real y confiable el simulador. Además, producto del mismo estudio que se realizara en cuanto a algoritmos se refiere, se dará una propuesta de cual se entiende que sea el más eficaz para ser utilizado en nuestra universidad teniendo en cuenta todos los problemas y limitaciones que se tienen producto del bloque económico.

A continuación se explicará brevemente cómo es que funciona un simulador quirúrgico y cuán importante es la parte de la detección de colisiones para que su funcionamiento tenga gran calidad y éxito.

2.2 Funcionamiento de un simulador quirúrgico

La simulación de cirugía, como cualquier otro tipo de simulación, consiste en la ejecución repetitiva de un determinado algoritmo. Como se muestra en la siguiente figura, en cada uno de los ciclos se ha de leer la posición de los instrumentos de interacción, detectar las posibles interacciones de cada uno de los instrumentos respecto a los otros y de los objetos presentes en el entorno, calcular la respuesta física de los tejidos a dichas interacciones y la fuerza de realimentación que dicha deformación genera, dibujar en 2D la escena resultante y realimentar las fuerzas que ejercen los dispositivos de interacción en función de las fuerzas de realimentación obtenidas Todo esto se tiene que llevar a cabo en menos de 50 milisegundos para el refresco visual y en menos de 3.3 milisegundos en el refresco de fuerzas de los dispositivos de interacción para cumplir los requerimientos de realismo [4].

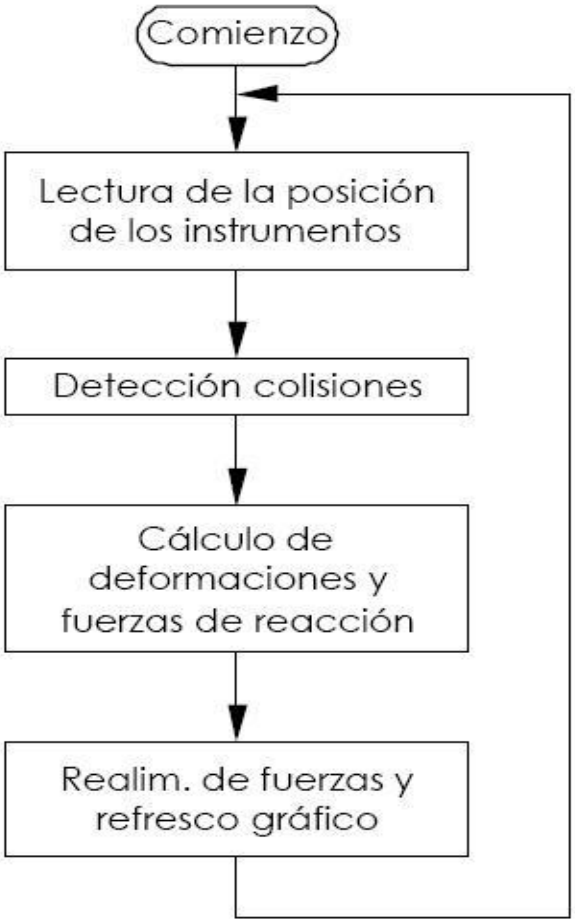


Fig. 10 Ciclo básico de simulación quirúrgica

La detección de colisiones, por otro lado, es una tarea compleja en sí misma. Algunas de las técnicas desarrolladas están orientadas a la detección de colisiones entre objetos volumétricos y utilizan mapas de ocupación 3D del espacio disponible, el cual se encuentra discretizado bien de forma fija [6] o variable [7]. Estos métodos son capaces de detectar cualquier tipo de posible colisión, aunque a un coste computacional excesivamente elevado (varios centenares de milisegundos para objetos de tamaño razonable), incluso cuando únicamente se considera la parte externa de los objetos, estos tiempos no pueden ser reducidos a menos de una decena de milisegundos [6].

Otros métodos de detección de colisiones están orientados a la representación de objetos mediante superficies poligonales. Para reducir al mínimo el número de comprobaciones de colisión entre superficies, se suelen utilizar cajas de inclusión de modo que únicamente se comprueban los pares de caras cuyas cajas de inclusión intersectan [8]. Ahora bien, muchos de estos métodos únicamente permiten la detección de colisiones entre objetos convexos y rígidos, no permiten autointersecciones y cambios en la topología del objeto, lo que implica la repetición de costosos precálculos. Para evitar este tipo de problemas, en simulación quirúrgica normalmente se propone la utilización del método de cajas de inclusión simplificado.

Este método únicamente calcula la intersección de órganos simples deformables con instrumentos rígidos (instrumental quirúrgico) que vienen representados mediante un único punto en su extremo. Esta técnica únicamente requiere de 3 milisegundos para detectar la posible colisión entre un objeto deformable de razonable tamaño y el instrumental quirúrgico [9].

2.3 Detección de colisiones para objetos deformables

La detección de colisiones para objetos deformables es un componente esencial en la interactividad, físicamente basada en la simulación y animación, lo cual permite un rápido crecimiento de la investigación en estas aéreas con un incremento del número de aplicaciones interesantes. La siguiente figura ilustra una de las mejores aplicaciones para el descubrimiento de colisión deformable: *"la simulación de paño"*. En la misma, se hacen acercamientos (aproximaciones) producto de deformar la ropa dinámicamente, esto tiene que ser combinado con algoritmos eficaces que se ocupan de las autocolisiones.

Fig. 11 Simulación interactiva de paño.

La simulación de la cirugía mostrada en la siguiente ilustración es una segunda área de aplicación importante para la detección deformable de la colisión. En tales ambientes, las colisiones entre órganos deformables tienen que ser detectadas y resueltas. Además, las colisiones entre las herramientas quirúrgicas y el tejido fino deformable tienen que ser procesadas. En la caja de cambios topológicos debido al corte, las auto colisiones del tejido fino pueden ocurrir y por tanto tienen que ser dirigidas. Puesto que el comportamiento interactivo de los ambientes de la simulación de la cirugía es esencial, se requieren algoritmos eficientes para la detección de colisiones en objetos deformables.

Fig. 12 Interacción de tejidos finos deformables y herramientas quirúrgicas

Si hacemos una comparación aprisionada para los objetos rígidos encontraríamos varios aspectos que se complicarían en el caso de objetos deformables.

Collisions and Self-collisions: Para simular interacciones realistas en objetos deformables, todos los puntos de contacto incluyendo aquellos que resultan de las autocolisiones tienen que ser considerados. Esto está en contraste con la detección de colisiones en cuerpos rígidos, donde las autocolisiones se descuidan comúnmente.

Pre-processing: Los algoritmos eficientes en la detección de colisiones son acelerados por las estructuras de datos espaciales incluyendo jerarquías basadas en volúmenes de inclusión, campos de distancia, o alternativas de repartir espacialmente los elementos. Tales representaciones del objeto se construyen en una etapa del proceso de simulación y se realizan comúnmente eficientemente para los

objetos rígidos. Sin embargo, en el caso de modelos deformables se torna mucho mas complicado debido a que estas estructuras, tienen que actualizarse con frecuencia. Por lo tanto, estas estructuras son menos eficientes para deformar objetos y su sentido práctico tiene que ser examinado cuidadosamente.

Collision Information: Los algoritmos de detección de colisiones para los objetos deformables tienen que considerar que una respuesta realista de la colisión requiere de la información apropiada. Por lo tanto, no es suficiente con apenas detectar la interferencia de objetos.

Performance: En usos interactivos, tales como simulación quirúrgica, juegos, y simulación del paño, la eficacia de los algoritmos en la detección de la colisión para los ambientes deformables que se modelan es particularmente importante. La interactividad es una de las características dominantes en estos usos, dando por resultado las altas demandas para la eficacia que computa de los algoritmos en la detección de la colisiones.

En este capitulo, se discuten los acercamientos de la detección de la colisión que tratan los problemas anteriormente mencionados para resolver los requisitos de los ambientes de la animación y de la simulación para objetos dinámicamente deformables. Aunque todos los algoritmos discutidos son apropiados para objetos deformables, no se restringen a ellos, sino que también trabajan con cuerpos rígidos.

2.3.1 Estructuras de datos utilizadas en la detección de colisiones

Las jerarquías de volúmenes contenedores (BVHs) han demostrado estar entre las estructuras de datos más eficientes para la detección de colisiones. Sobre todo, se han aplicado a la detección de las colisiones de cuerpos rígidos. Generalmente, una BVH se construye para cada objeto, en un paso del proceso anterior, antes de comprobar si hay detección de colisiones. La idea de la BVHs es repartir el sistema de primitivos del objeto recurrentemente hasta que se resuelve un cierto criterio de la hoja. Los primitivos son las entidades que muestran la trayectoria de los objetos gráficos, que pueden ser polígonos, remiendos de NURBS, etc. desde el nodo principal hasta la hoja.

En general, se define BVHs como cada nodo en el árbol que se asocia a un subconjunto de los primitivos del objeto, junto con un BV (cajas contenedoras) que incluya este subconjunto con un caso que contiene más pequeño de una cierta clase especificada de formas.

Una de las opciones del diseño con los árboles de BV es el tipo de BV. En el pasado se exploró una abundancia de los tipos de BV, por ejemplo: esferas, OBBs, DOPs, Boxtrees, AABBs, cáscaras esféricas y cascos convexos

Aunque una variedad de BVs se ha propuesto, dos tipos merecen la mención especial: OBBs y k-DOPs. OBBs tiene la característica de que, bajo ciertas determinaciones, su tirantez aumenta linealmente mientras que el número de polígonos disminuye. Los k-DOPs, por otra parte, se pueden hacer para aproximar el casco convexo arbitrariamente aumentando k. Además, los k-DOPs especiales tales como el 26-DOP se pueden construir y poner al día muy eficientemente.

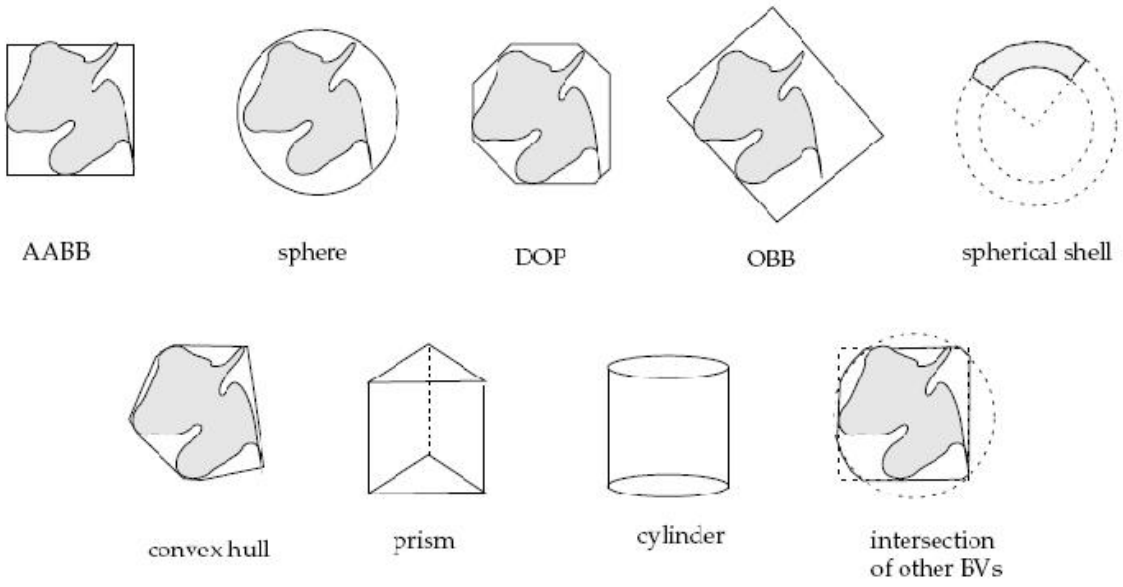
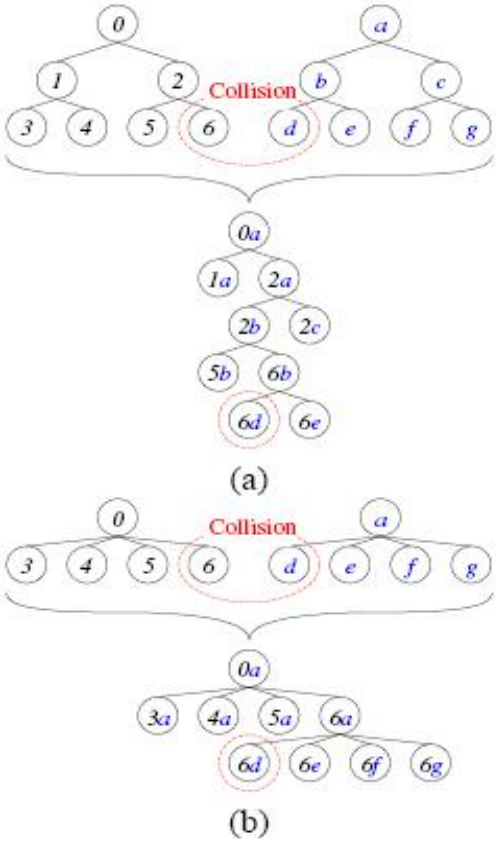


Fig. 13 Variedades de BVs

Para la prueba de la colisión de dos objetos o la prueba de la colisión de un mismo objeto, el ***BVHs*** la realiza de arriba hacia abajo atravesado y los pares de nodos del árbol se prueban recurrentemente para la superposición. Si los nodos superpuestos son hojas, entonces los primitivos incluidos se prueban para la intersección. Si un nodo es una hoja mientras que el otro es un nodo interno, el nodo de la hoja se

prueba contra cada uno de los hijos del nodo interno. Si ambos nodos son nodos internos se intenta reducir al mínimo la probabilidad de la intersección, tan rápido como sea posible. Por lo tanto, prueba el nodo con el volumen más pequeño contra los hijos del nodo con el volumen más grande.

Para dos objetos dados con el ***BVHs***: A y B, la mayoría de los algoritmos de la detección de colisiones ponen el esquema general siguiente del algoritmo en ejecución:



traverse (A, B)

if A and B do not overlap **then**

return

end if

if A and B are leaves **then**

return intersection of primitives enclosed by A and B

else

for all children A[i] and B[j] **do**

traverse (A[i], B[j])

end for

end if

Fig. 14 Mecanismo recursivo que usan los árboles binarios

El algoritmo que se muestra en la figura anterior se enfoca rápidamente dentro de pares de polígonos próximos. Nótese que se mezclaron cajas donde está una hoja y un nodo y se omite un nodo interno. Las características de diversos algoritmos jerárquicos para la detección de colisiones se sitúan en el tipo de ***BV*** usado, la prueba de superposición para un par de nodos y el algoritmo para la construcción de los árboles de ***BV***.

2.3.2 Construcción de BVHs

Para los cuerpos rígidos la meta es construir ***BVHs***, tales que todas las preguntas subsecuentes de la detección de colisiones se pueden contestar tan rápido como sea posible en promedio. Tal ***BVHs*** se le llama bueno en el contexto de la detección de colisiones.

Con los objetos deformables, la meta principal es desarrollar los algoritmos que pueden poner al día o reinstalar rápidamente el ***BVHs*** después de que haya ocurrido una deformación. Al principio de una simulación, un buen ***BVH*** se construye para no deformar el objeto apenas se edifica como para cuerpos rígidos. Entonces, durante la simulación la estructura del árbol se guarda generalmente, y solamente los grados del ***BVs*** son actualizados, puesto que para DOPs es generalmente más rápido ponerse al día con los objetos deformable, se prefiere OBBs.

Existen tres diversas estrategias para construir ***BVHs***, a saber: *de arriba hacia abajo*, *bottom-up* y *la inserción*. Sin embargo, la estrategia *de arriba hacia abajo* es la más comúnmente utilizada en la detección de colisiones. La idea de la construcción *de arriba hacia abajo* es partir recurrentemente un sistema de primitivos del objeto hasta que se alcanza un umbral. Esta partición es dirigida por un criterio usuario-especificado o heurístico que rinda buen ***BVHs*** con respecto al criterio elegido.

En el contexto de la simulación quirúrgica han aparecido varios métodos de detección de colisiones. En el caso de las técnicas computacionales empleadas en la simulación de cirugías para tiempos reales con realimentación de fuerza [18], se introduce el concepto de predicción de colisiones para disminuir el tiempo de detección de estas y, por tanto, permite aumentar el tiempo destinado a las deformaciones de los modelos.

Otro algoritmo es el propuesto en el artículo *"Real-time Collision Detection for Virtual Surgery"* [19], el cual aprovecha las características del hardware de representación gráfica para calcular la colisión entre la herramienta y los modelos de la escena en el recorrido del instrumental en dos pasos de simulación

consecutivos, obteniendo resultados muy eficientes. En relación con los algoritmos basados en hardware gráfico, se han propuesto algoritmos que alcanzan mayor rapidez que los tradicionales pero con limitaciones. Por ejemplo, *RECODE* [20], diseñado para objetos convexos, es más lento en el caso de modelos con pocos polígonos.

Un segundo ejemplo es el propuesto en *"Practical collision detection in rendering hardware for two complex 3D polygon objects"* [21], el cual no es capaz de detectar qué polígonos colisionan. Por lo tanto, se debe recurrir a algoritmos clásicos si se requiere esa información.

En el proyecto *Sinergia*, se optó por esta segunda opción y es que en un instante de cada ciclo de simulación, se considera "congelado" el estado del sistema como si fuera una fotografía, y a partir de ahí se realiza una detección de colisiones con métodos tradicionales adaptados y optimizados para un mundo 3D. De esta forma puede despreocuparse de si un determinado agente es rígido o es deformable, ya que en ese fotograma todos los agentes son considerados rígidos.

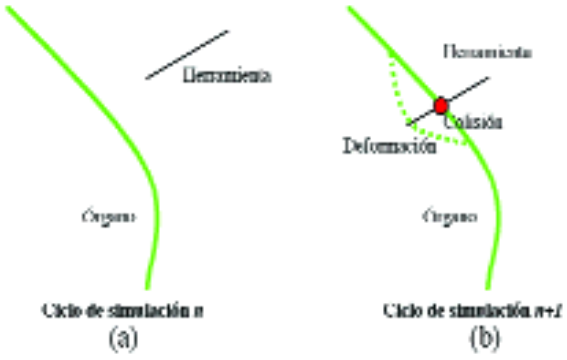


Fig. 15 Fotograma del simulador (a) instante de invocación del detector de colisiones en un ciclo de simulación n (b) invocación del detector de colisiones en el instante n+1.

Tal y como se muestra en la figura, en el instante de invocación del detector de colisiones en el ciclo n se "congela" el estado del simulador, se calculan las colisiones, y se detecta que no ha existido colisión

ninguna. En el ciclo de simulación n+1, se vuelve a “congelar” el estado del simulador, se calculan las colisiones y se detecta que hay una colisión, con lo que se genera una lista de puntos que transcriben las restricciones que se imponen a la geometría del órgano. A continuación y en el mismo ciclo, se deformará el órgano para adaptarse, según su modelo biomecánico, a la nueva geometría impuesta.

Es muy importante tener en cuenta dos detalles que avalan este planteamiento:

- La visualización de la escena no se lleva a cabo hasta el final de cada ciclo, con lo que los efectos visuales “anormales” que puedan aparecer debidos a este tipo de actuación no son visibles en ningún caso, ej. introducción de alguna herramienta en un órgano.

- Se supone que la frecuencia promedio de ciclos es de 25Hz (restricciones conocidas para que exista realismo visual en el simulador) por lo que las variaciones en la escena entre dos ciclos consecutivos son muy pequeñas, y por tanto las “anomalías” visuales que pudieran aparecer debido a la falta de continuidad tienen un efecto despreciable.

2.4 Algunas técnicas básicas de detección de colisiones

2.4.1 Fuerza Bruta

La fuerza bruta consiste en comprobar todos los pares de polígonos posibles para determinar si hay o no colisión entre ellos.

Ventajas: Es posible detectar tanto las colisiones entre objetos distintos como las de un objeto consigo mismo. Esto es importante para simular nudos en las suturas.

Inconvenientes: Requiere de una gran cantidad de cálculo, lo que le hace poco aplicable a la simulación quirúrgica puesto que no permite, en general, el funcionamiento del sistema en tiempo real.

2.4.2 Volúmenes de inclusión (Boundary Volumes)

Se definen volúmenes de geometría simple en los que se inscriben los objetos de interés y se determina si hay colisión entre los volúmenes. En el caso en que la haya, se aplica el algoritmo de fuerza bruta a los polígonos contenidos en el volumen.

Ventajas: Es muy rápido y discrimina de manera sencilla cuándo dos objetos no entran en colisión.

Inconvenientes: Requiere gran cantidad de cálculo cuando se detecta colisión entre las cajas de inclusión o en la actualización de los volúmenes en presencia de deformaciones.

2.4.3 Esferas de Inclusión

Los objetos se inscriben en esferas y se comprueba si éstas colisionan.

Ventajas: La intersección entre dos esferas es muy sencilla y rápida de calcular, simplemente debe compararse la distancia entre los centros con la suma de los radios de ambas esferas. Si la distancia es mayor que la suma de los radios no existe contacto entre ellas; caso contrario, sí existe.

Inconvenientes: Cuando los objetos son planos, o alargados y delgados, esta técnica produce muchos falsos positivos, es decir, aunque haya intersección entre las esferas no existe tal entre los objetos. Es necesario recalcular las cajas cuando se produce una deformación.

2.5 Métodos utilizados en la detección de colisiones

2.5.1 Algoritmos de Lin-Canny y V-Clip

Los algoritmos de Lin-Canny [22] y V-Clip [23] sirven para objetos poliédricos y convexos. Las colisiones se detectan únicamente en base a vértices, aristas o caras. El algoritmo de Lin-Canny mantiene un par con los elementos más cercanos (vértices, aristas o caras) entre dos poliedros convexos que se mueven a través del espacio. La distancia entre dos poliedros se tiene fácilmente una vez que se conozcan los elementos más cercanos. Se declara una colisión cuando esta distancia se encuentra por debajo de un cierto parámetro. Explotando el hecho de que los elementos más cercanos en un instante de tiempo dado están probablemente muy próximos a los elementos más cercanos del instante anterior, en la práctica esto se traduce en una reducción del tiempo de búsqueda.

2.5.2 Diagramas de Voronoi

Los diagramas de Voronoi son una de las estructuras fundamentales dentro de la geometría computacional, puesto que almacenan toda la información referente a la proximidad entre puntos

La idea este diagrama es construir, a partir de un número finito de puntos en el plano, planos en el lugar donde equidisten dichos puntos formando una malla.

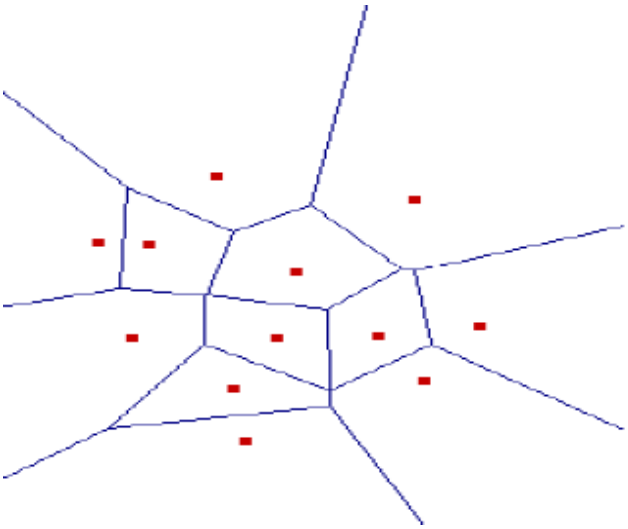


Fig. 16 Diagrama de Voronoi

2.5.3 GJK y GJK mejorado

GJK y GJK mejorado [24] fueron desarrollados por Gilbert, Jonson y Keerthi. Se basan en programación lineal para determinar la distancia euclídea entre dos objetos convexos. Para ello definen los objetos mediante la envolvente convexa de sus vértices y el conjunto convexo se representa según sus propiedades de soporte.

Gracias a esta representación es posible eliminar muchos de los cálculos de distancias entre las caras de los poliedros convexos, con lo que se consigue calcular las distancias más eficientemente. Nótese que muchos órganos no son convexos y si lo fueran al sufrir deformación perderían esa propiedad. Este algoritmo, en laparoscopia, no alcanza su máxima eficiencia.

2.5.4 Árboles OBB

El cálculo del OBB óptimo es bastante más complicado que el del AABB (es básicamente un problema de minimización). Debido a que el tiempo de cálculo es relativamente grande, se suelen obtener al principio del programa cuando se carga cada uno de los objetos, o directamente se cargan desde el fichero donde están los modelos.

Después de esta breve panorámica acerca de algunas técnicas y algoritmos, estos se explicarán con un poco más de detalle.

Se presentó un método simple para realizar en tiempo real la detección de colisión en un ambiente virtual de la cirugía. El método confía en el hardware de los gráficos para probar la interpenetración entre un órgano deformable virtual y uno rígido, herramienta controlada por el usuario. El método permite tomar en consideración el movimiento de la herramienta entre el tiempo consecutivo en dos pasos. Para nuestro uso específico, el nuevo método funciona cientos veces más rápido que el límite orientado bien conocido encaja un método del árbol.

Así, la idea básica de nuestro método es especificar un volumen de la visión que corresponde a la forma de la herramienta (o alternativamente al volumen cubierto por la herramienta entre dos pasos consecutivos del tiempo). Utilizamos el hardware para "dibujar" el objeto principal (el órgano) relativamente a esta "cámara fotográfica". Si nada es visible, después no hay colisión.

Varios problemas ocurren; en primer lugar, la forma de la herramienta no es tan simple como volúmenes generalmente de la visión. En segundo lugar, no se desea conseguir una imagen, sino que necesitamos la información significativa. Más exacto, se quiere saber si las caras implican una colisión, y en cuáles coordinan.

2.6 Algoritmos encontrados para el tratamiento de las colisiones en objetos complejos

2.6.1 Volúmenes de la visión

El frustum (zona final de la escena tomada), más común proporcionado por OpenGL, definidos por una cámara fotográfica orthográfica y por una cámara fotográfica de la perspectiva. En ambos casos, los volúmenes que ven son hexahedra, respectivamente una caja y una pirámide truncada, especificadas por seis valores escalares.

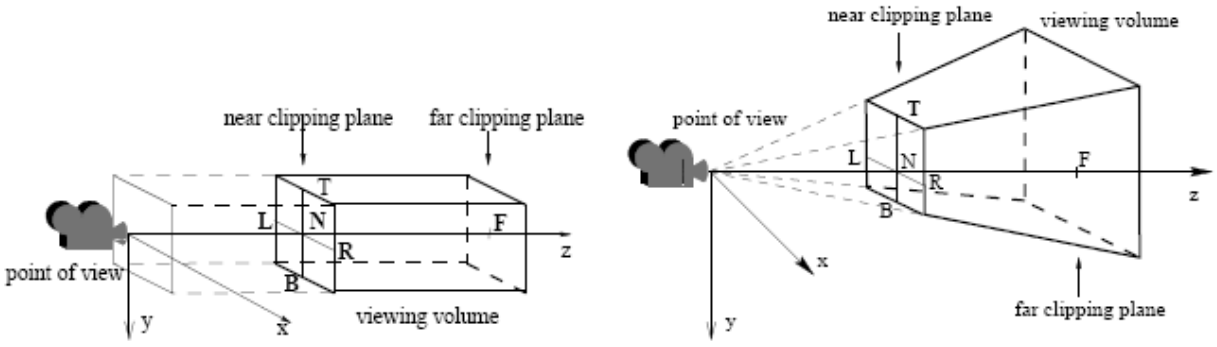


Fig. 17 La cámara fotográfica orthográfica de OpenGL (izquierda) y la cámara fotográfica de la perspectiva de OpenGL (derecha).

Los volúmenes de la visión que son una caja o una pirámide truncada, son caracterizados por las distancias entre los planos del truncamiento (el plano mas cercano y el mas lejano) y por los dos intervalos (izquierda y derecha) y (tope y fondo) que definen su sección en el plano cercano del truncamiento.

2.6.2 Detección de colisiones tipo estática

Las herramientas para la cirugía de laparoscopia se pueden ver como cilindros de sección constante y de longitud variable, puesto que el usuario puede tirar o empujar de ellos sobre el abdomen del paciente. En este documento, se le llama ***P0*** al centro de la abertura pequeña por la que la herramienta atraviesa y ***P*** al extremo de la herramienta. La detección estática de colisiones entre una herramienta y el acoplamiento poligonal que representa al órgano puede ser realizado fácilmente asociando una cámara fotográfica orthográfica a la herramienta.

En verdad, aquí se muestra una solución simple que consiste en probar la colisión existente entre una posición estática de la herramienta y el paso del órgano en un momento dado. Esto sufre de los defectos clásicos de la discretización del tiempo: si las manos del usuario se mueven rápidamente, la herramienta puede penetrar profundamente dentro del órgano antes de ser repelido. Puede incluso cruzar una parte fina del órgano sin que ninguna colisión sea detectada.

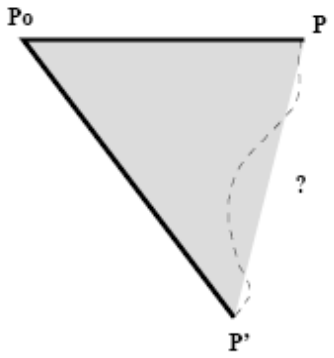


Fig. 18 Movimiento de la herramienta entre dos pasos de la simulación

Para evitar estos problemas, se presenta una extensión que considere el volumen cubierto por la herramienta durante un paso del tiempo (todavía se descuidan las deformaciones dinámicas del órgano durante este período). En este modelo, la herramienta pasa a través de la pared abdominal del paciente en el punto fijo ***P0*** y puede resbalar a través de este punto, así que su longitud varía en un cierto plazo. Se asume que la extremidad activa de la herramienta sigue una línea recta ***P, P'***. El área cubierta por el eje de la herramienta es así: el triángulo ***P0, P, P'***.

Puesto que la herramienta se puede ver como un cilindro, el volumen cubierto por la herramienta durante el intervalo de tiempo es obtenido agrandando y espesando el triángulo por la distancia.

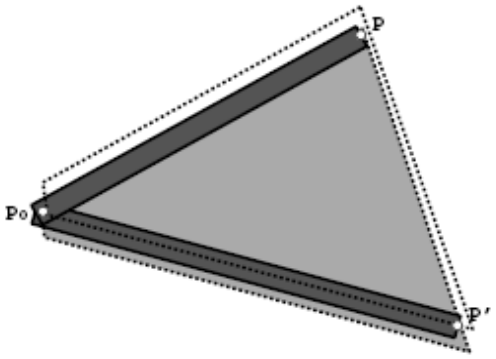


Fig. 19 Movimiento de la herramienta entre dos pasos de la simulación.

Este es un método simple y muy eficiente para detección de colisiones entre un modelo poligonal general y una o varias herramientas cilíndricas. Debido a su funcionamiento, el método es directamente aplicable en el contexto de un simulador en tiempo real de la cirugía. Además, es extremadamente fácil de poner en ejecución (solamente pocas líneas de códigos en uso con OpenGL para la visualización), es portátil (OpenGL existe en la mayoría de las plataformas) y ofrecen beneficios para los diferentes hardwares gráficos.

2.6.3 AABB árboles y modelos deformables

Los árboles de ***AABB*** se prestan fácilmente para su utilización en los modelos deformables. Un ejemplo típico de un modelo deformable es un acoplamiento del triángulo en el cual las coordenadas locales de los vértices son dependientes del tiempo.

En vez de reconstruir el árbol después de una deformación, por lo general es mucho más rápido reinstalar las cajas en el árbol.

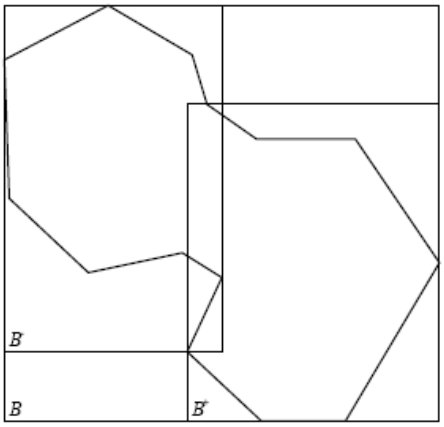


Fig. 20 El AABB más pequeño de un sistema de primitivas incluye al AABBs más pequeño de los subconjuntos en una partición del sistema.

Esta característica de los ***AABBs*** permite que un árbol de ***AABB*** sea reinstalado eficientemente de una manera *bottom-up*. Además, rinde un método directo para reinstalar una jerarquía de ***AABBs*** después de una deformación. Esta operación se puede poner en ejecución como un árbol del *post-order* trasversal, es decir, para cada nodo interno, visitan a los hijos primero, después de lo cual la caja de limitación es recalculada. Sin embargo, para evitar los gastos indirectos de las llamadas de función recurrentes, se ponen en ejecución de forma diferente.

En nuestra puesta en práctica las hojas y los nodos internos de un árbol de ***AABB*** se asignan como órdenes de nodos. Se puede hacer esto, puesto que el número de primitivas en el modelo es estático y a priori, sabido. Además, el árbol se construye tal que el número de índice de cada nodo interno del hijo, en el conjunto, es mayor que el número de índice de su padre. De esta manera, los nodos internos son reinstalados correctamente iterando sobre el conjunto de nodos internos en orden invertida. La reinstalación de un árbol de ***AABB*** de un acoplamiento del triángulo toma menos de 48 operaciones aritméticas por triángulo. Los experimentos han demostrado que para los modelos integrados por sobre 6000 triángulos, la reinstalación de un árbol de ***AABB*** es cerca de diez veces más rápida que reconstruyéndolo.

Hay, sin embargo, una desventaja a este método de reinstalación. Debido a los cambios relativos de la posición de primitivas en el modelo después de una deformación, las cajas en un árbol reinstalado pueden tener un grado más alto de superposición que las cajas en un árbol reconstruido.

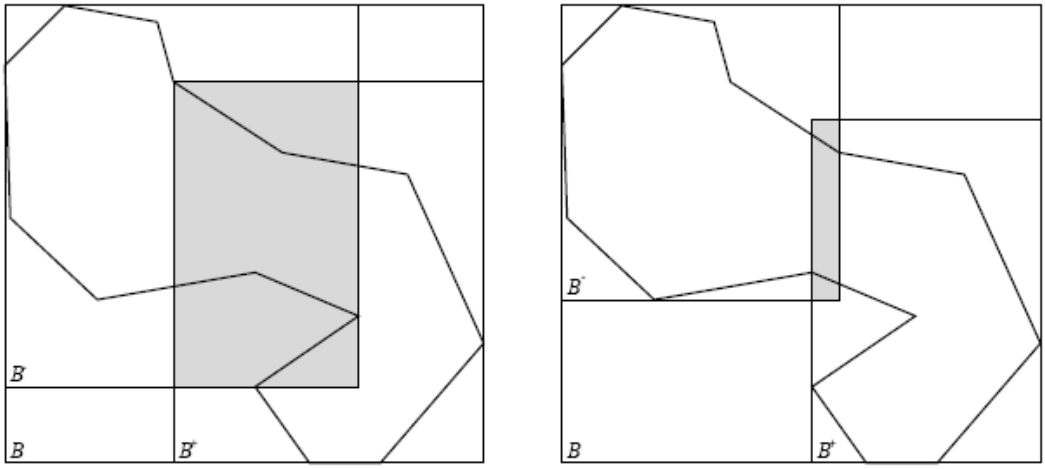


Fig. 21 Reinstalación contra la reconstrucción del modelo después de una deformación (a) Reinstalado (b) Reconstruido

La figura anterior (Fig. 21) ilustra este efecto para el modelo presentado en la Fig.20. Un grado más alto de la superposición de cajas en el árbol da lugar a más nodos que son visitados durante una prueba de la intersección, y así, un funcionamiento peor de la prueba de intersección.

Observamos un grado más alto de superposición entre las cajas en un árbol reinstalado sobre todo para las deformaciones radicales tales como torceduras excesivas. Esto es, debido al hecho de que el grado de superposición aumenta sobre todo para las cajas que se mantienen arriba en el árbol, mientras que la mayoría de las cajas que se prueban son las que se mantienen cerca de las hojas.

2.6.3.1 Funcionamiento

El coste total de probar un par de modelos representados por jerarquías de limitación del volumen se expresa en la función de coste siguiente. [4] [11]

$$T_{total} = N_b * C_b + N_p * C_p,$$

Donde

T_{total} Es el coste total de probar un par de los modelos para la intersección

N_b Es el número de los pares de limitación del volumen probado para la superposición

C_b Es el coste de probar un par de limitar los volúmenes para la superposición

N_p Es el número de los pares primitivos probados para la intersección

C_p Es el coste de probar a un par de los primitivos para la intersección.

Los parámetros en la función de coste que son afectados por la opción del volumen de limitación son N_b, N_p y C_b. Un tipo de limitación ajustado del volumen, tal como el ***OBB***, da lugar a una N_b y a un N_p bajos, pero tiene un C_b relativamente alto, mientras que un ***AABB*** dará lugar a más pruebas que son realizadas, pero el valor de C_b será más bajo.

2.6.4 V-clip

El *V-Clip* es un algoritmo para la detección de colisiones de objetos poliédricos especificados por una representación del límite. El *V-Clip* sigue el par más cercano de características entre los poliedros convexos, usando un acercamiento evocador del algoritmo más cercano a sus características, *Lin-Canny*. El *V-Clip* es una mejora sobre este último en varios aspectos: se reduce la complejidad de la codificación, y la robustez es significativamente mejorada; la puesta en práctica no tiene ninguna tolerancia numérica y no exhibe problemas que completan un ciclo. El algoritmo también maneja los poliedros penetrantes, haciéndola útil para la detección poliédrica convexa de la colisión. Este estudio presenta los principios teóricos del *V-Clip* y da una descripción del pseudocódigo del algoritmo. También documenta las varias pruebas que comparan el *V-Clip*, *Lin- Canny* y el algoritmo realizado de *GJK*.

Los resultados demuestran el *V-Clip* puede ser un competidor fuerte en esta rama, comparándose favorablemente con los otros algoritmos en la mayoría de las pruebas, en términos del funcionamiento y de la robustez.

2.6.4.1 Características básicas del algoritmo

1) maneja el caso de la penetración, describiendo la característica típica que muestra la penetración en el mismo tiempo requerido para el caso de la desunión.
2) es robusto. Las configuraciones degeneradas no son problemáticas, y la puesta en práctica del algoritmo no contiene una sola tolerancia numérica. Las divisiones son raras, y nunca implican un divisor de una magnitud más pequeña que el dividendo. El V-Clip no presenta problemas en el ciclo.
3) El código para el V-Clip es significativamente más simple que el de Lin-Canny. La especificación del V-Clip no implica ninguna de las clases de condiciones que hagan al Lin-Canny difícil de poner en ejecución.

El límite de un poliedro convexo adentro en $\mathcal{R}^3$ contiene vértices, bordes, y las caras; estas características son sistemas convexos, aquí denotados por las letras mayúsculas. Las letras minúsculas de la negrilla denotan puntos y vectores adentro $\mathcal{R}^3$. Si P es un plano y **y** es un punto, ambos en $\mathcal{R}^3$, Dp (Y) denota la distancia entre **y** y P. Si

$$P = \{x \in \mathcal{R}^3 : \hat{n} \cdot x + w = 0\},$$

Donde ñ es la unidad normal a P, entonces

$$D_p(y) = \hat{n} \cdot y + w.$$

Los bordes del Poliedro se tratan como vectores de un punto t posterior hacia un punto principal h adentro$\mathcal{R}^3$. Un punto a lo largo de la arista se puede parametrizar por un escalar λ.

$$e(\lambda) = (1 - \lambda)t + \lambda h, 0 \leq \lambda \leq 1. \qquad (1)$$

La topología de los poliedros convexos donde los vértices cercanos suceden a las aristas a ese vértice. Los vecinos de una cara son las aristas que limitan la cara. Una arista tiene exactamente cuatro vecinos: los dos vértices en sus puntos finales y las dos caras que limita. Observe que la relación vecina es simétrica. Las características Poliedras se tratan como sistemas cerrados. Por lo tanto, una cara incluye las aristas que la limitan, y una arista incluye sus puntos finales de los vértices. Las regiones y los planos de Voronoi son centrales al algoritmo del V-Clip.

Definición 1: para la característica X en un poliedro convexo, la región VR(X) de Voronoi es el sistema de los puntos fuera del poliedro que están cerca de X en cuanto a cualquier otra característica en el poliedro. El Voronoi VP(X plano; Y) entre las características vecinas X y Y es el plano que contiene VR(X) \ VR (Y).

Todas las regiones de Voronoi son limitadas por los planos de Voronoi, y las regiones cubren colectivamente el espacio entero fuera del poliedro. Los planos de Voronoi entre las características vecinas vienen en dos variedades: planos de vértice-arista y de cara-arista. Los planos del vértice-arista contienen el vértice y son normales a la arista, mientras que los planos de la cara-arista contienen la arista y son paralelos a la normal de la cara (Fig.22). La región de Voronoi de un vértice V es limitada por un sistema homogéneo de planos: cada uno es un plano del vértice-arista entre V y uno de sus aristas vecinas. La región de Voronoi de una arista E es limitada por cuatro planos: dos planos del vértice-arista y dos planos del cara- arista. La región de Voronoi de una cara s-echada a un lado F se limita por s + los planos 1: un plano del cara-arista para cada arista que limita F, más el plano F por sí mismo ayuda; VR (F) es un semi-infinito prisma poligonal.

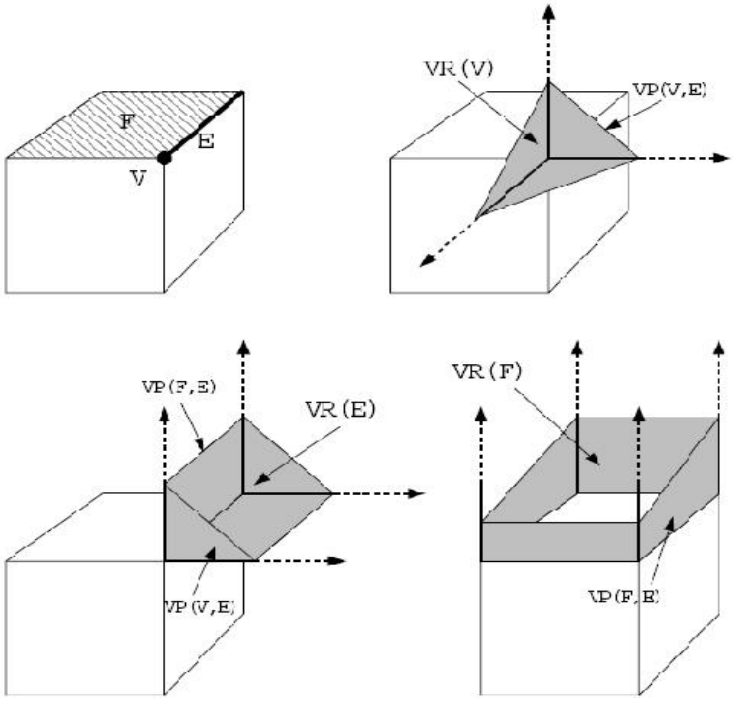


Fig. 22 Regiones de Voronoi.

En la izquierda superior de la figura se muestra un poliedro cúbico. Entre sus características están la cara F, la arista E, y el vértice V. La derecha superior: La región VR (V de Voronoi). Uno de los planos de Voronoi que limitan esta región es VP (V; E), correspondiendo a la arista más cercana a E. En la izquierda inferior: La región VR (E) de Voronoi. Dos de los planos de Voronoi que limitan esta región son VP (V; E) y VP (F; E), correspondiendo respectivamente a las características vecinas de V y F. En la derecha inferior: La región VR (F) de Voronoi. Uno de los planos de Voronoi que limitan esta región es VP (F; E), correspondiendo a la arista vecino E. El plano de soporte F se delimita a si mismo VR (F).

El teorema 1 sean X y Y un par de características de disolver poliedros convexos, y sea $x \in X$ y $y \in Y$ ser los puntos más cercanos entre X y Y. Si $\mathbf{x} \in$ VR (Y) y $\mathbf{y} \in$ VR(X), entonces **x** y **y** son un par global más cercano de puntos entre los poliedros.

El teorema 1 no requiere que los puntos más cercanos en X y Y sean únicos, y que en situaciones degeneradas no existan. Si las condiciones del teorema se resuelven, se puede decir que hay un par de

puntos de los dos poliedros más cercano que X y Y. Como Lin-Canny, el algoritmo del V-Clip es esencialmente una búsqueda para dos características que satisfagan las condiciones del teorema 1.

En cada iteración, el V-Clip prueba si el par actual de características satisface las condiciones, y si no, actualiza una de las características, generalmente con la más cercana. Si la nueva característica está en una dimensión más alta que la anterior, entonces la distancia de la inter-característica disminuye terminantemente. Si la nueva característica tiene una dimensión más baja que la vieja, se mantiene la misma distancia.

(Cuando se actualiza una característica cercana dimensionalmente más baja, no hay ninguna esperanza de disminuir la distancia, puesto que la anterior característica incluye el nuevo. Sin embargo, tal actualización mejora la localización del punto más cercano, y puede accionar las actualizaciones subsecuentes que reducen terminantemente la distancia de la inter-característica.)

En contraste con el algoritmo Lin-lin-Canny, el V-Clip nunca computa realmente el punto más cercano de una característica a otra, aunque si el anterior es un vértice, se torna trivial. Esto le da al algoritmo su robustez en la cara de la degeneración. Además, la iteración del V-Clip no depende de los poliedros que se desune, y converge correctamente a un par conveniente de características que muestran cuando hay penetración.

El diagrama de estado del (Fig.22) ilustra el algoritmo. Cada estado corresponde a una combinación posible de los tipos de la característica, por ejemplo, el V - el estado de F significa que una característica es un vértice, y la otra es una cara. Las flechas denotan pasos posibles de la actualización a partir de un estado a otro. Las flechas sólidas marcan las actualizaciones que disminuyen la distancia de la inter-característica; las flechas rayadas marcan las actualizaciones para las cuales la distancia de la inter-característica permanece igual.

Los cuatro estados primarios del algoritmo son V - V, V - E, E-E, y V - F; puede terminar en estos estados. El quinto estado, E-F, es especial por que el algoritmo no puede terminar en este estado a menos que los poliedros sean penetrantes. La (Fig.22) denota que el algoritmo debe terminar, ya que no hay ciclos en el gráfico que abarquen solamente flechas rayadas, cualquier trayectoria infinita a través del gráfico contendría un número infinito de flechas sólidas, cada una denota una estricta reducción en la distancia de la inter-característica. Dado que solo hay definitivamente muchos pares de características, lo convierte en imposible. La coherencia es explotada depositando el par de las características más cercanas a partir de

una invocación del algoritmo al siguiente. El método de inicializar el par de la característica no es crítico, puesto que el algoritmo converge a un par de las características más cercanas de cualquier par que comienza.

El problema 1: dado un par de las características, X y Y, una de cada poliedro, se determina si el punto más cercano en Y a X situado dentro de VR(X). Si no, se actualiza X de una manera que disminuya la distancia de la inter-característica, o baja la dimensión de X mientras que guarda la constante de la distancia de la inter-característica.

Básicamente, este problema es el de comprobar una de las condiciones simétricas del teorema 1, y de poner al día el par de la característica cuando no se resuelve. Si Y es un vértice, la solución es trivial: Y contiene solamente un solo punto que se comprueba fácilmente sabiendo si hay lateralidad contra los planos que limitan VR(X). Cuando Y no se encuentra en VR(X), los planos se superponen lo que indica cómo X debe ser actualizado.

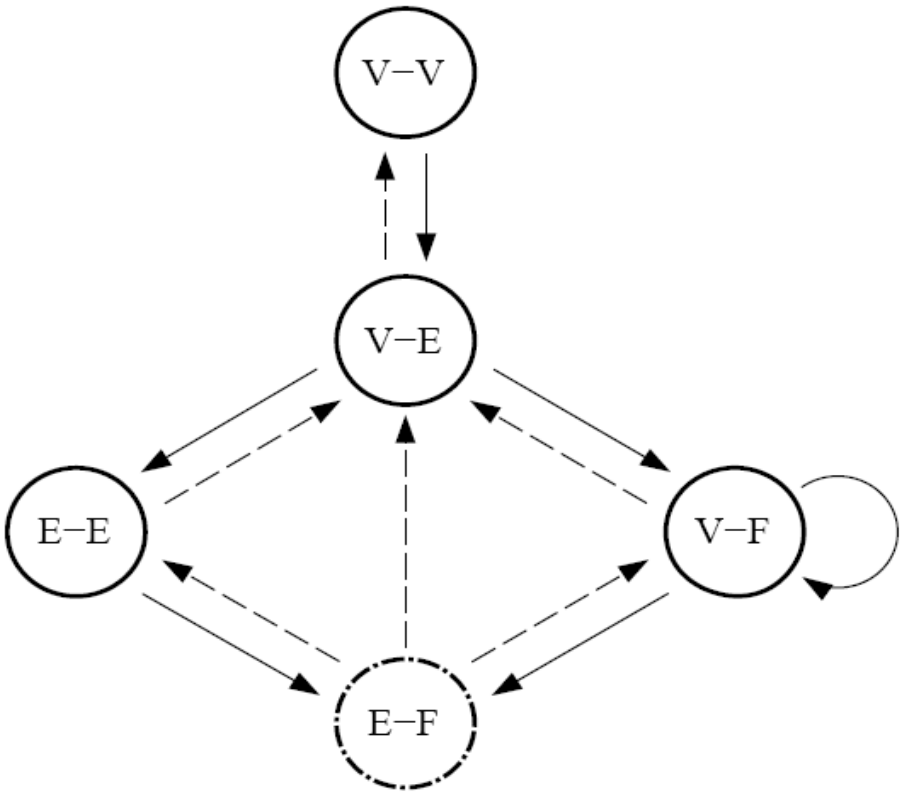


Fig. 23 Estados y transiciones del algoritmo V-Clip

Estados y transiciones del algoritmo del V-Clip: Estados de E-E, y de E-F de la (Fig.23), donde está un vértice, una arista y una cara X, respectivamente. Este procedimiento generalmente se llama truncamiento de Voronoi.

Deje S ser un subconjunto del sistema de limitación orientada de los planos VR(X). Cada plano en S impone una exigencia linear de la desigualdad que señale en VR(X) satisfaga. Colectivamente, los planos en S definen una región convexa $K \supseteq VR(x)$. Si E es una arista de t a h, entonces $E \cap K$ es vacían, o una línea segmento a lo largo de E:

$(1 - \lambda)t + \lambda h, \underline{\lambda} \leq \lambda \leq \overline{\lambda}$

El algoritmo 1 computa los valores $\underline{\lambda}$ y $\overline{\lambda}$ además de las características vecinas $\underline{N}$ y $\overline{N}$ de X eso

corresponde a los planos que acortan E: E incorpora K como ella cruza VP(X; N), y las salidas como él cruzan VP(X; N). La figura demuestra un ejemplo.

Si $t \in K, \underline{N} = \emptyset : Si\ h \in K, \overline{N} = \emptyset . Si\ E \cap K = \emptyset$, entonces el algoritmo devuelve FALSO,

si no devuelve VERDADERO. Las divisiones que ocurren en los pasos 11 y 18 son las únicas divisiones que ocurren en el algoritmo entero del V-Clip. En estos casos, la magnitud del divisor debe ser distinta a cero y nunca menos que la magnitud del dividendo, así el desborde del excedente puede ocurrir no.

Si X es un vértice, se hará una sola invocación a la arista podada la cual se utiliza para disminuir una arista contra todos los planos de Voronoi (K = VR(X)). Si X es una arista, acortando contra

Algoritmo 1 (Poda de arista). Acorte de la arista t a h contra los planos de Voronoi en S. Devuelve Falso si la arista se acorta totalmente, si no devuelve Verdadero.

```
1: underline(λ) ← 0; overline(λ) ← 1
2: underline(N) ← overline(N) ← ∅
3: for all Voronoi planes P in S do
4:    N ← neighbor feature of X corresponding t
5:    d_t ← D_P(t)
6:    d_h ← D_P(h)
7:    if d_t < 0 and d_h < 0 then
8:       underline(N) ← overline(N) ← N
9:       return FALSE
10:   else if d_t < 0 then
11:      λ ← d_t/(d_t − d_h)
12:      if λ > underline(λ) then
13:         underline(λ) ← λ
14:         underline(N) ← N
15:         if underline(λ) > overline(λ) then
16:            return FALSE
17:   else if d_h < 0 then
18:      λ ← d_t/(d_t − d_h)
19:      if λ < overline(λ) then
20:         overline(λ) ← λ
21:         overline(N) ← N
22:         if underline(λ) > overline(λ) then
23:            return FALSE
24: return TRUE
```

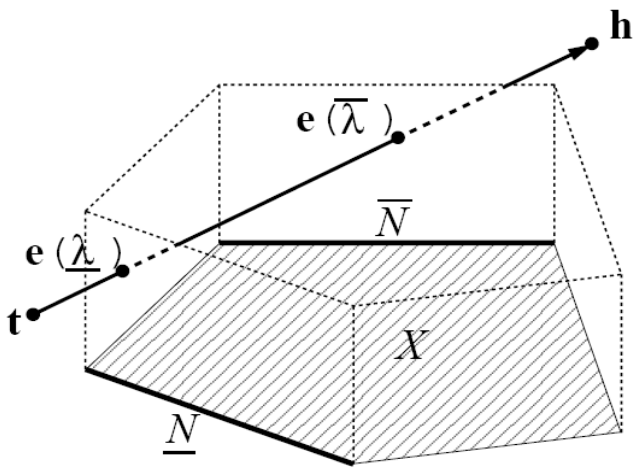


Fig. 24.- Los planos de vértice-arista y de la cara-arista que limitan VR(X) se hacen por separado. Finalmente, si X es una cara, S es el sistema de todos los planos entre X y sus aristas vecinas; al acortar contra el plano de soporte de X se maneja por separado. Después de que se acorte una arista E, el paso siguiente es determinarse la posición del punto más cercano en E a X dentro $K \supseteq VR(x)$ y cómo actualizar X.

Caso de intersección

Primero suponer que la arista E interseca K, de modo que la arista trucada devuelva VERDADERO.

Definición 2: Dejando que E sea una arista, y e (λ) un punto parametrizado a lo largo de él, como adentro de la ecuación mostrada en (1). Para una característica polyhedral X, la función de distancia de la arista $D_{E,X}(\lambda): \mathcal{R} \to \mathcal{R}$ es definido como:

$$D_{E,X}(\lambda) = \min_{X \in X} \|x - e(\lambda)\|$$

$D_{E,X}(\lambda)$ Es la distancia entre e (λ) y X. Algunas características importantes de esta función son dadas por el teorema siguiente

Teorema 2: La función de distancia de la arista $D_{E,X}(\lambda)$ es continua y convexa. Si, Luego $e(\lambda_0) \notin X$ es diferenciable en λ_0 La pregunta actual es

de si el valor mínimo$D_{E,X}(\lambda)$, sobre [0,1] ocurre en la gama $[\underline{\lambda}, \overline{\lambda}]$. Debido al teorema 2, se contesta esta pregunta simplemente comprobando las muestras de su derivado en $\underline{\lambda}$ y $\overline{\lambda}$. El mínimo ocurre en el intervalo $[0, \underline{\lambda})$ si y solamente si $D'_{E,X}(\underline{\lambda}) > 0$; ocurre en el intervalo $(\overline{\lambda}, 1]$ si y solamente si $D'_{E,X}(\overline{\lambda}) < 0$ A la luz de esto, el algoritmo 2 realiza una necesaria actualización. Si X se actualiza a una característica vecina, entonces la distancia de la inter-característica.

Chequeo de derivación del Post-truncamiento del algoritmo 2.

1: **if** $\underline{N} \neq \emptyset$ and $D'_{E,X}(\underline{\lambda}) > 0$ **then**
2: Update X to $\underline{N}$
3: **else if** $\overline{N} \neq \emptyset$ and $D'_{E,X}(\overline{\lambda}) < 0$ **then**
4: Update X to $\overline{N}$

Permanece igual si la característica vecina está de una dimensión más baja que X, y disminuye terminantemente, si la característica vecina está de una dimensión más alta que X. Si X no es actualizado, el punto más cercano en E a X esta situado dentro de K.

Las fórmulas para evaluar las muestras de $D'_{E,X}$ siguiendo la geometría básica. Siendo **u** un vector dirigido a lo largo de E, de la cola a la cabeza. Si X es un vértice en la posición **v**, entonces

$$sing\left[D'_{E,X}(\lambda)\right] = sing[u \cdot (e(\lambda) - v)]. \qquad (2)$$

Si X es una cara en el plano P con $\hat{\mathbf{n}}$ el normal exterior, entonces

$$sing\left[D'_{E,X}(\lambda)\right] = \begin{cases} +sing(u \cdot n), & D_p[e(\lambda)] > 0 \\ -sing(u \cdot n), & D_p[e(\lambda)] < 0 \end{cases} \qquad (3)$$

Una fórmula para el caso donde una arista X es innecesaria. El derivado se puede evaluar con respecto a la característica vecina relevante $\underline{N} = \overline{N}$, que debe ser un vértice o una cara. El derivado con respecto a esta característica vecina es igual al derivado con respecto a X en el punto donde E cruza el plano de Voronoi.

Caso de no intercepción

Considere después el caso donde la arista podada se vuelve FALSA, indicando que la posición de E es totalmente afuera VR(X). Hay dos maneras que esto pudo ser detectado. El primer, llamado de exclusión simple, se detecta cuando ambas puntos finales de E se encuentran en la posición **"**fuera**"** de un solo plano de Voronoi. El segundo, llamado de exclusión compuesta, se detecta cuando $\underline{\lambda}$ excede $\overline{\lambda}$; esto significa que ningún punto en E satisface simultáneamente las restricciones impuestas en dos planos de Voronoi. Los dos casos se distinguen cerca si o no $\underline{N} = \overline{N}$ En cualquier caso, X debe ser actualizado.

Un cierto cuidado se requiere debido a una sutil diferencia entre el vértice-arista y los planos de Voronoi de la cara-arista.

Dos espacios medios del plano Voronoi define vértice-arista: los puntos en un medio espacio están estrictamente más cercanos a la arista mientras que los puntos en el otro sea equidistante de la arista y del vértice. Un plano de Voronoi de la cara-arista no reparte el espacio de esta manera. Los puntos de cualquier lado del plano de Voronoi pueden estar terminantemente más cercano a la cara que a la arista (Fig.25) A consecuencia de

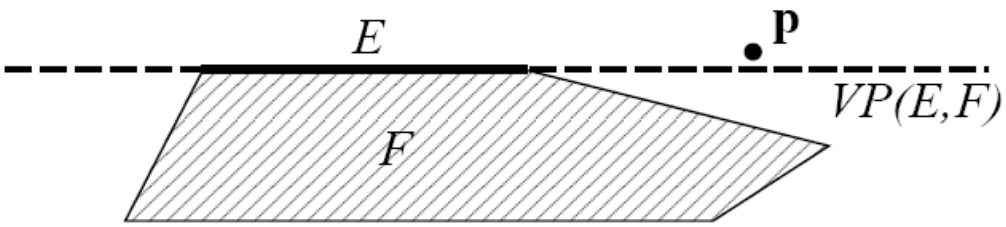


Fig. 25.- Una ensambladura de la arista-cara. Esto es una proyección 2-D en el plano de F. Aunque el punto p esta situado en el lado de la arista del plano de Voronoi, está terminantemente más cercano a la cara.

Estos hechos son producto de las derivaciones de chequeo y los pasos de la actualización del algoritmo 2, sea dondequiera que las intersecciones de una arista sean válidas al plano de Voronoi del vértice-arista, incluso si el camino no está en el límite entre las regiones correspondientes de Voronoi. En contraste, si una arista cruza un plano de Voronoi de la cara-arista, los resultados del derivado y los pasos de la actualización son válidos solamente si el camino se encuentra en el límite entre las regiones de Voronoi.

Exclusión de la región de Voronoi del vértice

Si E es falsa totalmente fuera del vértice V's de la región Voronoi, la actualización es directamente mejor. Para la exclusión simple, la arista E es falsa en el lado de la arista de un plano de Voronoi del vértice-arista. Cada punto en E está terminantemente más cercano a la arista vecina que a V, y así que V se actualiza a esta arista. Para la exclusión compuesta, puesto que ambos caminos anversos están con los planos de Voronoi del vértice-arista, se utiliza el algoritmo 2. Para ver que una actualización debe ocurrir en este caso, considere la función de distancia convexa $D_{E,V}(\lambda)$ definido por [0,1]. Si es compuesto la exclusión ocurre, entonces $0 < \overline{\lambda} < \underline{\lambda} < 1$. Si $D_{E,V}$ logra su mínimo adentro $[0, \overline{\lambda}]$ entonces la condición de la línea 3 es verdadera. Finalmente si $D_{E,V}$ logra su mínimo adentro $(\overline{\lambda}, \underline{\lambda})$, ambas condiciones son verdaderas. Por lo tanto V siempre se actualizara por su arista vecina, y la distancia para E ira estrictamente descendiendo.

Durante el proceso de la actualización de la característica, es posible obtener que este alojado en un mínimo local en el estado del vértice-cara. Esto sucede cuando el vértice satisface las restricciones del plano de la cara-arista de VR (F), situado “abajo” del plano de F, y tiene aristas vecinas que todos se dirijan lejos de F. De esta situación, cualquier actualización a una característica vecina o aumentaría la distancia de la inter-característica, o aumente la dimensión de una característica mientras que guarda la constante de la distancia de la inter-característica. Ni unos ni otros de éstos son actualizaciones válidas. Cuando el algoritmo alcanza tal estado, los poliedros pueden no ser penetrantes (Fig.26).

Si son penetrantes, el algoritmo debe terminar y generalizar esto. Si no, un nuevo par de las características debe ser encontrado en el mínimo local. El algoritmo 3 maneja esta situación. Prueba a V contra cada plano de la cara del poliedro de F. Si V esta situado en el lado negativo de todos, está dentro del poliedro y se vuelve la penetración AG. Si no, F se actualiza a la cara que V tiene distancia finita. Si el algoritmo 3 actualiza F a F0, V está por lo menos como cerca de F0 en cuanto a cualquier otra cara en el poliedro de F, y está terminantemente más cercano a F0 que a F. Por lo tanto no hay posibilidad de pendiente nuevamente dentro del mismo mínimo local entre V y F.

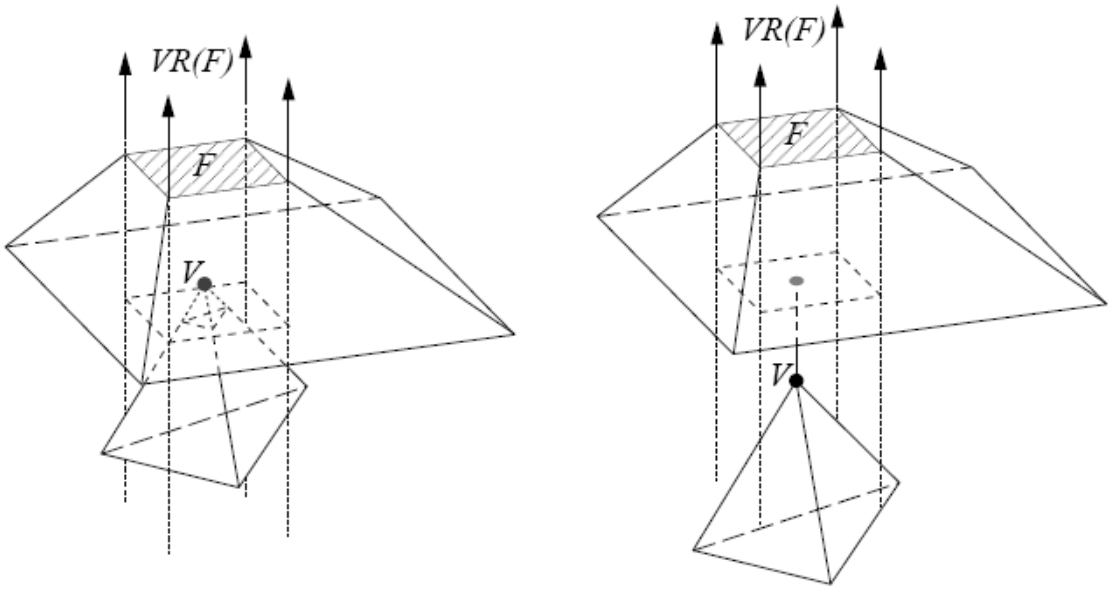


Fig. 26.- Estados mínimos locales con y sin la penetración.

El V-Clip detecta la penetración en uno de dos estados: en el estado de la arista, cuando la arista perfora la cara; y en el estado del vértice-cara, cuando un vértice esta situado adentro del otro poliedro. La prueba basada en un par del testigo de la arista-cara está muy rápidamente puesto que examina solamente la geometría local cerca de un punto de la penetración; la prueba es independiente de la complejidad de los poliedros. La prueba basada en un par de vértice-cara es mucho más lenta, puesto que el vértice se debe a probar su lateralidad contra todos los planos de n del poliedro de la cara: un cálculo de O(n). Afortunadamente, los testigos de la arista-cara a la penetración son mucho más comunes en la práctica.

Algoritmo 3: Mínimo local manual. Detecta la penetración o el desplazamiento de un mínimo local.

```
1: d_max ← −∞
2: for all faces F' on F's polyhedron do
3:     P' ← plane(F')
4:     d = D_P'(V)
5:     if d > d_max then
6:         d_max ← d
7:         F_0 ← F'
8: if d_max ≤ 0 then
9:     return PENETRATION
10: F ← F_0
11: return CONTINUE
```

Cuando se penetra una cantidad pequeña concerniente a sus tamaños, el testigo del par será la arista-cara. Solamente si un poliedro se mueve y atraviesa aproximadamente en su totalidad al otro con la fuerza del vértice-cara entonces la penetración de los testigos ocurre casi totalmente. La (Fig.27) demuestra una secuencia en la cual un par de poliedros pasan uno sobre otro, y los testigos de la penetración ocurren en el vértice-cara momentáneamente. Generalmente, los poliedros pasan uno sobre otro por fuera nunca generando testigos de la penetración del vértice-cara.

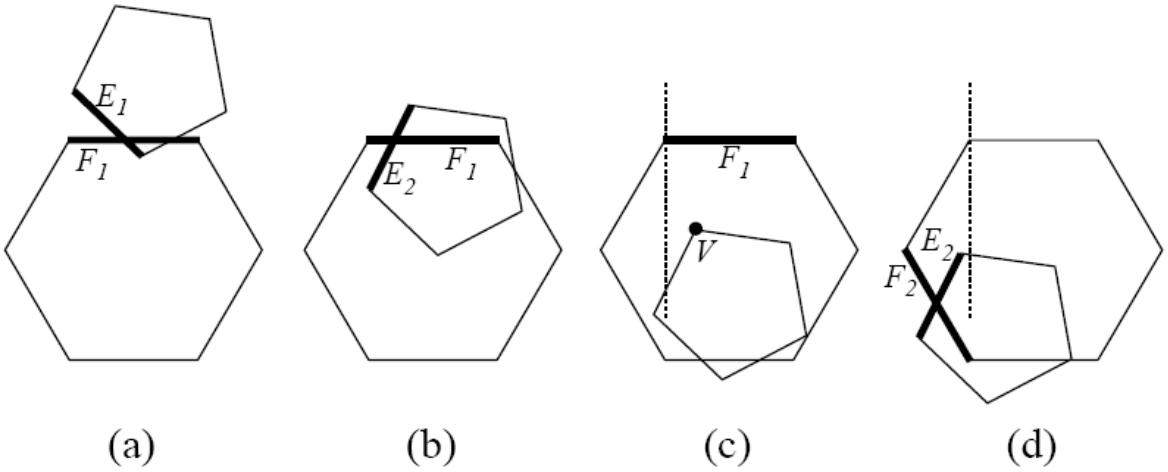


Fig. 27- Poliedros que pasan uno a través de otro.

Esto es una proyección 2-D con las caras que aparecen como línea segmentos. En (a) y (b), los testigos de la penetración son pares del vértice-cara. En (c), un poliedro se ha movido casi totalmente con el otro, y un par del vértice-cara se convierte en el testigo de la penetración. La línea discontinua representa uno de los planos de Voronoi de VR (F1). V cruza una vez este plano, como en (d), el algoritmo sale del V - mínimo local de F1, y localiza otra vez un par del testigo de la penetración del vértice-cara.

A manera de conclusión sobre este algoritmo tenemos que el V- Clip ha hecho una significativa mejora al Lin-Cany por tres razones. Por un mejor manejo de las penetraciones y por lo tanto los objetos no convexos, son más fácil de cifrar, y es más robusto. La robustez proviene del hecho de que el V-Clip no construye explícitamente los puntos más cercanos entre las características mientras que itera. Mas bien, localiza los puntos más cercanos a las varias regiones del interés a través de operaciones simples del truncamiento y de pruebas derivadas escalares. Es decir el V-Clip utiliza los primitivos topológicos [29] mientras que los nuevos objetos del cálculo Lin-lin-Canny de la necesidad: los puntos más cercanos entre las características. La arista-cara y la cara-cara son los dos casos más problemáticos de Lin-lin-Canny; explican la mayoría de la complejidad de la codificación y de los problemas que completan un ciclo. En V-Clip, la caja de la arista-cara es mucho simplificada, y se elimina la caja de la cara-cara.

2.6.5 GJK

El algoritmo de la distancia de Gilbert-Johnson-Keerthi (GJK) es un método iterativo para computar la distancia entre los objetos convexos [30].

- Aplicable a cualquier tipo de politopo
- Construye una secuencia de *simplices* cuyos vértices se toman de los vértices de la diferencia de Minkowski de ambos *BB*s.
- Cada *simplex* generado está más cercano al origen que el previo.
- Con un test determina cuando las *simplices* más cercanas se han encontrado.
- Este proceso termina cuando se encuentra un *simplex* que contenga al origen.
- Esta distancia final es la distancia entre los *BB*.
- En caso de politopos, calcula la distancia en tiempo O (n).
- Forma la diferencia de Minkowski entre ambos objetos P y Q

2.7 Librerías mayormente utilizadas por estos algoritmos

2.7.1 I-Collide: Tipo de modelo (Modelos o conjunto de modelos convexos)

I-Collide desarrollada en la Universidad de Carolina del Norte (UNC), es capaz de determinar con gran rapidez y exactitud las colisiones presentes en un entorno virtual de N elementos en movimiento rígido (rotación y traslación). I-Collide está implementada en [32] y emplea la coherencia temporal para conseguir una mayor eficiencia. Nótese que la existencia de coherencia temporal (el estado de la simulación no cambia significativamente entre dos pasos de simulación consecutivos) implica coherencia geométrica (el tiempo de simulación es tan pequeño que hace que los objetos no se muevan largas distancias entre dos pasos de simulación).

El algoritmo implementado en I-Collide consta de dos partes:

• Determinar los pares de objetos que se pueden solapar en la escena. Para ello emplea intersección de cajas de inclusión AABB. El resultado de este paso es una lista con los pares de objetos de la escena que pueden solaparse.

• Determinar cuáles son las primitivas (puntos, líneas o caras) de cada uno de los objetos de la pareja más cercana. Para ello, la librería tiene implementado el algoritmo de Lin-Canny. Aunque este algoritmo es sólo para objetos convexos, I-Collide lo adapta mediante el uso de un árbol jerárquico, de tal manera que los objetos no convexos los tratará como un árbol donde todos los elementos pueden ser convexos o no convexos excepto en el último nivel que tienen que ser convexos. Al suponer coherencia temporal, se considera que los dos elementos más cercanos son la pareja de primitivas resultado del paso de simulación anterior. Si no es así, el algoritmo considera que las primitivas más cercanas están cerca, a su vez, de las primitivas anteriores y por tanto estudia la distancia en las primitivas vecinas.

2.7.2 V-COLLIDE: Tipo de modelo (Modelos triangulares)

V-Collide [34] es una librería implementada en C++ por la UNC, para determinar cuál de los objetos en movimiento en una escena virtual de VRML [33] están en contacto potencial. La arquitectura de V-Collide, al igual que en el caso del I-Collide, se divide en dos niveles:

• En el primer nivel se determina qué pares de objetos se solapan en la escena. Para ello se emplea el mismo método que en el caso del I-Collide, la aproximación por cajas de inclusión AABB.

• En el segundo nivel se estudia la colisión de los pares de objetos seleccionados en la fase anterior. Para ello, el sistema calcula el árbol jerárquico de cajas de inclusión orientadas (OBB) mediante el mismo algoritmo que en RAPID. El sistema primero determina cuál de las cajas de inclusión del último nivel están en contacto y posteriormente estudia el solapamiento de los triángulos correspondientes. Por tanto, al contrario que en RAPID, V-Collide determina qué parejas de objetos de la escena virtual colisionan pero no la distancia entre ellos (como es el caso de I-Collide). Sin embargo, tiene mejores prestaciones que ICollide en el caso de objetos no convexos.

2.7.3 RAPID: Tipo de modelo (Modelos triangulares)

La librería RAPID es la implementación en C++ del método de detección de colisiones presentado por Gottschalk, Manocha y Lin en [35], método descrito para pares de objetos rígidos. El algoritmo es capaz de determinar de manera eficiente qué polígonos de dichos objetos se solapan. Para la búsqueda de la colisión RAPID pre calcula árboles jerárquicos de cajas orientadas (cajas con alineamiento no paralelo

OBB) de los dos objetos de la escena bajo estudio. Posteriormente estudia el solapamiento de cada uno de los volúmenes de inclusión. El algoritmo presentado es capaz de calcular todos los contactos entre geometrías muy complejas permitiendo interactividad.

2.7.4 SWIFT: Tipo de modelo (Geometrías cerradas y convexas)

SWIFT es una librería de detección de colisiones, cálculo de distancias y determinación de contactos de objetos poligonales tridimensionales en movimiento rígido. Ha sido implementada en C++. Soporta geometrías cerradas y convexas (aunque la versión SWIFT ++ sí que soporta modelos poliédricos no-convexos). Para el cálculo de distancias hace uso de una jerarquía de modelos multiresolución de los objetos implicados (LOD-Hierarchy) que, junto a la utilización de un algoritmo derivado en las regiones de Voronoi, hace que la librería tenga gran velocidad de cómputo [36]. También hace uso de cajas de inclusión para una primera aproximación en el cálculo de las colisiones. Esta librería es más rápida que otras librerías como ICollide y V-Clip y además, presenta buenas prestaciones incluso cuando no hay coherencia temporal.

2.7.5 SOLID: Tipo de modelo (Objetos convexos)

La librería Solid [37] sólo se considera aquí en su versión 2.0, también llamada freeSOLID. Esta librería, implementada en C++ y bajo licencia GNU, sirve para la detección de colisiones entre objetos convexos y/o formas geométricas básicas (cilindros, cubos, esferas, etc.) bajo movimientos rígidos y deformaciones y en especial para entornos virtuales VRML. Esta librería tiene implementados los siguientes algoritmos:

• El algoritmo GJK (Gilbert, Johnson y Keerthi), implementado por Van den Bergen. Sirve para el cálculo de distancias entre objetos convexos y para calcular distancias de penetraciones entre objetos

• Algoritmo para la detección de colisiones para modelos complejos deformables mediante el uso de árboles de cajas de inclusión con lados paralelos propuesto por Gino Van Den Bergen. Una de las principales ventajas del mismo reside en la gran rapidez en la actualización del árbol jerárquico de los objetos. Al trabajar con objetos convexos, Solid hace uso de la librería QHull, la cual permite calcular la superficie convexa de un conjunto de puntos

2.7.7 V-CLIP: Tipo de modelo (Modelos convexo)

EL algoritmo de V-CLIP (Voronoi Clip) es un algoritmo para la detección de colisiones entre objetos poliédricos convexos [38]. La librería V-Clip es una implementación en C++ del algoritmo V-Clip,

realizada por MERL (Mitsubishi Electric Research Lab) [39] que además facilita la manipulación de geometrías. Es capaz de calcular los puntos más cercanos de dos poliedros y la distancia entre los mismos en tiempo constante. Si existe penetración, V-Clip devuelve la profundidad de la misma. Esta implementación presenta mayor eficiencia y robustez. Frente a los algoritmos de Lin- Canny y GJK

Con este capitulo se logró sintetizar toda la literatura, que en cuanto a algoritmos de detección de colisiones se refiere. También se consiguió hacer un estudio a cerca de dichos algoritmos y técnicas, para en el capitulo siguiente mostrar cuales de estos métodos podrían ser mas fructíferos para su desarrollo aquí en nuestra universidad.

Capítulo III: Propuesta del algoritmo

3.1 Introducción

En este capítulo tenemos como objetivo proponer uno o varios algoritmos y técnicas de detección de colisiones para objetos complejos. Basándonos en las investigaciones realizadas en el capitulo anterior y teniendo en cuenta las ventajas, desventajas y diferencias que muestran estos métodos para la detección. En cuanto a las técnicas que en este trabajo de diplomado se exponen, pensamos que son corrientes de avanzada a nivel mundial que consideramos que para lograr la excelencia en cuanto a la calidad del simulador deberían de aplicarse junto con los algoritmos que en este capitulo presentamos.

Con este trabajo se podrá tener un conocimiento sobre cuales son los procedimientos para detectar colisiones en objetos complejos utilizando las técnicas mencionadas para que en un futuro ya no muy lejano estas reflexiones sean utilizadas e implementadas en nuestro simulador quirúrgico.

Fundamentándonos en las investigaciones realizadas nos tomaremos la libertad de seleccionar la forma de los volúmenes delimitadores que consideramos más eficaz, lo más interesante será utilizar los que permitan detectar colisiones entre ellos rápidamente. Además, lo habitual es utilizar el mismo tipo de formas para todos los objetos.

3.2 Desarrollo de la propuesta

Un primer ejemplo de volúmenes delimitadores son las cajas (cubos o prismas cuadrangulares). Determinar si dos cajas colisionan es más sencillo que detectar si un objeto de forma potencialmente irregular colisiona con otro. Sin embargo si no ponemos ninguna restricción a la orientación de las cajas la operación de detectar colisiones entre ellas puede complicarse.

Una opción que facilita la tarea es obligar a que las cajas estén alineadas con los ejes absolutos utilizados en el entorno (Axis-Aligned Bounding Boxes, AABBs)

3.2.1 Algoritmo AABB TREE

• El almacenamiento de cada AABB consiste en guardar los vértices de dos esquinas opuestas.

• Comprobar la colisión entre dos AABBs es sencillo. Basta comparar los rangos en los que se encuentra cada uno de ellos en cada eje. Si la intersección en los tres ejes absolutos no es vacía, los volúmenes colisionan.

• El cálculo del AABB que encierra a un objeto es muy sencillo. Es suficiente recorrer todos los vértices que lo forman, y obtener el mayor y menor valor de cada coordenada para obtener las dos esquinas opuestas que forman el AABB más pequeño que encierra al objeto.

• La creación de los niveles inferiores de la jerarquía también es simple. Suele realizarse dividiendo el objeto en ocho partes, mediante tres planos paralelos a los planos del sistema de coordenadas, cada uno de ellos dividiendo al AABB en dos partes iguales. Si alguno de los nuevos AABBs más pequeños no contiene ningún polígono del objeto original, se elimina de la jerarquía

• Un AABB se ajustará más o menos al objeto en función de la orientación de éste con respecto al sistema de coordenadas del mundo.

• El AABB de un objeto debe ser recalculado si el objeto al que delimita rota.

3.2.2 Esferas de inclusión

Para contrarrestar los problemas de los AABB's se pueden utilizar esferas. Detectar si dos esferas colisionan es realmente sencillo: basta comprobar la distancia entre sus centros: si es mayor que la suma de los radios, no hay colisión.

El método intuitivo de conseguir la esfera delimitadora consiste en hacer que el centro sea la media de todos los vértices del objeto, y el radio la distancia desde éste hasta el vértice más alejado del objeto. Sin embargo, existe otro método que obtiene esferas más ajustadas (realmente obtiene la esfera óptima), pero es bastante lento. Por tanto, si el cálculo de estos volúmenes se realiza en una fase previa a la ejecución del programa, podría valer la pena invertir un tiempo adicional en su cálculo con tal de conseguir beneficios durante la ejecución, por lo que utilizaríamos este segundo método. Sin embargo, si el cálculo de las esferas ha de realizarse en tiempo de ejecución (por ejemplo debido a deformaciones interactivas

de los objetos que suponen volver a construir la jerarquía de esferas), podría resultar más interesante hacer uso del primer algoritmo que, aunque menos exacto, es más rápido

Una ventaja adicional de las esferas de la que carecen las cajas alineadas con los ejes es que, al ser simétricas, no necesitan volver a ser calculadas ante giros del objeto. Bastará con desplazar o rotar la posición de los centros de todas las esferas de la jerarquía del mismo modo que se modifican los vértices del objeto.

El modo de generar los niveles inferiores de la jerarquía es semejante al caso de los AABBs. La única diferencia substancial es que cada esfera de los niveles inferiores puede realmente estar colisionando con alguna de sus esferas hermanas. Eso no es problema para el funcionamiento del algoritmo recursivo.

No obstante, construir los niveles inferiores dividiendo la esfera inicial en ocho partes no ajusta en exceso las esferas del nuevo nivel con la forma del objeto, como puede verse en la (Fig.28). Esto es debido a que los niveles inferiores son construidos dividiendo de forma preestablecida los niveles superiores, sin utilizar la información disponible sobre la estructura del objeto.

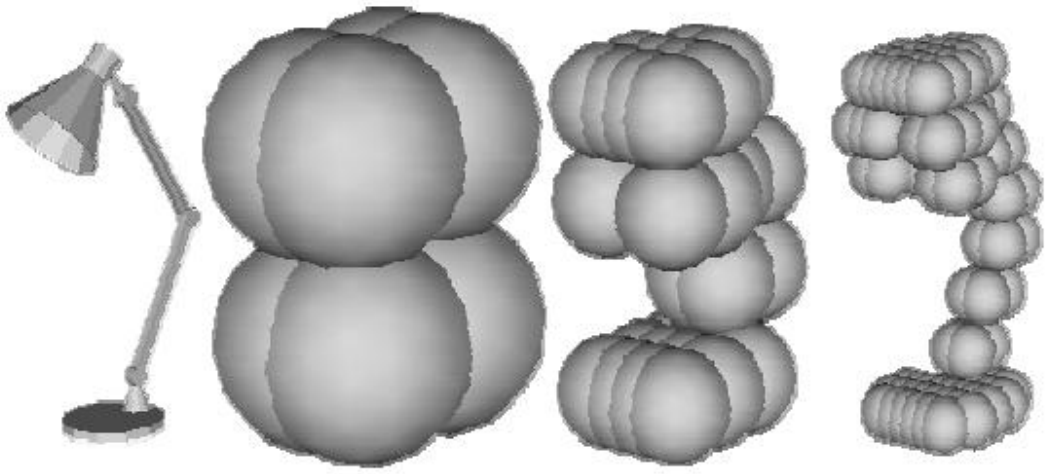


Fig. 248 Esferas delimitadoras mediante la división tipo octree

Este mal ajuste en los niveles inferiores es una desventaja, pues cada esfera contendrá mucho “volumen vacío” que no pertenecerá al objeto. Eso supone que el porcentaje de colisiones falsas encontradas por el algoritmo será mayor, lo que hará disminuir su rendimiento

3.2.2.1 Medial-Axis Surfaces

Para tratar de solucionar este problema también proponemos otro método para construir la jerarquía. La idea es calcular los ejes que recorren el objeto, considerando como eje a una superficie que es equidistante a dos caras del objeto (se les denomina *medial-axis surfaces)*. Obtener esos ejes es costoso, por lo que en realidad se calcula una aproximación

Las esferas son colocadas en función de los ejes con diferentes niveles de precisión para conseguir la jerarquía. Al hacer uso de la información geométrica del modelo, este proceso construye unos árboles de esferas mucho más ajustados a los objetos, como puede verse en la (Fig.29). Su principal problema es que resultan bastante más costosas de calcular

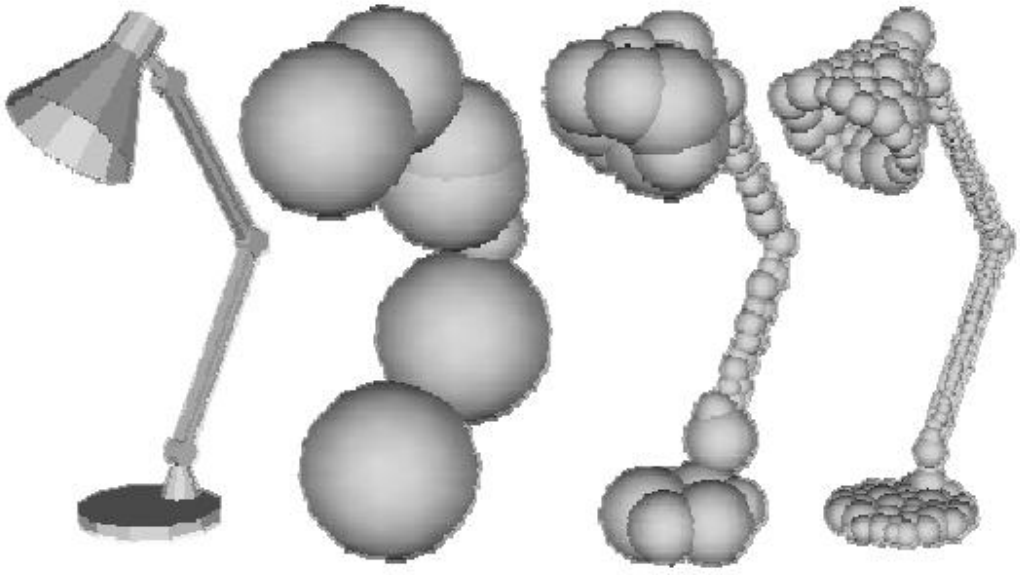


Fig. 29.- Esferas delimitadoras más ajustadas

3.2.3 OBB TREE

La diferencia entre las cajas delimitadoras orientadas (Oriented Bounding Boxes, OBBs) y los AABBs reside en que los OBB no tienen la restricción de tener sus ejes alineados con los ejes absolutos del entorno

Su obtención es más complicada que la de los AABBs, por lo que únicamente debería hacerse en fases previas a la ejecución del programa. Además, averiguar si dos OBBs colisionan es más complicado que su contraparte en AABBs o esferas. El método intuitivo requiere comprobar si las aristas de uno de los OBBs corta a alguna de las caras del otro, y viceversa. En total, 144 comprobaciones de colisiones entre una arista y una cara.

Sin embargo tienen la ventaja de que pueden ajustarse muy bien a los objetos (desde luego, mejor que los AABBs, y también mejor que las esferas cuando se llega a las hojas de la estructura jerárquica donde se tratan los polígonos), y no tienen los problemas de los AABBs ante rotaciones, por lo que merece la pena tratar de sacarles partido.

El OBBs tiene una mejora para colisiones. Tiene un método para construir el OBB de un objeto, y construye la jerarquía dividiendo cada OBB en dos a través de su eje más largo. En cada paso, se calcula el OBB asociado a los polígonos que quedan en cada una de las mitades del OBB original. En la (Fig.30) puede verse este proceso en dos dimensiones, donde el objeto es un conjunto de segmentos y los OBB son rectángulos.

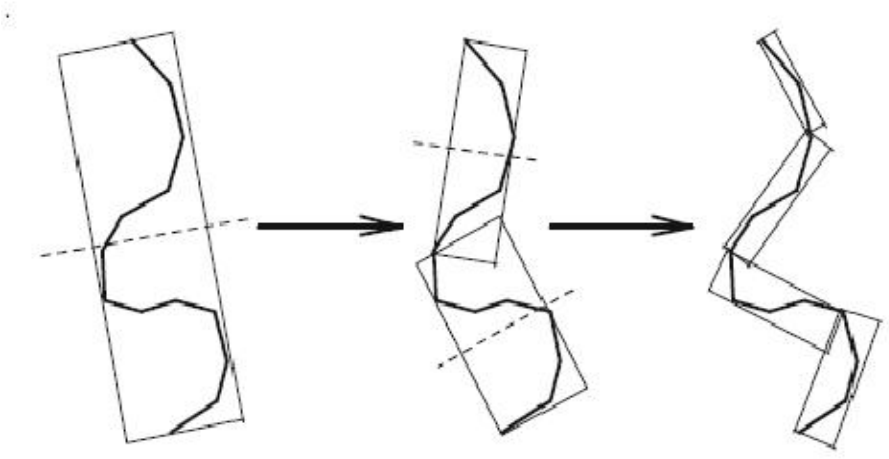


Fig. 30.- División de un objeto bidimensional mediante OBBs

También presenta una nueva forma de detectar colisiones entre OBBs. Utiliza la idea de que si dos OBBs no colisionan, existirá al menos un plano sobre el que las proyecciones de los dos OBBs no colisionen

¿Con cuantos, y con qué planos tenemos que probar para asegurar que los OBBs colisionan?

Para responder a esta pregunta, nos apoyamos en otra idea: si dos OBBs no colisionan, existirá un plano separador (diferente al plano anterior) que deja un OBB a un lado, y otro al otro. Además, ese plano será paralelo a alguna de las caras de los OBBs o paralelo a un eje de cada uno de los OBBs.

Con estas ideas, se consigue comprobar si dos OBBs colisionan en, como mucho, 200 operaciones aritméticas (comparaciones, sumas, restas, multiplicaciones, y valores absolutos), un número mucho menor que las necesarias para las 144 comprobaciones de colisión entre una arista y una cara.

Naturalmente sigue siendo necesario un mayor número de cálculos que los necesarios en el caso de las esferas o de los AABBs. Sin embargo en muchas ocasiones un OBB se ajusta más al objeto original que los otros volúmenes. Otro punto a su favor es que han implementado su sistema en una librería en C++ denominada RAPID (Robust and Accurate Polygon Interference Detection) ([RAPID]), que puede ser utilizada libremente para usos no comerciales. Es posible que en algunas ocasiones utilizar esferas o AABBs sea mejor a utilizar OBBs, pero teniendo en cuenta que la librería está disponible, el tiempo ahorrado en la implementación podría compensar una ligera pérdida de velocidad en tiempo de ejecución.

Las estructuras jerárquicas como las presentadas hasta el momento tienen otra ventaja adicional para la detección de colisiones en tiempo real. El tiempo de cálculo necesario para búsqueda de colisiones puede ajustarse en función del tiempo disponible, si se admite un cierto margen de error o inexactitud en las colisiones

Esto es debido a que si se detecta colisión en los niveles altos de la jerarquía, el algoritmo comprobará si la colisión se ha realmente producido o no mirando en los niveles inferiores. En función del tiempo disponible, este proceso podría ser detenido en cualquier momento, de tal forma que se considerara que los objetos realmente han colisionado aunque esto pueda que no sea correcto.

3.3 Propuesta final de algoritmo

Basados en los estudios y en lo antes planteado pensamos que el mas optimo para utilizar seria el OBBs debido a la exactitud que nos brinda en el momento de la colisión, además de que han implementado su sistema en una librería de C++ y esta puede ser utilizada libremente. Además también ponemos a

consideración los algoritmos AABBs y volumen de inclusión con esferas, debido a que son mas agiles en cuanto tiempo de ejecución por la facilidad de cálculo de sus colisiones.

En el transcurso de este trabajo de diplomado hemos expuesto las nociones básicas que se deben tener en cuenta a la hora de realizar tareas de detección de colisiones y a manera de resumen les podemos decir que en este trabajo hemos hablado sobre:

• Los objetos se han supuesto rígidos (geometría constante), es decir, que sólo experimentan rotaciones y traslaciones;

• Las herramientas del instrumental quirúrgico son objetos rígidos (geometría constante) y por lo tanto no se deforman;

• Sin embargo los órganos son objetos deformables (geometría cambiante), acorde con determinados criterios biomecánicas, e incluso pueden sufrir cortes (desaparición de polígonos en su geometría.).

• existen herramientas del instrumental quirúrgico, como la aguja, que tiene que poder perforar y atravesar un órgano (acciones que constituirían excepciones del sistema de detección de colisiones);

• Otra herramienta del instrumental, el hilo de sutura, tiene unas propiedades físicas muy distintas: no es deformable (en cuanto a que no posee propiedades elásticas), pero sí que es moldeable (no es rígido). También el hilo de sutura debe ser incorporado al sistema de detección de colisiones para permitir la sutura y, lo que es mucho más complejo, hacerse un nudo consigo mismo;

• Todo ello, naturalmente, debe ser realizado en un espacio tridimensional. Esto no supone un problema conceptual, pero sí computacional

En fin hemos tenido en cuenta todo lo referente al ambiente de de una cirugía, porque como antes planteamos es de vital importancia que el grado de realismo del simulador sea el mayor, debido a que la preparación que este software le brindará a los residentes de la carrera de medicina será una antesala de lo que nuestros futuros médicos realizaran en el quirófano el día de mañana. Decir también que nos sentimos muy orgullosos de contribuir con este granito de arena que es el estudio de la detección de colisiones en lo que en un futuro se convertirá en un logro más de la medicina revolucionaria cubana. La detección de colisiones se realiza una vez en cada iteración del bucle de simulación, es decir, cada cierto tiempo (*tsim).* Como consecuencia, si una colisión se produce en un instante intermedio no será detectada, hecho que o curre en el caso que una acción ocurra muy rápidamente.

Conclusiones

Después de haber hecho un estudio minucioso a los algoritmos que se emplean en la detección de colisiones en un simulador quirúrgico así como todo lo concerniente al entorno de la detección de colisiones se ha llegado a las siguientes conclusiones:

- Se logró hacer un resumen de la bibliografía relacionada a la detección de colisiones en un simulador quirúrgico
- Se trataron algunos algoritmos que también se pueden utilizar para detectar colisiones, pero por su alto coste computacional o por su margen de error, no lo señalamos como propuesta pero sirven para entender la detección de colisiones
- Se explicó el problema de detectar colisiones entre objetos complejos
- Se encontraron técnicas novedosas que se utilizan para detectar colisiones
- Se recogieron tendencias actuales para detectar colisiones en los simuladores quirúrgicos

Con todo este estudio realizado y con la propuesta presentada de un algoritmo para detectar colisiones en un simulador quirúrgico tratando con los objetos complejos que sea eficiente y fácil su entendimiento así como su implementación, también que en un futuro sea utilizado por el proyecto de la facultad 5 de La Universidad de las Ciencias Informáticas simuladores quirúrgicos

Recomendaciones

A continuación se presentan un grupo de recomendaciones para un futuro desarrollo de un simulador quirúrgico comenzando por la realización e implementación de algoritmos que sean capas de tener la máxima eficiencia para detectar colisiones

1. Realizar un estudio detallado en el campo de la simulación visual, centrándose específicamente en la detección de colisiones
2. Profundizar mas en el estudio de algoritmos como AABBs y OBB los cuales son unos de los más eficientes en la detección de colisiones
3. Implementar el algoritmo propuesto en el proyecto simulador quirúrgico de la facultad 5 de la Universidad de Ciencias informáticas
4. Destinar un presupuesto para la puesta en práctica de las primeras simulaciones utilizando el algoritmo propuesto

Referencias Bibliográficas

1. A.Gorostiaga, D.Andía. APRENDIZAJE DIAGNÓSTICO Y QUIRÚRGICO MEDIANTE ENDOSCOPIA VIRTUAL Y TUTORÍA INTELIGENTE: PROYECTO LAHYSYTOTRAIN. [En línea] I.Brouard Unidad de Endoscopía Servicio de Obstetricia y Ginecología Hospital de Basurto Bilbao. Osakidetza, 11 de 12 de 2001. [http://www.sitiomedico.com.uy/endoscopia/actual/enero1.htm].

2. Vázquez Estévez, Juan. EL FUTURO DE LA HOSPITALIZACIÓN PEDIÁTRICA: Estructura y función de los Hospitales. [En línea] 2007. http://www.sld.cu/print.php?idv=7976.

3. INSTRUMENTACIÓN BIOMÉDICA. Realidad Virtual y Simuladores virtuales quirúrgicos aplicaciones doctorado. [En línea] http://www.depeca.uah.es/docencia/doctorado/cursos04_05/83190/Documentos/RV-Simuladores.pdf.

4. Jukes, Nick. InterNICHE Tipos de alternativas y su impacto pedagógico. [En línea] InterNICHE. [http://www.interniche.org/book/spanish/artigoA1.html].

5. Centro de Investigacion e Innovacion en Bioinganieria. Simulación quirúrgica en realidad virtual. [En línea] Centro de Investigación e Innovación de Bioingeniería, 2006. http://www.upv.es/crib/docs/ficha_ci2b_simrv.pdf.

6. Padrón Arredondo, Luis Jesús. Las Nuevas Tecnologías de la Información (NTIC) en la . [En línea] 2007. http://www.revistaesalud.com/ojs/index.php/revistaesalud/article/view/104/253.

7. ZACHMANN G., LANGETEPE E. *Geometric data structures for computer graphics.* 2003.

8. GOTTSCHALK S., LIN M., MANOCHA D. *OBB-Tree: A hierarchical structure for rapid interference detection.* s.l. : ACMSIGGRAPH,AddisonWesley, 1996.

9. G., VAN DEN BERGEN. *Efficient collision detection of complex deformable models using AABB trees,.* s.l. : Journal of Graphics Tools 2, 4 .

10. ROUSSOPOULOS N., LEIFKER D. *Direct spatial search on pictorial databases using packed R-trees. .* Texas : ACMSIGMOD, 1985.

11. GOLDSMITH J., SALMON J. *Automatic creation Graphics.* s.l. : SIGGRAPH, 2004.

12. G, ZACHMANN. *Minimal hierarchical collision detection.* Hong Kong, China : ACM Symposium on Virtual Reality Software and Technology (VRST)., 2002.

13. VOLINO P., MAGNENAT-THALMANN N. *Efficient Self-Collision Detection on Smoothly Discretized Surface Animations using Geometrical Shape.* s.l. : Computer Graphics Forum.

14. —. *Collision and Self-Collision Detection: Efficient and Robust Solutions for Higly Deformable Surfaces.* s.l. : Comp. Animation and Simulation, 1995.

15. X., PROVOT. *Collision and Self-Collision Handling in Cloth Model Dedicated to Design Garments. .* Canadá : Canadian Human-Computer Communications Society, 1997.

16. LARSSON T., AKENINE-MÖLLER T. *Collision detection for continuously deforming bodies.* s.l. : Eurographics, 2001.

17. VAN DEN BERGEN, G. *Efficient collision detection of complex deformable models using AABB trees.* s.l. : Journal of Graphics Tools 2, 4 , 1997.

18. Kim, J., Srinivasan, A. *Computationally Efficient Techniques for real time surgical simulation with force feedback.* Orlando, Florida, USA. : IEEE Computer Society, 2002.

19. Lombardo, C., Cani, M. P y Neyret, F. *Real-time Collision Detection for Virtual Surgery.* s.l. : Proc. Computer Animation, 1999.

20. Baicu, G., Wong, W. y Sun, H. *RECODE: An Image-Based Collision Detection Algorithm.* s.l. : Journal of Visualization and Computer Animation, 1999.

21. Dinerstein, J. y Egbert, P. *Practical collision detection in rendering hardware for two complex 3D polygon objects.* s.l. : Eurographics UK , 2002.

22. Lin, M. y Canny, J. *Efficient Collision Detection for Animation.* Cambridge, England. : Third Eurographics Workshop on Animation and Simulation, 1991.

23. Mirtich, B. *V-Clip: Fast and Robust Polyhedral Collision Detection.* s.l. : ACM Transactions on Graphics, 1998.

24. Oxford University. Computing Laboratory. Computing the Distance between Objects. [En línea] Oxford University, 2003. http://web.comlab.ox.ac.uk/oucl/work/stephen.cameron/distances/index.html.

25. Christophe, Jean, Neyret, Fabrice y Cani, Marie Paule. Real-time Collision Detection for Virtual Surgery Jean-Christophe Lombardo. [En línea] 2004. http://www-sop.inria.fr/epidaure/FormerCollaborations/aisim/CompteRendu/aisim3/lombardo.pdf.

26. Shoemake, K. *Uniform random rotations.* Boston : Graphics Gems III, 1992.

27. Gottschalk, S. RAPID: Robust and accurate polygon interference detection system. [En línea] software library, 1996. http://www.cs.unc.edu/˜geom/OBB/OBBT.htm.

28. M¨oller, T. *A fast triangle-triangle intersection test.* s.l. : Journal of Graphics Tools, 1997.

29. Edelsbrunner, Herbert y Mucke, Ernst Peter. *Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms.* s.l. : ACM Transactions on Graphics, 1990.

30. Gilbert, E. G, Johnson, D.W y Keerthi, S. S. *A fast procedure for computing the distance between complex objects in three-dimensional space.* s.l. : IEEE Journal of Robotics and Automation, 1988.

31. Rabbitz, R. *Fast collision detection of moving convex polyhedra.* Boston : Graphics Gems IV.

32. Cohen, J. D, y otros. *I-COLLIDE:An interactive and exact collision detection system for large-scale environment.*

33. Bell, G., Carey, R. y Marrin, C. VRML97: The Virtual Reality Modeling Language. [En línea] 1997. http://www.vrml.org/Specifications/VRML97.

34. **Hudson, T., y otros.** *V-COLLIDE: Accelerated Collision Detection for VRML.* s.l. : VRML, 1997.

35. Gottschalk, S., Lin, M. y Manocha, D. *OBB-Tree: A Hierarchical Structure for Rapid Interference Detection.* s.l. : Department of Computer Science, University of N. Carolina, Chapel Hill, 1996.

36. Ehmann, S. A. y Lin, M. *Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching .* s.l. : Proceedings International Conference on Intelligent Robots and Systems, 2002.

37. Librería paradetección de colisiones SOLID. Software library for Interferection Detection . [En línea] http://www.win.tue.nl/~gino/solid/. SOLID.

38. Mirtich, B. *Fast and Robust Polyhedral Collision Detection.* s.l. : ACM Transactions on Graphics, 1998.

Librería de detección de colisiones mediante V-Clip. *Detection Algorithm.* [En línea] Mitsubishi Electric Research Laboratories. http://www.merl.com/projects/vclip/V-Clip Collision.

39. Cohen, J.D, y otros. *ICOLLIDE: An Interactive and Exact Collision Detection System for Large-Scaled Environments.* s.l. : ACM Int. 3D Graphics Conference, 1995.

Bibliografía

1- Orqueda, Ing. Omar A.A.orqueda@ieee.org. *Detección de Colisiones en Ambientes.*

2-U. Meier, C. Monserrat, M. Alcañiz, MC. Juan, V. Grau, JA. Gil. *SIMULACIÓN QUIRÚRGICA.* 2003.

3-López, Francisco Javier Roa. *Detección de Colisiones.* 2004.

4-Moctezuma Ramirez, Julio Guadalupe. *Manipulación Tridimensional de Objetos Deformables Virtuales.* Mexico : Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2006.

5-Cameron, S. A. y Culley, R. K. *Determining the minimum translational distance between convex polyhedra.* s.l. : IEEE Int. Conf. on Robotics and Automation, 1986.

6-Chung, K. y Wang, W. *Quick collision detection of polytopes in virtual environments.* s.l. : ACM Symposium on Virtual Reality Software and Technology, 1996.

7-Ong, C. J. y Gilbert, E. G. *The Gilbert-Johnson-Keerthi distance algorithm: A fast version for incremental motions.* s.l. : IEEE Int. Conf. on Robotics and Automation, 1997.

8-Barber, C. B., Dobkin, D. P. y Huhdanpaa, H. *The Quickhull algorithm for convex hull.* s.l. : ACM Transactions on Mathematical Software, 1996.

9-Kim, J. y Srinivasan, A. *Computationally Efficient Techniques for real time surgical simulation with force feedback.* Florida : 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002.

10-Lombardo, C., Cani, M. P. y Neyret, F. *Real-time Collision Detection for Virtual Surgery.* s.l. : Computer Animation, 1999.

11-Baicu, G., Wong, W. y Sun, H. *RECODE: An Image-Based Collision Detection Algorithm.* s.l. : Journal of Visualization and Computer Animation, 1999.

12-Dinerstein, J y Egbert, P. *Practical collision detection in rendering hardware for two complex 3D polygon objects.* s.l. : Eurographics UK Conference, 2002.

GLOSARIO

1) Frustum (zona final de la escena tomada)
2) BV (Volúmenes Contenedores)
3) AABBs (Axis-Aligned Bounding Boxes)
4) BVHs (Jerarquías de Volúmenes Contenedores)
5) GJK (Gilbert-Johnson-Keerthi)
6) RAPID Robust and Accurate Polygon Interference Detection
7) I-COLLIDE: An interactive and exact collision detection system for large-scale environment
8) UNC (Unbalanced operation Normal response mode Class)
9) VRML (Virtual Reality Modeling Language)

1