CEIG'08, Barcelona, Sept. 3–5 (2008)
L. Matey and J. C. Torres (Editors)

Optimización del Cálculo de Colisiones para Mallas Deformables mediante Voxelización de Primitivas

J. Gascón[1], M.A. Otaduy[1], J.M. Espadero[1], A. Rodríguez[2]

[1]Universidad Rey Juan Carlos (URJC)
[2]Universidad Politécnica de Madrid (UPM)

Abstract

Este artículo propone una optimización para la detección de colisiones de objetos deformables, consistente en la voxelización eficiente de primitivas en particiones espaciales. Nuestra técnica tiene aplicabilidad en el corte de objetos deformables, el cálculo de intersección de superficies y la detección del instante de colisión entre objetos móviles. El artículo incluye un análisis de rendimiento de la técnica propuesta frente a técnicas tradicionales basadas en cajas envolventes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism[Animation and Virtual Reality]

1. Introducción

El cálculo de colisiones [GASF94] es una de las etapas principales de la simulación de objetos móviles. En esta etapa se analizan los objetos de la escena en busca de contactos entre ellos. Esos contactos sirven para calcular después las fuerzas de repulsión que impiden que dos objetos interpenetren entre sí.

Uno de los mayores retos para las técnicas de detección de colisiones es su aplicación a escenarios donde están involucrados objetos deformables. El comportamiento de este tipo de objetos se caracteriza por encontrarse comprendido entre el campo de los sólidos articulados y el de los fluidos. Los objetos deformables no disponen de ningún esqueleto rígido, por lo que pueden sufrir cambios muy notables en su forma antes de volver a su apariencia original. Esta propiedad de los objetos deformables implica que no sean fácilmente aplicables las técnicas de detección de colisiones que descargan el peso computacional en la fase de preprocesamiento, tales como los volúmenes contenedores. En su lugar, es común optar por técnicas como las particiones espaciales [TKH*05], que no presuponen ninguna característica particular en el movimiento o deformación.

El test de colisión entre dos objetos requiere, en principio, comprobar la existencia de intersección entre todos los pares de primitivas de ambos objetos (p.ej., todas las facetas de un objeto contra las aristas del otro, y viceversa). El uso de una partición espacial reduce drásticamente el número de pares a comprobar, pues sólo se tienen en cuenta aquellos pares que comparten alguna celda de la partición. En este artículo se considerarán particiones espaciales regulares en celdas cúbicas o voxels. El primer paso de la detección de colisiones mediante particiones espaciales requiere identificar las celdas ocupadas por las distintas primitivas de los objetos. Tradicionalemente, dicha identificación se realiza de manera conservadora, calculando una caja envolvente (AABB) para cada primitiva y voxelizando la caja. Si las cajas son excesivamente conservadoras, la detección de colisiones identifica multitud de pares de primitivas como colisiones potenciales, pero que resultan ser falsas (se les conoce como *falsos positivos*).

Contribución del artículo: El trabajo presentado en este artículo se centra en torno a un método para acelerar la detección de colisiones en mallas deformables utilizando particiones espaciales regulares. En concreto, se describe una optimización de la voxelización de primitivas de las mallas, que permite reducir el número de falsos positivos entregados por la partición espacial, y reducir así el tiempo de ejecución de la detección de colisiones.

El trabajo incluye también el análisis de la técnica presentada sobre tres distintos ejemplos de aplicación:

1. **Corte de mallas de tetraedros.** La voxelización eficiente del filo de corte, representado como la unión de segmentos, reduce de manera drástica el número de tests elementales a realizar entre pares tetraedro-segmento de corte.
2. **Detección continua de colisiones.** En este caso, la inserción de primitivas en la partición mediante voxelización no ha resultado rentable, debido quizás a que el número de tests depende del volumen ocupado por las trayectorias de las facetas, no por las de los vértices.
3. **Intersección de superficies.** En mallas con primitivas de tamaño no homogéneo, la mejora obtenida también es muy considerable utilizando la voxelización de aristas.

Como se discute en el artículo, el grado de eficiencia de la técnica presentada depende de la morforlogía de las mallas utilizadas, pero en general proporciona aceleraciones notables en los problemas de corte e intersección, y aceleraciones mínimas en la detección continua.

A continuación se exponen algunos de los trabajos previos más relacionados con la problemática abordada en este artículo. Posteriormente, en el Apdo. 3 se detallan los algoritmos de particionado espacial y de voxelización utilizados en los experimentos. Los Apdos. 4–6 describen y analizan los casos de prueba realizados sobre objetos deformables: corte progresivo, detección de colisiones continua e intersección de mallas de triángulos. Finalmente se presentan las principales conclusiones extraidas.

2. Trabajos previos

El proceso de detección de colisiones conlleva un coste computacional muy alto si se sigue un enfoque de resolución mediante fuerza bruta. No obstante, paulatinamente han ido apareciendo diferentes técnicas que han permitido ir reduciendo el coste asociado a este proceso, entre las que se pueden mencionar los volúmenes contenedores, los métodos estocásticos, las técnicas basadas en los campos de distancia, las basadas en tests de visibilidad o las técnicas de subdivisión espacial. Se puede profundizar en el análisis de las distintas técnicas consultando alguna de las recopilaciones de trabajos previos existentes [LG98, JTT01, LM04, Ber04, Eri04, TKH*05].

Las primeras técnicas de subdivisión que aparecieron dividían el espacio en retículos regulares. Estas aproximaciones desperdiciaban mucho espacio en memoria, por lo que rápidamente aparecieron técnicas que dividían el espacio utilizando otras estructuras más eficientes, entre las que se pueden destacar las particiones binarias del espacio (BSPs), los *octrees* o el *spatial hashing*.

Centrándose en las técnicas basadas en *spatial hashing,* el trabajo de Teschner *et al.* [THM*03] quizás haya sido el que más ha contribuido a difundir la técnica basada en el uso de tablas *hash* para reducir el espacio de búsqueda sobre el particionado espacial que previamente se ha realizado, aunque fue Turk [Tur90] el primero en mencionar la utilización del término *spatial hashing* para la detección de colisiones. Teschner *et al.* aplican su técnica a objetos discretizados mediante tetraedros, pero sugieren que esta solución puede ser fácilmente extendida para trabajar con otras primitivas, aunque en el trabajo de Spillman *et al.* de nuevo se presentan resultados solamente con tetraedros [SBT07]. En sus análisis, Hastings *et al.* llegan a la conclusión de que interesa tener la mayoría de los objetos de un tamaño menor que el de la celda básica del grid de ocupación en el que se subdivide la escena [HMG05].

Kooten *et al.* [vKvdBT07] utilizan *spatial hashing* para realizar simulaciones con *smoothed particle hydrodynamics* (SPH) y encontrar los átomos vecinos. Se basan en el trabajo de Teschner *et al.* pero introducen dos mejoras: (i) Consideran argumentos representados en aritmética de coma flotante como argumentos de la función *hash*; (ii) En la consulta, utilizan la CPU para almacenar la tabla hash y la GPU para almacenar otra tabla con una lista de los átomos, haciendo una indirección desde esta tabla a la de la CPU.

Mesit *et al.* utilizan un modelo masa-muelle para representar los objetos deformables de la escena, sobre la cual aplican una función hash que tiene en cuenta los volúmenes contenedores jerárquicos calculados sobre los objetos deformables [MGH06].

Tradicionalmente, la técnica de voxelización en sus diferentes variantes ha sido un algoritmo fundamental ya sea en la visualización de modelos en general o para el renderizado específico de primitivas elementales [KS87, VKK*03]. Trabajos recientes aprovechan las nuevas posibilidades ofrecidas por el hardware gráfico para implementar la voxelización de los objetos mediante GPUs [ED06, PTTK07]. También recientemente, se han utilizado este tipo de soluciones para simplificar escenarios de realidad virtual mediante la utilización de impostores [BNV06]. Desde un punto de vista de la simulación, algunos autores proponen soluciones basadas en representaciones volumétricas de los objetos para resolver la etapa de detección de colisiones. Así, He y Kaufman proponen una técnica probabilista de detección de colisiones sobre objetos volumétricos [HK97]. Chen *et al.* utilizan la GPU para implementar una técnica de detecccción de colisiones basada en la voxelización de los modelos [CWZ*04]. La voxelización de primitivas en el contexto de detección de colisiones también ha sido utilizada por Heidelberger *et al.* [HTK*04], aunque en situaciones muy específicas.

3. Voxelización de primitivas en particiones espaciales

Esta sección describe en más detalle el algoritmo básico de detección de colisiones utilizando particiones espaciales. Seguidamente, se describe la voxelización de primitivas para su inserción eficiente en la partición espacial, en particular para la detección de colisiones con segmentos.

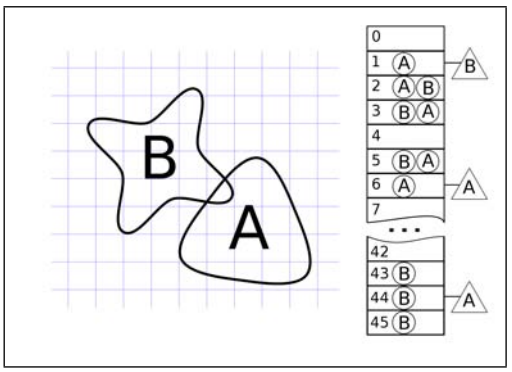

Figure 1: *Ejemplo de dos objetos deformables colisionando, y la inserción de sus primitivas en celdas de la tabla hash.*

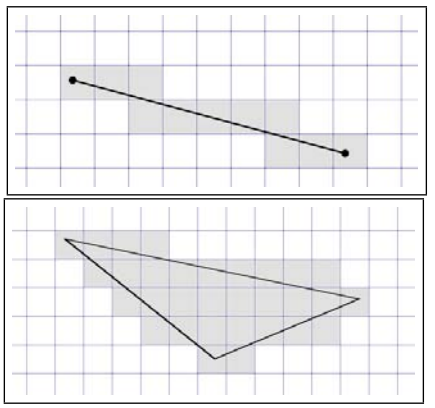

Figure 2: *Ejemplos de voxelización de un segmento y un triángulo.*

3.1. Partición espacial y tabla hash

En la Figura 1 se contempla un ejemplo con dos objetos *A* y *B* en colisión. Para detectar las colisiones, las primitivas (vértices, aristas, facetas, y/o tetraedros) que forman los objetos *A* y *B* se insertan en una partición espacial. De esta manera, se pueden eliminar fácilmente multitud de pares de primitivas, y sólo es necesario examinar en detalle aquellas que residen en una misma celda de la partición. Siguiendo el algoritmo de Teschner *et al.* [THM*03], y como representa la figura, las celdas de la partición espacial se asignan a posiciones de una tabla hash.

La forma bruta de insertar las primitivas en la tabla hash consiste en calcular su caja contenedora, y después insertar todas las celdas contenidas en la caja. Esta operación es trivial, pues sólo requiere el cálculo de las celdas máxima y mínima, y la ejecución de tres bucles (para los tres ejes cartesianos), que visitan las celdas contenidas. Sin embargo, como se aprecia en la Figura 5, el volumen de la caja contenedora puede ser notablemente mayor que el volumen de las celdas estrictamente circunscritas a la primitiva. El inconveniente de utilizar las cajas contenedoras es que la partición espacial devuelve multitud de pares de primitivas que comparten alguna celda y que requieren un test de colisión más detallado. Sin embargo, estos pares de primitivas son *falsos positivos* producidos por el volumen innecesario de las cajas contenedoras.

A continuación se describe la voxelización eficiente de primitivas. Se discute en detalle un algoritmo para la voxelización de segmentos o aristas, ya que los experimentos realizados han demostrado que son las primitivas que dan lugar a una mayor ganancia de rendimiento. La Figura 2 muestra el resultado de voxelización en 2D para un segmento y un triángulo.

3.2. Voxelización de segmentos

En lugar de utilizar cajas contenedoras (AABBs) para determinar las celdas de la partición ocupadas por un segmento, se empleará un algoritmo de rasterización basado en la voxelización incremental de Amanatides y Woo [AW87]. La mejora introducida por este método es patente cuando el segmento de corte no está alineado con los ejes, donde el número de celdas a recorrer se reduce de forma considerable (Figura 5)

Este algoritmo emplea una partición regular del espacio y un método para recorrer de forma eficiente las celdas de la partición que son atravesadas por el segmento. El algoritmo se divide en dos partes:

1. Un preproceso que identifica el punto de entrada del segmento a la partición y prepara los contadores a emplear en la segunda parte del algoritmo. Esta inicialización requiere entre 33 y 40 operaciones de coma flotante.
2. El proceso de recorrido incremental de las celdas atravesadas por el segmento. En cada celda se calcula la distancia que puede recorrer el segmento antes de volver a cambiar de celda, así como la próxima celda atravesada por el mismo. Este proceso toma 4 operaciones de comparación y 2 sumas en cada iteración.

El algoritmo es similar al algoritmo de rasterizado de Bresenham [Bre65], aunque el último trabaja con la secuencia de celdas recorridas por un segmento que va del centro de la celda inicial al centro de la celda final. Este comportamiento del algoritmo de Bresenham introduciría posible errores de muestreo que llevarían a omitir celdas que son recorridas por el segmento.

4. Corte progresivo de objetos deformables

Esta sección describe los resultados experimentales obtenidos en la simulación del corte de objetos deformables

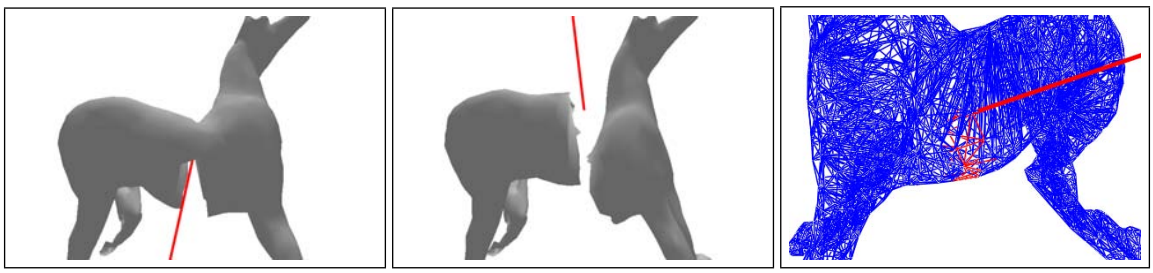

Figure 3: *En la izqda. y centro, dos fotogramas de la animación del corte de un caballo con* 6378 *tetraedros. En la dcha., en azul aristas de tetraedro intactas, y en rojo aristas que ya han sido cortadas.*

empleando detección de colisiones mediante la voxelización de las primitivas. La sección comienza con una breve descripción del proceso de corte.

4.1. Descripción del corte

En este experimento se consideran objetos deformables discretizados mediante mallas de tetraedros. La operación de corte del objeto deformable consiste en avanzar una curva de corte (es decir, el filo) y detectar los tetraedros del modelo que se seccionan. Las fuerzas internas del objeto deformable han de reconfigurarse en función de los tetraedros cortados. La simulación precisa de corte y fractura requiere otras operaciones tales como el remallado de la superficie, la rediscretización del modelo deformable, etc. [BG00, MBF04, SHGS06], pero este experimento se limita a evaluar la eficiencia de la detección de primitivas seccionadas.

En el experimento se ha utilizado como modelo deformable el model masa-muelle con conservación de volumen de Teschner *et al.* [THMG04]. Las fuerzas internas están compuestas por las proporcionadas por los muelles asociadas a las aristas de los tetraedros y por las fuerzas de conservación de volumen asociadas a los propios tetraedros. Como se reflaja en la Figura 4, al avanzar el filo de corte se han de detectar las aristas cortadas. Cuando se corta una arista se desactiva la fuerza de muelle correspondiente a dicha arista, así como las fuerzas de conservación de volumen de los tetraedros incidentes en la arista.

El filo de corte se representa en general como una curva en 3D, descompuesta en segmentos lineales. Para el experimento se ha utilizado un filo con un único segmento. En cada paso de muestreo de la simulación se desplaza el segmento de corte y se aproxima el área correspondiente al desplazamiento realizado mediante dos triángulos. Las aristas cortadas se pueden detectar calculando la intersección entre las aristas del modelo y los dos triángulos. Este test conlleva el cálculo de la intersección línea-plano junto con la comprobación del estado de la intersección en los rangos válidos de la arista y del triángulo utilizando coordenadas baricéntricas. Obviamente, se desea minimizar el número de intersecciones línea-plano a realizar, para lo cual se han empleado la partición espacial y el algoritmo de *spatial hashing*.

4.2. Test con voxelización eficiente

El test de corte de aristas efectúa los siguientes pasos:

1. Introducir en la partición espacial el segmento de corte en las posiciones actual y anterior, limitándose el avance del segmento en cada paso de muestreo a una celda de la partición.
2. Por cada arista del modelo, se introduce la arista en la partición espacial y se detecta si alguna de las celdas ocupadas lo está también por el segmento.
3. Si la arista y el segmento comparten alguna celda, se debe realizar el test entre la arista y los dos triángulos que representan el avance del segmento.

La resolución de la partición espacial se ha seleccionado de manera que cada arista del modelo ocupe como media una celda de la partición. En esta situación, el número de candidatos arista-segmento con los que se ha de realizar el test de intersección línea-triángulo depende en gran medida de las celdas de la partición ocupadas por el segmento de corte. Para minimizar dicho número de celdas se ha empleado el algoritmo descrito en la Sección 3.2 para la voxelización de segmentos. En principio, dicho algoritmo se podría utilizar también para introducir las aristas de la malla, pero el número de celdas ocupadas por las aristas es muy pequeño y la ganancia sería mínima.

4.3. Resultados y discusión

La Figura 5 compara el número de celdas ocupadas al insertar un segmento de corte en la partición, empleando la caja contendora del segmento (izqda.) o la voxelización eficiente de primitivas (izqda.). Como se puede apreciar, cuando el segmento de corte no está alineado con los ejes cartesianos, el número de celdas producido por la voxelización eficiente es notablemente menor, lo cual resulta en muchos menos candidatos arista-segmento a evaluar en detalle.

Se han aplicado ambas técnicas de inserción de segmen-

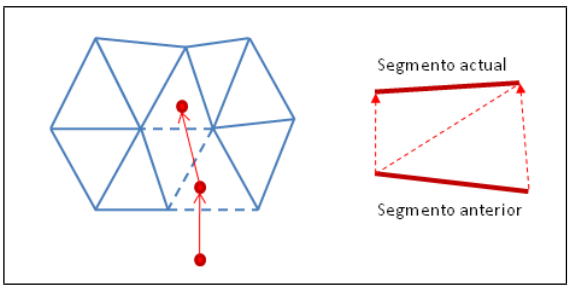

Figure 4: *Durante el corte progresivo, la superficie recorrida por el segmento de corte en un periodo de muestreo se aproxima mediante dos triángulos, y se comprueba la intersección entre dichos triángulos y las aristas de la malla.*

tos en la partición (mediante caja contenedora y por voxelización eficiente) en el corte progresivo del caballo deformable mostrado en la Figura 3. El caballo consta al comenzar la simulación de 6 378 tetraedros y 10 235 aristas. Si bien las imágenes (y el vídeo) muestran el corte combinado con deformación, en el estudio comparativo se ha mantenido el caballo en reposo para garantizar la reproducibilidad de los resultados. El corte se ha producido con un segmento oblicuo movido de forma horizontal a lo largo del caballo.

La Tabla 1 compara estadísticas de la simulación del corte del caballo insertando el segmento de corte mediante caja contenedora (columna central) y mediante voxelización de primitivas (columna dcha.). Como se puede apreciar, el número medio de celdas ocupadas es casi 1 000 veces menor voxelizando el segmento. Pero el dato más significativo es la reducción en el número de tests candidatos arista-segmento obtenidos de la partición: es 12 veces menor con voxelización eficiente, dando lugar a una notable reducción en el tiempo total de ejecución del corte.

	Caja Contenedora	*Voxeliz. Primitivas*
Celdas ocupadas	5 411	55
Tests candidatos	5 254	424
Tests positivos	0.6	0.6
T. medio	43	33

Table 1: *Estadísticas medias por fotograma, en el test de corte presentado en la Figura 3. La magnitud de los tiempos obtenidos viene expresada en ms.*

La conclusión alcanzada es que, en el problema del corte, la voxelización de primitivas proporciona una ganancia computacional muy alta cuando los segmentos de corte son relativamente largos respecto a la resolución definida en la malla y no se encuentran alineados con los ejes cartesianos.

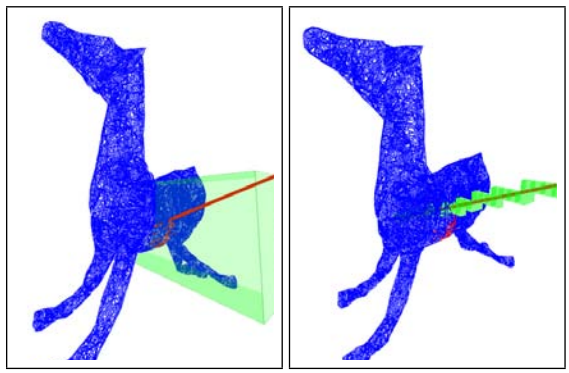

Figure 5: *En la izqda., volumen de la caja contenedora del segmento de corte; y en la dcha., celdas de la partición espacial obtenidas con la técnica de voxelización propuesta.*

5. Detección de colisiones continua

Esta sección describe la incorporación de la voxelización de primitivas en la detección de colisiones continua aplicada a la simulación de telas, así como los resultados obtenidos.

5.1. Test continuo entre vértices y facetas

La detección de colisiones continua entre dos objetos consiste en identificar el instante en que dichos objetos colisionan por primera vez [RKLM04]. Esta información es útil, por ejemplo, en la simulación de ropas, donde no se puede permitir que los objetos se intersequen [Pro95]. A nivel de primitivas de las mallas superficiales, el test continuo implica que para cada periodo de muestreo de la simulación se obtenga la detección de la posible intersección en las trayectorias de pares vértice-faceta o arista-arista, como se muestra en la Figura 6. En ambos casos, el test require la solución de una ecuación cúbica para obtener el instante de intersección [Pro95].

5.2. Test con voxelización eficiente

Se ha realizado un experimento de detección de colisiones continua entre las dos telas mostradas en la Figura 7, cada una con 5 000 triángulos. La detección se ha limitado a comprobar la intersección de vértices de la tela A con facetas de la tela B y viceversa, sin atender a la intersección entre aristas. Para minimizar el número de ecuaciones cúbicas a resolver, se han insertado las primitivas en una partición espacial. En concreto, se han seguido los siguientes pasos:

1. Introducir en la partición espacial las trayectorias de las facetas de A.
2. Por cada vértice de B, se introduce en la partición espacial el segmento trazado en un periodo de muestreo, y se detecta si alguna de las celdas ocupadas está ocupada también por alguna faceta de A.

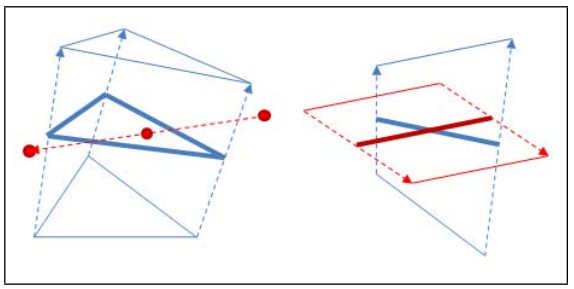

Figure 6: *Izqda: instante de intersección entre una faceta y un vértice; Dcha: instante de intersección entre dos aristas.*

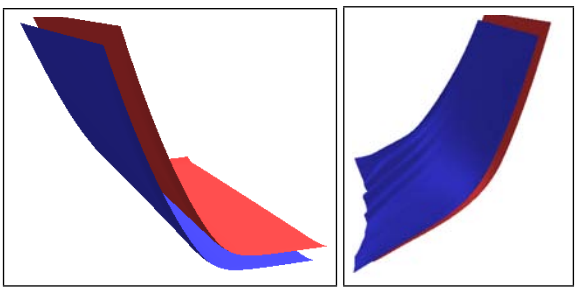

Figure 7: *Simulación de dos telas en movimiento con detección de colisiones continua. El test de intersección entre facetas y vértices sirve de prueba de rendimiento de la voxelizacón de primitivas.*

3. Si el vértice comparte alguna celda con alguna faceta, se realiza el test de intersección de sus trayectorias.
4. Se repite el mismo proceso intercambiando facetas y vértices de A y B.

La resolución de la partición espacial se ha seleccionado de manera que la trayectoria de cada faceta ocupe como media una celda de la partición. Se ha utilizado el algoritmo de voxelización de primitivas de la Sección 3.2 para la inserción de las trayectorias de los vértices. Cabe recalcar que, en este test, las trayectorias de los vértices dan lugar a segmentos.

5.3. Resultados y discusión

En este experimento, la utilización de la voxelización de segmentos no ha proporcionado prácticamente ninguna mejora de rendimiento. Pese a que el número de celdas de la voxelización es algo menor que el número de celdas obtenido mediante cajas envolventes, el número de tests candidatos faceta-vértice se reduce en menos de un 1 %. Es interesante analizar por qué no existe mejora en este caso. La hipótesis más factible es que el número de tests candidatos viene condicionado por el volumen ocupado en la partición espacial por las trayectorias de las facetas, no por las trayectorias de los vértices. La trayectoria de una faceta es una forma geométrica notablemente más compleja, y no existe un algoritmo de voxelización eficiente como los presentados en la Sección 3. La conclusión extraída es que cuando se considera el problema de detección de colisiones continua, la inserción de primitivas en la partición mediante voxelización precisa no es computacionalmente rentable.

6. Intersección de mallas de triángulos

Esta sección describe la aplicación de la voxelización de primitivas en el cálculo de la intersección de dos mallas de triángulos, así como las mejoras de rendimiento obtenidas.

6.1. Cálculo de la curva de intersección

La intersección de dos mallas de triángulos cerradas está definida por una curva (o varias) formada por segmentos rectos. Dichos segmentos están delimitados por los puntos de intersección entre aristas y facetas de ambas mallas, como se muestra en la Figura 8. En el caso de mallas deformables, el cálculo de los puntos de intersección se puede acelerar notablemente utilizando particiones espaciales y reduciendo así el número de candidatos arista-faceta a examinar en detalle.

6.2. Test con voxelización eficiente

El cálculo de los puntos de intersección entre dos mallas A y B requiere los siguientes pasos:

1. Introducir en la partición espacial las facetas de A.
2. Se introduce en la partición espacial cada arista de B y se detecta si alguna de las celdas ocupadas lo está también por alguna faceta de A.
3. Si la arista comparte alguna celda con alguna faceta, se realiza el test de intersección entre ambas.
4. Se repite el mismo proceso intercambiando facetas y aristas de A y B.

La resolución de la partición espacial se ha seleccionado de manera que cada faceta ocupe como media una celda de la partición.

6.3. Resultados y discusión

La Figura 9 muestra dos configuraciones de intersección entre modelos de una taza y una cuchara. La taza consta de 4 000 triángulos y 8 932 aristas, y la cuchara de 336 triángulos y 732 aristas. El tamaño de los triángulos de la escena es muy irregular, con lo cual no existe un tamaño de celda de partición que se adapte bien a todas las primitivas.

La Tabla 2 compara estadísticas de la intersección de los modelos de taza y cuchara insertando las aristas mediante caja contenedora (columna central) y mediante voxelización de primitivas (columna dcha.). Como se puede apreciar, el

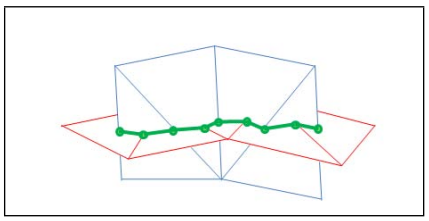

Figure 8: *La intersección de dos mallas de triángulos da lugar a una curva formada por segmentos rectos.*

número medio de celdas ocupadas es un 17 % mayor utilizando la caja contenedora. Igualmente, el número de tests candidatos arista-faceta aumenta un 40 %.

	Caja Contenedora	*Voxeliz. Primitivas*
Celdas ocupadas	25 156	21 505
Tests candidatos	7 983	5 677
Tests positivos	41	41
T. medio	61	47

Table 2: *Estadísticas medias por fotograma, en el test de intersección de superficies presentado en la Figura 9. La magnitud de los tiempos obtenidos viene expresada en ms.*

La conclusión extraída es que la voxelización de aristas proporciona ganancias de rendimiento considerables en la intersección de superficies cuando las mallas son irregulares.

7. Conclusiones

En este artículo se ha propuesto una técnica de detección de colisiones de objetos deformables basada en la voxelización eficiente de aristas combinada con *spatial hashing*. La combinación de ambas técnicas permite ahorrar gran cantidad de cálculos reduciendo el conjunto de colisiones candidatas si se utilizara solamente una de ellas.

Se ha analizado la respuesta de esta técnica comparándola con la de la utilización de una caja contenedora para el segmento de corte en tres casos bastante significativos en los que están involucrados objetos deformables: corte progresivo, detección de colisiones continua e intersección entre mallas de triángulos. Salvo en el segundo caso donde se ha obtenido un resultado neutro, los resultados han sido muy favorables a la técnica aquí planteada. Destacar que en el primer caso analizado, se ha logrado una mejora de más de un orden de magnitud en el número de tests candidatos a procesar.

Como trabajo futuro se plantea el análisis de la eficacia de esta técnica en la voxelización de facetas.

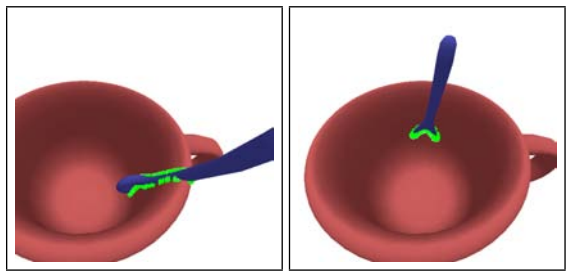

Figure 9: *Cálculo de la intersección entre los modelos de una taza y una cuchara, en dos configuraciones distintas. La voxelización de primitivas aporta ganancias de eficiencia en este escenario de mallas irregulares.*

Agradecimientos

Este trabajo ha sido financiado parcialmente con fondos del Ministerio de Educación y Ciencia (proy. TIN2007-67188) y por la Consejería de Educación de la Comunidad de Madrid (proy. S-0505/DPI/0235; GATARVISA).

References

[AW87] AMANATIDES J., WOO A.: A fast voxel traversal algorithm for ray tracing. In *Eurographics '87*. Elsevier Science Publishers, Amsterdam, North-Holland, 1987, pp. 3–10.

[Ber04] BERGEN G.: *Collision Detection in Interactive 3D Environments*. Morgan Kauffman Publishers, 2004. ISBN 1-55860-801-X.

[BG00] BIELSER D., GROSS M.: Interactive simulation of surgical cuts. In *Proc. of the Pacific Graphics 2000 Conference* (Oct. 2000), Niessen W., Viergever M., (Eds.), IEEE Computer Society Press, pp. 116–125.

[BNV06] BRUNET P., NAVAZO I., VINACUA A.: Modeling very complex geometric assemblies: The use of discrete and hierarchical models. *Computer Aided Desing & Applications 31*, 5 (2006), 639–644.

[Bre65] BRESENHAM J.: Algorithm for computer control of a digital plotter. *IBM Systems Journal 4*, 1 (1965), 25–30.

[CWZ*04] CHEN W., WAN H., ZHANG H., BAO H., PENG Q.: Interactive collision detection for complex and deformable models using programmable graphics hardware. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2004), ACM, pp. 10–15.

[ED06] EISEMANN E., DÉCORET X.: Fast scene voxelization and applications. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 71–78.

[Eri04] ERICSON C.: *Real-Time Collision Detection*. Morgan Kauffman Publishers, 2004. ISBN 1-55860-732-3.

[GASF94] GARCIA-ALONSO A., SERRANO N., FLAQUER J.: Solving the collision detection problem. *IEEE Comput. Graph. Appl. 14*, 3 (1994), 36–43.

[HK97] HE T., KAUFMAN A.: Collision detection for volumetric objects. In *VIS '97: Proceedings of the 8th conference on Visualization '97* (Los Alamitos, CA, USA, 1997), IEEE Computer Society Press, pp. 27–ff.

[HMG05] HASTINGS E., MESIT J., GUHA R.: Optimization of large scale, real-time simulations by spatial hashing. In *Proceedings of the 2005 Summer Computer Simulation Conference* (July 2005), vol. 37, The Soc. For. Model. & Sim., pp. 9–17.

[HTK*04] HEIDELBERGER B., TESCHNER M., KEISER R., MÜLLER M., GROSS M. H.: Consistent penetration depth estimation for deformable collision response. In *VMV* (2004), Girod B., Magnor M. A., Seidel H.-P., (Eds.), Aka GmbH, pp. 339–346.

[JTT01] JIMÉNEZ P., THOMAS F., TORRAS C.: 3D collision detection: a survey. *Computer & Graphics 25*, 2 (Apr. 2001), 269–285.

[KS87] KAUFMAN A., SHIMONY E.: 3d scan-conversion algorithms for voxel-based graphics. In *SI3D '86: Proceedings of the 1986 workshop on Interactive 3D graphics* (New York, NY, USA, 1987), ACM, pp. 45–75.

[LG98] LIN M. C., GOTTSCHALK S.: Collision detection between geometric models: A survey. In *Proceedings of IMA Conference on Mathematics of Surfaces* (1998), vol. 1, pp. 602–608.

[LM04] LIN M. C., MANOCHA D.: Collision and proximity queries. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, Goodman J. E., O'Rourke J., (Eds.). Chapman and Hall/CRC Press, New York, 2004, pp. 787–807.

[MBF04] MOLINO N., BAO Z., FEDKIW R.: A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. 23*, 3 (2004), 385–392.

[MGH06] MESIT J., GUHA R., HASTINGS E.: Multilevel sb collide: Collision and self-collision in soft body objects. In *Proceedings of the 8th International Conference on Computer Games: AI and Mobile Systems, CGAMES 2006* (July 2006).

[Pro95] PROVOT X.: Deformaiton constraints in a mass-spring model to describe rigid cloth behavior. In *Proc. Graphics Interface '95* (1995), pp. 147–154.

[PTTK07] PASSALIS G., THEOHARIS T., TODERICI G., KAKADIARIS I. A.: General voxelization algorithm with scalable gpu implementation. *Journal of Graphics Tools 12*, 1 (2007), 61–71.

[RKLM04] REDON S., KIM Y. J., LIN M. C., MANOCHA D.: Fast continuous collision detection for articulated models. In *SM '04: Proceedings of the ninth ACM symposium on Solid modeling and applications* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 145–156.

[SBT07] SPILLMANN J., BECKER M., TESCHNER M.: Non-iterative computation of contact forces for deformable objects. *Journal of WSCG 15*, 1–3 (2007), 33–40.

[SHGS06] STEINEMANN D., HARDERS M., GROSS M., SZEKELY G.: Hybrid cutting of deformable solids. In *VR '06: Proceedings of the IEEE conference on Virtual Reality* (Washington, DC, USA, 2006), IEEE Computer Society, p. 5.

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANTES D., GROSS M. H.: Optimized spatial hashing for collision detection of deformable objects. In *VMV* (2003), Ertl T., (Ed.), Aka GmbH, pp. 47–54.

[THMG04] TESCHNER M., HEIDELBERGER B., MÜLLER M., GROSS M.: A versatile and robust model for geometrically complex deformable solids. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 312–319.

[TKH*05] TESCHNER M., KIMMERLE S., HEIDELBERGER B., ZACHMANN G., RAGHUPATHI L., FUHRMANN A., CANI M.-P., FAURE F., MAGNENAT-THALMANN N., STRASSER W.: Collision detection in deformable objects. *Computer Grapchis Forum* (2005), 61–81.

[Tur90] TURK G.: *Interactive Collision Detection for Molecular Graphics*. Tech. Rep. 90-014, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1990.

[VKK*03] VARADHAN G., KRISHNAN S., KIM Y. J., DIGGAVI S., MANOCHA D.: Efficient max-norm distance computation and reliable voxelization. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 116–126.

[vKvdBT07] VAN KOOTEN K., VAN DEN BERGEN G., TELEA A.: *Point-based visualization of metaballs on a GPU*. Addison-Wesley, 2007, ch. 7.