CEIG - Spanish Computer Graphics Conference (2014)
Pere-Pau Vázquez and Adolfo Muñoz (Editors)

A System Proposal for Interactive Deformation of Large Medical Volumes

A. Rodríguez[1]* and A. León[1] and L. López[1] and M. García[1]

[1]Laboratorio de Realidad Virtual, Universidad de Granada, España

Abstract

In the field of volume deformation, an open research topic is the interactive and physically plausible deformation and rendering of large medical volumes. Many approaches to deform volumetric models have been proposed, offering a trade-off between realism and model resolution depending on the goal.
In this paper, we study the main techniques to deform volumetric models, focusing on the works that address interactive realistic deformation of large models and we outline the requirements needed to build an integrated system to interactively deform and visualize large volumes using the GPU. We also present a prototype of application that shows the viability of implementing such a system. For this prototype, we propose an enhanced deformation technique and a new fast deformed volume visualization scheme, assuring the system interactivity at any time.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms, I.3.1 [Computer Graphics]: Hardware Architecture—Parallel processing

1. Introducción

El campo de la deformación de volúmenes es un área de investigación con gran actividad en la que, desde hace varias décadas, se proponen diversos métodos de deformación de volúmenes para multitud de aplicaciones. Gibson et al. [GM97] presentaron un completo estado del arte de los distintos métodos existentes hasta la fecha en el campo de deformación de volúmenes. Meier et al. [MLM*05] presentaron otro estado del arte centrado en modelos de deformación aplicados a sistemas de simulación de procedimientos quirúrgicos.

Las técnicas propuestas, en función de la aplicación para la que se desarrollan, sacrifican resolución en los modelos con el objetivo de ofrecer mayor fidelidad biomecánica en las deformaciones o, por el contrario, ofrecen modelos de mayor resolución a expensas de perder realismo en la deformación. Entre las propuestas enfocadas en el desarrollo de sistemas aplicados a cirugía virtual, los métodos más empleados son los sistemas de simulación física FEM (*Finite Element Method*) y *Mass-Spring*. Por otra parte, en las propuestas enfocadas a mejorar la exploración de los datos originales en procesos de diagnóstico o herramientas de docencia, los métodos más empleados son los sistemas de deformación espacial, donde destacan las técnicas de FFD (*Free Form Deformation*), y los sistemas basados en restricciones geométricas, siendo el más destacado el algoritmo *ChainMail 3D*.

En este trabajo mostramos una propuesta de sistema que permite realizar deformaciones físicamente plausibles de grandes modelos volumétricos y visualizar de forma realista dichos modelos, teniendo como requisito fundamental la respuesta interactiva del sistema frente a las operaciones del usuario. Con este objetivo presentamos un análisis de los principales métodos empleados en este campo, centrándonos en las propuestas que hacen uso del algoritmo *ChainMail 3D*. Estudiando los problemas que presentan estas últimas propuestas, hemos extraído un conjunto de requisitos que debería cumplir un sistema interactivo de deformación y visualización realista de volúmenes.

Para comprobar la validez de nuestros requisitos, hemos desarrollado un prototipo de nuestro sistema que, aprovechando las últimas características disponibles en la programación de propósito general en GPU (GPGPU) mediante el uso de OpenCL, pone de manifiesto la viabilidad de crear tal sistema. Para la implementación del prototipo, proponemos una versión modificada del algoritmo *ChainMail 3D* y una

* Este trabajo ha sido parcialmente financiado por la Universidad de Granada bajo el programa "Formación de Profesorado Universitario del Plan Propio".

DOI: 10.2312/ceig.20141111

técnica ágil de remuestreo de los datos volumétricos adaptada para este algoritmo, junto con un visualizador de volúmenes basado en *Ray-casting*.

El presente trabajo se estructura de la siguiente forma: La sección 2 muestra un estudio de las principales técnicas que se han propuesto en el campo de deformación de volúmenes, realizando una clasificación en función del principal objetivo perseguido por dichas propuestas. La sección 3 muestra un análisis de los sistemas completos de deformación de volúmenes usando el algoritmo *ChainMail*, analizando los problemas que presentan. La sección 4 presenta nuestra propuesta de sistema, junto con los algoritmos desarrollados. La sección 5 presenta los resultados experimentales obtenidos con el prototipo desarrollado frente a los resultados ofrecidos por propuestas anteriores. Finalmente, la sección 6 concluye este trabajo recogiendo las aportaciones del mismo y las líneas de trabajo futuro posibles.

2. Trabajos Previos

A lo largo de los últimos años, se han propuesto varias técnicas de deformación de volúmenes médicos con la intención de mejorar la exploración de los datos o con la intención de crear herramientas de entrenamiento o enseñanza. Chen et al. [CCI*07] recopilaron en un extenso trabajo decenas de técnicas para deformar objetos definidos por muestreo discreto, entre los que destacan los volúmenes médicos.

En función de la meta que persiguen, se pueden diferenciar tres grandes grupos de técnicas de deformación de volúmenes médicos: técnicas de simulación física mediante integración temporal, técnicas de deformación espacial y técnicas basadas en restricciones geométricas.

2.1. Simulación física

Un importante conjunto de técnicas de deformación orientadas a conseguir el mayor realismo posible en las deformaciones aplican algoritmos de simulación física mediante integración temporal a modelos derivados de los datos volumétricos. Entre estas técnicas destacan los sistemas basados en el método de elementos finitos (FEM) y los sistemas basados en el modelo *Mass-Spring*.

2.1.1. Sistemas FEM

Los métodos basados en elementos finitos operan sobre el modelo tratándolo como un volumen continuo sobre el que se aplican fuerzas. Discretizando el volumen mediante una malla de nodos, se resuelve un sistema de ecuaciones derivadas de la teoría de la elasticidad para las posiciones de dichos nodos.

Este método ofrece deformaciones muy realistas, y se ha usado ampliamente en los sistemas de cirugía virtual. Sagar et al. [SBMH94] proponen un sistema de cirugía ocular empleando el modelo de elementos finitos para las deformaciones. Bro-Nielsen et al. [BNC96] establecen la teoría del modelo lineal de elementos finitos aplicado a deformación interactiva de tejidos, junto con un sistema de cirugía virtual empleando dicha teoría. Cotin et al. [CDA99] proponen un sistema de cirugía virtual con modelos de mayor resolución y comportamiento más realista precalculando deformaciones mediante elementos finitos. Müller et al. [MDM*02] proponen una técnica para aplicar grandes deformaciones a los sistemas lineales de elementos finitos, ya que estos presentaban artefactos visuales ante dichas deformaciones.

2.1.2. Sistemas Mass-Spring

En los sistemas basados en *Mass-Spring* el volumen se representa como un conjunto discreto de nodos con masa conectados por muelles, creando una malla para la deformación. De esta forma, las fuerzas se aplican sobre elementos discretos y se propagan por el volumen a través de las fuerzas que transmiten los muelles al deformarse, calculadas mediante la ley de Hooke. Estos sistemas son más simples de implementar y pueden trabajar con modelos más grandes que los basados en FEM, ya que las ecuaciones utilizadas para la resolución de la deformación son más sencillas pero, como contrapartida, ofrecen un menor grado de realismo, ya que el volumen se trata como un conjunto discreto de elementos, y las ecuaciones que los relacionan se definen de forma heurística.

A pesar de la dificultad de reproducir comportamientos reales de los tejidos, este método también se ha empleado en muchos sistemas de simulación de este tipo. Nedel et al. [NT98] proponen el uso de sistemas *Mass-Spring* para simular deformaciones musculares. Montgomery et al. [MBW01] emplean un sistema *Mass-Spring* para simular disecciones en ratas. Mollemans et al. [MSVCS03] proponen estructuras *Mass-Spring* tetraédricas para simular tejidos volumétricos.

Estas técnicas, y otras derivadas de las mismas, se basan en la integración temporal iterativa de funciones físicas definidas sobre los modelos y ofrecen una simulación física más próxima al comportamiento real de los tejidos a costa de trabajar con un conjunto de datos reducido, debido al coste computacional de dichos algoritmos. Varios trabajos recientes proponen versiones aceleradas de estas técnicas usando la GPU. Por ejemplo, Courtecuisse et al. [CJA*10] proponen una versión paralela de FEM aplicada a un sistema de cirugía virtual del hígado y Mosegaard et al. [MS05] proponen una versión paralela de *Mass-Spring* aplicada a cirugía virtual del corazón. Estas propuestas permiten incrementar la resolución de los modelos deformados, pero aún no ofrecen la velocidad necesaria para trabajar con los datos médicos originales, además de requerir una etapa de preprocesado supervisada costosa que impide su uso inmediato tras la adquisición de los datos.

2.2. Deformación espacial

Siguiendo una estrategia diferente, se han propuesto técnicas para aplicar las deformaciones sobre el espacio de los datos, en lugar de sobre los datos en sí. De este grupo de técnicas, destacan por su uso aquellas basadas en *Free-Form Deformation* (FFD). El funcionamiento básico de este método, presentado formalmente por Sederberg et al. [SP86], consiste en aplicar deformaciones locales o globales al espacio en el que se encuentra el modelo, de forma que al visualizar dicho espacio, el modelo se ve afectado por las deformaciones.

Esta técnica y sus posteriores mejoras y adaptaciones han sido muy usadas en el campo de la exploración y manipulación de modelos volumétricos con una resolución comparable a la del conjunto de datos original. Westermann et al. [WRS01] proponen operadores locales y globales de FFD para deformar volúmenes en tiempo real. McGuffin et al. [MTB03] proponen operadores de manipulación espacial para explorar datos volumétricos. Singh et al. [SSC03] proponen un esquema de deformación espacial basado en esqueletos para deformar volúmenes en tiempo real. Correa et al. [CSC06] proponen manipuladores espaciales para crear ilustraciones médicas a partir de volúmenes de manera interactiva.

Estas técnicas ofrecen esquemas de deformación muy rápidos ya que las funciones de deformación se aplican sobre el espacio y no sobre los datos en sí, pero, por el mismo motivo, no consiguen deformaciones acordes con las propiedades físicas de los modelos y provocan que las operaciones de deformación realizadas por el usuario y la respuesta obtenida no sean intuitivas.

2.3. Restricciones geométricas

Con el objetivo de mantener un comportamiento físico razonable en la deformación de los modelos, pero evitando el excesivo cálculo de integración de los esquemas de simulación, se han propuesto una serie de técnicas basadas en restricciones geométricas, siendo *ChainMail 3D* el algoritmo más empleado.

El algoritmo *ChainMail 3D*, originalmente propuesto por Gibson [Gib97], y posteriormente mejorado por Schill et al. [SGBM98] para soportar materiales heterogéneos, propone relacionar los elementos del volumen con sus vecinos mediante restricciones geométricas. De esta forma, el movimiento de un elemento provoca una reacción en cadena que resuelve las restricciones que se incumplen tras el movimiento de cada elemento.

Este esquema de deformación es muy apropiado para trabajar con los conjuntos de datos originales porque, aunque no consigue un realismo físico similar a los métodos de simulación de deformaciones del tipo FEM y *Mass-Spring* debido a que se basa en un modelo geométrico, tiene en cuenta las propiedades físicas del modelo y se puede aplicar a grandes volúmenes por su velocidad varios órdenes de magnitud superior a dichos métodos. A pesar de no seguir un esquema de simulación física, se ha aplicado con éxito a sistemas de cirugía virtual, como un sistema de cirugía ocular [SGBM98] o cirugía de rodilla [GSM*97], dado que realizar cambios topológicos en los modelos es muy sencillo mediante la modificación de enlaces entre vecinos y la eliminación de elementos. También se ha aplicado en otros contextos como la generación de ilustraciones médicas [MRH08]. Este esquema no necesita preprocesamiento complejo, ya que se puede aplicar directamente sobre los datos originales, deduciendo las relaciones entre elementos y el comportamiento físico de los mismos en función de las distintas densidades de los elementos.

El los últimos trabajos sobre la deformación de volúmenes, los autores se han planteado aprovechar la potencia de cálculo de las GPUs para incrementar la respuesta interactiva de las deformaciones. Siguiendo esta idea, Schulze et al. [SBH07] presentan un sistema basado en *ChainMail 3D* que permite manipular grandes volúmenes de elementos y utiliza la GPU para realizar una visualización mediante *Raycasting*. Rossler et al. [RWE08] presentan un sistema similar, proponiendo una versión paralela del algoritmo *Chainmail 3D*, de forma que todo el cauce de trabajo se ejecuta en la GPU.

Tras analizar estos trabajos previos, hemos constatado que para poder trabajar de forma interactiva con conjuntos de datos volumétricos sin sacrificar resolución, cosa que hasta la fecha no es posible con los métodos FEM y *Mass-Spring*, es razonable optar por un método basado en restricciones geométricas para nuestro sistema, ya que además, permite realizar una simulación físicamente plausible, cosa que no obtenemos claramente con los métodos FFD. En la siguiente sección presentamos los fundamentos del algoritmo *ChainMail 3D* y proponemos un conjunto de requisitos que debería cumplir un sistema interactivo de deformación físicamente plausible de grandes volúmenes que permita además una visualización realista del modelo.

3. Deformaciones interactivas con ChainMail 3D

El algoritmo *ChainMail* [Gib97] define una estructura de malla sobre los elementos volumétricos del modelo. Cada elemento se conecta con sus 6 vecinos adyacentes. Para cada vecino se define una región válida de movimiento de forma que, mientras un vecino permanezca dentro de esa región, el algoritmo considera que cumple las restricciones. En el momento en el que sale de dicha región, se violan las restricciones y esto obliga a una reubicación de vecinos con el objetivo de que se cumplan de nuevo las restricciones.

Bajo este esquema, representado en la Fig. 1, el proceso de propagación de la deformación consiste en que cuando un elemento se desplaza, comprueba secuencialmente si

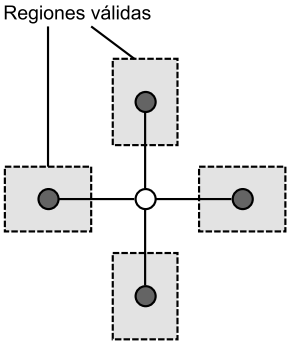

Figura 1: *Representación en 2D de la estructura de vecinos generada para el algoritmo ChainMail. El elemento central se conecta a sus 4 vecinos adyacentes, y define una región de posicionamiento válida para cada uno.*

sus vecinos respetan las restricciones y, en caso negativo, los desplaza a sus respectivas regiones válidas. Cuando estos vecinos son desplazados, pueden a su vez provocar nuevas violaciones de restricciones y provocar nuevos desplazamientos de vecinos, de manera que la deformación se propaga por el volumen como se puede ver en la Fig. 2. El orden de resolución de restricciones propuesto en el trabajo original garantiza que cada elemento sólo se desplaza una vez, con lo que no es necesario revisitar elementos.

Una vez resueltas las restricciones, se introduce una etapa de relajación que desplaza iterativamente los elementos a sus posiciones de equilibrio en el punto medio de sus vecinos. Definiendo las restricciones de los elementos y el esquema de relajación, este método permite simular comportamientos elásticos, plásticos y rígidos.

El algoritmo original no opera correctamente con materiales heterogéneos, por lo que Schill et al. [SGBM98] proponen el algoritmo *Enhanced ChainMail*, una versión mejorada que modifica el orden de resolución de las restricciones en función de la gravedad de la violación de restricción, de manera que la deformación se propaga antes por medios más rígidos, permitiendo de esta manera gestionar modelos volumétricos heterogéneos.

El sistema propuesto por Schulze et al. [SBH07] opera sobre grandes volúmenes médicos a máxima resolución haciendo uso del algoritmo *Enhanced ChainMail* para deformar los modelos y empleando un algoritmo de visualización de volúmenes basado en *Ray-casting* en GPU. No obstante, el sistema presenta varios problemas que le restan interactividad:

- El algoritmo de deformación está implementado en CPU, mientras que la visualización se realiza en la GPU, por lo que la transferencia de información introduce un retardo al trabajar con grandes cantidades de información.
- El tiempo necesario para resolver una deformación depende del número de elementos involucrados. Esto provoca que en deformaciones que impliquen un gran número de elementos (grandes desplazamientos al realizar una operación de deformación), el sistema pierda interactividad mientras se resuelve dicha deformación.
- Para visualizar las deformaciones, los autores proponen un esquema de remuestreo mediante un algoritmo de búsqueda ejecutado en la CPU, que ofrece una representación realista de las deformaciones, pero que, de la misma forma que en el caso anterior, al realizar deformaciones que impliquen un gran número de elementos, el tiempo requerido para realizar el remuestreo y posterior envío a la memoria de la GPU, reduce en gran medida la interactividad del método.

Rossler et al. [RWE08] proponen un sistema similar para el que emplean una implementación paralela del algoritmo *Chainmail 3D* que se ejecuta completamente en la GPU, de forma que se elimina el problema de la transferencia de información entre memoria principal y la memoria de la GPU, pero los otros dos problemas siguen presentes, por lo que el sistema pierde la velocidad necesaria para mostrar resultados interactivamente para grandes deformaciones.

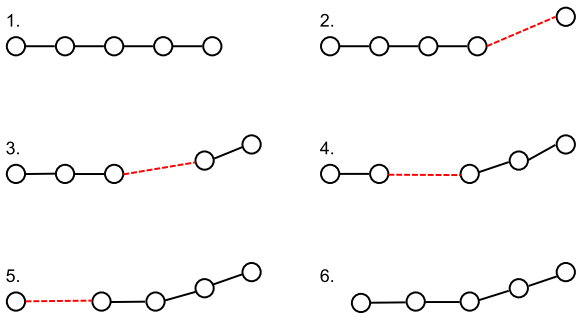

Figura 2: *Ejemplo de propagación de deformaciones con ChainMail. Tras el movimiento del elemento de la derecha, se viola la restricción geométrica con su vecino izquierdo, lo que produce una reacción en cadena de movimientos.*

Estos problemas, presentes en las propuestas de Schulze y Rossler, provocan que el sistema sólo responda de manera realmente interactiva para deformaciones relativamente pequeñas, ya que para deformaciones mayores, el sistema tarda varios segundos en ofrecer *feedback* visual ante la interacción del usuario. Este es un problema importante ya que, por lo general, la aplicación de una deformación deseada se consigue aplicando deformaciones sucesivas sobre el mismo o distintos elementos del volumen, siendo necesario tener *feedback* visual durante el proceso completo. Teniendo en cuenta nuestro análisis de las propuestas que consiguen acercarse más a un comportamiento interactivo en la deformación de grandes volúmenes, y sin perder de vista el problema de las grandes deformaciones en las soluciones aportadas, planteamos una serie de requisitos que debe cumplir

el diseño de un sistema para la deformación interactiva y físicamente plausible de grandes volúmenes

3.1. Requisitos del sistema

Tras analizar los sistemas propuestos hasta la fecha, y manteniendo las características que a priori podemos considerar solventadas en los sistemas estudiados, enumeramos los requisitos que debería cumplir un sistema de deformación y visualización de grandes volúmenes interactivo:

1. El sistema debe trabajar con una resolución similar, si no igual, a la del conjunto de datos original, reduciendo al mínimo el preprocesamiento necesario.
2. El sistema debe operar sobre los datos en la GPU directamente, ya que dado el volumen de información manipulado, la necesidad de transferencia entre memoria principal y memoria de la GPU impide la interactividad del sistema.
3. El sistema debe ser interactivo en todo momento, independientemente de que la deformación aplicada presente un carácter local o requiera un desplazamiento importante de parte del volumen.
4. El sistema debe mostrar *feedback* visual de las deformaciones interactivamente para facilitar la labor de deformación.

4. Propuesta de sistema

Tras identificar los requisitos que se han considerado fundamentales, así como definir el método de deformación que más se ajusta a los mismos, presentamos un prototipo de nuestro sistema que nos permite validarlos. Para realizar todas las operaciones en la GPU, aprovechando la potencia de cálculo de la misma y eliminando el problema de transferencia de información entre memoria principal y memoria de GPU, se ha optado por utilizar el framework OpenCL en la implementación de los distintos componentes del sistema. A continuación, presentamos una breve descripción de OpenCL, así como una descripción de los algoritmos que se han diseñado para implementar el prototipo.

4.1. OpenCL

OpenCL [SGS10] es un framework de desarrollo de aplicaciones de propósito general en entornos heterogéneos de computación con distintos tipos de dispositivos de procesamiento. En el caso de usarse para realizar operaciones en la GPU, el funcionamiento se puede simplificar de la siguiente manera.

Una aplicación consistirá en un *host*, encargado de gestionar la ejecución, y uno o más dispositivos, encargados de realizar el cálculo que les indique el *host*. En nuestro caso, la CPU actuará como *host* de la aplicación y se encargará de preparar el contexto de los dispositivos y de gestionar el cálculo en el dispositivo. Para esto, el *host* define búferes en la memoria de la GPU, que posteriormente inicializa con los valores necesarios. Estos búferes son *arrays* de datos que ofrecen el soporte para el almacenamiento de la entrada/salida de las funciones *kernel*, que son el código que se ejecuta en la GPU siguiendo un esquema SIMD. Cuando el *host* solicita la ejecución de un *kernel*, se lanzan tantas hebras en la GPU como elementos de cómputo se hayan indicado, y cada hebra de GPU ejecutará el código del *kernel* sobre los datos indicados, realizando esta ejecución en paralelo. Para esto, cada hebra de ejecución puede consultar su identificador dentro del conjunto, lo que le permitirá saber sobre qué datos debe operar.

OpenCL permite gestionar de manera explícita los diferentes espacios de memoria de la GPU (global, constante, local y privada), así como permite gestionar de manera explícita el número de hebras lanzadas y el esquema de agrupación de las mismas, de manera que se pueden optimizar estas configuraciones en función del algoritmo a ejecutar. De igual forma, permite realizar operaciones atómicas desde los *kernels* sobre la memoria de la GPU, y se pueden realizar transferencias de datos entre el *host* y el dispositivo en cualquier momento de la ejecución. Estas características hacen posible explotar al máximo la potencia de cálculo de las tarjetas gráficas modernas para una gran variedad de aplicaciones.

4.2. Deformación Interactiva

Para desarrollar el método de deformación interactiva en GPU, partimos del algoritmo propuesto por Rossler en [RWE08]. Este algoritmo, al igual que el algoritmo *ChainMail 3D* original, divide las deformaciones en dos etapas.

La primera es la etapa de propagación, que modifica la idea original de *ChainMail 3D* a la hora del cumplimiento de las restricciones geométricas para que el algoritmo sea paralelizable. En este caso, se comprueba para cada elemento si un vecino se ha movido en la iteración anterior y, si es así, se recoloca el elemento para que cumpla la restricción geométrica existente con su vecino. Este proceso se repite iterativamente hasta que todos los elementos respetan las restricciones geométricas del modelo. Una vez acaba este proceso, se ejecuta la etapa de relajación, similar a la original, pero en paralelo.

Esta solución hace que el sistema no muestre *feedback* visual ni permita una nueva interacción hasta que todas las etapas de la deformación han terminado, como se indica en la Fig. 3. Esto provoca que, al igual que con el algoritmo original, las deformaciones grandes dejen el sistema en un estado inoperable durante un tiempo variable. Por consiguiente, no se obtiene la respuesta interactiva necesaria en la fase de deformación, salvo para deformaciones relativamente pequeñas.

Para resolver este problema, proponemos una versión mejorada del algoritmo. Empleando un *flag* de actividad por

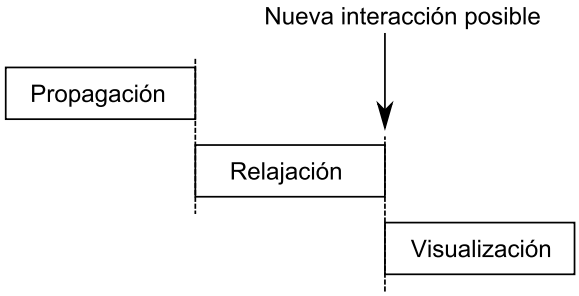

Figura 3: *Cauce de deformación del algoritmo propuesto por Rossler. El resultado de las deformaciones se visualiza una vez que todo el proceso de deformación ha terminado. Una nueva interacción sobre el modelo, de igual manera, solo es posible una vez ha acabado el cauce de deformación completo.*

cada elemento para llevar un control de aquellos que ya han sido alcanzados por la propagación, subdividimos las etapas de propagación y relajación en iteraciones.

Este *flag* indicará si la propagación provocada por la última interacción del usuario ha alcanzado a cada elemento. Empleando esta información, conseguimos distinguir cada una de las iteraciones de las etapas de propagación y relajación, de manera que las iteraciones de relajación se aplican a los elementos que ya cumplen las restricciones geométricas y han sido alcanzados por la propagación.

De esta manera, empleando dos copias de la información de deformación al igual que en [RWE08], una para leer y otra para escribir (al ser un proceso paralelo hay que garantizar el acceso a los datos correctos, ya que se pueden producir lecturas y escrituras concurrentes en caso de no emplear copias diferentes), las iteraciones de deformación crean una hebra por cada elemento. Cada hebra comprueba si algún vecino del elemento se ha movido en la iteración anterior y, en caso afirmativo, el elemento se marca como alcanzado por la propagación, comprueba la restricción con dicho vecino y en caso necesario se desplaza para cumplir la restricción y se marca como movido.

De manera similar, las iteraciones de relajación crean una hebra por cada elemento que comprueba si todos sus vecinos han sido ya alcanzados por la propagación actual y en caso afirmativo, aplican la función de relajación (explicada en [Gib97]). Al final de cada iteración, ya sea de propagación o de relajación, la copia de información empleada para escribir pasa a ser la copia de lectura y viceversa.

De esta forma, nuestro método permite solapar ambos procesos, y permite introducir visualizaciones intermedias sin necesidad de esperar a la completa finalización de las etapas, pudiendo decidir el número de iteraciones de cada etapa que se deben ejecutar antes de cada visualización. Este nuevo esquema se representa en la Fig. 4.

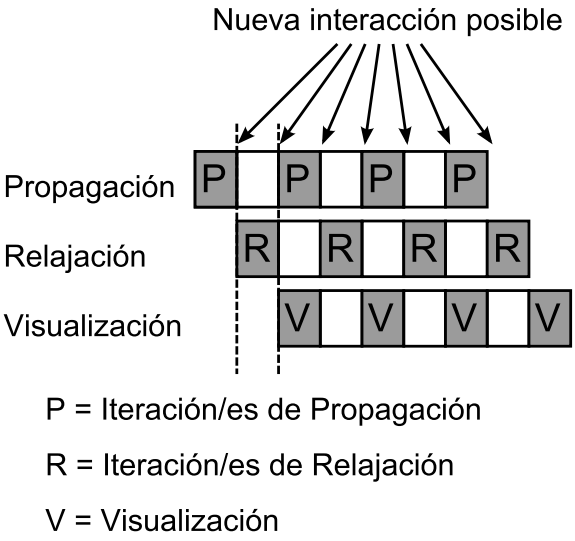

Figura 4: *Al subdividir las etapas del cauce en iteraciones, se consigue obtener visualizaciones intermedias durante las deformaciones. De igual manera, es posible aplicar nuevas interacciones sobre el modelo en cualquier punto del cauce de deformación al eliminar la secuencialidad entre las etapas.*

Esta subdivisión de las etapas en iteraciones permite obtener la interactividad deseada en el sistema, ya que se pueden visualizar estados intermedios de las deformaciones al no ser necesario esperar a que se propague completamente una deformación. Estas visualizaciones intermedias pueden no ser consistentes con el modelo ya que, al no haber acabado la propagación, no todos los elementos cumplirán las restricciones. No obstante, al propagarse la deformación desde el punto de aplicación de la misma, la zona cercana al punto de aplicación de la deformación, que se considera la zona de interés en ese momento, presenta un estado consistente en pocas iteraciones, de manera que se visualizan los resultados de la deformación en la zona de interes con mayor interactividad, permitiendo además la aplicación de una nueva deformación sin necesidad de esperar a que acabe la anterior, por lo que se agiliza el proceso de interación por parte del usuario.

4.3. Visualización Interactiva

Para garantizar el *feedback* visual durante las deformaciones, es necesaria una técnica de visualización que permita convertir el modelo deformado en cada iteración a una representación adecuada para visualización realista en tiempo real en GPU. Como se explica en [RWE08], la visualización de volúmenes deformados aplicando las deformaciones en un paso previo a la consulta de los datos proporciona la mejor visualización, ya que los datos originales permanecen

inalterados, pero se requiere invertir la función de deformación.

Esta función inversa es fácil de obtener cuando las deformaciones aplicadas son analíticas, como es el caso de las FFD, pero en el caso de deformaciones físicas basadas en las propiedades de los materiales, las inversas de las funciones de deformación no están definidas de forma analítica. Para aproximar este resultado, en [RWE08] proponen un algoritmo de optimización que aproxima la deformación inversa, pero este método es costoso computacionalmente y puede caer en mínimos locales, lo que dificulta el *feedback* visual de resultados intermedios durante el proceso de deformación.

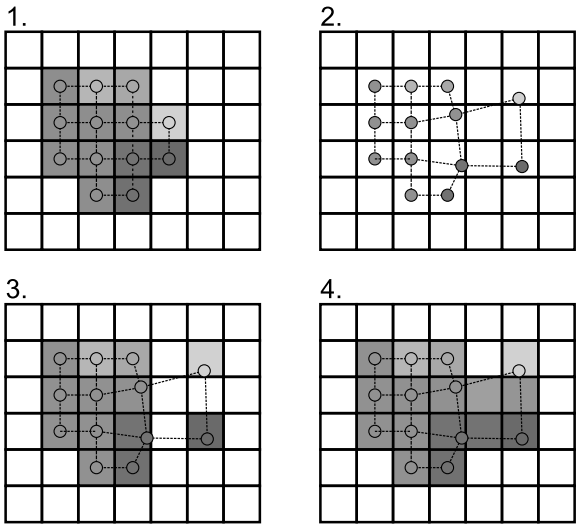

Figura 5: *Simplificación 2D del algoritmo de remuestreo. En 1) se representa la rejilla original junto con la estructura ChainMail subyacente. En 2) se realiza una deformación sobre los elementos, que en 3) son remuestreados sobre la rejilla, y en 4) se rellenan los huecos que aparecen.*

Para nuestro sistema proponemos un algoritmo paralelo de remuestreo del modelo, de manera que en las etapas intermedias se puedan visualizar los cambios que se van produciendo. En la implementación de este algoritmo empleamos las operaciones atómicas de OpenCL y la capacidad de escribir sobre una textura 3D.

Se define una rejilla regular uniforme inicializada con densidades cero de la misma resolución que la de los datos originales, y se lanza una hebra por cada elemento del modelo deformable. Cada hebra comprueba en qué *vóxel* de la rejilla cae dicho elemento y realiza la operación atómica MAX sobre la densidad del *vóxel* con la densidad del elemento, de manera que si varios elementos caen en el mismo *vóxel*, el de mayor densidad impone su valor, lo cual garantiza que, en caso de compresiones, los materiales más rígidos (de mayor densidad) pierden menos volúmen que los más elásticos.

Tras esto, se emplea la información de vecindad entre elementos para rellenar *vóxeles* intermedios. Para esto, cada una de las hebras comprueba para cada vecino en las direcciones crecientes de cada dimensión (en el caso 2D se comprueban los vecinos derecha y arriba, ya que los vecinos abajo e izquierda serán comprobados por otras hebras) el *vóxel* en el que cae, y sobre los *vóxeles* intermedios en la dirección del vecino se realiza la operación atómica MAX con la densidad interpolada de ambos elementos. La Fig. 5 ilustra un ejemplo simplificado en 2D del funcionamiento de este algoritmo, en el que se puede ver la configuracion inicial, con los elementos del modelo deformable centrados en los vóxeles correspondientes y, tras la aplicación de una deformación, se da valor de densidad a los vóxeles de la rejilla vacía en los que caen los elementos del modelo deformable y finalmente se da valor a los vóxeles intermedios entre vecinos.

Esta etapa de remuestreo se puede aplicar tras cualquier iteración de propagación o relajación, empleando como entrada la copia de los datos que dicha iteración haya usado como escritura. Una vez se obtienen los nuevos valores de los *vóxeles*, se escriben en una textura 3D, sobre la que se realiza una visualización mediante *Ray-casting* también implementada en OpenCL.

4.3.1. Precisión de Subvóxel

Un problema del muestreo es que al trabajar a nivel de *vóxel*, las deformaciones no se visualizan hasta que los elementos del modelo deformable pasan de un *vóxel* a otro. Esto provoca que las deformaciones que ocurren dentro de un *vóxel* no se visualizan y el resultado final de una deformación presenta una forma escalonada. Este efecto se puede apreciar en la Fig. 6.

Para paliar este problema, proponemos una técnica para alcanzar precisión de subvóxel. La idea es aplicar la técnica de la deformación inversa propuesta en [RWE08], pero de forma localizada dentro de cada *vóxel*, incluyendo en los datos del *vóxel* la inversa de la distancia entre el centro del *vóxel* y el elemento que da valor a dicho *vóxel*. Empleando esta técnica, cuando se realiza el proceso de *Ray-casting* sobre el volúmen, en una primera consulta, se lee este desplazamiento, que se aplica a la posición de muestreo del rayo y se toma una nueva muestra en la posición desplazada. De esta manera, el paso de un elemento por un *vóxel* se visualiza de forma suavizada, mejorando el *feedback* visual de la deformación y reduciendo el escalonamiento como se puede comprobar en la Fig. 7.

Esta técnica de visualización provoca pérdidas de información volumétrica debido al proceso de muestreo, pero garantiza el *feedback* necesario durante la aplicación de las deformaciones. Una vez que la deformación se completa, se puede aplicar un proceso más preciso de visualización de deformaciones, que mantenga toda la información original.

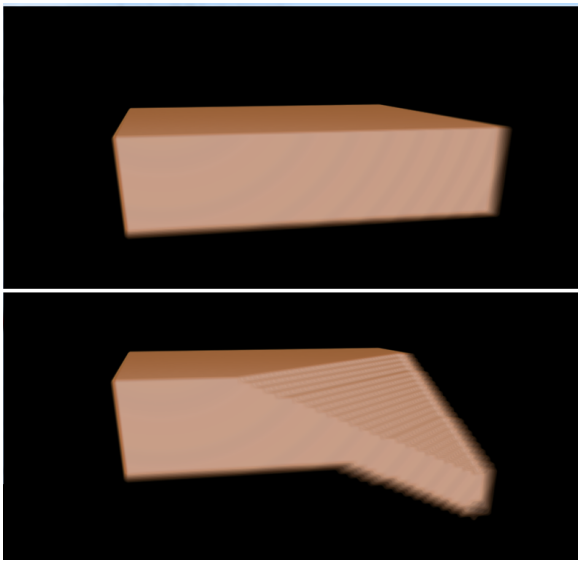

Figura 6: *Visualización de una deformación mediante remuestreo de vóxeles donde se aprecia la visualización escalonada.*

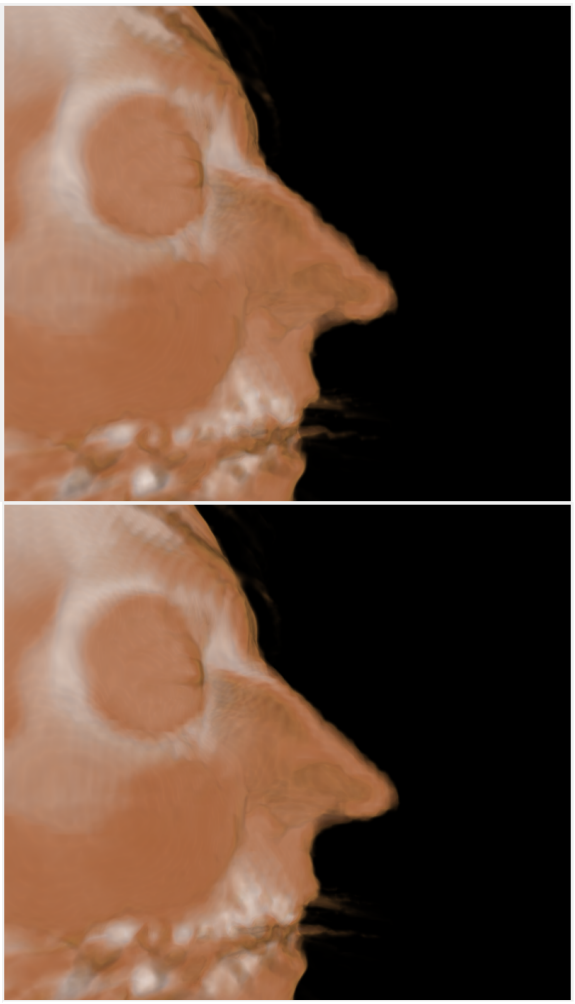

Figura 7: *Ejemplo de la mejora visual obtenida al usar precisión de sub-vóxel. El mismo modelo deformado se visualiza sin emplear la técnica (arriba) y empleando la técnica (abajo).*

5. Resultados

Los métodos descritos en la sección anterior, junto con un método de selección similar al propuesto en [SBH07] se han integrado en un prototipo de sistema para comprobar experimentalmente la interactividad de las operaciones de deformación.

También se ha implementado una versión del algoritmo de deformación propuesto en [RWE08] con objeto de comparar ambos sistemas.

Para realizar las pruebas se ha diseñado un experimento, consistente en aplicar deformaciones sucesivas a un volúmen sintético de tamaño $144 \times 144 \times 144$, variando el número de elementos afectados por dichas deformaciones. En cada caso, se ha medido el tiempo que el algoritmo de deformación tarda en devolver el control al sistema, así como los *frames* por segundo (FPS) que el sistema completo, incluyendo la visualización, consigue en cada caso.

Las pruebas han sido realizadas en un equipo con procesador Intel Core i5-3570 (3.4 GHz) y una tarjeta gráfica AMD Radeon R9 270X.

En la tabla 1 se muestran los resultados del experimento usando el algoritmo de deformación propuesto en [RWE08], donde se aprecia que a medida que se incrementa el número de elementos afectados por la deformación, el sistema pierde interactividad drásticamente debido al tiempo que el algoritmo de deformación permanece en ejecución.

En la tabla 2 se muestran los resultados del mismo test usando nuestro algoritmo de deformación. En este caso, se puede apreciar que se mantiene estable el número de imágenes generadas por segundo, ya que el algoritmo de deformación devuelve el control al sistema con una frecuencia independiente del tamaño de la deformación, lo cual permite una interacción mucho más ágil sobre los datos.

La gráfica de la Fig. 8 muestra el número de *frames* por segundo que se obtienen en función del número de elementos afectados por la deformación. Claramente se puede observar que incluso para un número bajo de elementos afectados por la deformación, el algoritmo de Rossler no consigue un número de *frames* por segundo aceptable, decrementándose este ratio conforme aumenta el número de elementos afectados, hasta una cantidad que no permite la interactividad del sistema de deformación. Sin embargo, nuestro méto-

Tabla 1: *Resultados del experimento para el algoritmo de deformación propuesto en [RWE08].*

Elementos deformados	Tiempo en fase de deformación (ms)	FPS del sistema
4.096	16	34.7
32.768	43	18.3
262.144	168	6.2
2.097.152	837	1.2
2.985.984	1055	0.7

Tabla 2: *Resultados del experimento para nuestro algoritmo de deformación.*

Elementos deformados	Tiempo en fase de deformación (ms)	FPS del sistema
4.096	2	61
32.768	3	61
262.144	5	60.2
2.097.152	8	55.4
2.985.984	10	52.3

do permite mantener de forma estable el número de *frames* por segundo, independientemente del número de elementos afectados por la deformación.

Para probar el sistema sobre un volúmen médico real, se ha simulado una incisión sobre una sección de una pierna. El volúmen empleado, que se puede ver en la Fig. 9 tiene un tamaño de $256 \times 255 \times 256$ elementos. Durante todo el proceso de deformación, el sistema ha respondido siempre entre 8 y 25 FPS, permitiendo refinar interactivamente el corte, que se puede apreciar en la Fig. 9.

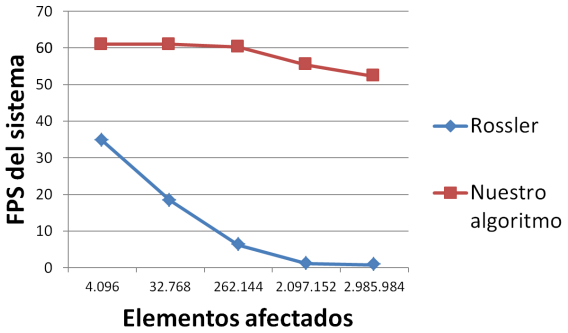

Figura 8: *Gráfica comparativa de los frames por segundo mantenidos por ambos algoritmos en función del número de elementos afectados por la deformación.*

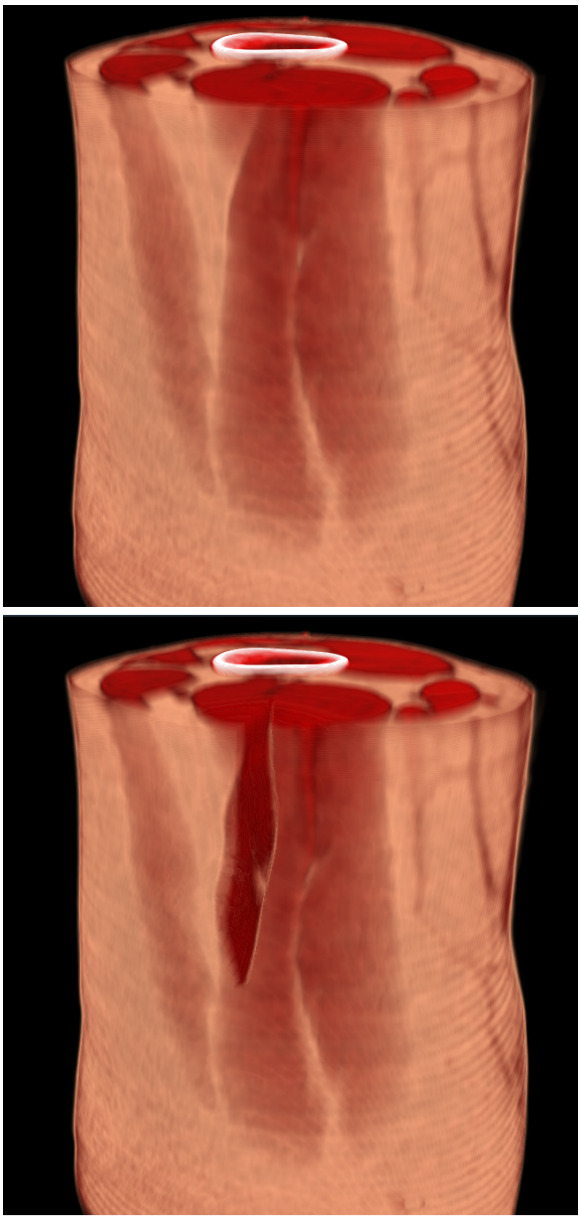

Figura 9: *Simulación de incisión sobre volúmen médico. Arriba se ilustra el volúmen original, una sección de pierna formada por* $256 \times 255 \times 256$ *elementos. Abajo, el resultado de generar interactivamente la incisión sobre el volúmen.*

6. Conclusiones y trabajos futuros

En este trabajo hemos recopilado las técnicas de deformación de volúmenes más empleadas en distintos ámbitos, centrando la atención en el algoritmo *ChainMail*, que proporciona deformaciones físicamente plausibles en tiempo razonable, capaz de trabajar sobre datos volumétricos médicos a máxima resolución. Hemos analizado los sistemas propuestos que emplean esta técnica, estudiando los problemas que presentan, y hemos esbozado los requisitos que debería cumplir un sistema completamente interactivo.

Hemos presentado un prototipo de sistema que, cumpliendo estos requisitos, garantiza la interactividad en todo momento. Para ello, hemos propuesto un algoritmo de deformación iterativo paralelo basado en *ChainMail* más rápido que propuestas anteriores y que permite visualizar resultados intermedios. También hemos propuesto un sistema de remuestreo de volúmenes paralelo para permitir la visualización interactiva de las deformaciones haciendo uso de las últimas características ofrecidas por los lenguajes de GPGPU.

Tras comprobar la viabilidad de un sistema de estas características, queremos extender el prototipo para gestionar volúmenes de mayor tamaño.

De igual forma queremos mejorar el algoritmo de deformación para trabajar de forma más rápida y aproximar mejor las propiedades físicas de los modelos, ya que el algoritmo propuesto en [RWE08], y por consiguiente el nuestro, no gestiona eficientemente los materiales heterogéneos. Además, queremos implementar las operaciones de modificación topológica descritas originalmente para el algoritmo e integrarlas en el sistema.

También queremos mejorar el algoritmo de visualización, ya que actualmente genera artefactos visuales que dificultan el análisis de los datos, y sólo ofrece una visualización básica basada en acumulación de luz por *Ray-casting*.

References

[BNC96] BRO-NIELSEN M., COTIN S.: Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer graphics forum* (1996), vol. 15, Wiley Online Library, pp. 57–66. 2

[CCI*07] CHEN M., CORREA C., ISLAM S., JONES M. W., SHEN P.-Y., SILVER D., WALTON S. J., WILLIS P. J.: Manipulating, deforming and animating sampled object representations. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 824–852. 2

[CDA99] COTIN S., DELINGETTE H., AYACHE N.: Real-time elastic deformations of soft tissues for surgery simulation. *Visualization and Computer Graphics, IEEE Transactions on 5*, 1 (1999), 62–73. 2

[CJA*10] COURTECUISSE H., JUNG H., ALLARD J., DURIEZ C., LEE D. Y., COTIN S.: Gpu-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in Biophysics and Molecular Biology 103*, 2 (2010), 159–168. 2

[CSC06] CORREA C., SILVER D., CHEN M.: Feature aligned volume manipulation for illustration and visualization. *Visualization and Computer Graphics, IEEE Transactions on 12*, 5 (2006), 1069–1076. 3

[Gib97] GIBSON S. F.: 3d chainmail: a fast algorithm for deforming volumetric objects. In *Proceedings of the 1997 symposium on Interactive 3D graphics* (1997), ACM, pp. 149–ff. 3, 6

[GM97] GIBSON S. F., MIRTICH B.: A survey of deformable modeling in computer graphics. *MERL Internal Report* (1997). 1

[GSM*97] GIBSON S., SAMOSKY J., MOR A., FYOCK C., GRIMSON E., KANADE T., KIKINIS R., LAUER H., MCKENZIE N., NAKAJIMA S., ET AL.: Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback. In *CVRMed-MRCAS'97* (1997), Springer, pp. 367–378. 3

[MBW01] MONTGOMERY K., BRUYNS C., WILDERMUTH S.: A virtual environment for simulated rat dissection: a case study of visualization for astronaut training. In *Proceedings of the conference on Visualization'01* (2001), IEEE Computer Society, pp. 509–514. 2

[MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), ACM, pp. 49–54. 2

[MLM*05] MEIER U., LÓPEZ O., MONSERRAT C., JUAN M. C., ALCANIZ M.: Real-time deformable models for surgery simulation: a survey. *Computer methods and programs in biomedicine 77*, 3 (2005), 183–197. 1

[MRH08] MENSMANN J., ROPINSKI T., HINRICHS K.: Interactive cutting operations for generating anatomical illustrations from volumetric data sets. 3

[MS05] MOSEGAARD J., SØRENSEN T. S.: Gpu accelerated surgical simulators for complex morphology. In *Virtual Reality, 2005. Proceedings. VR 2005. IEEE* (2005), IEEE, pp. 147–153. 2

[MSVCS03] MOLLEMANS W., SCHUTYSER F., VAN CLEYNENBREUGEL J., SUETENS P.: Tetrahedral mass spring model for fast soft tissue deformation. In *Surgery Simulation and Soft Tissue Modeling*. Springer, 2003, pp. 145–154. 2

[MTB03] MCGUFFIN M. J., TANCAU L., BALAKRISHNAN R.: Using deformations for browsing volumetric data. In *Visualization, 2003. VIS 2003. IEEE* (2003), IEEE, pp. 401–408. 3

[NT98] NEDEL L. P., THALMANN D.: Real time muscle deformations using mass-spring systems. In *Computer Graphics International, 1998. Proceedings* (1998), IEEE, pp. 156–165. 2

[RWE08] RÖSSLER F., WOLFF T., ERTL T.: Direct gpu-based volume deformation. *Proceedings of Curac* (2008), 65–68. 3, 4, 5, 6, 7, 8, 9, 10

[SBH07] SCHULZE F., BÜHLER K., HADWIGER M.: Interactive deformation and visualization of large volume datasets. In *GRAPP (AS/IE)* (2007), Citeseer, pp. 39–46. 3, 4, 8

[SBMH94] SAGAR M. A., BULLIVANT D., MALLINSON G. D., HUNTER P. J.: A virtual environment and model of the eye for surgical simulation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM, pp. 205–212. 2

[SGBM98] SCHILL M. A., GIBSON S. F., BENDER H.-J., MÄNNER R.: Biomechanical simulation of the vitreous humor in the eye using an enhanced chainmail algorithm. In *Medical Image Computing and Computer-Assisted Interventation?ICCAI?8*. Springer, 1998, pp. 679–687. 3, 4

[SGS10] STONE J. E., GOHARA D., SHI G.: Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering 12*, 3 (2010), 66. 5

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. In *ACM Siggraph Computer Graphics* (1986), vol. 20, ACM, pp. 151–160. 3

[SSC03] SINGH V., SILVER D., CORNEA N.: Real-time volume manipulation. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics* (2003), ACM, pp. 45–51. 3

[WRS01] WESTERMANN R., REZK-SALAMA C.: Real-time volume deformations. In *Computer Graphics Forum* (2001), vol. 20, Wiley Online Library, pp. 443–451. 3