UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

ugr | Universidad de Granada

Departamento de Lenguajes y Sistemas Informáticos

Programa de Doctorado en Tecnologías de la Información y la Comunicación

Métodos GPGPU para simulación y visualización de modelos volumétricos interactivos

GPGPU methods for simulation and visualization of interactive volumetric models

Tesis Doctoral

Alejandro Rodríguez Aguilera

Director: Alejandro José León Salas

Editor: Universidad de Granada. Tesis Doctorales
Autor: Alejandro Rodríguez Aguilera
ISBN: 978-84-9163-491-1
URI: http://hdl.handle.net/10481/48269

La memoria titulada "Métodos GPGPU para simulación y visualización de modelos volumétricos interactivos", que presenta Don Alejandro Rodríguez Aguilera, para optar al grado de Doctor, ha sido realizada en el Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada bajo la dirección del Doctor Alejandro José León Salas.

El doctorando Alejandro Rodríguez Aguilera y el director de la tesis Doctor Alejandro José León Salas, garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección del director de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, Junio de 2017

El doctorando

Alejandro Rodríguez Aguilera

El director

Alejandro José León Salas

Agradecimientos

Junto con esta memoria, se cierra un capítulo importante de mi vida que, echando la vista atrás, ha dado lugar a muchas experiencias positivas con las que he crecido personal y profesionalmente. Hay mucha gente que me ha acompañado durante esta etapa, y me gustaría dedicar este espacio a agradecer a todos ellos el tiempo que hemos compartido, y sobre todo, la buena amistad que hemos acabado labrando de forma inevitable.

Tengo que empezar agradeciendo a Alejandro por abrirme el camino de la investigación. Con su confianza en mi buen hacer, su dirección y sobre todo, su amistad, creo sinceramente que ha conseguido que mi doctorado haya sido un proceso muy fructífero en todos los sentidos. De igual manera, agradezco a todos los miembros del grupo de informática gráfica, mi grupo, su apoyo a lo largo de este exigente proceso. También quiero agradecer a todos los compañeros del CITIC con los que he compartido horas de trabajo y de ocio a partes iguales, o casi.

Una de las cosas que más valoro de esta etapa es la oportunidad que me ha ofrecido de conocer y colaborar con muchas personas que siempre han ofrecido su ayuda desinteresada, y me siento un privilegiado por ello. Todos los compañeros del grupo DEFROST, que me acogieron de la mejor manera en Lille, con especial mención a Christian por confiar en mí y darme la oportunidad de trabajar en un proyecto tan interesante. Todos los miembros del grupo GMRV, que de igual manera me acogieron la primera vez que fui a visitarles, y tan bien fue la cosa que ahora son mis nuevos compañeros de trabajo y aún mejores amigos. Mención especial también tengo que hacer a Miguel Ángel, que ha confiado en mí en ya múltiples ocasiones y que no ha dudado en compartir su amplia experiencia conmigo.

Tampoco me olvido en estos momentos de todos los grandísimos amigos, los que siguen aquí y los que ya no están, que me han acompañado desde mucho antes de empezar esta etapa, que me enriquecen como persona y que espero que sigan caminando junto a mí mucho tiempo más.

Por último, y por ello más importante, gracias a mis papis, a mis hermanitos y a Inés, por ser incondicionales y ofrecerme todo lo importante. Estoy muy orgulloso de vosotros y os quiero un montón.

A todos vosotros, ¡Gracias!

Contents

Abstract

In this thesis we address different computationally demanding problems in the fields of simulation and visualization, exploring novel GPGPU methods to provide solutions for interactive applications with a reduced time budget.

In the first part of this work, we tackle different limitations of the ChainMail algorithm for deforming volumetric medical models. We propose a novel parallel ChainMail algorithm for the efficient handling of large heterogeneous models, also accounting for interactive topology changes. The use of a novel scheduling method for the GPU addresses the sparse nature of the computation, allowing the use of much larger models in interactive applications.

The second part is devoted to the online motion planning of soft robots addressed as a computational simulation problem. Specifically, we propose an accurate and efficient model for hydraulic actuators and its usage in an inverse kinematics framework for the interactive control of actuated soft robots. An efficient estimation of the fluid weight is provided by a novel parallel algorithm exploiting modern GPU capabilities.

The last part of this work addresses problems related to the visualization of medical datasets through direct volume rendering. First, we propose a novel resampling algorithm for the interactive rendering of deformable medical models, capable of interactively providing a regular grid representation of the deformed dataset as input for the standard volume rendering pipeline. Next, we present a new tool for the the selection and visualization of regions of interest avoiding any kind of parameter tuning, aiming to eliminate trial-and-error approaches typically required for this task.

Chapter 1

Introduction and overview

1.1 Introduction

Modeling the physical behavior of real world phenomena has been of scientific interest throughout human history. It is a key tool not only to improve our understanding of the world, but also to provide a means to reproduce and predict the behavior of many kinds of physical processes, providing important insights that can then be employed for better decision-making, detailed study or training, among other applications. Computational modeling and simulation are among the most significant developments devoted to scientific and engineering inquiry in the last decades and a plethora of methods for physical simulation are extensively used in a wide range of disciplines, as diverse as astrophysics, molecule folding, civil engineering and both film and video-game industries.

A major issue of computational simulation methods arises with the complexity of model employed to describe the phenomenon. A model too complex to compute or too large to store may require an unacceptable amount of time to perform the desired simulation. This limitation becomes especially apparent for interactive applications, in which the time budget to perform a simulation step can be limited to a few milliseconds. Many of the interactive simulation applications also require visual feedback, thus the connection between the simulation model and its visual representation must be performed in an efficient manner to allow for the complete application pipeline to comply with the strict time budget.

In the specific case of medical simulations, such as the simulation of physiological processes or the simulation of surgical procedures (commonly known as virtual surgery), a complex model of the human anatomy is required. The tissue distribution of biological forms is typically captured through 3D sensing technologies such as computerized tomography (CT) procedures or magnetic resonance imaging (MRI) techniques. As a result, the acquired data includes up to several million voxels of information representing the tissue distribution inside the different organs and anatomical structures. This huge amount of data can be used to obtain both the

bio-mechanical and the visual models for interactive applications, becoming a very computationally demanding problem that has received a lot of attention from the scientific community.

The efficiency in the computation of these applications has naturally been raised over the years, both by improvements in the hardware and in the software. The introduction of the graphics processing units (GPUs) allowed to accelerate the visual feedback of these applications, but a major breakthrough was reached with the appearance of the programmable GPU pipeline, and more importantly, the subsequent development of frameworks for general purpose computing on graphics processing units (GPGPU) that extended their usage to many fields of modern computation, including the physical simulation field.

In this thesis, we explore GPGPU techniques aiming to develop new methods and improved algorithms for the simulation and visualization of interactive applications, with special focus on medical applications and soft robotics. Let us first give a brief overview to each of the main fields involved in the scope of this thesis.

1.2 Parallel GPU computing

The computing industry moved away from manufacturing processors with increased clock frequency toward increasing the number of processors several years ago. This event posed a great change in software development, and was regarded as a new software revolution [1].

Along with this change, and in part motivated by it, we have seen the extension of the use of GPUs from their original application, i.e., rasterization of visual geometric primitives, to a wide range of applications in most computation fields, since their massively parallel architecture offered the potential for dramatic speedups. Starting with the inclusion of the programmable graphics pipeline and currently with the development of general purpose platforms and languages such as CUDA [2, 3] and OpenCL [4, 5] to program the GPUs, their application to accelerate computationally demanding problems is expanding, and this is a trend expected to further develop during the coming years [6].

Modern GPUs are designed following a highly parallel *single instruction multiple data* (SIMD) architecture, motivated by the original rendering pipeline performed on GPUs, which followed a very simple SIMD computation of geometric primitives [2, 7]. However, most general problems have more complex requirements regarding the structure of the computation, the dependence between different stages and the data sharing between different processes [8]. For this reason, many different strategies regarding computation scheduling, data layout and memory access among others have been developed over the years to efficiently leverage the GPU resources in a variety of situations.

In the scope of this thesis, we focus on *stencil computation* and *irregular paral-*

lel programs, two ubiquitous classes of computation schemes that appear in many disciplines involving computation of large amounts of data, and play an important role in the solutions developed on this thesis as will become apparent later.

The stencil computation scheme [9, 10] operates over an input array, structured as a regular multidimensional grid of cell elements, typically 2- or 3-dimensional. A sequence of iterations are performed over the grid, and the cells are updated using the information of local neighboring cells in a fixed pattern, called the *stencil*. Many problems across different disciplines can be adapted to this computation pattern, such as finite difference computation [11], weather prediction [12] and partial differential equations solvers [10] among others. The structured and parallel nature of the approach perfectly adapts to the computation paradigm of modern GPUs, however, this approach is not free of limitations, and optimization strategies regarding shared data access and memory bandwidth have been widely studied [13, 9, 14]. Another issue studied in many problems is the sparse nature of the computation, i.e., the number of cells that require an update in an average iteration is significantly lower than the total number of cells in the grid, and several solutions have been proposed to handle these cases and avoid the waste of GPU resources [15, 16].

Other common issue found when running parallel algorithms on the GPU is the irregular parallel workload problem [17, 18], i.e., the workloads of the different parallel tasks are notably different. These cases require more intelligent scheduling approaches to balance the load across the parallel units. This problem adopts many forms across many disciplines and efficient scheduling strategies must be found accounting for the specifics of each case. In computer graphics, a well known form of this problem arises in the central stage of rasterization, and many strategies have been proposed [19, 20, 21, 22], always comprising a trade-off between load balancing effectiveness and scheduling overhead.

1.3 Interactive simulation of deformable models

Many materials, including living tissue, exhibit a deformable and heterogeneous behavior to some extent, and a correct simulation of their dynamics is crucial in many applications. Additionally, some of these applications rely on interactive responses for proper functioning, e.g., the interaction between virtual tools and different tissues in virtual surgery applications or the compliant behavior of soft materials in simulation-based motion planing of soft robots. For this reason, a trade-off between physical accuracy and performance must be reached, and many deformable models have been proposed over several decades following different approaches [23, 24].

A great body of approaches propose physically-based models [25]. Many methods are based on particle systems [26, 27, 28], defining dynamic systems of discrete particles that interact with each other through a set of constraints that define forces or energies between the particles, approximating the properties of the modeled el-

ements. For instance, the popular mass-spring method defines a set of mass nodes interconnected with mass-less springs, and the dynamics of the system are then computed by solving the partial differential equations that emerge from this formulation [29, 30]. These methods are very general and can capture many physical properties of different kinds of materials, however, they typically require a complex parameter tuning to fit the exact behavior of a real world specimen. Approaches based on the laws of continuum mechanics [31] overcome these limitations by defining the deformable behavior of a solid object in the continuum space which is then discretized, typically using the finite element method (FEM) [32], to approximate the real solution with a finite set of elements. Although the problem formulation in this case is more complex, it has an increased accuracy of the modeled behaviors and the tuning of the model parameters to match the real counterpart is much easier as they emerge from real, measurable material properties widely studied. These approaches are capable of producing highly accurate simulations, however, the computational cost is typically very high and, for interactive applications, models with a reduced number of particles or elements must be used. These models, specially the FEM approach, have been extensively used in interactive medical applications, however, in some cases the disparity between the high resolution source dataset and the low resolution model leads to an inevitable loss of complexity and heterogeneity of the reference model.

1.3.1 The ChainMail algorithm

For cases where the loss of detail and complexity is critical, the ChainMail algorithm proposed by Gibson [33] enabled physically realistic deformations following a two-stage approach. A ChainMail model consists of a regular 3D grid of elements, each connected to its 6 nearest neighbors through geometrical constraints. When a deformation is applied, a first geometrical stage enforces the constraints through the grid, and a second relaxation stage performs an iterative energy minimization process on the grid until a rest state is reached. Its geometrical nature makes the algorithm unconditionally stable and very efficient to compute, and models with a number of elements several orders of magnitude greater than the approaches stated above can be used in interactive applications. Several limitations are found due to its simple formulation, however, by tuning the geometrical constraints and the relaxation stage, it is capable of reproducing complex living tissue behaviors such as non-linear stress-strain response, hysteresis and asymptotic relaxation [34]. It has been used in many medical applications [35, 36, 37, 38, 39, 40, 41, 42, 43] and several versions of the original method have been proposed [44, 45, 46, 47, 48, 49, 50, 51, 52].

The increase of resolution of interactive models for all the above mentioned methods has also been a subject of intensive research. The use of coarsening techniques for FEM models [53, 54, 55] for instance, allows the use of high resolution meshes by just a fraction of their computational cost. Naturally, the use of GPUs to accelerate

parts of the computations has also been widely explored in the last years, both for particle systems [56, 57] and FEM systems [58, 59], although for the ChainMail algorithm only a first attempt was made by Rößler et al. [60].

1.3.2 Soft robotics control

In the field of robotics, an alternative to traditional articulated rigid robots emerged, inspired on biological structures, under the name of soft robots [61, 62]. The salient feature of soft robots is their compliant nature as they are primarily made of soft materials such as fluids, gels and soft polymers, and the motivation for this choice is the intrinsic increased dexterity and flexibility that allows them to support strain while remaining operative. This is a desirable feature for situations with a certain degree of uncertainty and safety requirements, typical (but not exclusive) of human-centric operations where contact with the surroundings is sensible, although it may be even desirable for a better operation [63, 64]. Under these circumstances, minimizing stress concentrations in contact areas also minimize damage to the operated environment, and it can be achieved by *compliance matching*, i.e., guaranteeing similar mechanical rigidity of the contacting surfaces. While compliance matching may become complex to achieve using conventional robots, it becomes trivial for soft robots [62].

As an emerging field, it is not free of challenges [65]. One major issue lies in the task control and motion planning area; the compliant nature of the supporting structure of soft robots yields a theoretical infinite number of degrees of freedom, which leads to heavily underactuated designs where the actuators become coupled by the deformation, and direct control and motion planning become non-trivial problems [66, 67]. One successful approach to perform the direct control of such robots consists on creating a simulated counterpart of the soft robot using a FEM discretization for modeling the mechanical behavior and structure of the fabricated robot together with an accurate modeling of the actuators influence on the robot, performing then an inverse kinematics problem given a target configuration [68]. This inverse problem can be efficiently solved using a quadratic programming optimization procedure as demonstrated in [69], however, the models of both the robot structure and the actuators must allow near real-time computations yet provide accurate predictions to allow for interactive control of fabricated robots. For this reason, the accurate and efficient modeling of different kinds of actuators has been and still is subject of research [70, 71, 72].

1.4 Interactive direct volume rendering

As stated earlier, medical imaging techniques typically produce a volumetric representation of the interior of a body. As a consequence, many direct volume rendering (DVR) techniques have been proposed and improved over the years [73, 74, 75, 76].

The most extended DVR approach is the Ray-casting algorithm: given a viewpoint and a volume dataset, a set of rays (one per pixel in the final output image) are cast and the radiative energy is integrated by mapping the volume samples fetched along a ray to emission and absorption coefficients through the use of a transfer function, and the final composed value is set to the according pixel to form the output rendering. A comprehensive explanation of the method is found in [74]. Many advanced illumination techniques proposed over the years have increased the output quality of the final compositions [77] and, although this method was initially too costly for interactive applications, it's GPU parallelization allowed for interactive applications to use this rendering approach to provide visual feedback [78, 79].

1.4.1 Medical volume exploration

Direct volume rendering approaches provide useful information regarding spatial properties and contextual information of different features, however, its wide adoption for clinical applications is still limited [80, 81]. The main reason for this is the level of expertise required to obtain desired visualizations of the data since the basic tool to control the visible structures is the definition of transfer functions, which is a process widely recognized as complex and time-consuming [82, 83].

This issue has motivated many techniques to interactively explore medical volumes. Some approaches borrow ideas from traditional medical hand-drawn illustrations, aiming to provide new tools for the generation of *focus+context* visualizations of the data, such as novel manipulation operators [84] or context-preserving rendering models [85, 86]. Great efforts have also been directed to improve the generation of meaningful transfer functions, either easing the generation process with semi-automatic techniques [87, 88, 89] or fully automatic techniques [90, 91, 92, 93]. Other techniques aim to connect the control of 3D volume rendering parameters with the classical 2D pipeline used in clinical workflows in an attempt to ease the transition while exploiting the benefits of both metaphors, typically relying on features of interest selected on the 2D slices to infer good visualization parameters for the 3D rendering [94, 95, 96, 80].

These and other approaches bridge the gap between volume rendering and professional workflows, but a general, standard set of techniques for an easy exploration of medical volume datasets is yet to be reached.

1.4.2 Rendering of deformed volume data

The visual feedback for medical simulations is in many cases as important as the simulation itself [97, 98]. Despite the mature state of the Ray-casting algorithm, its usage for applications using deformed volume data remains a challenge. This is due to the fact that high quality volume rendering approaches account for a regular grid of data to achieve interactive frame rates.

The traditional strategy to address this problem consists in segmenting and meshing the original volume data to some desired isosurface levels and resort to standard surface rendering. Besides the additional segmentation stage, the full detail in the original data is lost and it is not possible to visualize isosurfaces not segmented beforehand. Another strategy is to perform volume rendering using unstructured meshes to avoid the loss of detail. Despite achieving interactive frame rates through GPU computing, as demonstrated by Georgii et al. [99], the cost of unstructured volume rendering grows with mesh resolution, thus for high resolution deformation methods, as the ones mentioned in Section 1.3, the computational cost impedes interactivity. In addition, many of the improvements and advanced rendering features have been developed for regular-grid volume data and their implementation for unstructured meshes is typically more computationally demanding and in many cases has not been studied, as stated by Correa et al. [100].

A different strategy was later proposed, motivated by the superior performance and more advanced rendering features of DVR methods, following the idea of resampling the deformed volume onto a regular grid which is then fed as input to a standard DVR method [38, 101, 102]. Although the resampling of a large volume is a costly operation, it is possible to perform it interactively through massive GPU parallelization for tetrahedral meshes as demonstrated by Gascón et al. [101]. Following this strategy, both the advanced rendering and exploration techniques developed for DVR can be applied to visualize the deformed volume. The main limitation of current methods following this strategy resides again in the performance, since the computation cost depends on the mesh resolution, thus for high resolution meshes interactivity is impeded.

1.5 Objectives

The main goal of this thesis aims to improve both the simulation and the visualization of deformable models for interactive applications applying novel GPGPU methods. Specifically, we propose objectives both in the simulation and the visualization fields for medical applications and soft robotics.

In the field of simulation, we propose the following objectives:

- To develop a GPU accelerated ChainMail algorithm for interactive deformation of large and heterogeneous medical models.
- To propose and apply an efficient method to compute the effect of fluid weight on deformable models for the correct simulation and control of hydraulic soft robots.

In the field of medical visualization, we propose the following objectives:

- To propose an efficient resampling method for high resolution volume deformation techniques.

- To develop an interactive method for intuitive selection and visualization of regions of interest.

1.6 Contributions

The main contributions of this thesis cover the proposed objectives and, as a secondary result of our solutions, we also present additional contributions in the GPGPU field. Our core contributions can be summarized as follows:

1. A novel ChainMail algorithm, obtained by reformulating the original algorithm as a parallel stencil computation problem, hence allowing its efficient computation using the GPU. Moreover, the proposed algorithm accounts for the sparse nature of it computation, and the propagation and the relaxation stages are modified to enable the use of heterogeneous materials on the model, even in the presence of interactive topology changes.

2. An accurate, real-time computation of the fluid weight distribution of hydraulic components which, together with a model of their mechanical behavior, is integrated withing a FEM simulation framework for online motion planning of hydraulic actuated soft robots.

3. A parallel resampling method for deformed volumes, independent of the underlying deformation method, more than one order of magnitude faster than previous algorithms when applied to high resolution deformation models.

4. A ROI selection and visualization method based on interactively generated spatial opacity maps, following a stencil computation approach.

5. A blocking method for efficient computation of sparse stencil computation problems. This blocking method drastically reduces the unnecessary GPU computation.

6. A new scheduling technique to address irregular-parallel workload problems, used on the computation of the fluid weight inside hydraulic components.

1.6.1 Publications

The results corresponding to the contributions directly derived from this thesis are published in five international journals indexed in JCR, including one ACM Transactions on Graphics presented at SIGGRAPH Asia, one peer-reviewed international conference and two peer-reviewed national conferences. The works here presented were done in collaboration with several colleagues, and my implication can be inferred from the position in the authors lists:

- **A. Rodríguez**, A. León, G. Arroyo, and J. M. Mantas (2015). "SP-ChainMail: a GPU-based sparse parallel ChainMail algorithm for deforming medical volumes". *The Journal of Supercomputing*, Volume 71, Issue 9, pp. 3482-3499.
  - Status: Published [103]
  - Impact factor (JCR 2015): 1.088
  - Subject category: Computer Science, Hardware and Architecture (Q2: 23/51), Computer Science, Theory and Methods (Q2: 47/105).
- **A. Rodríguez**, A. León and G. Arroyo (2016). "Parallel deformation of heterogeneous ChainMail models: Application to interactive deformation of large medical volumes". *Computers in Biology and Medicine*, Volume 79, pp. 222-232.
  - Status: Published [104]
  - Impact factor (JCR 2015): 1.521
  - Subject category: Biology (Q2: 35/84), Computer Science, Interdisciplinary Applications (Q3: 54/105), Mathematical and Computational Biology (Q2: 19/57), Engineering, Biomedical (Q3: 45/77).
- **A. Rodríguez**, A. León Salas, D. Martín Perandrés and M. A. Otaduy (2015). "A parallel resampling method for interactive deformation of volumetric models". *Computers & Graphics*, Volume 53, pp. 147-155.
  - Status: Published [105]
  - Impact factor (JCR 2015): 1.120
  - Subject category: Computer Science, Software Engineering (Q2: 41/106).
- R. Torres, **A. Rodríguez**, J. M. Espadero and M. A. Otaduy (2016). "High-resolution interaction with corotational coarsening models". *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2016*, Volume 35, Issue 6, Article No. 211.
  - Status: Published [106]
  - Impact factor (JCR 2015): 4.218
  - Subject category: Computer Science, Software Engineering (Q1: 1/106).
- K. A. Mountris, J. Bert, J. Noailly, **A. Rodríguez**, A. Valeri, O. Pradier, U. Schick, E. Promayon M. A. Gonzalez Ballester, J. Troccaz and D. Visvikis (2017). "Modeling the impact of prostate edema on LDR brachytherapy: a Monte Carlo dosimetry study based on a 3D biphasic finite element biomechanical model". *Physics in Medicine and Biology*, Volume 62, Issue 6, pp. 2087-2102.

  - Status: Published [107]
  - Impact factor (JCR 2015): 2.811
  - Subject category: Engineering, Biomedical (Q1: 19/76), Radiology, Nuclear Medicine and Medical Imaging (Q2: 32/124).

- **A. Rodríguez**, E. Coevoet and C. Duriez. "Real-time simulation of hydraulic components for interactive control of soft robots". *IEEE International Conference on Robotics and Automation* (ICRA 2017).
  - Status: Published [108]
  - CORE Conference Ranking 2014: CORE B
  - H index (SJR 2015): 105.

- **A. Rodríguez**, A. León, L. López Escudero and M. García Sánchez (2014). "A System Proposal for Interactive Deformation of Large Medical Volumes". *Proceedings of Spanish Computer Graphics Conference* (CEIG 2014).
  - Status: Published [109]

- **A. Rodríguez** and A. León (2016). "Spatial Opacity Maps for Direct Volume Rendering of Regions of Interest". *Proceedings of Spanish Computer Graphics Conference* (CEIG 2016).
  - Status: Published [110]

Other contributions, not directly related to the goals of this thesis have been developed for other projects and also published during the thesis period, including one international journal indexed in JCR, one international journal and one international conference:

- D. Martín and G. Arroyo and **A. Rodríguez** and T. Isenberg (2017). "A survey of digital stippling". *Computers & Graphics*, Volume 67, pp. 24-44.
  - Status: Published [111]
  - Impact factor (JCR 2015): 1.120
  - Subject category: Computer Science, Software Engineering (Q2: 41/106).

- **A. Rodríguez** and A. León (2016). "A framework for remote 3D interaction with handheld devices: Application to a 3D heritage gallery prototype". *Scientific Research and Information Technology* (SCIRES-IT), Volume 6, Issue 1, pp. 79-86.
  - Status: Published [112]

- **A. Rodríguez** and A. León (2015). "Smartphone-based remote 3D interaction for digital heritage applications". *Proceedings of Digital Heritage* (DH 2015), pp. 297-300.
  - Status: Published [113]

1.7 Outline

This dissertation is presented in the modality of "compendium", and the scientific contributions directly connected to the scope of this thesis, with me as leading author, have been organized in the following chapters.

Chapter 2 groups the contributions related to our proposed ChainMail algorithm, including details of the recasting of the ChainMail algorithm as a stencil computation problem, the blocking scheme for efficient computation of sparse stencil computations, the modified propagation and relaxation stages for the handling of heterogeneous materials and the introduction of interactive topology changes in the ChainMail models.

Chapter 3 includes our contribution regarding the modeling of hydraulic components for direct control of soft robots and the modeling of the mechanical behavior of the components, along with the formalization of our irregular-parallel leveraging algorithm.

Chapter 4 groups contributions regarding visualization techniques. First, our resampling method for large volumes is presented, including details of its parallel implementation and several use cases with different deformation methods. Then, our ROI selection and visualization method based on spatial opacity maps is presented, including details regarding its implementation, integration within the volume rendering pipeline and an analysis and discussion of its ease of use and robustness.

Finally, in chapter 5 we present the main conclusions derived from the work carried out in the development of this thesis and discuss future works.

Appendix A includes a brief summary of the present thesis to comply with the current regulation regarding the international theses written in English, including a brief introduction, the structure of the document and the conclusions already presented in chapter 5.

Chapter 2

Parallel ChainMail simulation of heterogeneous medical models

2.1 A system proposal for interactive deformation of large medical volumes

- **A. Rodríguez**, A. León, L. López Escudero and M. García Sánchez (2014). "A System Proposal for Interactive Deformation of Large Medical Volumes". *Proceedings of Spanish Computer Graphics Conference* (CEIG 2014).
  - Status: Published [109]

CEIG - Spanish Computer Graphics Conference (2014) , pp. 1–10
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".

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 *Ray-casting*. Rossler et al. [RWE08] presentan un sistema similar, proponiendo una versión paralela del algoritmo *Chain-mail 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 *Chain-Mail 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

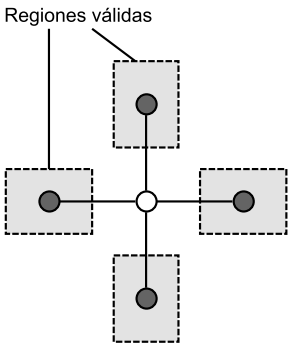

Figura 1: *Representación en 2D de la estructura de vecinos generada para el algoritmo ChainMail. El elemento central se conecta a sus 4 vecinos adyacentes, y define una región de posicionamiento válida para cada uno.*

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.

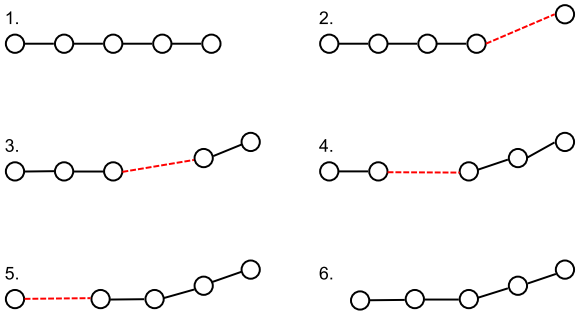

Figura 2: *Ejemplo de propagación de deformaciones con ChainMail. Tras el movimiento del elemento de la derecha, se viola la restricción geométrica con su vecino izquierdo, lo que produce una reacción en cadena de movimientos.*

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

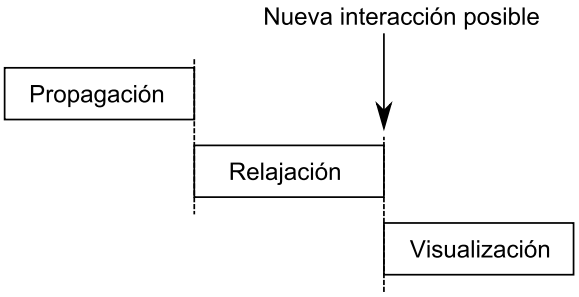

Figura 3: *Cauce de deformación del algoritmo propuesto por Rossler. El resultado de las deformaciones se visualiza una vez que todo el proceso de deformación ha terminado. Una nueva interacción sobre el modelo, de igual manera, solo es posible una vez ha acabado el cauce de deformación completo.*

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.

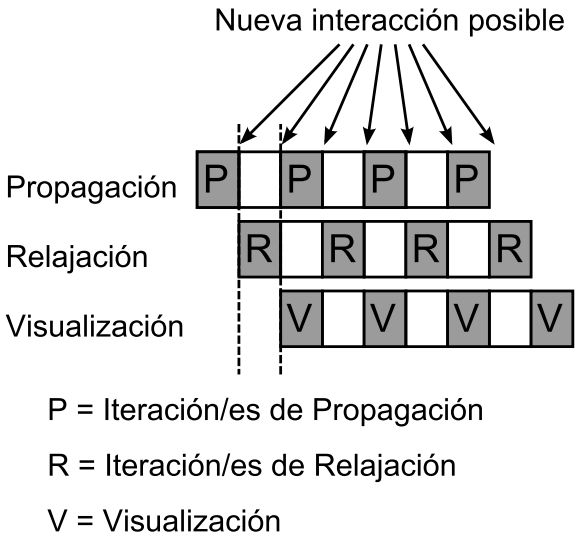

Figura 4: *Al subdividir las etapas del cauce en iteraciones, se consigue obtener visualizaciones intermedias durante las deformaciones. De igual manera, es posible aplicar nuevas interacciones sobre el modelo en cualquier punto del cauce de deformación al eliminar la secuencialidad entre las etapas.*

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.

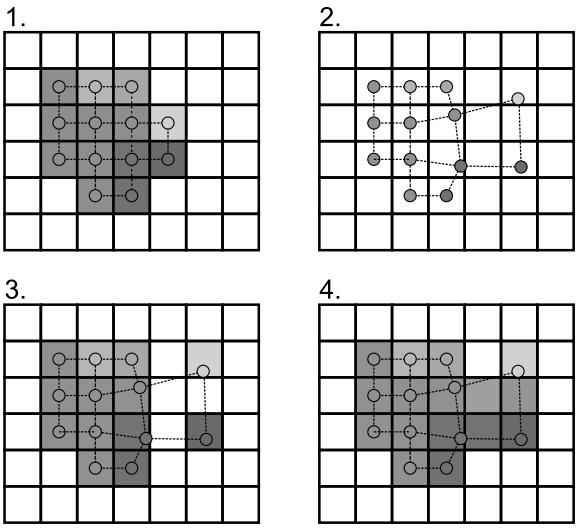

Figura 5: *Simplificación 2D del algoritmo de remuestreo. En 1) se representa la rejilla original junto con la estructura ChainMail subyacente. En 2) se realiza una deformación sobre los elementos, que en 3) son remuestreados sobre la rejilla, y en 4) se rellenan los huecos que aparecen.*

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.

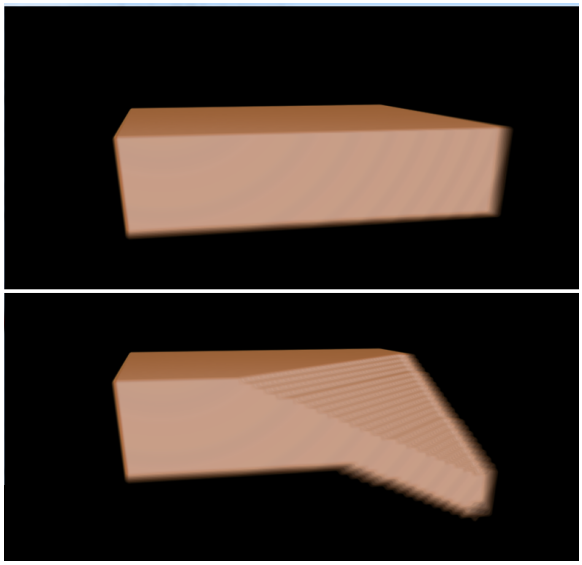

Figura 6: *Visualización de una deformación mediante remuestreo de vóxeles donde se aprecia la visualización escalonada.*

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

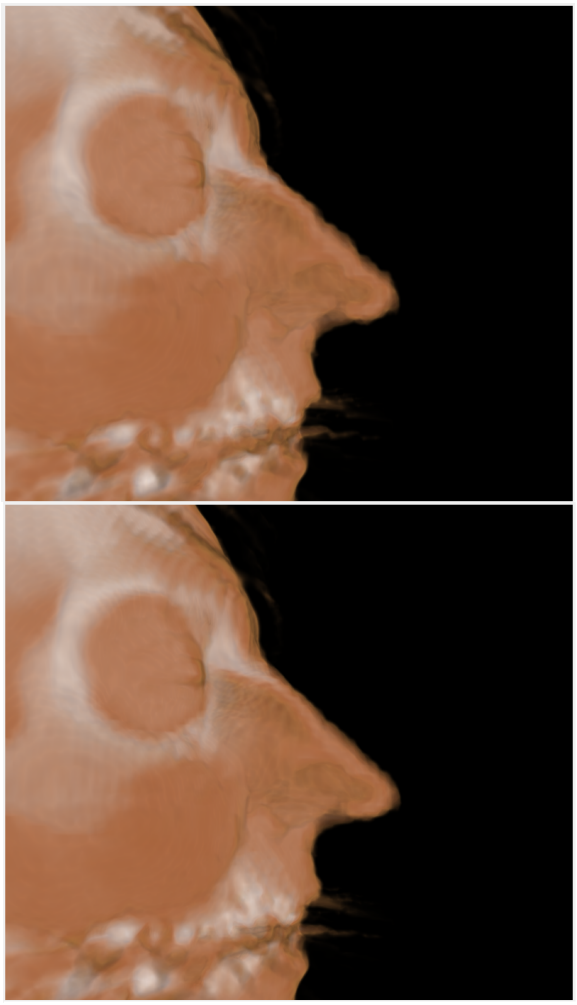

Figura 7: *Ejemplo de la mejora visual obtenida al usar precisión de sub-vóxel. El mismo modelo deformado se visualiza sin emplear la técnica (arriba) y empleando la técnica (abajo).*

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.

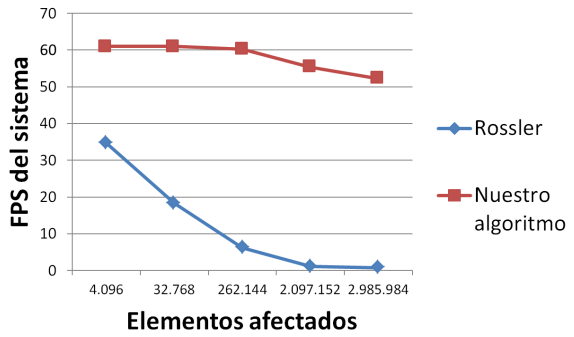

Figura 8: *Gráfica comparativa de los frames por segundo mantenidos por ambos algoritmos en función del número de elementos afectados por la deformación.*

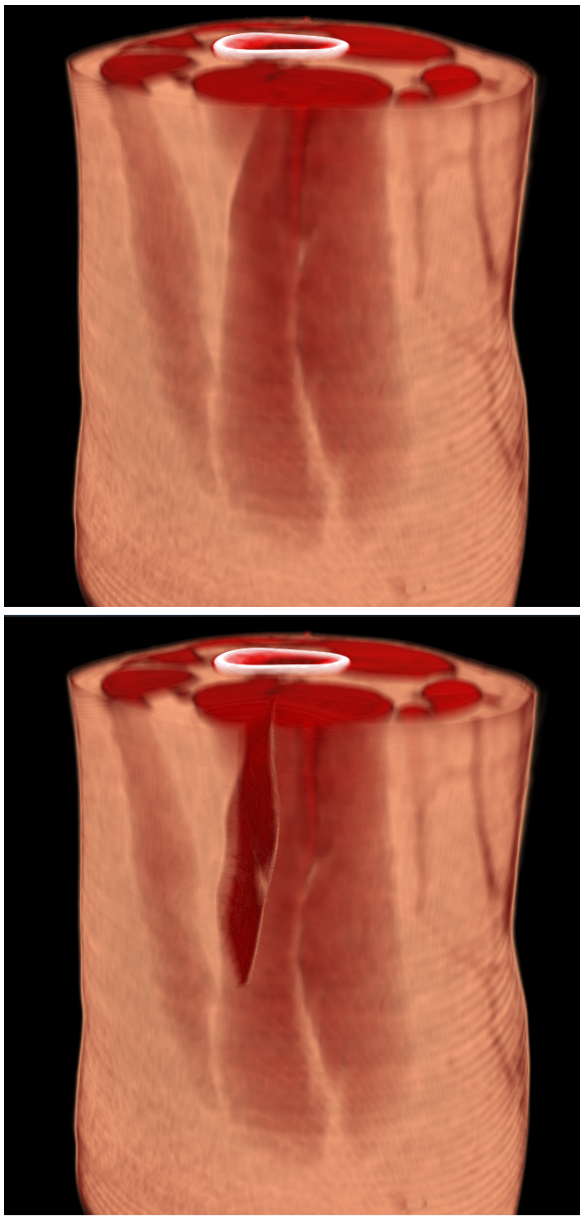

Figura 9: *Simulación de incisión sobre volúmen médico. Arriba se ilustra el volúmen original, una sección de pierna formada por* $256 \times 255 \times 256$ *elementos. Abajo, el resultado de generar interactivamente la incisión sobre el volúmen.*

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, 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

2.2 SP-ChainMail: a GPU-based sparse parallel ChainMail algorithm for deforming medical volumes

- **A. Rodríguez**, A. León, G. Arroyo, and J. M. Mantas (2015). "SP-ChainMail: a GPU-based sparse parallel ChainMail algorithm for deforming medical volumes". *The Journal of Supercomputing*, Volume 71, Issue 9, pp. 3482-3499.
  - Status: Published [103]
  - Impact factor (JCR 2015): 1.088
  - Subject category: Computer Science, Hardware and Architecture (Q2: 23/51), Computer Science, Theory and Methods (Q2: 47/105).

J Supercomput manuscript No.
(will be inserted by the editor)

SP-ChainMail: A GPU-based Sparse Parallel ChainMail Algorithm for Deforming Medical Volumes

Alejandro Rodríguez · Alejandro León · Germán Arroyo · José Miguel Mantas

Received: date / Accepted: date

Abstract ChainMail algorithm is a physically-based deformation algorithm that has been successfully used in virtual surgery simulators, where time is a critical factor. In this paper, we present a parallel algorithm, based on ChainMail, and its efficient implementation that reduces the time required to compute deformations over large medical 3D datasets by means of modern GPU capabilities. We also present a 3D blocking scheme that reduces the amount of unnecessary processing threads. For this purpose, this paper describes a new parallel boolean reduction scheme, used to efficiently decide which blocks are computed. Finally, through an extensive analysis, we show the performance improvement achieved by our implementation of the proposed algorithm and the use of the proposed blocking scheme, due to the high spatial and temporal locality of our approach.

Keywords GPU programming · stencil computation · physically-based deformation · parallel algorithms

1 Introduction

Over the last years, Graphics Processing Units (GPUs) have been widely used to accelerate a huge variety of algorithms in different fields. This is due to the fact that modern GPUs are designed following a highly parallel Single Instruction, Multiple Data (SIMD) scheme, containing hundreds or thousands of processors and dedicated memory.

Many approaches for physically-based deformation of medical volumetric models take advantage of these capabilities such as the parallel implementation of the Finite Element Method proposed by Comas et al. [2] or the parallel

A. Rodríguez · A. León · G. Arroyo · J.M. Mantas
University of Granada
Granada, Spain
E-mail: alejandrora@ugr.es

Mass-Spring system proposed by Georgii et al. [8], since they can be adapted to operate in parallel over a huge amount of data elements. The ChainMail algorithm, introduced by Gibson [10], is a two-stage physical deformation algorithm which, unlike other physically-based deformation algorithms, follows a purely geometrical approach that is therefore capable of handling several orders of magnitude more elements. It has been successfully used to simulate surgical procedures, such as a vitrectomy [19] or arthroscopic knee surgery [9], where a low response time is a strong requirement.

In this paper we present a parallel version of the ChainMail algorithm which efficiently handles deformations over large areas of the dataset. Our algorithm allows a parallel implementation of the tasks that are computationally intensive using the GPU, thus avoiding costly memory transfers to visualize the deformations since the rendering is also carried out using the GPU. Unlike previous approaches, our algorithm is capable of interleaving its two stages, allowing intermediate visualizations of the current state of the deformation. Therefore, our algorithm is able to provide a more interactive visual feedback.

We also propose a partitioning method that, taking into account the sparse nature of our algorithm, splits up the computation of the dataset elements into blocks that can be processed independently. This blocking method prevents the processing of blocks that do not require any computation, reducing the number of idle threads, thus decreasing the overall computation time.

Additionally, we present a novel parallel reduction approach that is limited to reduction of boolean sets, but improves the performance of the general parallel reduction approach.

This paper is organized as follows: In Section 2, previous related work is reviewed. Our parallel ChainMail algorithm is described in Section 3, which also includes a brief introduction to the original ChainMail algorithm, as well as details about the implementation. In Section 4, the blocking method is described and the required algorithms and data structures to efficiently handle the blocks are detailed. Also, the proposed boolean reduction mechanism is presented. In Section 5 we present an analysis of results, testing our approach under different blocking configurations and several datasets, comparing its performance against an optimized multithreaded implementation of the original ChainMail algorithm. We extend the analysis to several hardware configurations, demonstrating the portability and scalability of the blocking scheme and the benefits achieved by our approach. Finally, our conclusions are exposed in Section 6.

2 Related Works

In order to take advantage from General-Purpose Computing on Graphics Processing Units (GPGPU), it is necessary to map the algorithms to the graphics hardware, which is not always an easy task. Kirk et al. [11] presented an excellent introduction to massively parallel general-purpose computation using modern graphics hardware, compiling recent developments, common tech-

niques and several practical examples. Due to the SIMD nature of modern GPUs, a common approach to perform parallel computation is the iterative stencil computation scheme [3]. This scheme consists of a sequence of iterations over a given dataset, stored in a grid of cells. Each iteration performs local neighborhood computations to obtain new values for the cells. Examples of this approach are the stencil based GPU algorithm proposed by Micikevicius [14] to perform 3D finite difference calculations, and the iterative parallel approach proposed by De la Asunción et al. [4] to simulate shallow water systems on the GPU.

The original ChainMail algorithm [10] has been used by many authors for medical applications, such as angioplasty simulation [12] , heterogeneous deformation of medical datasets [20] and generation of medical illustrations [13]. Unfortunately, interactivity is only achieved if the amount of affected elements is relatively small. Since the original ChainMail algorithm presents an important computational stage which is inherently sequential, a direct mapping to parallel platforms computation has a very limited impact on the performance.

A two-stage parallel approach based on the ChainMail algorithm was introduced by Rößler [17], and has been also used in medical applications by Fortmeier et al. [5,6]. This parallel approach achieves good performance for small deformations, but suffers from a high amount of idle computation when large deformations are applied, hurting the overall performance. Moreover, the visualization can only be performed after the whole deformation is completed, decreasing the visual feedback and interactivity during large deformations.

Unlike previous approaches, our algorithm handles both the propagation and the relaxation stage at iteration level following a stencil computation scheme, allowing overlapping both stages in order to generate partial visualizations of the deformations. Moreover, the use of our blocking scheme avoids unnecessary computation, increasing the performance of the overall process.

Bandwidth and computation problems associated with the stencil computation approach have been widely studied. Many cache-based blocking schemes palliate bandwidth problems. An example is discussed in the work of Nguyen et al. [16], introducing a 3.5D spatial and temporal blocking scheme applied to the input grid into on-chip memory to optimize bandwidth bounded kernels. Brodtkorb et al. [1] proposed an early exit mechanism to avoid further computation of blocks marked as non-contributing in the previous iteration. Sætra [18] proposed methods to reduce the computational burden and required memory in order to perform stencil operations over sparse domains.

Our blocking scheme extends the work of Brodtkorb et al. [1] to efficiently handle the activation and deactivation of the blocks, further reducing the unnecessary computation performed in each iteration of our stencil computation.

3 SP-ChainMail

The original ChainMail algorithm [10] defines a mesh structure over the elements of the volumetric model. Each element is connected to its six adjacent

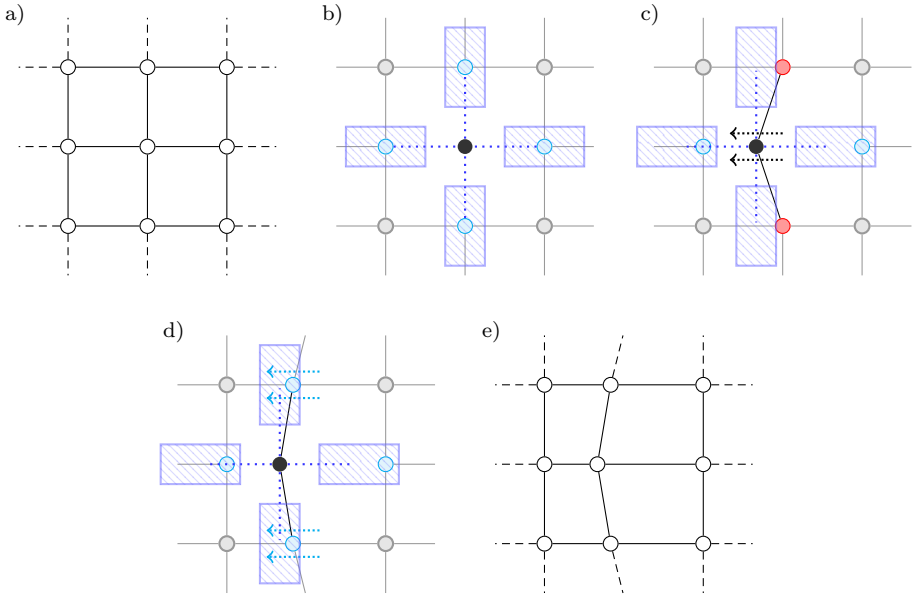

Fig. 1 2D depiction of an element deformation using the ChainMail algorithm. (a) Given an initial configuration: (b) the element defines valid regions for its neighbors; (c) when the element is displaced, it defines new valid regions; (d) the neighbors violating those new constraints are shifted to fulfill them. (e) The final stable state is reached when all the constraints are satisfied.

neighbors. A deformation is handled by two separate stages: propagation stage and relaxation stage.

In the *propagation stage*, a valid spatial region is defined for each neighbor of a given element. While a neighbor remains within that region, the state is valid and no updates are needed. Since the regions are defined relative to the current position of each element, the valid regions for the neighbors of an element are displaced when that element is displaced. Due to this displacement, a neighbor may be outside the new valid region. If this happens, the neighbor is shifted to a new location in order to fulfill the constraint, as shown in Fig. 1. The shifting of a neighbor may, in turn, lead to new constraint violations and cause further displacement of elements, propagating the deformation through the mesh elements.

Once all the constraints are satisfied, the *relaxation stage* begins: each element is iteratively displaced towards its equilibrium position based on a midpoint calculation of the positions of its neighbors. Rigid, plastic and elastic behaviors can be achieved by tuning up the geometric constraints between elements and modifying the relaxation scheme, as described by Gibson [7].

The ChainMail algorithm is implemented using the CPU since the propagation stage of the algorithm is inherently sequential, and the deformed mesh must be transferred to the graphics device memory to perform the visual-

ization. For large models, this memory transfer is expensive and impedes an interactive visualization of the applied deformations.

3.1 Sparse Parallel ChainMail

In our approach, the volumetric model is arranged as a regular, structured 3D grid of cells. Each cell corresponds to an element of the ChainMail mesh. Hence, a cell stores the 3D position of its associated element and the connections with its neighbors.

Two copies of the grid are used following a Jacobi sweep scheme [16]: one grid is designated to stencil read operations and other grid is designated to stencil write operations, swapping roles after each iteration. Both grids are stored in the device memory (dedicated memory of the GPU). Operations over cells corresponding to the propagation and relaxation stages are performed following a stencil computation approach:

- In the *propagation stage* the original propagation mechanism is inverted as explained by Rößler [17], adapting it to follow an iterative stencil-based approach: for each cell, if a neighbor has been displaced on the previous iteration, the new constraint is checked. If the constraint is not satisfied, the element is shifted to meet the existing geometric constraint with its neighbor. This process is repeated iteratively until all the constraints are satisfied.
- In the *relaxation stage*, a minimization process is applied based on the elastic and plastic properties of the model as explained by Gibson [7]. This energy minimization process also follows an iterative stencil-based approach, since each cell updates its position as a result of a computation regarding the current positions of its neighbors.

The computation performed for each cell during a propagation iteration, as well as during a relaxation iteration, is independent from the computation performed for the rest of the cells, allowing a parallel computation of each iteration.

Unlike the previous solutions, we introduce a mechanism to handle the stages at iteration level. This mechanism requires adding a control flag for each cell. This flag tracks whether the cell has already been reached by the current propagation front. Therefore, when a new external deformation is applied to a cell, it is flagged as *reached* and the rest of the cells are flagged as *not reached*. During subsequent propagation iterations, when an element is reached by the propagation front, it is flagged as *reached*.

After each iteration of the propagation stage, this flag allows to identify the cells that have already been reached by the propagation front. If a cell and its neighbors have already been reached, the cell is ready to perform the relaxation stage. Therefore, this flag allows overlapping the propagation and relaxation stages by alternating propagation and relaxation iterations. This overlapping mechanism allows to visualize partial results of the deformations and, since the

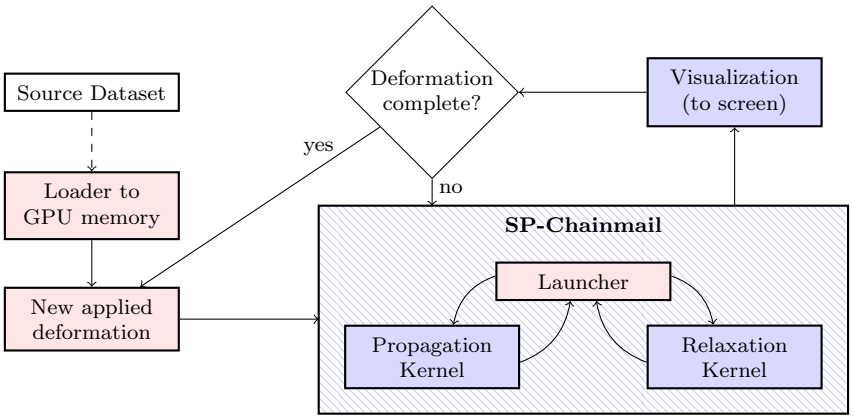

Fig. 2 Overview of the simulation system. The SP-ChainMail algorithm runs in parallel on the GPU, allowing partial visualizations of the deformations. Blue stages are carried out using the GPU, and red stages are carried out using the CPU.

updated data after any propagation or relaxation iteration is already present in the device memory, no memory transfers are required, allowing a more interactive visual feedback. An overview of the proposed algorithm integrated in a virtual surgery system is depicted in Fig. 2.

3.2 Parallel Implementation

In our approach, all the computationally intensive tasks are executed in parallel using the GPU. Hence, the 3D dataset is loaded into the device global memory as an array of cells. Each cell stores the following information:

- *Element data*: position and constraint values.
- *Neighbors flags*: a set of six flags indicating whether the element is connected or not with each of its neighbors (its six surrounding cells in the grid).
- *Activity flag*: a flag indicating if the element has been displaced in the preceding propagation iteration.
- *Reach flag*: a flag indicating if the element has already been reached by the current propagation.

In order to cope with the Jacobi sweep scheme, we duplicate the whole array supporting read and write operations. The current read array will be referred to as *global read array*, and the current write array will be referred to as *global write array*.

3.2.1 Propagation Stage

The propagation stage is implemented as a GPU kernel that is iteratively invoked. Each kernel invocation computes a single iteration of the propagation, generating one thread per cell. The kernel is described in Algorithm 1:

```
1  propagation_kernel (Cell readArray[], Cell writeArray[])
2      Integer id = get_thread_id();
3      Cell current = readArray[id];
4      current.activityFlag = False;
5      FOREACH neighbor IN activeNeighbors(current)
6          IF ( (neighbor.activityFlag == True) AND (
       restrictionsNotSatisfied(current, neighbor)) )
7              relocate(current);
8              current.activityFlag = True;
9              current.reachFlag = True;
10         ENDIF
11     ENDFOREACH
12     writeArray[id] = current;
13 END
```

Algorithm 1 Pseudo-code of the propagation kernel. During the kernel invocation, each instance of this kernel operates over a single cell, writing the resulting updated cell to the current global write array.

1. The cell data corresponding to the current thread is read from the global read array (lines 2-3).
2. For each neighbor, the following condition is checked (lines 5-6): the neighbor has been shifted in the previous propagation iteration and the new restrictions are violated.
3. If this condition is met, the current element is shifted in order to fulfill the new constraints, and it is flagged as *reached* and *active* (lines 7-9).
4. Otherwise, the current element is flagged as *inactive* (line 4).
5. Finally, the cell data is written to the global write array (line 12).

If no elements are shifted during the kernel invocation, the propagation stage finishes, and no more propagation iterations are needed.

3.2.2 Relaxation stage

The relaxation stage is also implemented as a GPU kernel that is iteratively invoked. Each invocation of the kernel computes a single relaxation iteration, generating one thread per cell. The kernel is described in Algorithm 2:

1. The cell data corresponding to the current thread is read from the global read array (lines 2-3).
2. The following condition is checked (line 4): the current element has already been reached but it is not active.
3. If this condition is met and all the neighbors of the current element have already been reached (lines 5-11), the relaxation process is applied to the current element (line 12).
4. Finally, the cell data is written to the global write array (line 15).

If none of the elements is shifted during a relaxation iteration and the propagation stage has already finished, the relaxation stage also finishes and the deformation is completed.

```
1  relaxation_kernel (Cell readArray[], Cell writeArray[])
2      Integer id = get_thread_id();
3      Cell current = readArray[id];
4      IF ( (current.reachFlag == True) AND (current.
       activityFlag == False ) )
5          Boolean continue = True;
6          FOREACH neighbor IN activeNeighbors(current)
7              IF (neighbor.reachFlag == False)
8                  continue = False;
9              ENDIF
10         ENDFOREACH
11         IF (continue == True)
12             applyRelaxationFunction(current);
13         ENDIF
14     ENDIF
15     writeArray[id] = current;
16 END
```

Algorithm 2 Pseudo-code of the relaxation kernel. During the kernel invocation, each instance of this kernel operates over a single cell, writing the resulting updated cell to the current global write array.

After an invocation of any of these kernels, the global arrays switch their roles, allowing the next kernel invocation to read from the updated array. Since the relaxation kernel only affects the elements already reached by the propagation, excluding those belonging to the current propagation front, both kernels can be interleaved. The visualization of the current state of the deformation is also possible by accessing the array data updated by the latest iteration.

Some details regarding our implementation have been omitted for the sake of clarity. In order to improve the efficiency of the GPU kernels we have adopted the following strategies:

- The *foreach* loops are completely unrolled.
- The GPU shared memory is used in order to optimize the access to neighboring cells.
- The actual data of the cells are stored in a Structure-of-Arrays fashion, more amenable to the regular memory access patterns of the kernels.

4 Computational Blocking Method

Our stencil approach presents a high spatial and temporal locality of the computational burden since the deformations applied to the model propagate iteratively through the regular grid following a wavefront pattern. This leads to a highly sparse computation in the propagation stage, resulting in a high amount of unnecessary computation.

This unnecessary computation is produced because many of the elements may have already been shifted in a previous iteration or have not yet been reached by the current propagation. A less severe sparse computation is also present in the relaxation stage because of the same reason. Due to this sparse

computation, many of the launched threads would be idle, wasting GPU resources since these threads also need to read from the device global memory to compute the data.

Since our solution follows an iterative stencil computation approach, we can introduce a blocking method to reduce the number of idle threads, optimizing the usage of the computation power offered by the GPU.

For this purpose, the computational domain is divided into blocks that can be computed independently. The storage of the dataset in the device memory remains the same, but each block is handled by an independent kernel launch instead of a single kernel launch over the whole dataset.

In order to maintain this structure, we store the corresponding 3D offset for each block. During a kernel launch for a particular block, the kernel receives this offset information to access the data of the cells in that block.

After each iteration, the blocks are flagged as *active* or *inactive*. *Active* means that the block may require further computation in the next iteration, while *inactive* means that the block will not need further computation in the following iteration. These flags allow launching the kernel only over active blocks in order to avoid unnecessary computation. This blocking scheme is applied to both stages of our algorithm in an efficient way as explained in the following subsections.

4.1 Efficient Activation and Deactivation of Blocks

In order to handle the activation and deactivation of blocks, we extend the solution proposed by Brodtkorb et al. [1], which involves the use of an auxiliary boolean buffer in order to indicate whether a block requires computation in the next iteration or not. In our approach, we use several of these buffers, referred to as *boolean maps*, to update and control the state of the blocks.

Each boolean map is stored as a global array on device memory containing one binary flag per block in the partition. A first boolean map is associated to the propagation stage. A second boolean map is associated to the relaxation stage.

In each iteration, for any of the both stages, the blocks that need to be computed in the next iteration are flagged as active in the corresponding boolean map. A new condition test, added to the end of the kernels code, decides whether a block requires further computation or not by checking if any element in the block has been updated. An element is considered updated by a propagation iteration if it has been reached by the propagation front. An element is considered updated by a relaxation iteration if it has been displaced by the relaxation function. On the other hand, if none of the elements in a block have been updated during the current iteration, the block is flagged as inactive. Fig. 3 shows a 2D illustration of this mechanism.

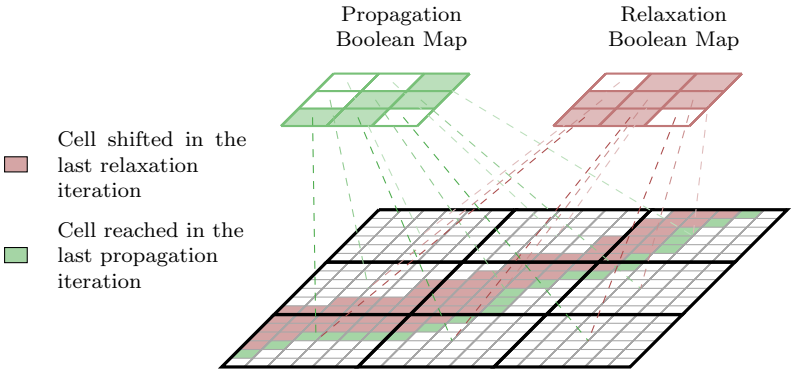

Fig. 3 2D simplification of the boolean map mechanism. Blocks containing cells reached in the last propagation iteration are flagged in the propagation boolean map, and blocks containing cells shifted in the last relaxation iteration are flagged in the relaxation boolean map.

4.1.1 Activation of Neighboring Blocks

When the propagation front or the relaxation process reaches the border of a block, the neighboring block must be activated. Six additional boolean maps are defined, each one associated to one of the borders for all the blocks.

Therefore, if an element belonging to the border of a block is updated in the current iteration, the position of that block in the boolean map corresponding to that border is set as *active*.

In the host memory (main memory), two lists of active blocks are maintained. At the end of each iteration, for any of the both stages, the corresponding boolean maps are copied to the host memory and the corresponding list is updated using the boolean maps as look-up tables. At the beginning of the next iteration, only the blocks indexed in the corresponding list are processed by the kernel.

Fig. 4 presents the steps and memory accesses during an iteration of the algorithm. Notice that the dataset is always stored in the device memory and operated from the GPU, and only the boolean maps are transferred to host memory.

4.2 Parallel Boolean Reduction

As mentioned earlier, the boolean maps are updated by the kernels but, since each launched thread handles only one cell, it is necessary to perform a gathering process regarding each block. Instead of a parallel reduction approach as in [1] and [18], we propose a novel two-step Parallel Boolean Reduction (PBR) mechanism:

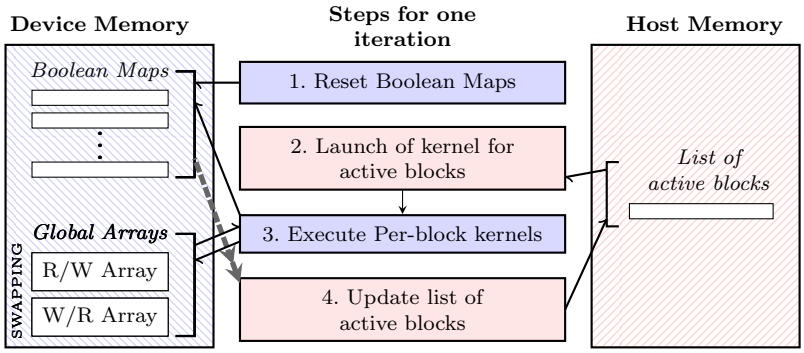

Fig. 4 Steps and memory accesses during an iteration. The blue steps (steps 1 and 3) run on the GPU, while the red steps (steps 2 and 4) run on the CPU. The only memory transfer between device memory and host memory is performed in the step 4 in order to update the active blocks list.

1. All the flags of the boolean maps are set as *inactive* before launching the kernels corresponding to the active blocks, assuming that none of the blocks will need further computation in the next iteration.
2. The kernels corresponding to the active blocks are launched. If an element of a block is updated, a write operation is performed to set as *active* the position of that block in the corresponding boolean map.

Although the concurrent writing of several threads to the same variable does not guarantee the integrity of data, in this case all the threads write the same value. This fact ensures the final state of the boolean values while avoiding the additional latency introduced by a parallel reduction approach. A depiction of both approaches operating over the same set of values is shown in Fig. 5. A performance comparison of both methods is presented in the next section.

5 Experiments and Results

In order to demonstrate the benefits of the proposed methods, several tests have been conducted using different hardware configurations to also evaluate the portability and scalability of the proposed blocking scheme. Three hardware configurations have been used:

- **GTS-250** configuration: Intel Core i3-530 2.93 GHz, 4 GB RAM, Nvidia GeForce GTS 250 (Tesla microarchitecture, 128 cores) with 1 GB of video memory GDDR3. OpenCL 1.1 driver included in CUDA 6.
- **R9-270X** configuration: Intel Core i5-3570 3.4 GHz, 8 GB RAM, AMD Radeon R9 270X (1280 cores) with 2 GB of video memory GDDR5. OpenCL 1.2 driver.

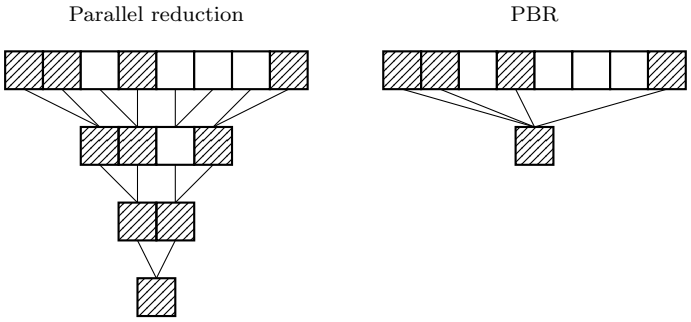

Fig. 5 Left: parallel reduction of eight boolean values using the binary OR operation. Right: parallel boolean reduction (PBR) of the same eight boolean values.

– **GTX-670** configuration: Intel Core i7-3770 3.4 GHz, 16 GB RAM, Nvidia GeForce GTX 670 (Kepler microarchitecture, 1344 cores) with 2 GB of video memory GDDR 5. OpenCL 1.1 driver included in CUDA 6.

Two different datasets have been used for the tests. The first dataset, referred to as *Cube dataset*, is a synthetic regular 3D cube, consisting of $96 \times 96 \times 96$ elements. The second dataset, referred to as *Leg dataset* is a section of a leg from the Visible Human Project of the National Library of Medicine (see Fig. 6), consisting of $160 \times 160 \times 160$ elements.

5.1 SP-ChainMail Performance

To evaluate the performance of our approach, the SP-ChainMail algorithm has been implemented, together with the blocking scheme, using OpenCL [15], integrating it into a virtual surgery system prototype.

The original ChainMail algorithm [10] has also been implemented as a reference. The propagation stage of the original algorithm is inherently sequential and cannot be parallelized for multicore processors, but the relaxation stage has been parallelized using OpenMP by dividing the ChainMail elements in balanced groups and assigning the computation of each group to one thread.

A deformation has been applied to each dataset, causing a propagation-relaxation through the whole dataset affecting all the elements. For the Cube dataset, the SP-ChainMail algorithm took 285 propagation iterations and 468 relaxation iterations until a completely stable configuration was reached. The original ChainMail algorithm also required 468 relaxation iterations after the propagation (not measurable in iterations). For the Leg dataset the SP-ChainMail algorithm took 477 propagation iterations and 788 relaxation iterations. The original ChainMail algorithm also required 788 relaxation iterations after the propagation. Each test has been repeated five times, although no noticeable differences were encountered through the different executions due to the deterministic behavior of the algorithms. The results presented here report the average of the measured times.

Fig. 6 Left: The Leg dataset, used in the performance tests. Right: The Leg dataset deformed by the Sparse Parallel ChainMail algorithm, using the developed virtual surgery system.

The original ChainMail was tested only on the GTX-670 configuration, our most modern hardware configuration. Using 8 threads for the relaxation computation (grouping the elements in 8 groups), the stable state was reached after 10,648 ms for the Cube dataset and 49,283 ms for the Leg dataset.

We tested the SP-ChainMail implementation on both datasets using different block sizes for the blocking scheme, resulting in different rates of reduction on the total number of launched GPU threads. As can be seen in Table 1, a smaller block size always implies a higher reduction in the number of launched threads, which is expected since the smaller block sizes lead to a finer adjustment of the active blocks to the actual propagation front.

Table 2 shows the measured times using the Cube dataset and Table 3 shows the measured times using the Leg dataset. The times reported represent the time taken to reach the stable state for the same applied deformation to the dataset. The speed-up with respect to the original ChainMail (running on the GTX-670 configuration) is also reported in both tables.

The tests reveal that the SP-ChainMail outperforms the original ChainMail even using relatively old GPUs, achieving notable speed-up factors higher than 20x when using a modern GPU. Interestingly, the results show that smaller blocks do not always lead to a higher speed-up although the number of launched threads is smaller. This is due to the fact that the smaller kernel launches do not create enough parallel threads to fully hide the memory access latency, and this overhead, added to the overhead of managing more kernel launches, gradually decimates the gain of the reduced computation load.

Table 1 Thread launch reduction achieved using different block sizes.

	Cube dataset			Leg dataset		
Block size	**# of Blocks**	**Launched threads**	**Thread launch reduction**	**# of Blocks**	**Launched threads**	**Thread launch reduction**
No blocks	-	666,206,208	-	-	5,181,440,000	-
32x32x32	27	234,553,344	64.79%	125	1,018,888,192	80.33%
32x16x16	108	148,873,216	77.65%	500	639,262,720	87.66%
16x16x16	216	107,683,840	83.83%	1,000	465,821,696	91.01%
32x8x8	432	103,868,416	84.40%	2,000	451,835,904	91.27%
16x8x8	864	67,950,592	89.80%	4,000	297,355,264	94.26%
8x8x8	1,728	49,827,840	92.52%	8,000	219,824,128	95.75%

Table 2 Measured times of our SP-ChainMail implementation using the Cube dataset. Speed-up factors relative to the original ChainMail are also shown.

	GTS-250		R9-270X		GTX-670	
Block Size	**Time (ms)**	**Speed-up**	**Time (ms)**	**Speed-up**	**Time (ms)**	**Speed-up**
No blocks	8,159	1.31x	1,131	9.41x	1,545	6.89x
32x32x32	4,293	2.48x	693	**15.36x**	705	15.10x
32x16x16	3,476	**3.06x**	815	13.06x	635	16.76x
16x16x16	3,720	2.86x	1,015	10.49x	598	**17.80x**
32x8x8	3,923	2.71x	1,484	7.17x	893	11.92x
16x8x8	4,363	2.44x	1973	5.39x	1,064	10.01x
8x8x8	5,668	1.87x	2,678	3.97x	1,443	7.37x

Table 3 Measured times of our SP-ChainMail implementation using the Leg dataset. Speed-up factors relative to the original ChainMail are also shown.

	GTS-250		R9-270X		GTX-670	
Block Size	**Time (ms)**	**Speed-up**	**Time (ms)**	**Speed-up**	**Time (ms)**	**Speed-up**
No blocks	61,271	0.80x	7,601	6.48x	11,448	4.30x
32x32x32	20,658	2.39x	2,683	**18.36x**	2,810	17.53x
32x16x16	16,130	**3.05x**	3,207	15.36x	2,428	20.29x
16x16x16	17,261	2.85x	3,910	12.60x	2,243	**21.97x**
32x8x8	18,128	2.72x	5,742	8.58x	3,438	14.33x
16x8x8	19,030	2.59x	7,718	6.39x	4,207	11.71x
8x8x8	25,291	1.95x	11,240	4.38x	5,960	8.27x

5.2 Blocking Method Portability

For the GTS-250 configuration, any of the tested block sizes leads to a significant speed-up with respect to the non-partitioned case (i.e., the SP-ChainMail witouth using the blocking scheme), being the speed-up factor higher when using the Leg dataset, since the thread launch reduction is higher. As already

mentioned, the gain is gradually decimated as the block size is reduced, due to the added overhead, as can be seen in Fig. 7.*a* and Fig. 7.*d*.

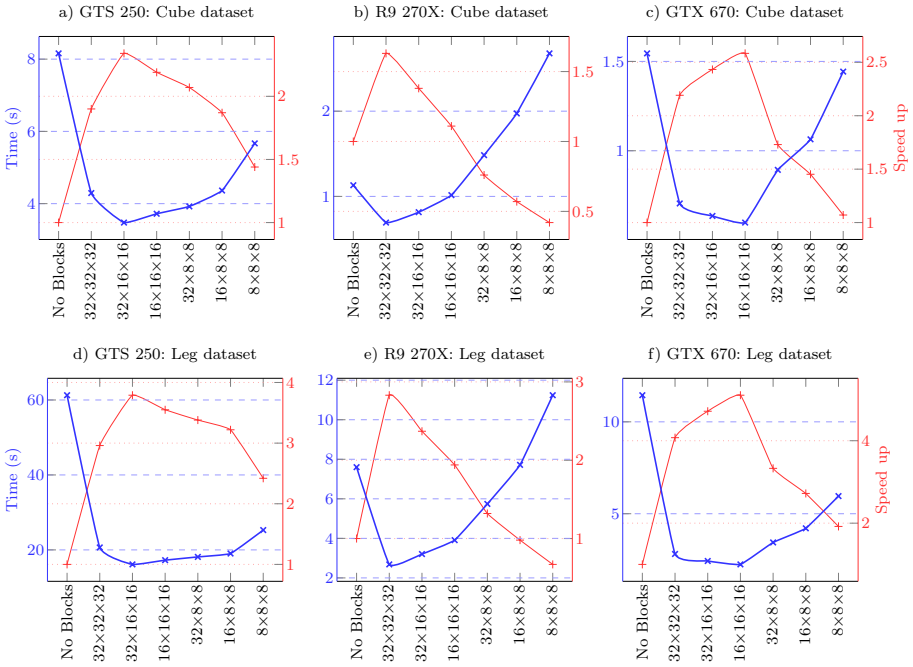

Fig. 7 Plots showing the time reduction and speed-up factor measured for the different block sizes with respect to the non-partitioned SP-ChainMail case. The top row shows the results using the Cube dataset. The bottom row shows the results using the Leg dataset.

R9-270X and GTX-670 configurations exhibit a similar behavior (Figs. 7.*b*, 7.*c*, 7.*e* and 7.*f*) but, since the more recent GPUs present in those configurations have a much higher amount of stream processing units, they require an even higher amount of parallel threads to hide memory latency, and the smaller block sizes cannot even fully populate the GPU cores, leading to a loss of effective computation power. This loss is most severe in the case of the R9-270X configuration, on which the use of small block sizes even yields a worse performance than the non-partitioned case.

Despite this effect, the use of a reasonable block size (which depends on the particular GPU architecture) leads to a noticeable speed-up using any of the three configurations, showing the portability of the proposed blocking method and the performance gain obtained through its use.

5.3 Scalability Test

Notice that the previous tests on the Leg dataset achieve a higher speed-up than their counterparts using the Cube dataset, suggesting a good scalability

of the blocking method regarding the dataset size. In order to further analyze the scalability of our blocking method with respect to the dataset size, a second test using synthetic regular datasets, with dimensions ranging from $32 \times 32 \times 32$ to $224 \times 224 \times 224$, has been performed.

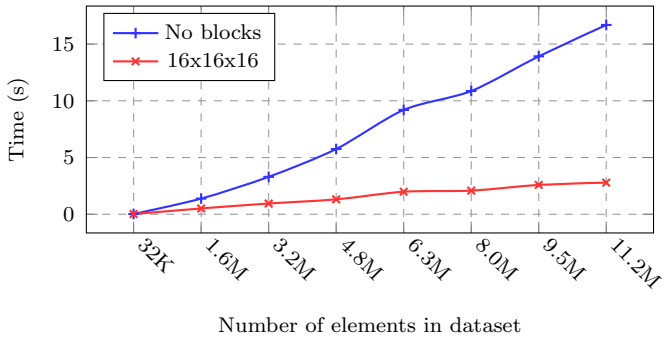

Fig. 8 Measured times of the scalability test, using the GTX-670 configuration.

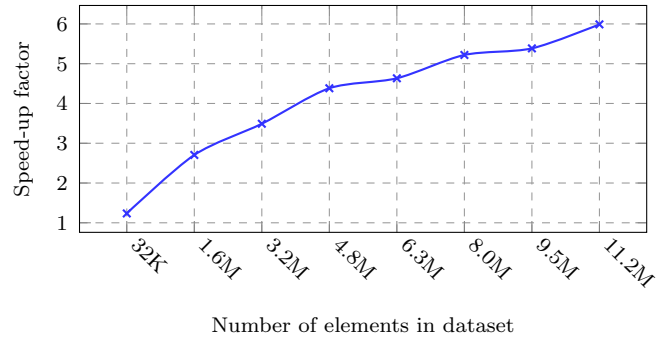

Fig. 9 Speed-up factor achieved for the increasing dataset size of the scalability test, using the GTX-670 configuration.

The most modern configuration (the GTX-670 configuration) has been used to perform this test. A deformation affecting all the elements of the dataset has been applied, measuring the propagation time without using the blocking method and measuring the propagation time of the same deformation using a block size of $16 \times 16 \times 16$, which achieved the best performance gain on the GTX-670 configuration.

The measured times corresponding to this second test, presented in Fig. 8, show a significant reduction of the propagation time for all the tested dataset sizes, and they also show a good scalability of the proposed method since the speed-up factor, shown in Fig. 9, also increases when increasing the dataset size.

5.4 Memory Requirements

5.4.1 SP-ChainMail Memory Requirements

The memory requirements of our SP-ChainMail algorithm (corresponding to the Global Arrays in Fig. 4) scale linearly with the number of elements in the input dataset.

For each element, 30 bytes of device memory are required, which leads to a total amount of 480 MB for an input dataset of $256 \times 256 \times 256$ elements, and a total amount of 3.75 GB for an input dataset of $512 \times 512 \times 512$ elements, an amount currently offered only by high-end GPUs. However, this limitation is not reached in most scenarios, such as virtual surgery applications, since the simulation is usually performed on a sub-region of the dataset, and SP-ChainMail information would only be generated for the elements of the sub-region in those cases.

5.4.2 Blocking Method Memory Requirements

The blocking method has very low host and device memory requirements.

In host memory (List of active blocks in Fig. 4), 64 bytes are required per block. In device memory, only 8 bytes are required per block.

In our most memory demanding test (the Leg dataset with a $8 \times 8 \times 8$ block size, generating 8,000 blocks in the partition) required 500 KB of device memory and 62.5 KB of device memory. As mentioned in section 4.1.1, only the boolean maps are transferred from device memory to host memory at the end of each iteration. Even in our most memory demanding test, this transfer consumes less than 1 ms, which is a negligible overhead considering the achieved gain.

5.5 PBR performance Test

A performance test comparing the proposed Parallel Boolean Reduction (PBR) algorithm with a general parallel reduction algorithm has been conducted. Both algorithms have been applied to reduce several arrays of boolean elements of a wide range of sizes.

The measured times of both algorithms using the GTX-670 configuration are shown in Fig. 10. The PBR algorithm shows a better performance for all the array sizes, since less read/write operations are needed and no synchronization steps are required.

6 Conclusions and Future Work

In this work we have presented a Sparse Parallel ChainMail algorithm. The proposed algorithm has been implemented and integrated into a virtual surgery

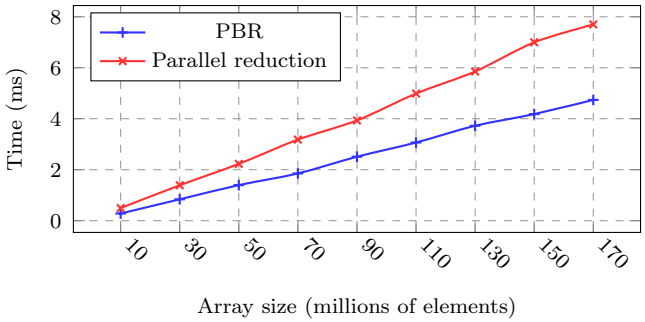

Fig. 10 Comparison of the times required by the parallel reduction algorithm and the PBR algorithm to perform a reduction over several boolean arrays, using the GTX-670 configuration.

system, allowing an interactive visual feedback during the manipulation of large volumetric models. Following a stencil computation approach, our algorithm adapts to the modern GPU computation paradigm.

We have proposed and implemented a 3D blocking method to deal with the sparse nature of the SP-ChainMail computation, drastically reducing the amount of idle GPU threads created.

A novel parallel boolean reduction mechanism has been used to efficiently handle the activation and deactivation of blocks. This reduction approach has been proven faster than a generic parallel reduction approach, and it can be used in any context in which the reduced value has a boolean nature, i.e., there are only two possible output values.

The tests conducted in this work show that our implementation considerably outperforms a parallel multithreaded implementation of the original ChainMail algorithm, and our blocking method effectively reduces the computation time required for the deformations, enhancing the interactivity of the simulation system. The tests also show a good portability and scalability of the blocking scheme, which increases its effectiveness as the dataset size increases, while the required additional memory is negligible.

As future lines of research, we intend to include an auto-tuning mechanism to determine the optimal block size automatically for each hardware and software configuration. Another interesting future line of work is the generalization and further testing of the blocking scheme for stencil computation approaches. Moreover, it would be interesting to test the use of dynamic parallelism to perform the handling and launching of the blocks directly from the GPU.

Acknowledgements This work is supported by the "Formación de Profesorado Universitario, Plan Propio de Investigación" program of the University of Granada. This work is also supported by the project TIN2014-60956-R of the Spanish Ministry of Economy and Competitiveness. JMM acknowledges the Spanish MINECO project MTM2014-52056-P.

References

1. Brodtkorb, A.R., Sætra, M.L., Altinakar, M.: Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. Computers & Fluids **55**, 1–12 (2012)
2. Comas, O., Taylor, Z.A., Allard, J., Ourselin, S., Cotin, S., Passenger, J.: Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA. In: Biomedical Simulation, pp. 28–39. Springer (2008)
3. Datta, K., Murphy, M., Volkov, V., Williams, S., Carter, J., Oliker, L., Patterson, D., Shalf, J., Yelick, K.: Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, p. 4. IEEE Press (2008)
4. De La Asunción, M., Mantas, J.M., Castro, M.J.: Simulation of one-layer shallow water systems on multicore and CUDA architectures. The Journal of Supercomputing **58**(2), 206–214 (2011)
5. Fortmeier, D., Mastmeyer, A., Handels, H.: Image-based palpation simulation with soft tissue deformations using chainmail on the GPU. In: Bildverarbeitung für die Medizin 2013, pp. 140–145. Springer (2013)
6. Fortmeier, D., Mastmeyer, A., Handels, H.: An image-based multiproxy palpation algorithm for patient-specific VR-simulation. Studies in health technology and informatics **196**, 107 (2014)
7. Frisken-Gibson, S.F.: Using linked volumes to model object collisions, deformation, cutting, carving, and joining. Visualization and Computer Graphics, IEEE Transactions on **5**(4), 333–348 (1999)
8. Georgii, J., Echtler, F., Westermann, R.: Interactive simulation of deformable bodies on GPUs. In: SimVis, pp. 247–258 (2005)
9. 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, pp. 367–378. Springer (1997)
10. Gibson, S.F.: 3D ChainMail: a fast algorithm for deforming volumetric objects. In: Proceedings of the 1997 symposium on Interactive 3D graphics, pp. 149–ff. ACM (1997)
11. Kirk, D.B., Wen-mei, W.H.: Programming massively parallel processors: a hands-on approach. Morgan Kaufmann (2012)
12. Le Fol, T., Acosta-Tamayo, O., Lucas, A., Haigron, P.: Angioplasty simulation using ChainMail method. In: Medical Imaging, pp. 65,092X–65,092X. International Society for Optics and Photonics (2007)
13. Mensmann, J., Ropinski, T., Hinrichs, K.: Interactive cutting operations for generating anatomical illustrations from volumetric data sets (2008)
14. Micikevicius, P.: 3D finite difference computation on GPUs using CUDA. In: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, pp. 79–84. ACM (2009)
15. Munshi, A., et al.: The OpenCL specification. Khronos OpenCL Working Group **1**, l1–15 (2009)
16. Nguyen, A., Satish, N., Chhugani, J., Kim, C., Dubey, P.: 3.5-D blocking optimization for stencil computations on modern CPUs and GPUs. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–13. IEEE Computer Society (2010)
17. Rößler, F., Wolff, T., Ertl, T.: Direct GPU-based volume deformation. Proceedings of Curac pp. 65–68 (2008)
18. Sætra, M.: Shallow water simulation on GPUs for sparse domains. In: Numerical Mathematics and Advanced Applications 2011, pp. 673–680. Springer (2013)
19. 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 Intervention, pp. 679–687. Springer (1998)
20. Schulze, F., Bühler, K., Hadwiger, M.: Interactive deformation and visualization of large volume datasets. In: GRAPP (AS/IE), pp. 39–46. Citeseer (2007)

2.3 Parallel deformation of heterogeneous ChainMail models: application to interactive deformation of large medical volumes

- **A. Rodríguez**, A. León and G. Arroyo (2016). "Parallel deformation of heterogeneous ChainMail models: Application to interactive deformation of large medical volumes". *Computers in Biology and Medicine*, Volume 79, pp. 222-232.
  - Status: Published [104]
  - Impact factor (JCR 2015): 1.521
  - Subject category: Biology (Q2: 35/84), Computer Science, Interdisciplinary Applications (Q3: 54/105), Mathematical and Computational Biology (Q2: 19/57), Engineering, Biomedical (Q3: 45/77).

Parallel deformation of heterogeneous ChainMail models: application to interactive deformation of large medical volumes

Alejandro Rodríguez[a,*], Alejandro León[a], Germán Arroyo[a]

[a]*ETSIIT, University of Granada, Granada, Spain*

Abstract

In this work we present a new solution for correctly handling heterogeneous materials in ChainMail models, which are widely used in medical applications. Our core method relies on two main components: (1) a novel timestamp-based propagation scheme that tracks the propagation speed of a deformation through the model and allows to correct ambiguous configurations, and (2) a novel relaxation stage that performs an energy minimization process taking into account the heterogeneity of the model. In addition, our approach extends the SP-ChainMail algorithm by supporting interactive topology changes and handling multiple concurrent deformations, increasing its range of applicability. Finally, we present an improved blocking scheme that efficiently handles the sparse computation, greatly increasing the performance of our algorithm.

Our proposed solution has been applied to interactive deformation of large medical datasets. The simulation model is directly generated from the input dataset and an user defined material transfer function, while the visualization of the deformations is performed by rendering the resampled deformed model using direct volume rendering techniques. In our results, we show that our parallel pipeline is capable of interactively deforming models with several million elements. A comparison is finally discussed, analyzing the properties of our approach with respect to previous work. The results show that our algorithm correctly handles very large heterogeneous ChainMail models in an interactive manner, increasing the applicability of the ChainMail approach for more demanding scenarios both in response time and material modeling.

Keywords: ChainMail, Soft tissue deformation, Volume rendering, GPU

*Corresponding author. Tel.: +34 687 63 22 88; email: alejandrora@ugr.es

1. Introduction

The use of computational simulation methods is nowadays extensively used in the fields of biology and medicine. In particular, many simulation applications related to the medical field such as the simulation of physiological processes [1, 2] or the simulation of surgical procedures [3], require a complex simulation of the human anatomy. The deformation of human tissues exhibits complex heterogeneous and non-linear behaviors that have been approached by many different solutions over the years [4, 5]. In the case of interactive applications, a trade-off between performance and deformation accuracy has to be reached, since the requirements regarding to visual and interaction feedback impose strong response time restrictions [6].

Patient-specific data captured through CT or MRI scans include up to several million voxels, each one storing information related to different tissues and organs. Physically-based simulation methods such as mass-spring models or finite element methods reproduce many heterogeneous behaviors accurately, but the computational cost of these approaches limits the resolution of the models to a small fraction of the captured medical data resolution which is usually several orders of magnitude higher. This disparity hinders the modeling process and leads to an inevitable loss of heterogeneity and complexity of the source data.

In order to increase the resolution of the deformation models, Gibson proposed the ChainMail algorithm [7] by following a geometrical approach that enables physically realistic deformations. Since it is a purely geometrical approach, it is unconditionally stable and is computed in a well bounded execution time, making it suitable for interactive applications. Moreover, it is capable of simulating many complex human tissue behaviors such as non-linear stress-strain response, hysteresis and asymptotic relaxation. Thanks to these properties, its usage has spread to many medical applications, as we review in Section 2.

In our previous work, we proposed the SP-ChainMail algorithm [8], a parallel version of the original ChainMail algorithm. It allows to interactively deform models for very high resolutions by computing its propagation and relaxation stages using modern GPU parallel capabilities and integrating a blocking scheme to handle the sparse computation.

Both the original and the SP-ChainMail assume a homogeneous material for the model, thus the heterogeneity of the medical data is incorrectly handled in many situations. This limitation motivates our work, since an efficient and correct handling of the heterogeneity in the models would lead to a wider range of applicability of the ChainMail approach in medical contexts.

In this work, we extend the SP-ChainMail algorithm to correctly manage heterogeneous materials in the model, interactive topology changes and multiple concurrent deformations while preserving its efficient computation. We also propose a modification of the blocking scheme to further improve the performance of the sparse parallel computation.

We have integrated the proposed algorithm into a GPU-based pipeline for direct deformation of medical datasets. A ChainMail model is created from the input dataset, comprising all the heterogeneous information present in the data since it is created at the same resolution. After a deformation is applied, the deformed model is resampled in runtime into a voxel grid which is then visualized using standard direct volume rendering. All the main stages of the proposed pipeline are executed in parallel by the GPU, which allows to interactively deform and visualize models up to several million elements.

The remainder of this paper is organized as follows. In Section 2 we review related work, focusing on the ChainMail algorithm and its derived works. In Section 3 we present all the aspects related to our proposed ChainMail algorithm. Section 4 describes the application of the proposed algorithm for interactive manipulation of medical datasets, including the model creation process from a source medical dataset and issues related to visualization and interaction with the model. In Section 5, we present the results of several tests using the proposed pipeline, including a comparative experiment with the SP-ChainMail algorithm. We also analyze and discuss aspects related to the efficiency and memory requirements of our approach. Finally, our conclusions are listed in Section 6.

2. Related Work

Interactive simulation of deformable models with biomechanical behaviors has been of interest for several decades. During this period, many approaches have been proposed, although this paper focuses on the ChainMail algorithm and its derived works. For a more general overview of the different approaches, we refer the reader to the surveys of Meier et al. [4] and Moore et al. [5].

The original ChainMail algorithm is a physically motivated approach that defines a *ChainMail model* as a regular 3D mesh structure of elements arranged in a regular grid. Each element is connected to its six nearest neighbors. These elements define geometrical limits for their neighbors based on the current position of the elements and the properties of the tissue modeled

by them. Henceforth, we will refer to these geometrical limits simply as *constraints*. The valid regions that an element imposes to its neighbors are thus defined through geometric constraints for the position $P_n = (x_n, y_n, z_n)$ of the neighbor. The constraints are defined by the *maxD* and *minD* parameters, which control the stretching and contraction limits, and the $maxHorizD_1$ and $maxHorizD_2$ parameters, which control shear limits. Fig. 1 shows the parametrization for a right neighbor, yielding the constraints:

$$x_e + minDx \leq x_n \leq x_e + maxDx, \tag{1}$$

$$y_e - maxHorizDx_1 \leq y_n \leq y_e + maxHorizDx_1, \tag{2}$$

$$z_e - maxHorizDx_2 \leq z_n \leq z_e + maxHorizDx_2, \tag{3}$$

where $P_e = (x_e, y_e, z_e)$ is the current position of the element. These constraints are applied for a right neighbor, and analogous constraint equations are found for the other neighbors (back, front, top, bottom and left) using *Dy* and *Dz*-based parameters.

Thus when an element is displaced and the constraints are violated, the neighbors violating the constraints are shifted to valid positions, which may lead to new constraint violations, and the deformation is propagated through the model as a chain reaction by enforcing those constraints. After all the constraints are met, the resulting configuration is relaxed in a second stage by adjusting the elements towards an optimal energy configuration. Non-linear elastoplastic deformations and other complex behaviors can be achieved by tuning the geometric constraints between elements and by modifying the relaxation scheme, as described in [9]. This algorithm has been used in many medical applications such as arthroscopic knee surgery [10], simulation of an angioplasty procedure [11], liver biopsy simulation [12, 13], non-rigid image registration [14], automatic segmentation [15] and interactive generation of medical illustrations by cutting and deforming medical models [16].

Other versions of the original algorithm have been proposed for different applications, such as real time haptic rendering [17, 18, 19], biomechanical modeling of organs [20, 21, 22, 23] or patient palpation simulation [24, 25].

One of the most important lacking features of the ChainMail algorithm is the support of heterogeneous materials, since a homogeneous material is assumed for the entire model, i.e., *maxDx*, *minDx*, $maxHorizDx_1$, $maxHorizDx_2$, used in Eqs. (1) – (3) (and their analogous *Dy* and

Dz-based parameters) have to remain constant for all the elements of the model, which limits its application to real data.

In order to overcome this limitation, Schill et al. [26] proposed a modification of the order of propagation. A dynamic ordered list is maintained and, instead of a First-Moved-First-Processed criterion for shifting the candidate elements, the element with highest constraint violation is processed first. This propagation mechanism enables the use of heterogeneous materials in the models, but makes the processing very slow since, after processing each element, all the new candidate elements have to be inserted and sorted in the list prior to processing the next one. Moreover, the relaxation stage is not modified and a homogeneous material is still considered, which reduces the plausibility of the final achieved configuration. This algorithm has also been used in medical applications such as vitrectomy simulation [26, 27] and heterogeneous deformation of large medical datasets [28].

The SP-ChainMail algorithm [8] increases the performance of the original algorithm by more than one order of magnitude by mapping the computation of the propagations to the GPU. In this algorithm, the propagation process is inverted: when an element is shifted due to a new deformation, a parallel iterative process is launched on the GPU. For each iteration, a thread is launched per element. Each thread checks whether any of the neighbors of the current element has been moved in the previous iteration. If this condition is met, the new constraints imposed by that neighbor are checked and the element is shifted if necessary in order to satisfy the constraints. Each iteration is performed, and the process continues until no element is shifted during an iteration.

This iterative approach propagates the deformations through the model following a wavefront pattern, thus an element is always reached first by the propagation through the shortest path from the source of the propagation to the element. In the presence of heterogeneous materials, this treatment leads to incorrect and unpredictable behaviors, limiting its use to homogeneous materials.

Our proposal extends the SP-ChainMail algorithm, modifying both the propagation and the relaxation stages, generalizing its applicability to heterogeneous materials. Our solution also implicitly copes with interactive topology changes in the model.

3. Heterogeneous Parallel ChainMail

In this section we describe our extended ChainMail algorithm as well as many important aspects related to its parallelization.

After defining our specification of constraints and links in the model, we detail the new propagation and relaxation stages which ensure the correct handling of heterogeneous materials in the model. This is achieved by means of a novel timestamp system and a heterogeneous energy minimization process respectively. We also explain how interactive topological operations are managed by the algorithm.

Finally, we detail important improvements that allow the concurrent handling of different deformation wavefronts and a faster computation of the sparse blocking scheme.

3.1. Heterogeneous constraints

As in the original ChainMail algorithm, the elements in our models are initially arranged in a regular grid structure. Each element is connected to its six nearest neighbors (back, front, top, bottom, left and right) given this initial configuration.

We define the geometric constraints imposed by one element to a neighbor as shown in Fig. 2. A box center, relative to the current position of the element, is calculated attending to the initial separation of the elements. Three parameters, namely Dx, Dy and Dz, specify the dimensions of the axis aligned box that define a valid region for the neighbor, controlling the stretching and shearing limits, yielding the constraints:

$$x_e + x_{sep} - Dx \leq x_n \leq x_e + x_{sep} + Dx, \tag{4}$$

$$y_e + y_{sep} - Dy \leq y_n \leq y_e + y_{sep} + Dy, \tag{5}$$

$$z_e + z_{sep} - Dz \leq z_n \leq z_e + z_{sep} + Dz, \tag{6}$$

for the position $P_n = (x_n, y_n, z_n)$ of the neighbor, where $P_e = (x_e, y_e, z_e)$ is the current position of the element and $\Delta_e = (x_{sep}, y_{sep}, z_{sep})$ is the initial separation of the element and its neighbor.

Isotropic materials for an element can be defined by assigning the same constraints to each neighbor of its neighborhood, whereas anisotropic materials can be defined by assigning different constraints depending on the direction of the neighbor.

Since every element may be defined with a specific material, the actual constraint parameters are stored in the links, i.e., each link stores its own Dx, Dy and Dz parameters, which can be different for each link. Thus the constraint parameters stored in a link are computed as a contribution of the two connected elements by averaging their constraint parameters. These averaged parameters stored in the link determine the constraints that both elements impose to each other. For the sake of clarity we assume the same stretching and shearing limits for any material (i.e., $Dx_e = Dy_e = Dz_e \ \ \forall \boldsymbol{e} \in \boldsymbol{L}$, being $\boldsymbol{L}$ the set of elements in the model) and thus a single parameter per element is required, though the extension for generalized materials is straightforward.

An example of the computation of the heterogeneous constraints of a simple 2D model can be seen in Fig. 3: the constraint parameters of the elements are averaged and stored on the links between them (a); the averaged constraint parameter of elements A and B determines the constraints imposed by element A on element B (b) and by element B on element A (c, left); the averaged constraint parameter of elements B and C determines the constraints imposed by element B on element C (c, right) and by element C on element B (d).

3.2. Heterogeneous propagation

In order to properly deal with heterogeneous materials using the ChainMail approach, the propagation speed of a deformation through different tissues must be considered.

This evidence was first detected by Schill et al. [26]. Since the ChainMail algorithm follows a purely geometric approach, the propagation speed is inversely proportional to the constraint values. They addressed the issue using a priority queue; the neighbors of the last displaced element are added to the queue, which is sorted attending to the amount of constraint violation.

Although this approach correctly propagates the heterogeneous deformations through the volume, it is not suited for a parallel computation, since the next element to process is only known after the last element has been processed and the priority queue has been updated, which limits the processing to a strict serial computation.

To handle this phenomenon without losing the parallel nature of the SP-ChainMail algorithm, our algorithm modifies the propagation process by taking into account the time required by the wavefront to travel through the different materials. As we explain in the following subsection, this allows to adjust already deformed elements in order to obey the priority of the constraints.

3.2.1. Timestamp-based propagation

Our iterative algorithm propagates the deformation applied to an element as a wave through the model. Thus, when a deformation is applied, a *wavefront* is generated and evolved iteratively, which is composed by the elements reached by the propagation in the current iteration. For each iteration, the deformation propagates from the elements reached on the previous iteration to their neighbors, thus the wavefront advances in all the directions by one step. Due to the nature of this process, the elements are always reached first by the shortest path from the epicenter of the deformation to the element. For this reason, if an already reached element is later reached by the same wavefront through a path composed by links with stronger constraints (i.e., shorter propagation times), its position needs to be readjusted.

In order to detect and correct this casuistry, each link stores the time required to travel through it. This time is directly proportional to the constraint value assigned to it. As the wavefront iteratively travels through the model, the algorithm tracks the speed at each point of the wavefront by accumulating the time required in each step and storing it in the reached elements as a timestamp mark. This mark intuitively indicates the time elapsed since the beginning of the propagation until that element was reached, and depends on the stiffness of the links in the path that led the propagation from the source of the deformation to the element. A simple 2D example is shown in Fig. 4, where a deformation is applied on element A, setting its timestamp mark to 0 and initiating a propagation, first reaching element B, updating its timestamp mark to 1.5 = 0 + 1.5, and finally reaching element C, updating its timestamp mark to 4.5 = 1.5 + 3.

When an element reached by a previous iteration is reached again by the wavefront and the current accumulated time of the wavefront is smaller than the stored timestamp mark, the algorithm imposes the current propagation of the wavefront over the previous one, since it traveled through a path with stronger constraints.

Regarding to the implementation of this algorithm, this can be easily adapted as a GPU kernel that is iteratively launched over the elements of the model. For each element we add the *timestamp mark*, and for each link we add the *propagation time*, i.e., the time required to travel through it, which is directly derived from the constraint value of the link. When a new deformation is applied to the model, the timestamp of the element receiving the deformation is set to 0. After that, the iterative propagation process starts. In each iteration, the kernel is launched creating one GPU thread per element. The kernel is described in Algorithm 1. Each thread it-

erates over the neighbors of its assigned element (lines 2-11). For each neighbor, a candidate timestamp is calculated as the sum of the neighbor's timestamp and the propagation time of the link between them (line 4). If the candidate timestamp is smaller than the current timestamp of the element, it is assigned as the new timestamp and the constraint regarding that neighbor is checked, shifting the element if necessary (lines 5-10). If the candidate timestamp is greater or equal than the current timestamp of the element, the neighbor is ignored.

```
1  BEGIN propagationKernel(elem)
2   N = getNeighbors(elem);
3   FOREACH n IN N
4    newTS = n.tStamp + pTime(elem, n);
5    IF (newTS < elem.tStamp)
6     elem.tStamp = newTS;
7     IF constraintViolated(elem, n)
8      shift(elem);
9     END IF
10   END IF
11  END FOREACH
12 END
```

Algorithm 1: Timestamp-based propagation kernel launched over the elements in the model.

This new propagation mechanism allows to readjust previously deformed elements since the fastest path, attending to the propagation speed instead of the number of links, now enforces its propagation to those elements even when they are reached in later iterations. Consider for instance the "heterogeneous ring" example depicted in Fig. 5. A deformation is applied to the top-left element, initiating the propagation through the model to satisfy the violated constraints, indicated by the red links. Through the iterative process, the elements on the right are first reached through the elastic path. When the two sides of the wavefront reach the bottom-right element, (iteration 3) the behavior of the SP-ChainMail becomes unpredictable. Even in this simple case, two outcomes are possible depending on the attended neighbor. If the element attends to the constraints of its top neighbor, (bottom branch) the expected final configuration is achieved. If the element attends however to the constraints of its left neighbor (top branch), the propagation will continue through the left path, reaching the initially deformed element, leading to an undesired final configuration. As the model becomes more complex, the probability of

reaching the correct configuration is drastically decreased. Our novel approach, however, takes into account the propagation speed through the different materials and hence always reaches the expected configuration.

Fig. 6 shows a more complex model deformed by our approach considering a homogeneous material and a more realistic heterogeneous distribution of materials, showing the differences of the timestamps propagated through it and the final deformed configuration in each case. The heterogeneous distribution of materials is correctly handled due to the timestamp system, which tracks the faster propagation of the deformation through stiffer tissue and prevents the bone tissue to deform unrealistically.

If one or more of the materials used in the model apply different constraints for each dimension (different values for the Dx, Dy and Dz parameters), then the links store the different times required to travel for each dimension and the timestamp system independently tracks and handles the propagation for each dimension.

The algorithm as presented, requires resetting the timestamp of all elements whenever a new propagation starts. In practice, we use a different approach that avoids the need for actively resetting the timestamps and also allows to concurrently propagate the wavefronts of multiple applied deformations as we explain in Sec. 3.5.

3.3. Heterogeneous Relaxation

The relaxation stage of the SP-ChainMail algorithm has also been modified in order to cope with the heterogeneous material requirements and the parallel approach.

An energy minimization process is carried out during a relaxation iteration. We introduce the stiffness of the materials in the computation of the energy stored in the model to take the heterogeneity into account. The stiffness of a material is measured in terms of the constraints assigned to it. Thus for an element, each of its neighbors defines an optimal position, i.e., a position that would reduce the energy stored in the link between them to zero. Then, we define the energy based on the Hooke's law as follows.

Let $\boldsymbol{e}$ be an element connected to m neighbors, let $\boldsymbol{\rho}_e$ be its current position, let $\boldsymbol{c}_i$ be the constraint value of the link between $\boldsymbol{e}$ and its i^{th} neighbor, and let $\boldsymbol{p}_i$ be the optimal position for $\boldsymbol{e}$ defined by its i^{th} neighbor. Then, the potential energy of $\boldsymbol{e}$ is proportional to

$$U_e = \sum_{i=1}^{m} (\|\rho_e - p_i\|^2 \, w_i), \tag{7}$$

with $w_i = \frac{1}{c_i}$. Solving $\frac{\partial U_e}{\partial x} = 0, \frac{\partial U_e}{\partial y} = 0, \frac{\partial U_e}{\partial z} = 0$, we find that the new position $\boldsymbol{\rho}_e^*$ that minimizes the energy is given by a simple weighted mean of the optimal positions, computed as

$$\rho_e^* = \frac{\sum_{i=1}^{m} (p_i w_i)}{\sum_{i=1}^{m} w_i}. \tag{8}$$

In practice, we use $w_i = \frac{1}{c_i + \varepsilon}$ with ε a very small positive value, since $\boldsymbol{c}_i = 0$ for completely rigid materials. Using this weighting function, the materials experiment a hyperbolic weight gain as they become stiffer, reaching a virtually infinite weight for a rigid material. The weighted mean prevents the nodes belonging to the boundary between structures with different stiffness to deform unrealistically.

Note that we have modified the definition of the potential energy of the elements, but the minimization process, which is in turn a closed negative feedback system [9], remains the same as in the original formulation. The system energy, defined by $E = \sum_{i=1}^{n} U_i$, with n the number of elements in the model, is decreased in each iteration until a stable configuration is reached, thus the process is free of instability or divergence issues.

As explained by Gibson [9] the definition of the optimal position of an element regarding a neighbor permits to model different material properties, ranging from purely plastic to purely elastic behavior depending on whether a single position or a 3D region is considered as optimal. Moreover, different linear or non-linear functions can be used to measure internal stress for a given link, resulting in different stress-strain response. Hysteresis during loading and unloading can also be reproduced by using different functions for measuring internal stress during stretching and compression.

This stage is also easily implemented as a GPU kernel launched over the elements in every relaxation iteration.

3.4. Topology modification

As in the original ChainMail algorithm, topology changes such as cutting and carving can be introduced by removing links or elements respectively. The propagation and relaxation stages proposed in this work implicitly cope with interactive topology changes, since their impact is automatically detected by both the propagation and the relaxation iterative processes.

These topology changes, either introduced directly on the model or detected during the interaction with virtual tools, are easily computed through a parallel process over the model:

- When a cut is detected, one thread for each link in the model is launched. Each thread computes an intersection test between the corresponding link and the path defined by the cut. If the test is positive, the link is removed from the model.
- When an operation of carving is detected, one thread per element is launched. Each thread computes an inclusion test of the corresponding element with the carving volume. If the test is positive, the element is removed from the model.

Once the topology change has been introduced, new deformations applied to the model are correctly propagated even in the presence of heterogeneous materials. The timestamp system automatically takes into account the removed elements and links when the deformation is being processed. Take for instance the case shown in Fig. 7: On the left, the model is deformed by pulling from the skin tissue at the bottom, propagating the deformation to the internal structures. On the right, the same deformation is applied after performing a cut on the bottom part of the model. The propagation of the deformation takes into account the modified topology and the propagation has to travel around the cut (as shown by the propagation time heatmap). Thus, the deformation is absorbed by the top part of the model and the bottom right part remains undeformed as it is now reached later.

3.5. Concurrent Wavefronts

Since the original Chainmail algorithm was proposed, the application of several simultaneous nodal forces has been hard to achieve because it was not explicitly handled. For this reason, the interaction was commonly limited to a single nodal deformation. Our timestamp-based propagation algorithm as exposed in Sec. 3.2.1 requires to clear the residual timestamp marks from a previous propagation, considering they may interfere with a new wavefront generated by a new applied deformation. Instead of performing an explicit reset of the timestamps, we apply a different strategy that also allows to concurrently propagate several wavefront, i.e., allow for several deformations to propagate through the model at the same time.

We add a *wavefront counter*, a new property stored in each element. During the initialization process, a global counter and all wavefront counters are set to 0. When a new deformation is

applied to the model, the global counter is increased by 1, the wavefront counter of the element receiving the deformation is set to the current value of the global counter, and its timestamp value is set to 0. Using this wavefront counter, the propagation kernel is modified as shown in Algorithm 2 since three new possible cases arise:

1. If the wavefront counter of the neighbor is greater than the element's counter, the timestamp of the element is residual and is consequently ignored. The wavefront counter of the element is updated to the same value of the wavefront counter of the neighbor and the timestamp of the element is set as the sum of the timestamp of the neighbor and the time value of the link between them (lines 5-6). The constraint imposed by that neighbor is checked and the element is shifted if necessary (lines 7-9).
2. If the wavefront counter of the neighbor equals the element's counter, both elements have been reached by the wavefront and the normal algorithm (from Algorithm 1) is executed (lines 10-18).
3. If the wavefront counter of the neighbor is smaller than the element's counter, it is ignored since it has not yet been reached by the current wavefront.

```
1  BEGIN propagationKernel(elem)
2   N = getNeighbors(elem);
3   FOREACH n IN N
4    IF (n.wCounter > elem.wCounter)
5     elem.iCounter = n.wCounter;
6     elem.tStamp = n.tStamp + pTime(elem, n);
7     IF constraintViolated(elem, n)
8      shift(elem);
9     END IF
10   ELSE IF (n.wCounter == elem.wCounter)
11    newTS = n.tStamp + pTime(elem, n);
12    IF (newTS < elem.tStamp)
13     elem.tStamp = newTS;
14     IF constraintViolated(elem, n)
15      shift(elem);
16     END IF
17    END IF
18   END IF
19  END FOREACH
20 END
```

Algorithm 2: Complete propagation kernel allowing for concurrent wavefronts propagating through the model.

This complete propagation approach is particularly interesting for some common situations, such as several deformations applied consecutively on the same spot or different nodal deformations applied on different parts of the model that do not interfere, e.g., simultaneously separating both sides of an incision as shown in Fig. 8. In those cases, a single parallel propagation iteration advances all the concurrent wavefronts at once, thereby harvesting more parallel power from the GPU, since a higher amount of the launched threads perform useful computation. This strategy does not really handle the interaction between different wavefronts, since simply the latest will prevail, however, it is a step further into the handling of simultaneous deformations using the ChainMail approach.

Also, as already stablished for the SP-ChainMail algorithm, an additional flag, referred to as *activity flag*, indicating whether an element was updated in the previous propagation iteration, allows the relaxation process to identify elements ready to start the relaxation process. If an element is not active and its wavefront counter is equal to the wavefront counters of its neighbors,

applying the relaxation process to that element does not interfere with the propagation process.

The combination of these two mechanisms allows to apply new deformations to the model without having to wait for the previous deformations to fully propagate, and also enables the interleaving of propagation iterations with relaxation iterations, naturally blending both stages. This feature is of high interest for applications demanding a high feedback frequency, since it is possible to render partial results of the deformations on the model.

3.6. Improved sparse blocks computation

The SP-ChainMail algorithm [8] includes a blocking method applied to the computation of the iterative stages to avoid unnecessary computation for elements not yet reached by the propagation or already in a stable configuration. The computational domain is partitioned into blocks, so a 3D offset is necessarily calculated to reference each block. The active blocks, i.e., the blocks still requiring computation, are efficiently flagged in each iteration and the computation of the next iteration is only performed over these active blocks by making one GPU kernel call per block.

By choosing a smaller size for blocks in the partitioning, the amount of launched threads is reduced since the active elements are more effectively enclosed. However, the experiments show that after a given threshold, reducing the block size actually decreases the obtained gain, and can even lead to a worse performance than the non-partitioned case. As discussed in [8], this is produced because the GPU gets gradually misused as more kernel invocations with fewer concurrent threads are performed.

We also apply this blocking scheme to reduce the computational load of our algorithm, but in order to avoid the limitation mentioned above, we have introduced a small, yet significant modification to the launching approach, similar to the proposal of Sætra [29]. After the current iteration has finished and the boolean maps are updated, an array storing the 3D offsets corresponding to each active block is updated and sent to the GPU. Using this array, a single kernel launch computes all the blocks at once: the number of launched threads is simply calculated as the number of active blocks multiplied by the number of elements per block, and each thread uses its own *id* to deduce the corresponding block and read the corresponding 3D offset from the array to access the correct element.

As we demonstrate in our experiments, this approach is better suited for the GPU computing paradigm and small block sizes can be used without resulting in a performance penalty due to a

misuse of GPU resources.

4. Interactive deformation of medical models

The proposed algorithm has been applied to deform large models achieving interactive framerates. All the data structures required by the algorithm are generated from a source volume dataset.

In order to maintain the complexity of the source volume dataset, one ChainMail element is created per each voxel. The initial position of each element is calculated attending to the space existing between voxels in each dimension.

Attending to the density values or any defined segmentation process, elements corresponding to empty or undesired areas of the source dataset can be discarded.

We assign material properties to the ChainMail elements based on the density of the corresponding voxels in the source dataset in order to define the heterogeneous constraints for the links in the model. This is done by specifying a *material transfer function* as we show in our examples. We define the constraints as a percentage of the initial spacing between voxels in the dataset. Under this assumption, we can rapidly generate high resolution heterogeneous models from the source dataset. The relation between CT data and mechanical properties is known to be accurate for bone tissue [30, 31], however, the estimation of the properties of soft tissue from CT data is still an open problem. Thus, a more accurate estimation of these material properties could be obtained through a more complex segmentation process based on known material properties of the identified tissues, or other measuring techniques such as elastography [32].

In order to visualize the deformations interactively, we have coupled the proposed algorithm with the resampling method described in [33]. When a new deformation has been applied, the model is resampled into a voxel grid, which is then visualized using a standard ray-casting algorithm. Both the resampling and the ray-casting algorithms are also implemented in parallel using the GPU.

For direct interaction, we have implemented a simple isosurface mapping mechanism, relating the 3D position of the selected isosurface with the nearest ChainMail element, although more immersive interaction mechanisms such as virtual tools can be also used.

5. Results and discussion

We have tested our approach in two interactive virtual examples: a sheep heart exploration procedure and a wrist surgery simulation. We refer the reader to the video provided as additional material (Online Resource 1) for further results of the proposed examples.

We have also conducted a test to evaluate the performance of our heterogeneous parallel ChainMail algorithm and the improved blocking method, also comparing it with the SP-ChainMail algorithm.

The proposed algorithm was implemented with OpenCL 1.2. The performed experiments were run on an Intel Core i7-3770 machine, with 16 GB RAM and an Nvidia GeForce GTX 670 with 1344 cores, using a 800x700 viewport for the interaction and visualization.

5.1. Sheep heart exploration

For this setup, the inside of a sheep heart is explored. The source dataset is an MRI scan with 12.4 million voxels. The heart is accessed through the aorta, interactively exploring and probing its chambers. The model consists of 12.4 million ChainMail elements, one per voxel, although the air voxels are discarded.

The materials are defined through a material transfer function as shown in Fig. 9 (bottom), assigning different elastic materials based on the density values of the source dataset. We perform 10 propagation iterations and 10 relaxation iterations before a new frame is rendered.

The high resolution of our ChainMail model allows for the deformations to adapt to the complex topology in the inside of the heart as can be seen in Fig. 9 (top), when applying a deformation to an internal feature.

With this configuration, the example runs at 12-26 fps. Each frame, the deformation takes between 11 and 53 ms depending on the number of affected elements, the resampling of the deformed volume takes an average of 10 ms and the rendering using raycasting takes an average of 17 ms.

We have tested the same setup using a dynamic simulation based on the mass-spring model, using a tetrahedral mesh and explicit integration, parallelized on the GPU as proposed by Georgii et al. [34]. For a fair comparison, we set a mass-spring mesh leading to a running time similar to the one achieved by our method. Thus we set a mesh with 42,875 mass nodes, connected by 196,520 tetrahedra.

Table 1 shows the average simulation time and the average number of shifted nodes per simulation step for both methods when applying a small and a large deformation, affecting the 5% and 30% of the model respectively.

The computation time of our algorithm varies depending on the amount of shifted nodes. This is due to the local nature of the method, which performs computation only on the affected region, while the mass-spring approach displays a stable performance independently of the affected region. More importantly, in contrast to our method, the mass-spring mesh is unable to capture the fine topological features present in the dataset because of the disparity of resolution, which leads to an average of 290 data voxels modeled by each mass node. To illustrate this issue, Fig. 10 shows the same applied deformation using both methods. Using our method, shown on the top row, the right side of the tissue "ring" can be deformed while the left part remains undeformed. Using the mass-spring model, shown on the bottom row, the hole in the tissue is not correctly modeled and the deformation applied on the right part is transmitted to the tissue wall on the left.

5.2. Wrist surgery

In this setup, we explore a human wrist by performing a cut on the skin and applying deformations to reveal the internal structures. The source dataset is a CT scan of a human hand with 7.1 million voxels. Again, we generate one ChainMail element per voxel.

The materials are also defined through a material transfer function as shown in Fig. 11 (bottom). The ChainMail elements corresponding to bone tissue are configured as rigid. The rest of the tissues are modeled as elastoplastic materials with an elastic recovery corresponding to the 80% of the undergone strain. We perform 20 propagation iterations and 40 relaxation iterations between frames.

As can be seen in Fig. 11, our model correctly handles the interactive topology changes, and the different internal tissues deform along when the skin around the cut is deformed. The heterogeneous materials of the different tissues prevent bones from deforming during the relaxation stage.

The example runs at 6-21 fps. The deformation takes between 25 and 84 ms depending on the number of affected elements, the resampling takes an average of 7 ms and the rendering takes between 14 and 61 ms depending on the chosen transfer function and the camera position.

5.3. Performance analysis

To evaluate the performance of our approach, we have conducted a test comparing it with the SP-ChainMail algorithm. for a fair comparison, we have used a model with 2 million elements ($126x126x126$) and a homogeneous material so that both algorithms perform the same number of propagation and relaxation iterations.

A deformation affecting all the elements has been applied, and we have measured the time required for the deformation to fully propagate and relax to an stable configuration, without intermediate visualizations, requiring 376 propagation iterations and 619 relaxation iterations.

We have performed this measurement for both algorithms. We have used both algorithms with no blocking scheme and with several block size configurations. The results of the test are summarized in Fig. 12.

As the results show, the use of the timestamp-based heterogeneous propagation barely affects the performance since the SP-ChainMail completes the deformation after 3321 ms when the blocking scheme is not used, and our approach takes 3498 ms also with no blocking scheme, roughly increasing the computation time by 5.32%.

Attending to the blocking scheme, the SP-ChainMail achieves the best performance for a block size of $16x16x16$, completing the deformation after 1312 ms, achieving a speedup of 2.53x. When smaller block sizes are used, the achieved speedup is reduced, even leading to a worse performance than the non-partitioned case. As already mentioned, this behavior is expected because the GPU becomes misused for small block sizes since it leads to multiple small kernel invocations.

In our approach, these invocations are combined into a single invocation, leading to higher speedups and achieving the best performance for the smaller block size of $16x4x4$, completing the deformation after 458 ms, inducing a speedup of 7.63x.

Therefore, our approach not only allows for heterogeneous models, but also improves the performance with respect to the SP-ChainMail thanks to the enhanced blocking scheme.

5.4. Memory requirements

Our approach heavily relies on GPU dedicated memory, since all the main structures are stored and computed directly on the GPU. The memory requirements grow linearly with the number of ChainMail elements in the model. Each element requires 58 bytes of dedicated memory, including position, constraint values, timestamp mark, wavefront counter, activity flag and

neighbor flags, and some structures are moreover duplicated to accommodate the Jacobi sweep scheme [35]. All in all, 55.31 MB are required per each million elements.

The blocking scheme consumes a negligible amount of memory w.r.t. the main structures since only 12 bytes of GPU dedicated memory and 64 bytes of host memory are required per block. In our most demanding scenario, the sheep heart model using $16x4x4$ blocks, 50,460 blocks were created, leading to a memory consumption of 592 KB of GPU memory and 3.08 MB of host memory.

6. Conclusions and Future Work

In this paper, we have presented an enhanced parallel ChainMail algorithm supporting heterogeneous materials and interactive topology changes. We solve the issue of heterogeneity in the model by a novel timestamp-based propagation mechanism and a modified relaxation scheme. We have also improved the blocking scheme in order to increase the performance of our algorithm by a notable factor, efficiently handling the sparse nature of the GPU computation of our approach.

Our results demonstrate that the proposed algorithm, in conjunction with a direct assignment of material properties and a parallel resampling approach, allows to interactively visualize and deform models consisting of up to several million elements.

In addition to these contributions, our algorithm is able to deal with different deformations simultaneously for the same model. This is an important step further for ChainMail-based algorithms, which have been limited to single nodal interactions since the original algorithm was proposed.

Finally, although our algorithm improves the SP-ChainMail approach both in performance and material flexibility, providing a more general and applicable solution, there are certain limitations that we would like to address as future work.

An important limitation is found in the material characterization process. Although the ChainMail approach allows to model complex tissue behaviors, the mapping of material properties measured from real world specimens is not trivial, and our direct assignment of materials and model topology is oversimplified in order to achieve a minimum preprocessing effort as already mentioned. Moreover, we have compared our approach with the mass-spring method, however, we have focused on performance and resolution issues. As future work, we would

like to perform a comparison with other methods regarding behavior and model accuracy. Thus, assisted methods for ChainMail material characterization should be explored, which combined with the acquisition of real specimen measurements for validation would allow to perform such a comparative study.

Although we allow multiple deformations to simultaneously propagate through the model, when they overlap their interference is resolved simply prioritizing one of them. Therefore, a better management of their interference should be explored to address those cases. Lastly, the visualization of the interactive cuts with the used resampling algorithm leads to material loss, thus a better visualization of topology changes would improve the realism and feedback during the simulations.

Acknowledgements

This work is supported by the University of Granada, under the "Formación de Profesorado Universitario, Plan Propio de Investigación 2012" program. This work is also supported by the project TIN2014-60956-R of the Spanish Ministry of Economy and Competitiveness with FEDER funds. The funding entities had no involvement in any stage of the development of this work or in the decision to submit this manuscript. The heart and the hand models were obtained from The Volume Library (lgdv.cs.fau.de/External/vollib/) and the ImageVis3D (www.sci.utah.edu/software/imagevis3d) software package respectively. The authors would like to thank the anonymous reviewers for their insightful comments.

Conflict of interest statement

None declared.

References

[1] Rahimi-Gorji, M., Pourmehran, O., Gorji-Bandpy, M., Gorji, T.. Cfd simulation of airflow behavior and particle transport and deposition in different breathing conditions through the realistic model of human airways. Journal of Molecular Liquids 2015;209:121–133. doi:10.1016/j.molliq.2015.05.031.

[2] Rahimi-Gorji, M., Gorji, T.B., Gorji-Bandpy, M.. Details of regional particle deposition and airflow structures in a realistic model of human tracheobronchial airways: two-phase flow simulation. Computers in biology and medicine 2016;74:1–17. doi:10.1016/j.compbiomed.2016.04.017.

[3] Liu, A., Tendick, F., Cleary, K., Kaufmann, C.. A survey of surgical simulation: applications, technology, and education. Presence: Teleoperators and Virtual Environments 2003;12(6):599–614. doi:10.1162/105474603322955905.

[4] 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 2005;77(3):183–197. doi:10.1016/j.cmpb.2004.11.002.

[5] Moore, P., Molloy, D.. A survey of computer-based deformable models. In: Machine Vision and Image Processing Conference, 2007. IMVIP 2007. International. 2007, p. 55–66. doi:10.1109/IMVIP.2007.31.

[6] Kerdok, A.E., Cotin, S.M., Ottensmeyer, M.P., Galea, A.M., Howe, R.D., Dawson, S.L.. Truth cube: Establishing physical standards for soft tissue simulation. Medical Image Analysis 2003;7(3):283–291. doi:10.1016/S1361-8415(03)00008-2.

[7] Gibson, S.F.. 3d chainmail: a fast algorithm for deforming volumetric objects. In: Proceedings of the 1997 symposium on Interactive 3D graphics. ACM; 1997, p. 149–154. doi:10.1145/253284.253324.

[8] Rodríguez, A., León, A., Arroyo, G., Mantas, J.M.. Sp-chainmail: a gpu-based sparse parallel chainmail algorithm for deforming medical volumes. The Journal of Supercomputing 2015;71(9):3482–3499. doi:10.1007/s11227-015-1445-5.

[9] Frisken-Gibson, S.F.. Using linked volumes to model object collisions, deformation, cutting, carving, and joining. Visualization and Computer Graphics, IEEE Transactions on 1999;5(4):333–348. doi:10.1109/2945.817350.

[10] Gibson, S., Samosky, J., Mor, A., Fyock, C., Grimson, E., Kanade, T., et al. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback. In: CVRMed-MRCAS'97. Springer; 1997, p. 367–378. doi:10.1007/BFb0029258.

[11] Le Fol, T., Acosta-Tamayo, O., Lucas, A., Haigron, P.. Angioplasty simulation using ChainMail method. In: Medical Imaging 2007: Visualization and Image-Guided Procedures. 2007,doi:10.1117/12.709582.

[12] Villard, P.F., Boshier, P., Bello, F., Gould, D.. Virtual reality simulation of liver biopsy with a respiratory component. InTech; 2011.

[13] Villard, P.F., Vidal, F.P., ap Cenydd, L., Holbrey, R., Pisharody, S., Johnson, S., et al. Interventional radiology virtual simulator for liver biopsy. International Journal of Computer Assisted Radiology and Surgery 2013;9(2):255–267. doi:10.1007/s11548-013-0929-0.

[14] Castro-Pareja, C.R., Daly, B., Shekhar, R.. Elastic registration using 3d chainmail: application to virtual colonoscopy. In: Medical Imaging 2006: Image Processing; vol. 6144. 2006, p. 947–955. doi:10.1117/12.653644.

[15] Shekhar, R., Lei, P., Castro-Pareja, C.R., Plishker, W.L., DSouza, W.D.. Automatic segmentation of phase-correlated ct scans through nonrigid image registration using geometrically regularized free-form deformation. Medical physics 2007;34(7):3054–3066. doi:10.1118/1.2740467.

[16] Mensmann, J., Ropinski, T., Hinrichs, K.. Interactive cutting operations for generating anatomical illustrations from volumetric data sets. Journal of WSCG 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2008;16(1-3):89–96.

[17] Park, J., Kim, S.Y., Son, S.W., Kwon, D.S.. Shape retaining chain linked model for real-time volume haptic rendering. In: Volume Visualization and Graphics, 2002. Proceedings. IEEE / ACM SIGGRAPH Symposium on. 2002, p. 65–72. doi:10.1109/SWG.2002.1226511.

[18] Park, J., Kim, S.Y., Kwon, D.S.. Mechanical representation of shape-retaining chain linked model for real-time haptic rendering. In: Medical Simulation. Springer; 2004, p. 144–152. doi:10.1007/978-3-540-25968-8_16.

[19] Sang-Youn, K., Jinah, P., Dong-Soo, K.. The real-time haptic simulation of a biomedical volumetric object with shape-retaining chain linked model. IEICE transactions on information and systems 2005;88(5):1012–1020. doi:10.1093/ietisy/e88-d.5.1012.

[20] Li, Y., Brodlie, K.. Soft object modelling with generalised chainmail extending the boundaries of web-based graphics. Computer Graphics Forum 2003;22(4):717–727. doi:10.1111/j.1467-8659.2003.00719.x.

[21] Villard, P.F., Jacob, M., Gould, D., Bello, F.. Haptic simulation of the liver with respiratory motion. In: Proceeding of Medicine Meets Virtual Reality 17 (MMVR17); vol. 142. 2009, p. 401–406.

[22] Vidal, F.P., Villard, P.F., Lutton, E.. Tuning of patient-specific deformable models using an adaptive evolutionary optimization strategy. Biomedical Engineering, IEEE Transactions on 2012;59(10):2942–2949. doi:10.1109/TBME.2012.2213251.

[23] Vidal, F.P., Villard, P.F.. Development and validation of real-time simulation of x-ray imaging with respiratory motion. Computerized Medical Imaging and Graphics 2016;49:1–15. doi:10.1016/j.compmedimag.2015.12.002.

[24] Fortmeier, D., Mastmeyer, A., Handels, H.. Image-based palpation simulation with soft tissue deformations using chainmail on the GPU. In: Bildverarbeitung für die Medizin 2013. Springer; 2013, p. 140–145. doi:10.1007/978-3-642-36480-8_26.

[25] Fortmeier, D., Mastmeyer, A., Handels, H.. An image-based multiproxy palpation algorithm for patient-specific VR-simulation. Medicine Meets Virtual Reality 2014;:107–113doi:10.3233/978-1-61499-375-9-107.

[26] 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. Springer; 1998, p. 679–687. doi:10.1007/BFb0056254.

[27] Schill, M.A., Wagner, C., Hennen, M., Bender, H.J., Männer, R.. EyeSi – A Simulator for Intra-ocular Surgery; chap. Medical Image Computing and Computer-Assisted Intervention – MICCAI'99: Second International Conference, Cambridge, UK, September 19-22, 1999. Proceedings. Springer Berlin Heidelberg; 1999, p. 1166–1174. doi:10.1007/10704282_126.

[28] Schulze, F., Bühler, K., Hadwiger, M.. Interactive deformation and visualization of large volume datasets. In: GRAPP (AS/IE). Citeseer; 2007, p. 39–46.

[29] Sætra, M.L.. Shallow water simulation on gpus for sparse domains. In: Numerical Mathematics and Advanced Applications 2011. Springer; 2013, p. 673–680. doi:10.1007/978-3-642-33134-3_71.

[30] Helgason, B., Perilli, E., Schileo, E., Taddei, F., Brynjólfsson, S., Viceconti, M.. Mathematical relationships between bone density and mechanical properties: a literature review. Clinical biomechanics 2008;23(2):135–146. doi:10.1016/j.clinbiomech.2007.08.024.

[31] Taddei, F., Pancanti, A., Viceconti, M.. An improved method for the automatic mapping of computed tomography numbers onto finite element models. Medical engineering & physics 2004;26(1):61–69. doi:10.1016/S1350-4533(03)00138-3.

[32] Ophir, J., Alam, S.K., Garra, B.S., Kallel, F., Konofagou, E.E., Krouskop, T., et al. Elastography: imaging the elastic properties of soft tissues with ultrasound. Journal of Medical Ultrasonics 2002;29(4):155–171. doi:10.1007/BF02480847.

[33] Rodríguez, A., Salas, A.L., Perandrés, D.M., Otaduy, M.A.. A parallel resampling method for interactive deformation of volumetric models. Computers & Graphics 2015;53:147–155. doi:10.1016/j.cag.2015.10.002.

[34] Georgii, J., Westermann, R.. Mass-spring systems on the gpu. Simulation modelling practice and theory 2005;13(8):693–702. doi:10.1016/j.simpat.2005.08.004.

[35] Nguyen, A., Satish, N., Chhugani, J., Kim, C., Dubey, P.. 3.5-d blocking optimization for stencil computations on modern cpus and gpus. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society; 2010, p. 1–13. doi:10.1109/SC.2010.2.

Table 1: Measured average time and shifted nodes per simulation step during a small and a large deformation of the model using our approach and the mass-spring model. Total simulation nodes for each method are also shown.

		5% deformation		30% deformation	
	Nodes	Shifted nodes	Time (ms)	Shifted nodes	Time (ms)
Our approach	12,487,168	123,428	11	1,423,685	53
Mass-Spring	42,875	1,923	45	12,652	45

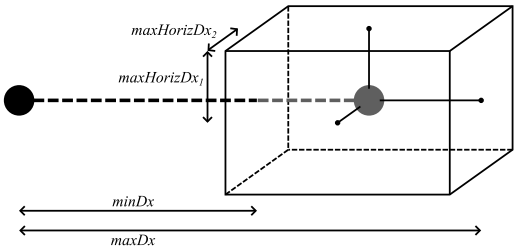

Figure 1: The valid region (box) that an element (black) imposes to a right neighbor (gray) with the original ChainMail algorithm is defined by the constraint parameters (*maxDx*, *minDx*, $maxHorizDx_1$ and $maxHorizDx_2$).

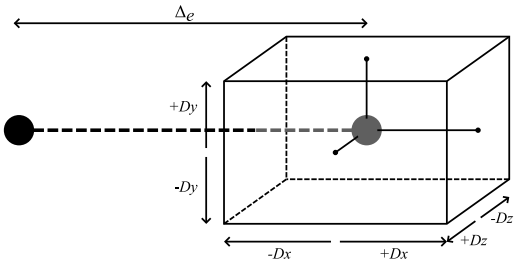

Figure 2: The valid region (box) that an element (black) imposes to a neighbor (gray), is defined by the constraint parameters (*Dx*, *Dy* and *Dz*) and by the initial separation of both elements Δ_e.

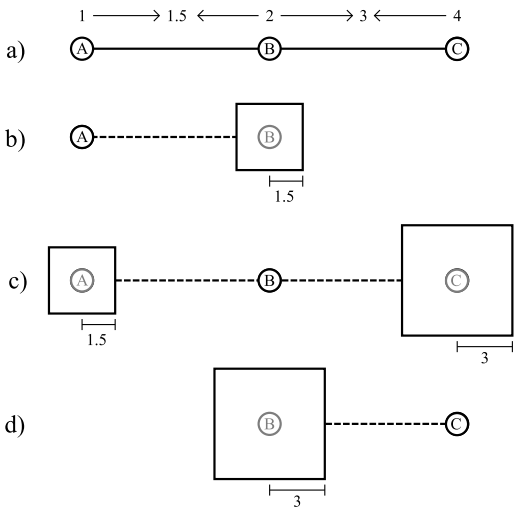

Figure 3: 2D example of heterogeneous constraints in a simple, heterogeneous model: the constraint values of each two linked elements, depicted on top of the elements, are averaged and stored in the link between them (*a*), and the constraints applied by each element to its neighbors are computed attending to these averaged values, leading to the 2D valid regions represented by the squares (*b*, *c* and *d*).

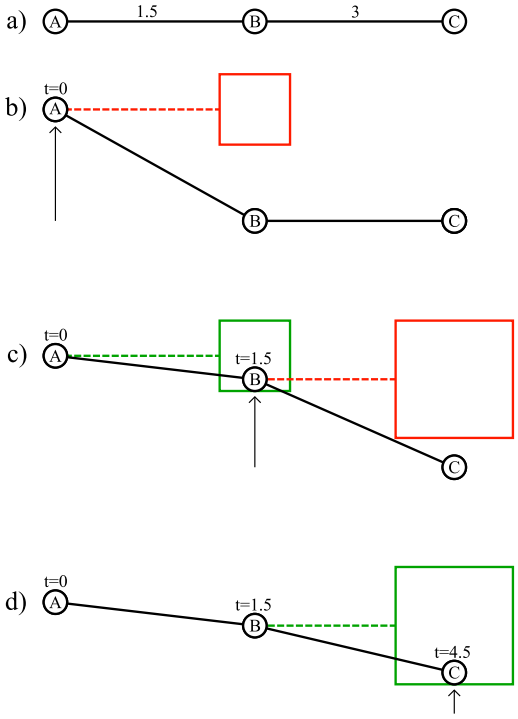

Figure 4: 2D example of the timestamp propagation system with heterogeneous materials (*a*). When a new deformation is applied to an element, the timestamp of the element is set to 0 (*b*), and the propagation starts. Again, the squares represent the valid regions. When a new element is reached through a link (violated constraints in red, satisfied constraints in green), its timestamp mark is updated as the sum of the timestamp of the linked neighbor and the propagation time stored in the link between them (*c* and *d*).

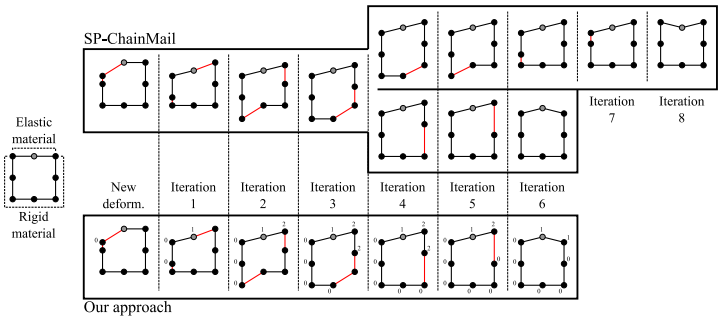

Figure 5: An "heterogeneous ring" model, composed by an elastic material at the top and a rigid material on the lateral and bottom parts, is deformed using both the SP-ChainMail algorithm and our algorithm. Even in this simple case, the SP-ChainMail algorithm has an unpredicted behavior, leading to two possible outcomes, which in turn may result in an undesired configuration. Our approach (below) correctly handles the heterogeneous materials due to the timestamp system.

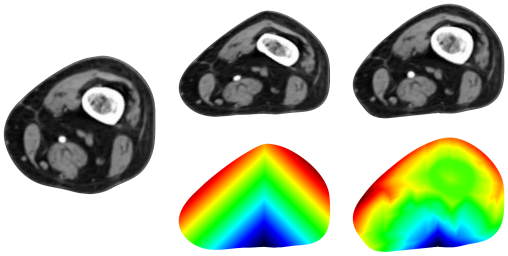

Figure 6: A 2D section of a leg (left) is compressed and deformed using a homogeneous material (middle) and a more realistic distribution of heterogeneous materials (right). The propagation time is shown below each case using a heatmap, ranging from dark blue at the source of the deformation, to a dark red at the last reached elements.

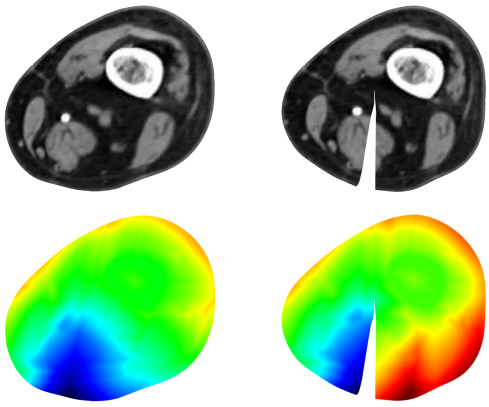

Figure 7: The same deformation is applied to a 2D section of a leg before and after applying a cut. The propagation time is shown below each case using a heatmap, ranging from dark blue at the source of the deformation, to a dark red at the last reached elements, showing that the topology modification is automatically taken into account by the timestamp system.

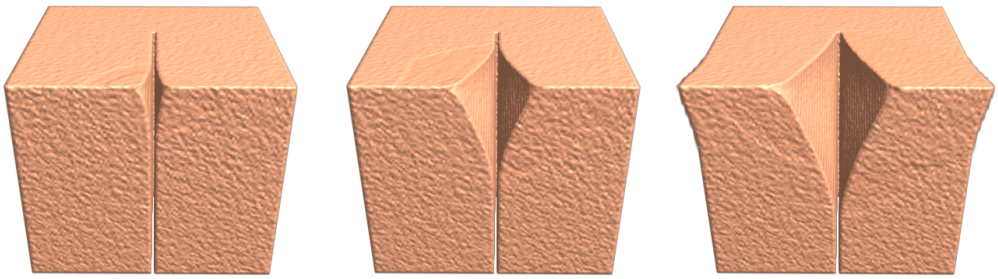

Figure 8: Two simultaneous deformations are applied to both sides of an incision in a tissue sample. Our propagation approach is able to handle and compute both deformations simultaneously.

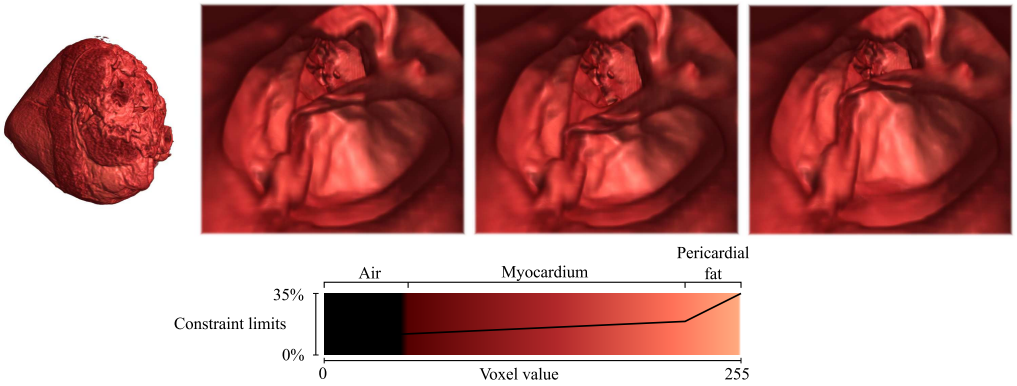

Figure 9: The inside of a sheep heart model with 12.4 million elements (top-left) is explored and several deformations are applied to an internal feature (top-right), which is correctly captured by the high resolution of our ChainMail model. The constraints of the materials assigned to the sheep heart model, ranging from a 15% to a 35% variation from the initial spacing, are specified using a material transfer function (bottom).

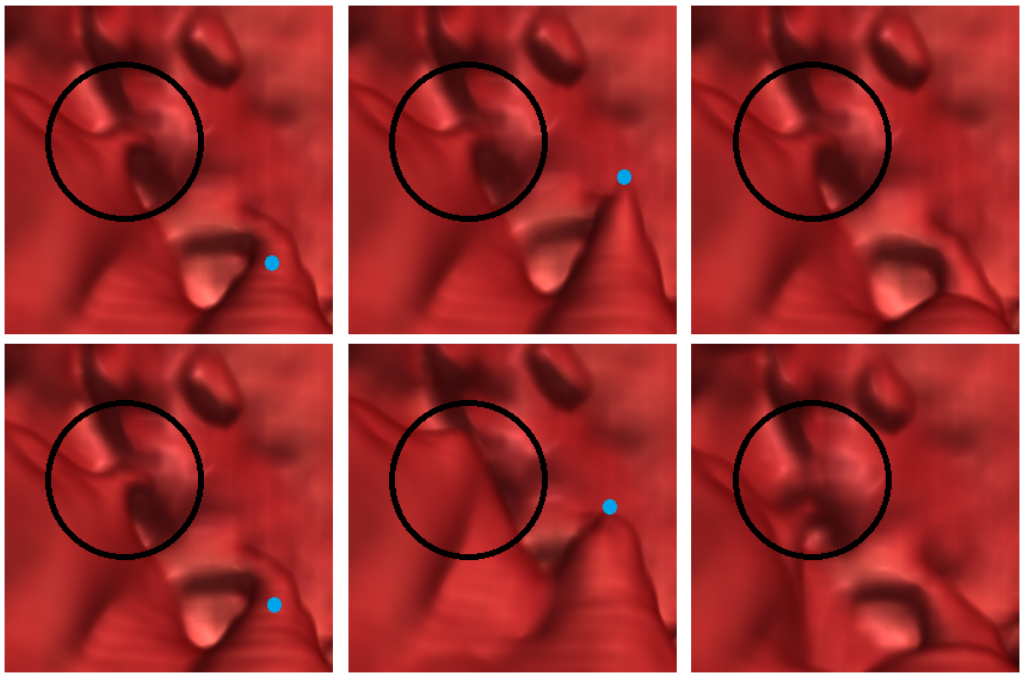

Figure 10: The same deformation is applied to a fine feature using our ChainMail approach (top row) and a mass-spring model (bottom row). The topology of the feature is incorrectly captured by the mass-spring model, leading to an incorrect transmission of the deformation from the right side of the tissue ring where the force is applied (blue dot) to the left wall (highlighted by the black circle). In contrast, the high resolution of our method correctly captures the topology and avoids that transmission.

Figure 11: A hand model with 7.1 million elements (left) is deformed by pulling from the skin on both sides of an applied incision (right). The skin pulling propagates the deformation to the internal structures. The constraints of the materials assigned to the hand model, ranging from a 0% to a 50% variation from the initial spacing, are defined through a material transfer function (bottom).

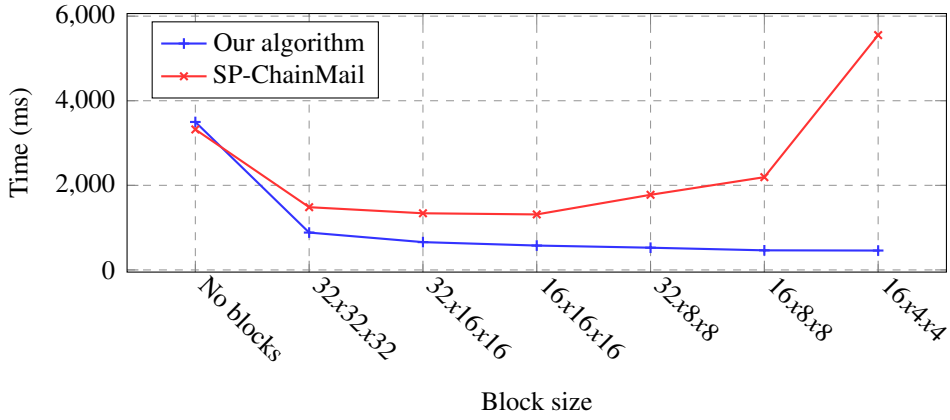

Figure 12: Performance comparison of our method and the SP-ChainMail algorithm using different block sizes for the blocking method. The SP-ChainMail algorithm fails to optimize the usage of small block sizes, leading to a misuse of the GPU resources. Our improved blocking scheme is able to efficiently manage the smaller block sizes, further improving the overall performance.

Chapter 3

Simulation-based control of soft robots

3.1 Real-time simulation of hydraulic components for interactive control of soft robots

- **A. Rodríguez**, E. Coevoet and C. Duriez. "Real-time simulation of hydraulic components for interactive control of soft robots". *IEEE International Conference on Robotics and Automation* (ICRA 2017).
  - Status: Published [108]
  - CORE Conference Ranking 2014: CORE B
  - H index (SJR 2015): 105.

Real-time simulation of hydraulic components for interactive control of soft robots

A. Rodríguez[1] and E. Coevoet[2] and C. Duriez[2]

***Abstract*— In this work we propose a new method for online motion planning in the task-space for hydraulic actuated soft robots. Our solution relies on the interactive resolution of an inverse kinematics problem, that takes into account the properties (mass, stiffness) of the deformable material used to build the robot. An accurate modeling of the mechanical behavior of hydraulic components is based on a novel GPU parallel method for the real-time computation of fluid weight distribution. The efficiency of the method is further increased by a novel GPU parallel leveraging mechanism. Our complete solution has been integrated within the open-source SOFA framework. In our results, we validate our simulation with a fabricated silicone cylinder and we demonstrate the usage of our approach for direct control of hydraulic soft robots.**

I. Introduction

The salient feature of soft robots is their compliant nature. As consequence of their compliance, soft robots exhibit a large number of passive degrees of freedom, leading to a higher operational flexibility even for underactuated configurations, and an intrinsic robustness to uncertainty. Due to these features, soft robots provide a real alternative to rigid robots for different environments and tasks, mainly involving safety conditions, or certain levels of uncertainty [1], [2].

In order to benefit from these features, advanced control systems must be developed. Unlike their rigid counterparts, the configurations and motions of soft robots are ruled by the deformable mechanics of their underlying soft structure and the properties of their base materials, thus a key component in the control pipeline is a precise yet interactive modeling of this compliant behavior. Additionally, a reliable simulation framework also provides a platform to design and validate prototypes prior to fabrication. Classic deformable solids simulation allows to predict the pose of a robot for a given state of its actuators. However, the inverse problem arises when aiming for a direct control of soft robots: find the necessary actuation to reach a desired robot pose. The fundamental interactive control capabilities of *point-to-point movement* or *path tracking* among others require a solution to this inverse problem in real time [3].

Duriez [4] introduced a novel method for simultaneous simulation and control of soft robots in an interactive manner based on an inverse kinematics problem using the Finite Element Method (FEM). This framework was further improved by Largilliere et al. [5], efficiently solving the inverse problem through a quadratic-programming (QP) algorithm on the constraint space. The complete framework is integrated within SOFA [6] and enables the design, simulation and interactive control of soft robots piloted by cable actuators controlled by servo-motors and/or pneumatic actuated internal cavities. As evidenced by recent works, hydraulic actuated soft robots exhibit a higher force output, and allow a higher frequency of actuation change [7], increasing the range of applicability of soft robots. Hydraulic actuation has also been proven effective for medical environments [8] among other fields [9].

In this work, we propose a real-time motion planning generation scheme in task-space for hydraulic actuated soft robots, allowing for both rapid prototyping and direct control or hydraulic soft robots. The correct and efficient modeling of hydraulic components arises as a necessary piece of this scheme, and requires a computationally demanding estimation of the fluid weight distribution on the model depending on the current configuration. We present a GPU parallel method for efficient fluid weight distribution inside dynamic cavities, along with a complete modeling of the dynamic behavior of hydraulic components. Moreover, we present a simple yet efficient leveraging mechanism for the computation of *irregular-parallel workloads*, which is applied to further increase the efficiency of the proposed method. We have integrated our entire solution within the Soft Robot framework of SOFA and we have validated our weight distribution algorithm with a real soft model.

II. FEM-QP soft robot control

We aim to develop a model for hydraulic components to be integrated within the FEM-based soft robot control framework proposed in [5]. In their work, cable and pneumatic actuators are handled interactively, but the computational complexity of estimating the fluid weight on the structure of the robots prevented their use within the framework.

Let us first recall the principles of the framework, which serves as base for our work. A soft robot is regarded as a FEM model accounting for its structure and material properties, a set of actuators and an arbitrary number of control DOFs in the form of end effectors placed on the structure. The configuration of the robot at a given time is obtained by solving the static equilibrium between the internal non-linear stress forces of the structure $f(x)$, the external and gravity loads f_{ext} and the contributions of each actuator $J_a^T \lambda_a$, yielding

$$f(x) = -f_{ext} - \sum_a (J_a^T \lambda_a), \tag{1}$$

*This work was not supported by any organization

[1]University of Granada, Spain alejandrora@ugr.es
[2]INRIA, University of Lille 1, France

with J_a^T the direction of the effort applied by the actuator on the FEM nodes and λ_a the contribution of the actuator.

We compute a linearization of the internal forces at each time step i of the simulation

$$f(x_i) \approx f(x_{i-1}) + K(x_{i-1})dx, \tag{2}$$

where $K(x)$ is the tangent stiffness matrix that depends on the current position of the FEM nodes, and dx is the difference between consecutive positions in time $dx = x_i - x_{i-1}$.

To enable a direct control through motion planning, the value of λ_a is unknown, and depends on the input desired constraints δ_e for the end effectors. Thus, a three-step strategy is followed.

Step 1 A configuration x^{free} of the model without actuators influence is found by solving Eq. 1 with $\lambda_a = 0 \; \forall a$. For the end effectors, also coupled to the model through J_e^T, a constraint violation δ_e^{free} is found.

Step 2 The actuators contribution λ_a that minimizes the violation is found by solving an inverse QP problem, efficiently defined by projecting the mechanics into the smallest possible constraint space

$$\begin{aligned} &\min_{\lambda_a} \left\| \delta_e = \sum_a (J_e K^{-1} J_a^T \lambda_a) + \delta_e^{free} \right\|^2 \\ &\text{s.t. } A \geq b \end{aligned} \tag{3}$$

with A and b the constraint matrices on actuators, such as limits on volume growth.

Step 3 The model configuration is corrected

$$x = x^{free} + \sum_a (K^{-1} J_a^T \lambda_a). \tag{4}$$

Non-actuating constraints, such as cables with fixed length or internal air chambers can also be introduced in the system, with their corresponding λ_a set to fixed values.

Within this framework, hydraulic components add two constraints $J_p^T \lambda_p$ and $J_w^T \lambda_w$, corresponding to the pressure term and the fluid weight term respectively. The pressure term is equal to that of the pneumatic components, but the fluid weight term is complex to obtain because it requires an accurate computation of the fluid weight distribution on a given configuration. We address this problem on Section III.

Moreover, both the pressure and the fluid weight terms must be merged prior to performing the optimization routine. Otherwise, they would be treated as independent constraints and lead to an incorrect simulation. We address this issue in Section IV.

III. Fluid weight distribution

The first step for a correct modeling of hydraulic cavity components is an accurate computation of the added fluid weight distribution on the cavity surface. Although this is an easy task on analytical shapes, the typical piecewise FEM models use unstructured triangle or quad meshes for their geometrical description, and the exact computation of the weight distribution in those cases become a highly complex geometrical problem.

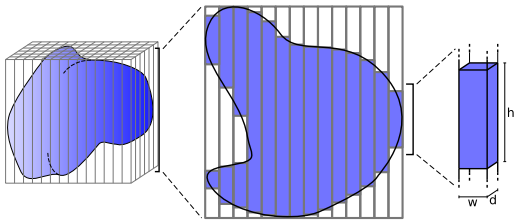

Fig. 1. Our method discretizes the fluid volume inside the cavity using a grid of regular columns covering the cavity geometry as shown on the left. A 2D lateral view of a slab of the computed grid is shown on the center. The parameters used to compute the volume of one column section are shown on the right. Note that in the presence of non-convex cavities, one column may generate more than one weight contribution.

Instead, we address this as a *mesh-on-grid* discretization problem. Two major advantages motivate this choice: we can easily control the trade-off between accuracy and computational cost independently of the particular model by setting the grid size, and more importantly, the problem becomes very amenable to parallel computation, allowing us to obtain very accurate approximations in real time, as we demonstrate later in this work.

A schematic depiction of our approach is shown in Fig. 1. Intuitively, we discretize the bounding box of the cavity into regular fluid columns with fixed cross section area $A = w \cdot d$. Then the problem is reduced to compute the height h of each *column section* bounded by the cavity and distributing its weight, easily computed per column as $W = w \cdot d \cdot h \cdot \rho \cdot g$, with ρ the fluid density and g the gravity force. Of course, for non-convex geometries a single column may produce several column sections contributing to different parts of the cavity (take for instance the 2D simplified case depicted in Fig. 1, center) and thus our algorithm also addresses this casuistry, as we explain below.

W.l.o.g., let us assume cavities defined by a closed triangular mesh. Our algorithm is composed by several consecutive steps:

Step 1 The mesh is transformed to fit a discrete 2D grid with arbitrary resolution. The grid is parallel to the XZ plane with with its normal parallel to the gravity force in the Y direction, and corresponds to the cross section of the grid of 3D columns.

Step 2 The mesh is partitioned in two groups: top triangles and bottom triangles, attending to whether their normals point upwards or downwards.

Step 3 The intersections between the top triangles and the columns are computed. For each candidate column-triangle pair, a barycentric test is performed to check whether the column center intersects the triangle. The barycentric test can be performed in 2D using the projected triangle on the grid plane. In case of intersection, the barycentric coordinates are used to compute the height where the intersection occurs. At the end of this stage, a list of intersecting heights per column is obtained.

Step 4 Similar to the previous step, the intersections between the bottom triangles and the columns are computed. The only difference is that for each intersection, not only the height is stored, but also the triangle *id* and the barycentric coordinates of the intersection point.

Step 5 Finally, per each column, the top-intersections and the bot-intersections are sorted and matched to compute the actual height of the different column sections. Then, the weight force of each column section is distributed to the nodes of its associated triangle, attending to the barycentric coordinates of the intersection to perform the distribution.

Our algorithm is very close in nature to the LDI method [10], which is used for similar purposes in the computer graphics field and could likely be adapted to its use for our purpose. Our method however is specifically designed to our problem and thus avoids unnecessary rendering-related steps present in the LDI method, but more importantly, the LDI method relies on the fixed graphics pipeline while ours is standalone, becoming thus more versatile and easily adaptable to different platforms and languages.

A. Parallel leveraging

We have tested our algorithm running on CPU, but, as we shown in our experiments, only small grid sizes can be used under strong time restrictions, thus obtaining inaccurate estimations. However, we can achieve much more accurate results in the same time using a GPU computation. In fact, all the stages of our algorithm are highly parallelizable, thus we have implemented our algorithm using CUDA.

Steps 1, 2 and 5 are trivially computed in parallel by launching one thread per mesh vertex, triangle and column respectively.

Steps 3 and 4 on the other hand require a more complex parallel leveraging. By computing in parallel the axis aligned bounding box (AABB) of each triangle in the grid space we easily discard many columns per triangle, greatly reducing the candidate column-triangle pairs. The computation of each column-triangle pair candidate is independent of each other and theoretically they could run in parallel. However, the presence of triangles of different sizes leads to an uneven number of candidate columns per triangle, turning into an *irregular-parallel workload* problem [11]. This is actually an ubiquitous problem in a variety of disciplines, and different approaches have been proposed attending to the specificities of each context.

Our case has a special resemblance with a well known issue in the computer graphics field, since a very similar problem arises in the triangle rasterization stage, which is central to the rendering pipeline. This stage was historically performed by the fixed hardware implementation of the GPUs, however, some recent software approaches aiming to provide a higher flexibility have been presented. A very easy approach consists on launching one thread per triangle that computes all the candidates for that triangle [12], [13], unfortunately, a very poor performance is observed for highly uneven configurations. Other approaches provide complex scheduling pipelines to dispatch pair candidates in a parallel manner [14], [15]. Although these approaches provide a stable behavior, their complex nature adds a scheduling overhead that hurts performance, most evidenly when handling very homogeneous cases, and also complicates their implementation, although this is regarded as a minor issue.

We propose a new approach which is as simple as the triangle-parallel approaches, but provides a fairly stable behavior in all cases without this resulting in a heavy computational overhead.

After computing the AABBs of the top or bottom triangles, we compute the average number c of columns per AABB. We then assign $k = \lfloor \alpha c \rfloor$ threads to each triangle (ensuring $k \geq 1$), and we perform a single gpu call of $k \cdot n$ parallel threads, with n the number of triangles. The α parameter allows to control the trade-off between parallel threads and memory read operations as will become apparent later. Through simple arithmetic using the threads *ids* and the already computed triangle-specific AABBs, we efficiently assign a subset of the candidates of each triangle to its assigned threads as we show in Algorithm 1, thus avoiding the construction of complex indirection structures required by other methods.

Algorithm 1 Leveraging kernel

$i_{tri} \leftarrow \lfloor ThreadID/k \rfloor$
$triangle \leftarrow trianglesArray[i_{tri}]$
$AABBSize \leftarrow triangle.AABBMax - triangle.AABBMin$
$offset \leftarrow triangle.AABBMin$
$i_{cur} \leftarrow ThreadID \mod k$
while $i_{cur} < AABBSize.x \cdot AABBSize.y$ **do**
  $i_{col}.x \leftarrow offset.x + i_{cur} \mod AABBSize.x$
  $i_{col}.y \leftarrow offset.y + (i_{cur}/AABBSize.x) \mod AABBSize.y$
  Compute column-triangle pair candidate
  $i_{cur} \leftarrow i_{cur} + k$
end while

This approach is very easy to implement with a negligible overload, yet it greatly homogenizes the workload among the threads. Additionally, this scheme also suits well the modern GPU programming paradigm since neighboring threads access the same triangle information thus memory read operations are efficiently performed, and each thread computes several column-triangle candidates for the same triangle thus this information only needs to be read once per thread. In our experiments we found the best outcome for $\alpha = 0.25$ given an enough amount of triangles, i.e., in the order of several thousands.

Fig. 2 shows the behavior of this approach in two very simple cases with three even and three uneven triangles respectively. In the first case the average number of candidates per triangle is 21, thus $k = 3$ and 9 threads are launched. In the second case, the average number of candidates per triangle is 20.3, thus $k = 5$ and 15 threads are launched. In both cases, all the threads are launched in a single GPU call, addressing all the column-triangle candidates without

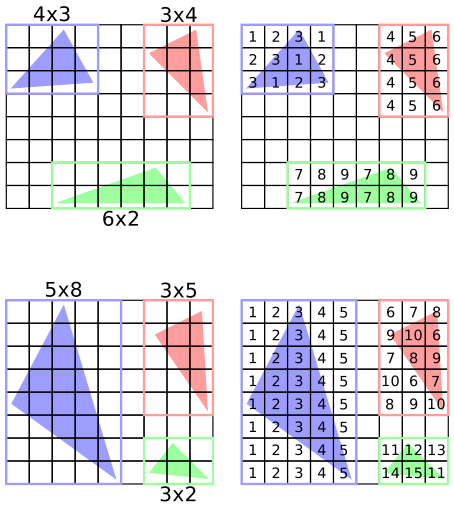

Fig. 2. Two simple cases with an even (top) and uneven (bottom) set of triangles. After computing the triangles AABBs (left), the average number of candidates per triangle is obtained and the number of threads to launch per triangle is computed. All the threads are launched in a single GPU call, which compute all the column-triangle candidates (right).

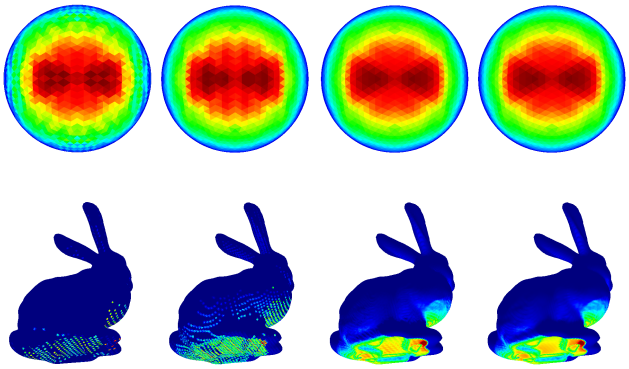

Fig. 3. Results of our weight distribution algorithm for different grid sizes (from left to right, 30^2, 50^2, 200^2 and 500^2) applied on a spherical cavity with 5120 triangles (top) and a bunny-shaped cavity with 90760 triangles (bottom). The nodal weight distribution is shown using a heatmap ranging from dark blue to dark red. Note that a distribution pattern appears due to the meshing of both the sphere and the bunny.

the need of additional indirection structures. In the first case, the workload on the threads is perfectly balanced. In the second case, the thread workload remains uneven, but the computational burden of each triangle is evenly split among its assigned threads, homogenizing the average workload with a negligible overload.

B. Performance and accuracy evaluation

We have tested our fluid weight distribution in order to evaluate the performance of the algorithm and the accuracy achieved. We use two models with different topological complexity: a convex sphere model, created with $1cm$ radius and 5120 triangles, and a non-convex bunny model of $2cm$ height with 90760 triangles. We have applied our method with different grid sizes to validate the volume estimation and test its performance using both the CPU and the GPU implementations using an Intel Xeon W3550 system equipped with a Nvidia GeForce GTX 960. The results are summarized in Table I.

TABLE I

VOLUME ESTIMATION OBTAINED AND TIME REQUIRED BY BOTH CPU AND GPU IMPLEMENTATIONS OF OUR METHOD FOR TWO CAVITY MODELS USING SEVERAL GRID SIZES.

	Sphere			Bunny		
Grid size	Volume (cm^3)	GPU (ms)	CPU (ms)	Volume (cm^3)	GPU (ms)	CPU (ms)
30^2	4.1835	0.29	0.63	1.1414	0.65	3.97
50^2	4.1828	0.31	0.93	1.6709	0.75	7.07
100^2	4.1804	0.37	2.24	1.7154	0.84	9.57
200^2	4.1799	0.57	7.63	1.7164	1.06	15.64
500^2	4.1798	0.86	41.77	1.7165	1.27	54.16
1000^2	4.1798	2.65	160.75	1.7165	2.79	186.66
1500^2	4.1798	5.29	358.81	1.7165	4.90	394.95

The theoretical volume of a sphere with $r = 1cm$ is $V = \frac{4}{3}\pi = 4.1887cm^3$, however our mesh is not analytic and its exact volume computed using the signed volume method [16] is $V = 4.1798cm^3$. Although the cavity volume estimation is very good for all the grid sizes, the distribution is not well approximated for low resolution grids, as seen in Fig. 3 (top), however, for the 200^2 grid both the volume estimation and the distribution estimation are very precise, and for larger grids the improvement is negligible. It is also worth noting that the meshing of the cavity plays an important role during the distribution process since larger triangles receive a higher amount of fluid weight.

For the bunny model we also compute the exact volume using the signed volume method, obtaining $V = 1.7165cm^3$. In this case, a very low grid resolution leads to a poor volume estimation, but for the 100^2 grid the cavity volume is accurately approximated. The distribution estimation requires higher grid resolutions as seen in Fig. 3 (bottom), mainly due to the increased number of triangles of the mesh, reaching a very precise estimation for the 500^2 grid. Again, meshing patterns appear during the distribution process.

Summing up, the method converges to the real solution by increasing the grid size. Naturally, the computation time also increases and the CPU version remains interactive only for low resolution grid sizes, however, the GPU version achieves very accurate estimations for both cases in less than 2 milliseconds, enabling its use for real-time simulation environments.

IV. UNIFIED HYDRAULIC CONSTRAINT

Hydraulic actuation applies two types of load on the deformable structure of the robot: pressure inside the cavity and additional weight distribution created by the fluid (computed thanks to the method presented above). Yet, in practice, these two terms are coupled when one activates the actuator by adding / removing the liquid. In the case of hydraulic components, this equates to differentiating the pressure and fluid weight terms w.r.t. the amount of contribution. Consequently, both terms must be coupled in a single constraint defining the behavior of the hydraulic component, and the function

governing this coupling has a highly non-linear nature. Our goal is to keep the same formulation $J_a^T \lambda_a$ (see equation 1) for the total contribution of hydraulic actuators. In the following, we present how this term is obtained.

The derivative of pressure forces is easy to obtain since it only depends on the geometry of the cavity walls. $p = \frac{F}{A}$, thus we define $J_p^T \lambda_p = pA$. Therefore, for a given configuration, we build J_p^T as $(J_p^T)_i = \sum_{t,i \in t} \frac{S_t n_t}{3}$, with n_t the normal and S_t the area of a triangle incident on the i^{th} node. With this definition, J_p^T is homogeneous to a force and λ_p represents the intensity of the pressure.

The derivative of the weight distribution with respect to the volume change, on the other hand, is complex because it depends on the cavity shape deformation. This deformation depends on the equilibrium between the pressure forces exerted from inside the cavity (which in turn depend on the geometry of the cavity walls as explained above), and the opposing stress forces exerted by the material surrounding the cavity (which in turn depend on the material properties and the current geometry configuration of the soft robot).

This is in fact a highly non-linear relation, but it can be linearized around the current configuration, assuming small weight variations. This assumption is valid since the soft-robot control pipeline is recomputed every few milliseconds, thus changes between iterations are indeed small. We also validate this hypothesis experimentally in this work. We define $J_w^T = \frac{\partial W}{\partial v}$ with W the fluid weight and v the cavity volume. For a given configuration, we compute the weight distribution per node w_i as explained in Sec. III and set the corresponding J_w entries with $\frac{w_i}{v_c}$, with v_c the current cavity volume. λ_w represents then the cavity volume change.

Lastly, we need to merge both pressure and weight terms because they are in fact coupled for hydraulic components and otherwise they would be treated as independent constraints during the optimization routine. As we explained earlier, $K^{-1} J_a^T \lambda_a = \Delta x$, thus $K^{-1} J_p^T \lambda_p$ gives the nodal displacements due to the pressure term. These displacements can be regarded as extrusion lengths for a given surface shape with area a, extruded along the surface normal direction, and would thus produce a volume change $a \cdot \Delta x$. Since this surface information is already stored in J_p, we linearize this relation as $\lambda_w = J_p K^{-1} J_p^T \lambda_p$ that couples both terms. Therefore, we merge both terms into a single hydraulic constraint λ_a with $J_a^T = J_p^T + J_w^T J_p K^{-1} J_p^T$. Again, this linearization remains accurate only for small changes from a given configuration and must be updated every iteration.

V. Results and Validation

We have integrated our solution within the SOFA soft robot framework [5]. To validate our model, we fabricated an empty silicone cylinder, we fill the cavity both with air and water and inflate it with an extra 15 ml of content in each case. We compare the obtained results with the simulated version of the same setup, using a FEM model with 375 nodes and computing the fluid distribution inside the cavity with a grid of 200^2 cells. The final results are shown in Fig. 4. Our model accurately reproduces the behavior of its fabricated counterpart in every case, indicating that our fluid weight distribution method achieves accurate and realistic estimations despite being.

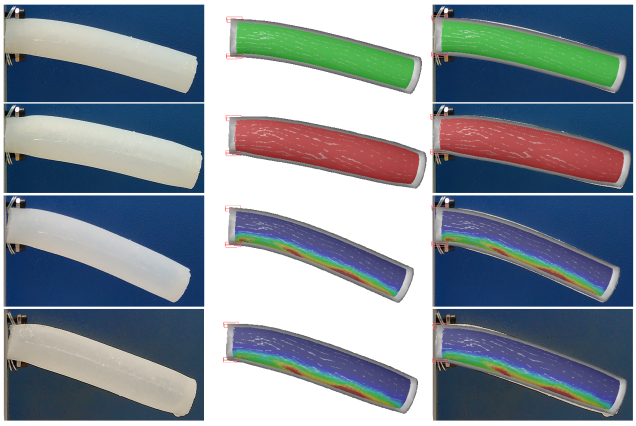

Fig. 4. Comparison of the fabricated silicon cylinder and the simulation of the same setup. From top to bottom, filled with air in rest state, inflated with 15 ml, filled with water in rest state and inflated with 15 ml.

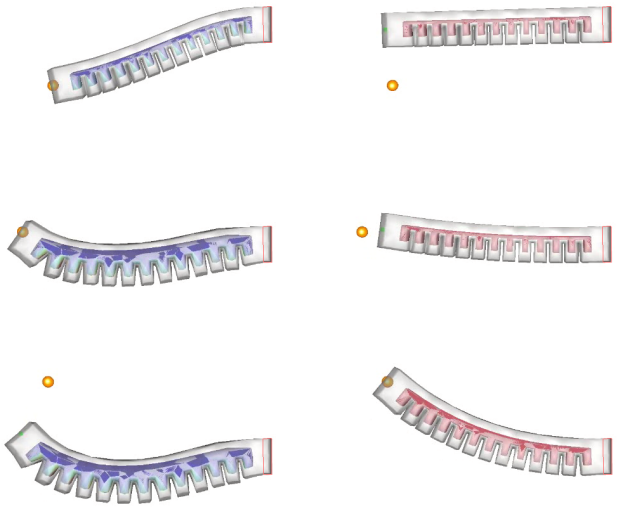

Fig. 5. Comparison of a soft finger operated by an hydraulic actuator (left) and a pneumatic actuator (right) for the same desired positions (yellow sphere). The models exhibit very different ranges of actuation, but our framework is able to reach the closest possible configuration in each case.

For the validation of our inverse problem modeling, we perform a direct control of an actuated soft finger. To further emphasize the differences between pneumatic and hydraulic actuated soft robots, we compare both actuators on the soft finger as shown in Fig 5. The user interactively inputs the desired position of an end effector placed on the tip of the finger and the inverse problem is solved using the constraints built as explained in Section IV. As it is clearly depicted, the range of possible configurations is very different in both cases, and the models are accurately actuated to reach the desired configuration, or reach the closest possible configuration when the desired position is unreachable.

We also test a more complex example of a hanging soft

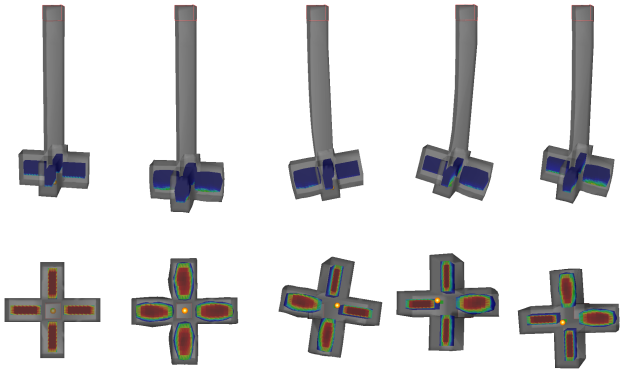

Fig. 6. Different configurations of a hydraulic actuated soft pendulum, shown from the front (top row) and from below (bottom row). The motion planning framework operates the actuators in real time to reach the desired position (yellow sphere) which is controlled by the user.

pendulum operated by four hydraulic actuators, shown in Fig 6, where the inflatable cavities can modify the weight of the bottom part and the center of gravity. Again, the user inputs the desired position of the end effector placed at the bottom of the pendulum. The fluid distribution inside each cavity is accurately computed with a grid of 200^2 cells, and the computation of the four cavities takes less than 2.1 ms.

The fluid weight is correctly considered during the motion planning process, allowing the pendulum to extend and contract along its vertical (Fig 6, first and second columns) and bend towards any direction in the horizontal plane (Fig 6, third to fifth columns) by inflating one or more of the cavities.

VI. Conclusions and Future Work

In this work we have presented an online simulation and motion planer for hydraulic actuated soft robots. The motion planer is based on an inverse solver that has been integrated within the SOFA framework. The main contributions are a novel method for real-time computing of fluid weight distribution and an accurate model for the dynamic behavior of hydraulic actuated cavities in soft robots. Moreover, we propose a novel GPU parallel leveraging algorithm for the efficient computation of irregular-parallel workload problems, increasing the efficiency of our fluid weight distribution approach. Our experiments and results show that both the weight distribution algorithm and the dynamic behavior model of the soft robots provide very accurate estimations with interactive times, enabling the direct control of hydraulic soft robots.

Our simulation matches the real behavior of a simple fabricated specimen and, although we can successfully control a simulated hydraulic soft robot, we would like to apply our solution to control a fabricated soft robot as future work. We also believe that a joint actuation of hydraulic and other components may increase the capabilities of soft robots and should be explored. Another interesting line for future work is the inclusion of contact with the environment in the motion planner, which could achieve an automatic avoidance of contact with obstacles or even consider them to modify the range of configurations of the robots. Moreover, we believe that our parallel leveraging algorithm may be of use in other contexts, such as software rasterization algorithms. We would like to perform a comparison with other current leveraging strategies and evaluate its performance and benefits.

Acknowledgements

This work is supported by the FPU 2012 program of the University of Granada and by the project TIN2014-60956-R of the Spanish Ministry of Economy and Competitiveness.

References

[1] A. D. Marchese, R. K. Katzschmann, and D. Rus, "Whole arm planning for a soft and highly compliant 2d robotic manipulator," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 554–560.

[2] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus, "Haptic identification of objects using a modular soft robotic gripper," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1698–1705.

[3] A. D. Marchese, K. Komorowski, C. D. Onal, and D. Rus, "Design and control of a soft and continuously deformable 2d robotic manipulation system," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2189–2196.

[4] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3982–3987.

[5] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, and C. Duriez, "Real-time control of soft-robots using asynchronous finite element modeling," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2550–2555.

[6] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "Sofa: A multi-model framework for interactive physical simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*. Springer, 2012, pp. 283–321.

[7] R. K. Katzschmann, A. D. Marchese, and D. Rus, "Hydraulic autonomous soft robotic fish for 3d swimming," in *Experimental Robotics*. Springer, 2016, pp. 405–420.

[8] K. Ikuta, H. Ichikawa, and K. Suzuki, "Safety-active catheter with multiple-segments driven by micro-hydraulic actuators," in *Medical Image Computing and Computer-Assisted InterventionMICCAI 2002*. Springer, 2002, pp. 182–191.

[9] M. De Volder and D. Reynaerts, "Pneumatic and hydraulic microactuators: a review," *Journal of Micromechanics and microengineering*, vol. 20, no. 4, p. 043001, 2010.

[10] B. Heidelberger, M. Teschner, and M. H. Gross, "Real-time volumetric intersections of deforming objects." in *Proceedings of Vision, Modeling, Visualization (VMV)*, vol. 3, 2003, pp. 461–468.

[11] S. Tzeng, A. Patney, and J. D. Owens, "Task management for irregular-parallel workloads on the gpu," in *Proceedings of the Conference on High Performance Graphics*. Eurographics Association, 2010, pp. 29–37.

[12] F. Liu, M.-C. Huang, X.-H. Liu, and E.-H. Wu, "Freepipe: a programmable parallel rendering architecture for efficient multi-fragment effects," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. ACM, 2010, pp. 75–82.

[13] K. Fatahalian, E. Luong, S. Boulos, K. Akeley, W. R. Mark, and P. Hanrahan, "Data-parallel rasterization of micropolygons with defocus and motion blur," in *Proceedings of the Conference on High Performance Graphics 2009*. ACM, 2009, pp. 59–68.

[14] C. Eisenacher and C. Loop, "Data-parallel micropolygon rasterization," *Eurographics 2010 Short Papers*, pp. 53–56, 2010.

[15] S. Laine and T. Karras, "High-performance software rasterization on gpus," in *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. ACM, 2011, pp. 79–88.

[16] C. Zhang and T. Chen, "Efficient feature extraction for 2d/3d objects in mesh representation," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 3. IEEE, 2001, pp. 935–938.

Chapter 4

Interactive medical visualization

4.1 A parallel resampling method for interactive deformation of volumetric models

- **A. Rodríguez**, A. León Salas, D. Martín Perandrés and M. A. Otaduy (2015). "A parallel resampling method for interactive deformation of volumetric models". *Computers & Graphics*, Volume 53, pp. 147-155.
  - Status: Published [105]
  - Impact factor (JCR 2015): 1.120
  - Subject category: Computer Science, Software Engineering (Q2: 41/106).

A parallel resampling method for interactive deformation of volumetric models

Alejandro Rodríguez[a], Alejandro León[a], Domingo Martín[a], Miguel A. Otaduy[b]

[a] *University of Granada, Granada, Spain*
[b] *URJC, Madrid*

Abstract

In this work, we propose a method to interactively deform high-resolution volumetric datasets, such as those obtained through medical imaging. Interactive deformation enables the visualization of these datasets in full detail using state-of-the-art volume rendering techniques as they are dynamically modified. Our approach relies on resampling the original dataset to a target regular grid, following a 3D rasterization technique. We employ an implicit auxiliary mesh to execute resampling, which allows us to decouple mapping of the deformation field to the volume from actual resampling. In this way, our method is practically independent of the deformation method of choice, as well as of the resolution of the deformation meshes. We show how our method lends itself nicely to an efficient, massively parallel implementation on GPUs, and we demonstrate its application on several high-resolution datasets and deformation models.

Keywords:
Volume data, volume deformation, 3D rasterization

1. Introduction

The tissue distribution of biological forms is often captured and visualized using volumetric representations, most notably in medical applications.

Email address: alejandrora@ugr.es (Alejandro Rodríguez)

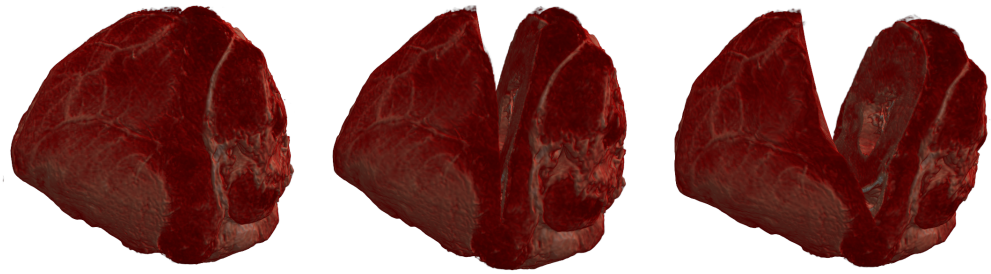

Figure 1: A cutting operation is applied to a sheep heart dataset consisting of 12.4 million voxels, deformed using a Mass-Spring model. Our proposed resampling algorithm allows us to deform and cut the model interactively. After each simulation step, our method outputs a resampled regular grid of the same resolution as the input, which is visualized with a standard Ray-casting algorithm, revealing internal features due to the open cut. Resampling of this high-resolution dataset is executed in just 26 ms per frame.

Biological tissue is soft and deformable for the most part, hence applications dealing with biological tissue require methods to deform such volumetric representations. Some common examples in the medical field include image registration, intra-operative navigation, or surgical planning. Nowadays practical implementations of these applications are often limited to rigid data or non-interactive solutions, due to the difficulty to deform volumetric representations in an interactive manner.

Regardless of the particular deformation method of choice, one major challenge for the development of applications using deformed volume data is the visualization of dynamic volumetric representations. Traditionally, there are mainly two strategies to address this problem: (i) Segment and mesh the original volume data, and resort to visualization of surface meshes. This strategy fails to capture the full detail in the original volume data, and it does not allow interactive modification of the visualized isosurfaces as offered by volume rendering techniques. (ii) Execute volume rendering using unstructured meshes. Despite achieving interactive framerates, as demonstrated by Georgii et al. [1], the cost of unstructured volume rendering grows with mesh resolution. In addition, as remarked by Correa et al. [2], advanced lighting such as gradient-based lighting, which is an important feature in medical applications, becomes complex even in the case of static unstructured meshes.

In our work, we follow a different strategy. We propose to deform the original volume data by dynamically resampling it onto a regular grid, and then

apply regular volume rendering. Of course, this strategy is not new to computer graphics, as it constitutes the fundamental pipeline of raster graphics with deferred shading. This strategy has already been applied to volume data deformation, by other authors, e.g. Schulze et al. [3] and Gascón et al. [4], but we achieve more than one order of magnitude speed-up compared to previous methods. As a result, we can deform and render high-resolution volumetric objects with high-resolution deformable meshes at interactive rates.

In our method, the input volumetric dataset is transformed into an implicit sampling mesh of the same resolution. At runtime, two steps are performed. First, after the deformation stage, the deformation field is applied to the positions of the nodes of the sampling mesh. And second, the deformed sampling mesh is resampled onto the target regular grid. Both steps are massively parallelized in our GPU implementation. With our two-step method, we decouple the resampling from the evaluation of deformations, and due to the implicit and regular nature of the sampling mesh, we eliminate the need for any type of indirection or auxiliary data structure, and thus we maximize throughput, achieving better performance than previous methods. Actual volume rendering is decoupled into yet a third step, which is executed efficiently on the target regular grid.

The decoupling of deformation mapping, resampling, and volume rendering has several advantages over previous approaches. In contrast to methods based on unstructured volume rendering, one major advantage is the possibility to adopt advanced lighting models efficiently, such as gradient-based volume lighting. Moreover, the size of the visualization viewport and the resolution of the deformation model affect performance separately, and there is no extra penalty in combining a large viewport with a high-resolution deformation model. When compared with all previous methods, one general advantage of ours is that it can be coupled with any deformation technique by means of a mapping of the underlying deformation structure to the implicit sampling mesh. And yet a final but important advantage is that the performance of the resampling algorithm is independent of the resolution of the underlying deformation structure, thus allowing interactive visualization when using deformation meshes several orders of magnitude denser than previous methods. This feature becomes critical to support deformation algorithms that support high-resolution meshes, such as GPU-parallel soft-tissue simulation algorithms based on implicit FEM [5] or Mass-Spring [6], coarsening algorithms that govern a heterogeneous high-resolution tetrahedral mesh through an efficient homogenized low-resolution simulation [7, 8, 9], or the

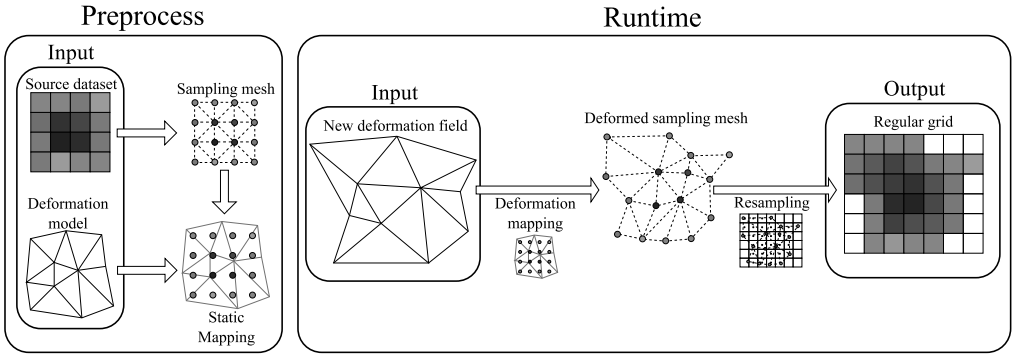

Figure 2: An overview of our method. In a preprocessing step, the sampling mesh is created using the input dataset, and coupled with the underlying deformation structure, generating static mapping information. After every deformation step, the resulting deformation is mapped to the sampling mesh, which is then resampled to a regular grid.

ChainMail algorithm [3, 10], which can handle interactively models with millions of nodes.

In our examples we have tested diverse models such as the Finite Element Method, the Mass-Spring Model and the ChainMail algorithm, and we also include extensions to handle topological changes. We show that we can successfully deform and visualize a volume with 12.4 million voxels, using a mass-spring model with 2.4 million tetrahedra, at a rate of more than 20 fps (Fig. 1).

The remainder of this paper is organized as follows: Section 2 reviews related work. Our parallel resampling method is described in Section 3, including the definition of the implicit sampling mesh, the mapping process and the resampling algorithm, together with implementation details. Section 4 presents several results, coupling our pipeline with several deformation methods. Several experiments to analyze the properties of the method are also presented. Finally, our conclusions are listed in Section 5.

2. Previous works

Existing techniques for visualizing deformable volumetric models can be grouped into three main strategies.

A first group of approaches relies on the concept of spatial deformation. Instead of applying a deformation to the model, an inverse deformation map is applied to the rays that traverse the space containing the volume. Rezk-

Salama et al. [11] proposed a spatial deformation technique dividing the space into a hierarchical set of deforming sub-cubes, thus defining a piecewise approximated inverse deformation field. Westermann et al. [12] proposed a set of local and global free-form space deformations, applying the inverse deformations through tessellated slicing planes. H. Chen et al. [13] applied Ray-casting to volumetric models deformed through a free-form deformation (FFD) approach using an inverse ray deformation technique. M. Chen et al. [14] introduced the concept of Spatial Transfer Functions as a tool to define FFD operations. Correa et al. [15] presented a set of FFD spatial operators enforcing an alignment with the features present in the models to generate medical illustrations. These spatial deformation schemes enable interactive frame rates, but at the price of low-resolution, non-physically based deformation.

A second group of approaches deforms an unstructured volumetric mesh, and then executes volume rendering on this unstructured mesh. Classic and current unstructured volume rendering techniques aim for an accurate visualization of non-regular volumetric models that can be deformed over time (see the works of Miranda et al. [16] and Okuyan et at. [17] for recent GPU-based unstructured volume rendering techniques). However, a direct application to physically deformed medical volumes impedes interactive frame rates under high-resolution deformation structures, since medical models may contain several orders of magnitude more elements than those handled by these techniques interactively. Inheriting many of the ideas of these approaches, Georgii et al. [1] proposed a system to perform unstructured volume rendering of tetrahedral meshes deformed by physical simulation schemes using the GPU rendering pipeline, also allowing the use of 3D texture mapping to increase the detail inside a tetrahedron. While the method achieves interactive framerates for relatively large models (i.e., 100 ms per frame for a tetrahedral mesh of 190,000 elements using a 512x512 viewport), its cost grows bilinearly with mesh resolution and viewport size. Nakao et al. [18] proposed the use of a dynamically refined proxy mesh to adapt the mesh complexity to the deformations applied to the model. This scheme handles large volumes and allows topological changes in the model but, as the authors point out, the performance of the method depends on the number of nodes of the deformable model, and can only support a few hundred interactively.

A third group of approaches, motivated by the superior performance of direct volume rendering (DVR) techniques over the unstructured volume rendering techniques, as well as the more advanced shading and illumina-

tion techniques available under DVR, performs a resampling of the deformed volume onto a regular grid which is later fed as input to a standard DVR pipeline. Schulze et al. [3] proposed a resampling algorithm based on a nearest neighbor search to relocate and interpolate the deformed voxels. However, the algorithm is designed to perform resampling only on small parts of the volume, and performance degrades badly if the deformed volume is large. In addition, the resampling process is highly dependent on the ChainMail deformation technique. Gascón et al. [4] proposed a GPU-based tetrahedral mesh rasterization algorithm with 3D texture mapping. After each simulation step, the deformed tetrahedra are rasterized onto a regular grid, and target voxels are mapped to the original volume for a texture lookup. The parallel implementation of the rasterization process achieves interactive frame rates for high-resolution volumes, but performance degrades under high-resolution meshes. In addition, the method only supports deformation techniques based on tetrahedral meshes.

Our method also falls in this group, as it performs a parallel resampling of the original volume data set onto a regular grid. However, thanks to the use of an intermediate implicit sampling mesh that decouples the computation of the deformation from the rasterization, we avoid costly indirections during the actual rasterization, and the performance is independent of the resolution of the underlying deformation structure.

3. Volume resampling pipeline

The key to the efficiency of our volume resampling method is the use of an implicit sampling mesh that effectively decouples the data structure of the particular deformation model of choice from the actual resampling of the volume. Fig. 2 outlines the full resampling pipeline.

In a preprocessing step, the sampling mesh is generated from the input dataset and it is coupled with the deformation method. At runtime, two steps are performed after each deformation stage on a massively parallel manner on the GPU: the mapping of the deformation to the sampling mesh, and the resampling of the deformed sampling mesh.

This section starts with a description of the implicit sampling mesh, and continues with descriptions of the deformation mapping and the actual resampling.

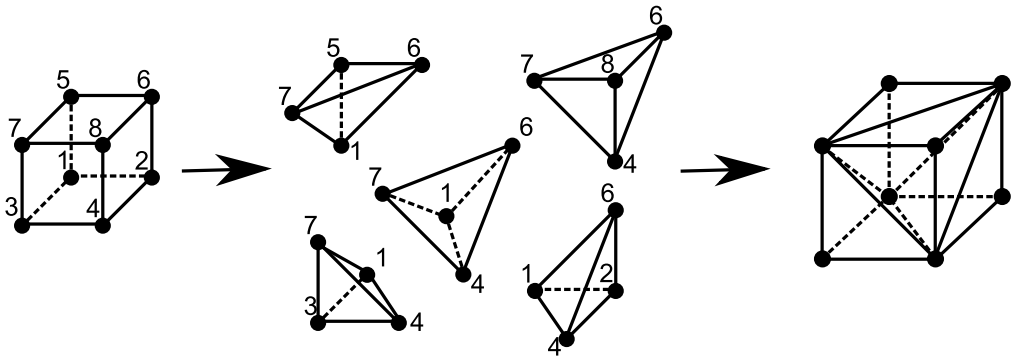

Figure 3: A 5T tetrahedral decomposition generates five adjoining tetrahedra covering the same volume as the original hexahedron.

3.1. Implicit sampling mesh

Given a regular grid as input dataset, we define a regular mesh of the same resolution that carries in its vertices the data associated to grid points. This regular mesh retains absolutely all the information present in the original dataset. When the mesh is deformed, a continuous field can be reconstructed on the entire volume occupied by the mesh through interpolation of the original data values inside each mesh element.

Specifically, given an input dataset stored as a regular grid of $l \times m \times n$ voxels, we create an array of $l \times m \times n$ vertices, each of them associated to one voxel. Each vertex is assigned the data value of its corresponding voxel, and is placed at a position in space given by its indices and the spacing of the model in each dimension.

Given the initial layout of the vertices, every eight adjacent vertices define a hexahedron. We implicitly decompose each hexahedron into five adjoining tetrahedra, following a 5T decomposition, as shown in Fig. 3.

Each hexahedron can be decomposed into five adjoining tetrahedra (5T decomposition) partitioning the entire volume of the hexahedron as shown in Fig. 3. We transform the hexahedral mesh into an adjoining tetrahedral mesh by applying mirrored 5T decompositions to adjacent hexahedra [19], using the shared face as a mirror plane.

The proposed decomposition scheme produces a mesh that is continuous and complete. These properties guarantee that, as long as the input deformation is self-intersection free, every point inside the mesh is bounded always by one and only one tetrahedron.

Thanks to the regularity of the mesh, we can infer the tetrahedra incident on each vertex simply from its indices. As a result, the sampling mesh is only implicitly defined, without the need to explicitly build it or store it. Therefore, the implicit sampling mesh does not add any overhead to the storage of the original dataset aside from the vertex positions, and it is visited on-the-fly during the resampling step.

We have considered other options for the definition of the implicit sampling mesh, in particular the hexahedral mesh defined inherently by the grid. However, we have opted for our tetrahedral mesh because inclusion tests and interpolation functions are simpler (non-planar faces are avoided), hence more efficient at run-time. Despite having more elements than the hexahedral mesh, the implicit definition of our tetrahedral mesh avoids storage penalties.

3.2. Deformation mapping

The sampling mesh acts as an intermediate representation between each particular deformation method and the resampling process. The deformation method produces a deformation field that can be evaluated at any location in space. In order to carry out this evaluation efficiently, we set a static mapping between the deformation method and the sampling mesh.

In a preprocessing step, for each vertex of the sampling mesh, we store static mapping information, i.e., appropriate pointers to the elements of the deformation method, as well as static weights or coefficients needed for evaluating the deformation field. The particular pointers, weights and/or coefficients stored per vertex depend on the particular deformation method of choice.

At runtime, once the deformation model is updated, the mapping process is executed to define the updated positions of the sampling mesh vertices, according to the new deformation field. This mapping is executed on a massively parallel manner on the GPU, and the actual functions to be evaluated depend again on the particular deformation method of choice.

In Section 4 we show examples with different deformation methods.

3.3. Volumetric resampling

At runtime, once the deformation is mapped onto the vertices of the sampling mesh, we execute the resampling of the original dataset onto a target regular grid. This process is executed in parallel for all the tetrahedra of the sampling mesh, and each tetrahedron contains all the information

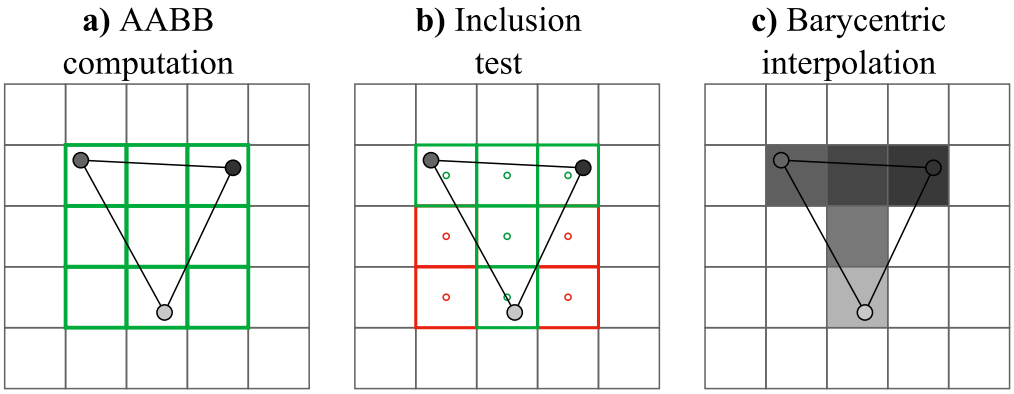

Figure 4: Steps of the resampling process on a 2D example. a) The AABB of the triangle is computed. b) An Inclusion test is performed for each voxel in the AABB, using the barycentric coordinates of its center. c) Output data values are assigned to the voxels lying inside the triangle through barycentric interpolation.

needed in the process, i.e., the target positions of its four vertices and the original data values to be interpolated.

For each tetrahedron of the sampling mesh, we perform the following process. First, we select candidate target voxels by computing an axis-aligned bounding box (AABB) of the vertices of the tetrahedron. Then, we traverse all the candidate voxels, and compute their barycentric coordinates. For voxels that lie inside the tetrahedron, we compute the output value through barycentric interpolation of the data values stored in the four vertices of the tetrahedron. A simplified 2D version of the resampling process is shown in Fig. 4.

3.3.1. GPU implementation

This algorithm is well suited for massive GPU parallelization. We have implemented it as a single GPU kernel that runs in parallel on the hexahedral decomposition of the sampling mesh, thus each thread visits the five tetrahedra defined implicitly on each hexahedron.

A high-level pseudo-code of the resampling kernel is outlined in Code 1. Each thread loads the eight vertices of a hexahedron (lines 7-14), labeled as indicated in Fig. 3. Then, to ensure that the complete mesh remains adjoining, the vertices are reordered as necessary, thus the labels of the vertices are exchanged (lines 15-23) depending on the parity of the indices of the first vertex (vertex 1 in Fig. 3). Note that this operation is handled at runtime,

```
1  kernel_resample(tex3D outGrid)
2    int x,y,z;
3    x = getThreadIndexX();
4    y = getThreadIndexY();
5    z = getThreadIndexZ();
6    vertex v1, v2, v3, v4, v5, v6, v7, v8;
7    v1 = getVertex(x, y, z);
8    v2 = getVertex(x+1, y, z);
9    v3 = getVertex(x, y, z+1);
10   v4 = getVertex(x+1, y, z+1);
11   v5 = getVertex(x, y+1, z);
12   v6 = getVertex(x+1, y+1, z);
13   v7 = getVertex(x, y+1, z+1);
14   v8 = getVertex(x+1 ,y+1, z+1);
15   IF (x % 2 == 1)
16     swap(v1, v2);swap(v3, v4);swap(v5, v6);swap(v7, v8);
17   ENDIF
18   IF (y % 2 == 1)
19     swap(v1, v5);swap(v2, v6);swap(v3, v7);swap(v4, v8);
20   ENDIF
21   IF (z % 2 == 1)
22     swap(v1, v3);swap(v2, v4);swap(v5, v7);swap(v6, v8);
23   ENDIF
24   SampleTetrahedron (v1, v3, v4, v7, outGrid);
25   SampleTetrahedron (v7, v8, v4, v6, outGrid);
26   SampleTetrahedron (v4, v2, v1, v6, outGrid);
27   SampleTetrahedron (v1, v5, v7, v6, outGrid);
28   SampleTetrahedron (v7, v4, v6, v1, outGrid);
29 END
30
31 SampleTetrahedron (vertex v1, v2, v3, v4, Tex3D outGrid)
32   aabb boundingBox = outGrid.computeAABB(v1, v2, v3, v4);
33   FOREACH (voxel IN boundingBox)
34     float4 baryCoords = computeBaryCoords(voxel.center, v1, v2, v3, v4);
35     IF (centerLiesInsideTetrahedron(baryCoords))
36       char newValue = interpolateValue(baryCoords, v1, v2, v3, v4);
37       setValue(voxel, dataValue);
38     ENDIF
39   ENDFOREACH
40 END
```

Code 1: Pseudo-code of the resampling GPU kernel. Each thread handles one hexahedron of the sampling mesh, the eight vertices of the hexahedron are loaded. After that, the corresponding tetrahedra are deduced on-the-fly and sampled onto the output grid.

without ever storing the tetrahedral mesh explicitly. Lastly, each tetrahedron is actually sampled on the target grid (lines 24-28) as explained earlier.

Thanks to the regular and structured nature of our sampling mesh, consecutive threads access consecutive array positions and thus we achieve coalesced global memory read operations (lines 7-14). This coalesced access scheme is achieved independently of the current configuration of the mesh since it depends on the GPU thread ids alone. For this same reason, the access to vertex-based data (such as deformed positions and volume data) is easily optimized using shared memory since neighboring threads belonging to the same thread warp access neighboring vertices. Moreover, the implicit definition of the mesh eliminates the need for any indirection scheme for the topology definition, saving both memory requirements and global and shared memory accesses.

4. Results and discussion

We have tested our volume deformation method with several different deformation models. In the following subsections we discuss implementation details for each deformation model, along with performance data. We refer the reader to the video provided as additional material for more results. We also present the results of several tests performed in order to analyze the properties and limitations of our method.

We have implemented our algorithm using OpenCL 1.2, running on an Intel Core i5-3570 machine with 8 GB RAM, equipped with an AMD Radeon R9 270X with 2 GB of video memory GDDR5.

4.1. ChainMail deformation

A parallel version of the ChainMail algorithm, similar to the one proposed by Rößler [20] has been implemented. The ChainMail algorithm, introduced by Gibson [21], applies elastic and plastic deformations to the model at the same resolution of the input dataset by defining geometric constraints between neighbor elements of the model. Heterogeneous deformations can be achieved by defining different restrictions to the elements, as explained in [22].

Since the ChainMail algorithm works at the resolution of the input dataset, in this case the vertices of the sampling mesh are co-located with the ChainMail elements and the mapping is straightforward.

We have integrated the possibility to execute simple topological changes on the ChainMail model by implementing cutting and carving operations,

Figure 5: A foot dataset consisting of 16.7 million voxels is deformed using a corotational finite element method using a mesh of 16,000 tetrahedra. Our resampling pipeline generates a resampled regular grid in 39 ms, which is then visualized with a standard Ray-casting algorithm.

following the approach in [23]. To apply the topological changes to the volume dataset for visualization purposes, we obtain acceptable results simply by deleting tetrahedra that are cut or carved, thanks to the high resolution of the implicit sampling mesh. We add a *bitfield* char value to each hexahedron to flag individual tetrahedra as active or inactive, and we check this bitfield as part of the resampling kernel in Code 1. When a new cutting/-carving operation is applied to the model, we perform an intersection test with the cut surface (or carving volume) for each tetrahedron, and flag new inactive tetrahedra accordingly. With simple cut surfaces or carving volumes, a brute-force GPU parallelization of per-tetrahedron tests turned out to be fast enough. We refer the reader to the survey by Wu et al. [24] for more information on advanced cutting methods.

Fig. 6 shows example applications of our method on two medical datasets. The knee model with 13.2 million voxels is deformed at 35 ms per frame. 11 ms are devoted to ChainMail deformation, 0.5 ms to map the deformation to the sampling mesh, and 23.5 ms to resample the volume. Cutting operations are performed in less than 21 ms, and they affect performance only at frames when the cut is actually executed, not in subsequent frames. The head model with 6.6 million voxels is deformed at 16 ms per frame. Carving operations are performed in less than 8 ms, and they also affect performance only while carving is executed.

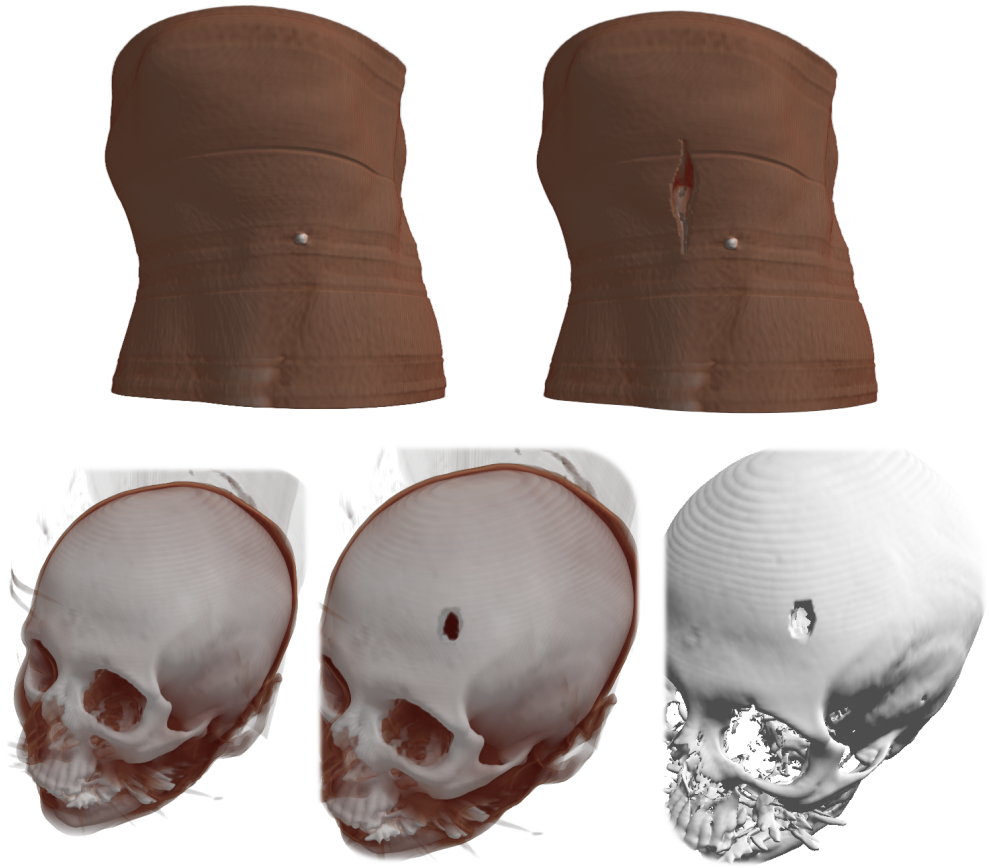

Figure 6: Interactive deformations of medical datasets using the ChainMail model.**Top:** A cutting operation is applied to a knee model consisting of 13.2 million voxels, followed by a deformation to visualize internal structures. Our pipeline resamples the volume in 24 ms. **Bottom:** A carving operation is applied to a head model consisting of 6.6 million voxels. Our pipeline resamples the volume in 11 ms. The output regular grid can be visualized with different direct volume rendering techniques, such as a standard Ray-casting volume rendering technique (top, bottom-left and bottom-center) or a Ray-casting-based isosurface extraction algorithm (bottom-right).

Table 1: Evaluation of the cost of the two steps of our algorithm (deformation mapping and resampling) for different deformation methods and deformation mesh resolutions. The cost of ray-casting the resampled model for a 700×700 viewport with 500 samples per ray is also shown. Finally, the size of the sampling mesh and its required GPU memory are also shown.

Deformation algorithm	**Deformation nodes**	**Deformation mapping**	**Parallel resampling**	**Ray-casting**	**Sampling mesh size (tetrahedra)**	**Sampling mesh memory**
ChainMail	13,200,705	0.4 ms	23.3 ms	8.2 ms	65,153,280	188.84 MB
Mass-Spring	2,000	5.7 ms	24.2 ms	7.5 ms	65,153,280	390.27 MB
Mass-Spring	512,000	7.4 ms	23.7 ms	8.3 ms	65,153,280	390.27 MB
FEM	125	5.9 ms	24.1 ms	7.3 ms	65,153,280	390.27 MB
FEM	3,200	5.9 ms	23.9 ms	7.2 ms	65,153,280	390.27 MB

4.2. Mass-Spring deformation

We have also tested our algorithm together with a dynamic simulation based on the mass-spring model, using tetrahedral meshes and explicit integration, parallelized on the GPU as proposed by Georgii et al. [25]. To implement the mapping from the mass-spring model to the implicit sampling mesh, as a preprocessing step we identify for each implicit node the mass-spring tetrahedron that contains it as well as its barycentric coordinates in the tetrahedron. At runtime we simply perform a barycentric combination of mass-spring node positions, which is trivially parallelized on the GPU.

We have also integrated simple cutting operations on the mass-spring model, separating adjacent tetrahedra by their shared face. To apply the cutting operations on the volume dataset, we follow the same approach as for the ChainMail model described above, using a bitfield of active tetrahedra.

Fig. 1 shows an example of a heart simulated with the mass-spring model that is deformed, cut, and resampled using our approach. With a volume dataset of 12.4 million voxels and a mass-spring model of 2.5 million tetrahedra, full volume deformation takes only 45.7 ms per frame. Dynamic deformations using explicit integration take 19 ms, deformation mapping takes 5.3 ms, and actual resampling takes 21.4 ms. Cuts applied to the model are computed in less than 50 ms.

4.3. FEM deformation

Finally, we have tested our algorithm with a corotational finite element method (FEM) [26], using a quasi-static solver with tetrahedral elements.

The deformation field is mapped to the sampling mesh using the exact same approach as for the mass-spring model described above. Fig. 5. shows an interactive deformation of a foot.

4.4. Performance analysis

We have carried out several tests to analyze the performance and scalability of our algorithm. The factors that we analyze are: the deformation method and its resolution, the resolution of the input volume dataset, the application of large deformations, and the size of the viewport. We also analyze preprocessing and memory costs.

4.4.1. Deformation methods and their resolution

We have compared performance on the same input volume dataset for the three deformation methods listed earlier in this section. We have used the knee MRI dataset shown on the top row of Fig. 6, with a resolution of $189 \times 305 \times 229 = 13,200,705$ voxels (leading to a sampling mesh of approximately 65 million tetrahedra). We have tested the ChainMail model at the same resolution as the volume, and the Mass-Spring and FEM models on two different resolutions each. Table 1 reports the resolutions of the deformation models, the time spent on mapping the deformation to the sampling mesh, and the time spent on actual resampling. Timings were averaged over several runs of the algorithm.

As the results indicate, the cost of deformation mapping is notably lower than the cost of resampling. Furthermore, the cost of resampling is practically independent of the deformation model and its underlying resolution. This result was expected, as our sampling mesh succeeds to decouple deformation mapping from resampling. The cost of deformation mapping is also fairly insensitive to the resolution of the deformation mesh.

With the ChainMail model, deformation mapping is negligible. With the Mass-Spring and FEM models it grows slightly with the resolution of the deformation model, but even with a Mass-Spring model with over half a million nodes the cost remains low. It is important to note that we used much coarser meshes with the quasi-static FEM solver running on the CPU because with high-resolution meshes the actual deformation solver becomes the bottleneck. This was not the case with our Mass-Spring solver because of explicit time integration.

For the case of tetrahedral deformation meshes (either with the Mass-Spring or FEM models), we also performed a more extensive scalability analysis as a function of mesh resolution. And we compared the results of our algorithm to the results of the rasterization algorithm by Gascón et al. [4]. For this purpose, we used the foot dataset shown in Fig. 5. Fig. 7 shows scalability plots with our method and with the method of Gascón et al. Ours is

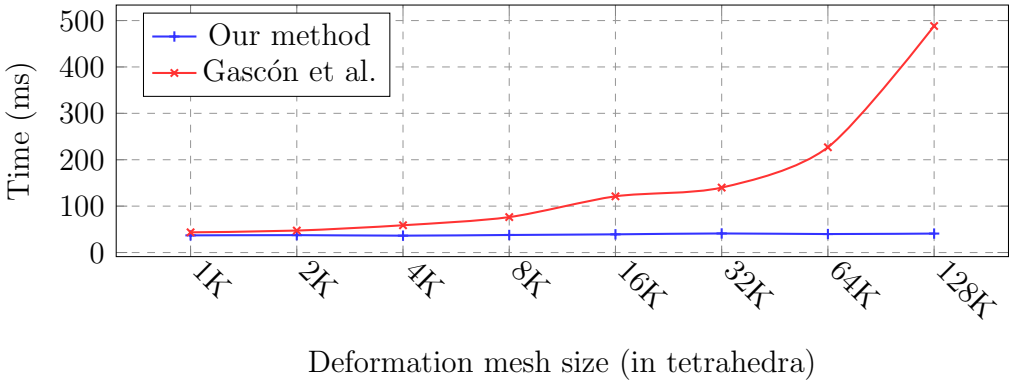

Figure 7: Scalability comparison of our method and the one by Gascón et al. [4] w.r.t. the resolution of the deformation mesh.

Table 2: Performance results of the large deformation test shown in Fig. 9. The table show resampling times for the five proposed scenarios, samples per tetrahedron (average, minimum, and maximum), resampling throughput measured as samples per millisecond, grid size, and required memory.

Deformation	**Resampling time**	**Average samples per tetrahedron (min, max)**	**Samples per millisecond**	**Output grid size**	**Output grid memory**
Undeformed	14.6 ms	1.07 (1, 3)	2,670,286	$256 \times 256 \times 113$	14.2 MB
Local deformation	14.8 ms	1.08 (1, 12)	2,667,913	$256 \times 256 \times 113$	14.2 MB
Rotated	23.6 ms	1.83 (2, 6)	2,821,122	$360 \times 372 \times 216$	55.2 MB
Spatially varying scaling	42.7 ms	3.25 (1, 8)	2,769,767	$384 \times 384 \times 170$	47.8 MB
Scaling 1.5x	51.2 ms	4.21 (3, 5)	2,997,839	$384 \times 384 \times 170$	47.8 MB

fairly independent of mesh resolution, while theirs is largely penalized under high-resolution meshes. The reason is that their algorithm uses indirection mechanisms that depend on the deformation mesh during resampling. Ours, instead, takes advantage of the implicit sampling mesh to decouple deformation mapping and resampling, avoiding costly indirections.

4.4.2. Volume resolution

In order to study the scalability of our resampling algorithm w.r.t. the resolution of the input volume dataset, we have measured the resampling time for several datasets. For this purpose, we have generated synthetically homogeneous data cubes of varying size. We have tested the three deformation methods on each dataset, using a mesh with 512,000 nodes for the Mass-Spring model and a mesh with 3,200 nodes for the FEM simulation.

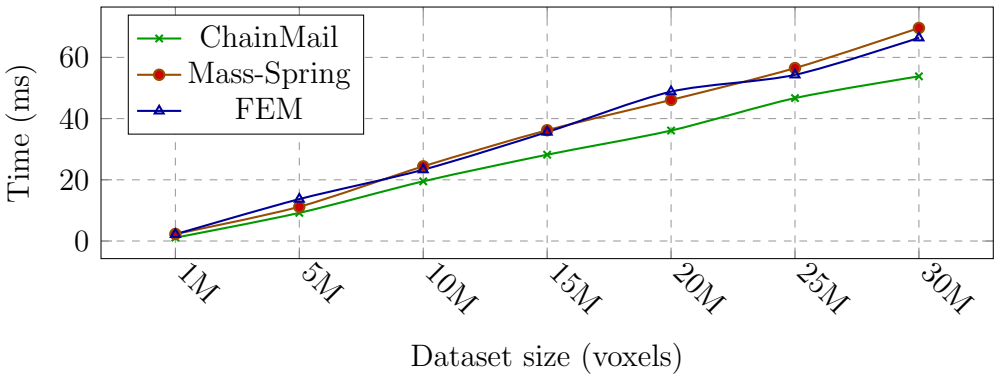

Figure 8: Scalability of our resampling pipeline w.r.t. the size of the input volume dataset. The three deformation methods have been applied to each tested dataset.

As shown in Fig. 8, our resampling pipeline (including times of both deformation mapping and resampling) exhibits a linear growth with respect to the size of the input dataset for the three tested deformation methods. Notice that when coupled with the ChainMail algorithm, the growth rate is slightly lower due to the simpler mapping scheme.

4.4.3. Performance under large deformations

We have conducted another experiment to analyze the behavior of our resampling algorithm when large deformations are applied to the model. For this purpose, we have used a head dataset consisting of $256 \times 256 \times 113$ voxels (Fig. 9.a)), and we have analyzed different deformed scenarios with respect to the undeformed configuration. The first scenario comprises a localized load on the nose (Fig. 9.b). The second scenario consists of a rotation of 45 degrees on each axis to maximize the misalignment of the sampling grid and the output grid (Fig. 9.c), increasing the size of the axis-aligned bounding box of the dataset by a factor of 3.9. The third scenario consists of a spatially varying scaled model, using a scaling factor going from 1.0 at the bottom to 1.5 at the top of the head (Fig. 9.d). The last scenario consists of a uniform scaling of the model by a factor of 1.5 (Fig. 9.e). In the last two cases, the volume of the axis-aligned bounding box of the dataset grows by a factor of 3.375.

Table 2 analyzes the cost of resampling for this experiment. Note that the amount of deformation affects only the cost of resampling, and the costs of deformation mapping and volume rendering are equal in all five cases. The

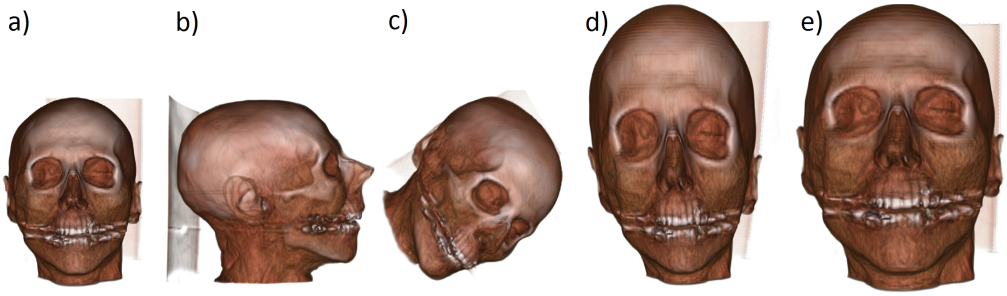

Figure 9: a) A head dataset consisting of 7.4 million voxels undergoes different large deformations. The model is b) deformed by a localized load on the nose, c) rotated 45 degrees on each axis, d) deformed using spatially varying scaling, and e) deformed using uniform scaling with a factor of 1.5.

localized load produces the highest peak in maximum samples per element for the elements affected but, due to its local scope, the performance of the algorithm is barely affected. The rotated configuration maximizes the misalignment of the sampling and the output grids, increasing the samples tested per element, and the resampling time grows by a factor of 1.61.

As expected, for the uniform scaling, the cost grows roughly linearly with the output volume. Both under uniform and spatially varying scaling, the size of the dataset grows from 7.4 million to 25.1 million voxels, but under spatially varying scaling the deformed model does not cover the entire grid. For this reason, it yields a slightly lower resampling cost. It is particularly interesting to analyze the throughput of the resampling algorithm, measured in terms of the grid samples handled per millisecond, and the uniform scaling case yields a higher throughput. The reason is that, under spatially varying scaling, the deformed tetrahedra do not have equal volumes. These size differences, together with the misalignment of the sampling and the output grids produced by the inhomogeneous scaling, lead to imbalanced workloads for the GPU threads, reducing the degree of parallelism. Notice how, for this example, the number of samples per tetrahedron may vary by a factor of 8. We can conclude that the performance of our method may become suboptimal under an extreme imbalance in the sizes of deformed tetrahedra, due to reduced parallelism.

4.4.4. Viewport size

Thanks to the decoupling of the resampling and the actual visualization, in our algorithm, the size of the viewport affects only the performance of visualization, not the resampling step. This is different from the behavior of unstructured volume rendering methods, where the size of the viewport affects the cost of all major steps, as analyzed by Okuyan et al. [17]. In addition, unstructured volume rendering methods pay a performance penalty when they use both high-resolution meshes and large viewports. This is not the case with our algorithm, because mesh resolution and viewport size affect the cost of disjoint steps.

In most of our experiments, we have used a viewport of 700×700 pixels, with 500 samples per ray. Under this resolution, the cost of ray casting was of 7.4 ms per frame on average, and went up to 24.1 ms per frame on average with the addition of on-the-fly gradient-based lighting. With a viewport of 1920×1080 pixels, the average cost per frame of ray casting was 40.3 ms, and with gradient-based lighting it rose to 107.7 ms. All the images in the paper were generated using gradient-based lighting.

4.4.5. Preprocessing

Although less relevant than the runtime cost, the preprocessing cost of our method is low. It grows linearly with both the size of the volume dataset and the resolution of the deformation method. In the most demanding scenario we have tested, a volume with 30 million voxels and a Mass-Spring mesh with 512,000 nodes, the preprocessing step took less than 15 seconds.

4.4.6. Memory

Last, our method is limited by GPU memory, as memory requirements grow linearly with the size of the dataset. In single precision, each voxel requires a total of up to 31 bytes: 2 to store its data value, 12 for its deformed position, 16 for indices and barycentric weights of deformation nodes to implement the deformation mapping, and 1 to flag topological changes. All in all, our algorithm requires 29.5 MB of memory per million voxels in the original dataset. For the ChainMail algorithm, the memory requirements are lower, as shown in Table 1, because there is no need to store deformation mapping information.

The GPU memory required by the output regular grid is comparatively lower, with only 2 bytes per voxel in our examples. This amounts to roughly 1.9 MB per million voxels.

If the memory limit is met and a higher visual detail is desired, a higher resolution volumetric model could be mapped to the sampling mesh by applying 3D texturing to its tetrahedra, as in [4], instead of using direct interpolation of the data values stored at their vertices.

5. Conclusions and future work

We have presented an algorithm to interactively resample deformable volumetric models onto a regular grid, which can be fed as input to standard direct volume rendering techniques. Our algorithm relies on an implicit sampling mesh, making the resampling process independent of the underlying deformation method.

This independence has been demonstrated in our experiments, and it grants two major advantages over previous approaches: First, a great variety of deformation methods can be coupled with our algorithm by means of a deformation mapping scheme. We have demonstrated the coupling with three deformation methods in this paper. Second, the cost of the resampling step is independent of the deformation method and its resolution, thus allowing the interactive visualization of datasets deformed using dense meshes.

All the stages of our implementation run in parallel using the GPU, and the execution time scales linearly with respect to the size of the input volume dataset. However, the amount of required dedicated memory also scales linearly with respect to the size of the dataset, hence for very large datasets it is possible to reach the memory limits of commodity GPUs.

Our algorithm admits several lines of future work to further enhance its performance and accuracy. They include the use of our regular sampling mesh at medium-high resolution combined with 3D texture mapping, adaptive resampling only in regions where deformation exceeds a threshold, more accurate handling of topological changes, or integration with advanced illumination techniques.

References

[1] Georgii J, Westermann R. A generic and scalable pipeline for gpu tetrahedral grid rendering. Visualization and Computer Graphics, IEEE Transactions on 2006;12(5):1345–52.

[2] Correa CD, Hero R, Ma KL. A comparison of gradient estimation methods for volume rendering on unstructured meshes. Visualization and Computer Graphics, IEEE Transactions on 2011;17(3):305–19.

[3] Schulze F, Bühler K, Hadwiger M. Interactive deformation and visualization of large volume datasets. In: GRAPP (AS/IE). Citeseer; 2007, p. 39–46.

[4] Gascon J, Espadero JM, Perez AG, Torres R, Otaduy MA. Fast deformation of volume data using tetrahedral mesh rasterization. In: Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation. ACM; 2013, p. 181–5.

[5] Allard J, Courtecuisse H, Faure F. Implicit fem solver on gpu for interactive deformation simulation. GPU Computing Gems Jade Edition 2011;:281–94.

[6] Etheredge C. A parallel mass-spring model for soft tissue simulation with haptic rendering in cuda. In: 15th Twente Student Conference. 2011,.

[7] Kharevych L, Mullen P, Owhadi H, Desbrun M. Numerical coarsening of inhomogeneous elastic materials. ACM Transactions on Graphics (TOG) 2009;28(3):51.

[8] Torres R, Espadero JM, Calvo FA, Otaduy MA. Interactive deformation of heterogeneous volume data. In: Biomedical Simulation. Springer; 2014, p. 131–40.

[9] Nesme M, Kry PG, Jeřábková L, Faure F. Preserving topology and elasticity for embedded deformable models. ACM Transactions on Graphics (TOG) 2009;28(3):52.

[10] Fortmeier D, Mastmeyer A, Handels H. Image-based palpation simulation with soft tissue deformations using chainmail on the gpu. In: Bildverarbeitung für die Medizin 2013. Springer; 2013, p. 140–5.

[11] Rezk-Salama C, Scheuering M, Soza G, Greiner G. Fast volumetric deformation on general purpose hardware. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware. ACM; 2001, p. 17–24.

[12] Westermann R, Rezk-Salama C. Real-time volume deformations. In: Computer Graphics Forum; vol. 20. Wiley Online Library; 2001, p. 443–51.

[13] Chen H, Hesser J, Männer R. Ray casting free-form deformed-volume objects. The Journal of Visualization and Computer Animation 2003;14(2):61–72.

[14] Chen M, Silver D, Winter AS, Singh V, Cornea N. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In: Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics. ACM; 2003, p. 35–44.

[15] Correa C, Silver D, Chen M. Feature aligned volume manipulation for illustration and visualization. Visualization and Computer Graphics, IEEE Transactions on 2006;12(5):1069–76.

[16] Miranda FM, Celes W. Volume rendering of unstructured hexahedral meshes. The Visual Computer 2012;28(10):1005–14.

[17] Okuyan E, Güdükbay U. Direct volume rendering of unstructured tetrahedral meshes using cuda and openmp. The Journal of Supercomputing 2014;67(2):324–44.

[18] Nakao M, Minato K. Physics-based interactive volume manipulation for sharing surgical process. Information Technology in Biomedicine, IEEE Transactions on 2010;14(3):809–16.

[19] Hacon D, Tomei C. Tetrahedral decompositions of hexahedral meshes. European Journal of Combinatorics 1989;10(5):435–43.

[20] Rössler F, Wolff T, Ertl T. Direct gpu-based volume deformation. Proceedings of Curac 2008;:65–8.

[21] Gibson SF. 3d chainmail: a fast algorithm for deforming volumetric objects. In: Proceedings of the 1997 symposium on Interactive 3D graphics. ACM; 1997, p. 149–ff.

[22] Schill MA, Gibson SF, Bender HJ, Männer R. Biomechanical simulation of the vitreous humor in the eye using an enhanced chainmail algorithm. In: Medical Image Computing and Computer-Assisted Intervention, MICCAI'98. Springer; 1998, p. 679–87.

[23] Frisken-Gibson SF. Using linked volumes to model object collisions, deformation, cutting, carving, and joining. Visualization and Computer Graphics, IEEE Transactions on 1999;5(4):333–48.

[24] Wu J, Westermann R, Dick C. A survey of physically based simulation of cuts in deformable bodies. In: Computer Graphics Forum (to appear). Wiley Online Library; 2015,.

[25] Georgii J, Echtler F, Westermann R. Interactive simulation of deformable bodies on gpus. In: SimVis. 2005, p. 247–58.

[26] Müller M, Gross M. Interactive virtual materials. In: Proceedings of Graphics Interface 2004. Canadian Human-Computer Communications Society; 2004, p. 239–46.

4.2 Spatial opacity maps for direct volume rendering of regions of interest

- **A. Rodríguez** and A. León (2016). "Spatial Opacity Maps for Direct Volume Rendering of Regions of Interest". *Proceedings of Spanish Computer Graphics Conference* (CEIG 2016).
  - Status: Published [110]

CEIG - Spanish Computer Graphics Conference (2016)
Alejandro García-Alonso and Belen Masia (Editors)

Spatial opacity maps for direct volume rendering of regions of interest

A. Rodríguez-Aguilera[1] and A. León[1]

[1]Virtual reality lab, University of Granada, Spain

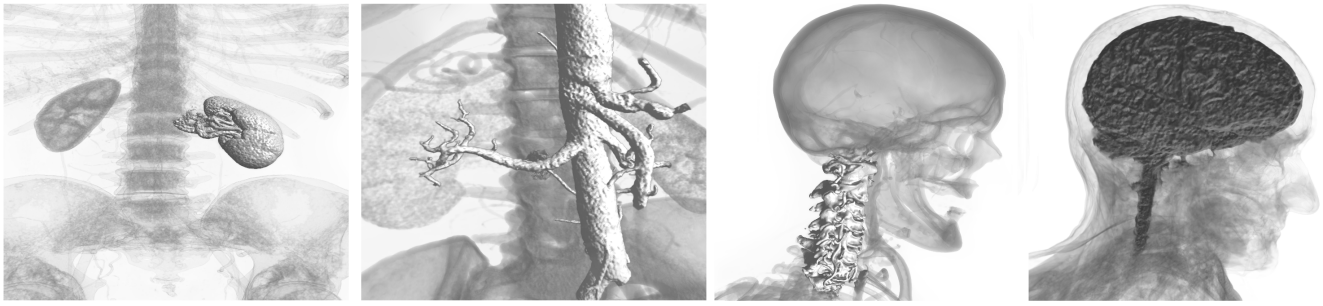

Figure 1: *A single point selection on the 2D slices of a medical dataset is the sole input for our method. Our spatial opacity maps highlight the selected feature and a simple automatic transfer function generation algorithm reveals relevant contextual information.*

Abstract

Despite the mature state of the volume rendering field, its adoption in medical applications is hindered by its complex parametrization and control, and 2D slice based tools are still preferred for clinical workflows.
In this paper, we introduce the concept of spatial opacity maps as an interactive tool for exploring volumetric data focusing on the rendering of features of interest. In a region growing fashion, the maps are dynamically created from a user input on the 2D slices, taking into account not only the density values of the structure but also the topology. Using this approach, an inexperienced user is able to generate meaningful 3D renderings with no need to tweak non-intuitive visualization parameters. The spatial opacity maps are independent of the current visualization parameters and they can be easily plugged into the volume rendering integral and combined with other approaches for region of interest (ROI) visualization. We combine our approach with a simple automatic transfer function generation algorithm to improve the visualization of the contextual data.

1. Introduction

During the last decades, the volume rendering pipeline has been gradually improved to a great level of quality and maturity. State-of-the-art techniques for volume rendering [EHK*06, BRGIG*14] allow to interactively generate 3D visualizations of volumetric data including advanced illumination, material and shadowing techniques that provide rich information regarding depth and spatial relationships in the data. However, as stated in recent works [BKKG08, WVFH12], its wide adoption for clinical applications is very limited, mainly due to the excessive level of expertise required to control the visualization parameters that, in the end lead the clinicians to work with classical, 2D based approaches, which are closer to their training knowledge and are thus more practical.

Despite this fact, volume rendering is undeniably useful in many contexts as it is able to provide additional information regarding spatial properties of features of interest and contextualized relations between different features in the data, which provide meaningful information. Therefore, during the last years, there is a great effort to minimize the complexity of controlling the visualization of features of interest.

One way to achieve this goal that has proven useful is to combine the classic 2D pipeline with the 3D volume rendering [WVFH12,

KBKK07], connecting them unidirectionally or bidirectionally in order to achieve a complementary control of both metaphors.

In this work, we present the spatial opacity map as a novel tool to easily highlight features of interest in volumetric data using the slice-based selection metaphor as sole input for the method. The opacity maps account for the spatial topology of the selected feature, which allows to preserve context information while focusing on the visualization of the region of interest, and they are seamlessly integrated into the volume rendering integral to highlight the feature of interest with no need to modify visualization parameters. Moreover, since the opacity maps are defined as a separated, easy-to-plug component for the volume rendering pipeline, they can be easily combined with existing volume exploration techniques and state-of-the-art rendering techniques.

The opacity maps are interactively generated through a parallel GPU computation on the volume data as we explain in this work. In our examples we show that even in a complete absence of manual or assisted preprocessing, they are able to easily provide a meaningful visualization regarding a selected structure of interest, but they can also benefit from complimentary techniques, such as automatic transfer function generation methods, as the one proposed in this work.

2. Related work

The study of techniques to ease the 3D exploration of volume data and the highlighting of regions of interest (ROIs) has spanned many approaches focusing on different parts of the control interface and volume rendering pipeline.

A large body of approaches provides new tools and models for the volume rendering stage. Viola et al. [VKG04] proposed an importance-driven approach to perform focus+context rendering by using pre-segmented object information to compose the final image. Correa et al. [CSC06] proposed a set of manipulation operators applied on the rendering stage, also relying on pre-segmented data. Bruckner et al. [BGKG06] proposed a rendering model, providing a substitute for the conventional clipping techniques by preserving some of the context information during the interaction. Since the most time-consuming manual task during 3D volume rendering is the search for a good transfer function, many works propose interactive techniques to ease this process. Good examples are the work of Kniss et al. [KKH01], where they proposed several manipulation widgets for an assisted specification of multidimensional transfer functions, and the work of Correa and Ma [CM11], introducing the concept of visibility-driven transfer functions as a semi-automatic method for easing the process of generating meaningful transfer functions to maximize the visibility of regions of interest. Roettger et al. [RBS05] proposed a feature selection tool introducing the spatialized transfer functions, which are automatically generated after classifying the different regions of the model to provide different colors to the different identified feature classes and allows the user to highlight the identified classes by selecting them on the transfer function. The main drawback of this approach is the non-intuitive selection mechanism, since the generated spatialized transfer functions lack a real physical or visual meaning w.r.t. the real feature and thus the selection process becomes difficult. Moreover, the method requires an initial manual setup of the *feature radius* parameter and similar features may become merged into the same class, thus impeding the selection of the individual features. These and other approaches provide very powerful tools for the direct volume rendering stage, but they either rely on pre-processed segmentation or assisted trial and error exploration procedures directly over the 3D visualization.

Other works rely on the 2D selection of features using the 2D slices of the model to generate 3D visualizations, exploiting the fact that this is the standard exploration approach used in clinical workflows. Kohlmann et al. [KBKK07, KBKG08] proposed a system to generate meaningful volumetric views by 2D-selecting features and automatically generating the camera position, zoom factor and clipping plane. However, their approach also relies in a good pre-defined transfer function for the model.

Our approach is also based on the selection of features on the 2D slices to generate the spatial opacity maps, however, it also avoids other inputs such as a manually tuned transfer function or any other manual pre-processing of the data. Sherbondy et al. [SHN03] proposed a similar interactive 2D-based exploration approach by obtaining the selected feature using a region growing approach. However, they apply the obtained binary segmentation as a mask to the rendering, impeding the visualization of any kind of contextual information aside from the highlighted feature. Our spatial opacity maps can be seamlessly plugged into the standard volume rendering pipeline, thus they can take advantage of any defined transfer function to display contextual information as we demonstrate in our results, and they can also be combined with existing focus+context techniques, such as the ones previously referred in this section. Huang and Ma [HM03] also exploit the region growing approach by generating a transfer function based on the selected feature, but generally, a lack of context information is observed since the transfer function is defined to only reveal the opacity values present in the feature, and undesired features with similar opacities to the selected feature may occlude the region of interest. Our opacity maps avoid these two issues because they highlight only the selected feature instead of all the features with similar opacity and, as already mentioned, their compatibility with the standard volume rendering pipeline offers the possibility to expose contextual data using different focus+context techniques.

3. Spatial opacity map

The principle of our spatial opacity maps and their generation procedure are founded on the well known seeded region growing segmentation approach [AB94]. We define a homogeneity criterion between neighboring voxels based on their properties regarding a seed voxel. In the original region growing approach this criterion is used to determine the membership of the new voxel to the selected region attending to the value of the voxels, generating a binary membership classification as output. Instead, we apply it to compute an opacity value for the new voxel based on the opacity value of its neighbor and the homogeneity factor, thus the output of our approach is a 3D scalar field of opacity values over the volume.

The generation process starts when the user selects a point in a 2D slice. The voxel containing the point is considered as the *seed*

voxel, its value on the opacity field is set to the maximum opacity value and the standard deviation of voxel values regarding its 1-ring neighborhood is computed and stored. After that, the opacity field is iteratively expanded by updating the 6 direct neighbors of the voxels that had their opacity modified in the previous iteration (in the first iteration, the 6 direct neighbors of the seed voxel are candidates for updating).

Candidate voxels are updated using an opacity extinction function. Let d_s be the density value of the *seed voxel* and let σ_s be its stored standard deviation, let v be a candidate voxel with density value d_v and current opacity o_v that needs to be updated due to a previously modified neighboring voxel w with current opacity o_w, a new candidate opacity is computed as

$$o_v^* = o_w - E_v, \tag{1}$$

with

$$E_v = \frac{|d_s - d_v| - \sigma_s}{\lambda \sigma_s}. \tag{2}$$

E_v automatically adapts the extinction behavior to the presence of noise in the data of the selected feature, and can be further tuned by setting the λ value. We have found however that with $\lambda = 30$ we obtain good results in all the scenarios tested in this paper, thus we have simply set it by default and the sole input in all the examples shown in this work is a 2D selection in the slices.

Since a voxel can be visited several times by several neighbors, its opacity value is only updated if $o_v^* > o_v$. If that is the case, the voxel opacity is updated, thus its neighbors will require an update on the next iteration.

3.1. Parallel implementation

The expansion process of a spatial opacity map is easily computed using the GPU by processing all the voxels requiring an update in parallel in each iteration. After setting the seed voxel, for each expansion iteration a kernel is invoked, launching one thread per voxel. The algorithm is shown in Algorithm 1.

Algorithm 1 Opacity expansion kernel

```
Input: Voxel v
o_v ← v.opacity
o_w ← 0
N ← v.neighbors
for each n in N do
    o_w ← max(o_w, n.opacity)
end for
E_v ←Equation 2
o_v^* ← o_v − E_v
o_v ← max(o_v, o_v^*)
```

Each thread loads the current opacity value of its assigned voxel and the highest opacity value of its neighbors. Then, the candidate opacity is computed using (1). If it is higher than the voxel's current opacity, it is updated.

The expansion reaches elements directly connected to the last influenced voxels, thus on the n^{th} iteration the kernel is launched on a $(n+1)^3$ region centered on the seed voxel, ensuring its reach to all the candidate voxels.

3.2. The volume integral

In order to visualize a generated spatial opacity map, we plug it into the volume rendering integral.

Let us first recall the base volume emission-absorption model [EHK*06]. The radiant energy C reaching the eye from a given direction is defined as an integral along the direction ray

$$C = \int_0^\infty c(t) \cdot e^{-\int_0^t \kappa(\hat{t}) d\hat{t}} dt,$$

with $c(s)$ and $\kappa(s)$ the emission and absorption coefficients, defined as scalar fields over the volume.

In practice, a numerical approximation of the integral is used, first approximating the emissions and absorptions along the ray as $C_i = c(i \cdot \Delta t)\Delta t$ and $A_i = 1 - e^{-\kappa(i \cdot \Delta t)\Delta t}$, now in the forms of colors and opacities respectively, to finally yield

$$\tilde{C} = \sum_{i=0}^{n} C_i \prod_{j=0}^{i-1} (1 - A_j), \tag{3}$$

with n the number of samples.

Our opacity maps are in fact scalar fields $o(s)$ that can be integrated along with the existing volume properties. Thus, we redefine the opacity component as $A_i^* = (1 - e^{-\kappa(i \cdot \Delta t)\Delta t}) o(i)$.

Substituting this new opacity component in (3) yields a modified volume rendering equation

$$\tilde{C} = \sum_{i=0}^{n} C_i \prod_{j=0}^{i-1} (1 - A_j^*).$$

With this formulation, we seamlessly integrate the opacity maps within the standard volume rendering pipeline, and thus we are able to apply state-of-the-art rendering along with other existing approaches applied to the standard pipeline.

4. Interactive visualization

The scalar field of a spatial opacity map can be configured to produce opacity values within the range $[o_{min}, o_{max}]$, with $0 \leq o_{min} < o_{max} \leq 1$. Thus when a seed voxel is selected, its opacity value is set to o_{max} and the opacity value of the rest of the voxels is set to o_{min} before the iterative expansion starts.

Opacity maps encompass enough information to produce meaningful renderings without modifying other visual parameters, but their true potential shows up when they are combined with other existing tools, both automatic and assisted. Moreover, the interactive control of the expansion of the selected feature also proves useful for further understanding of the anatomy, since it allows to gradually explore complex topologies, which is of interest for features such as vascular networks. In this section we explore these different options.

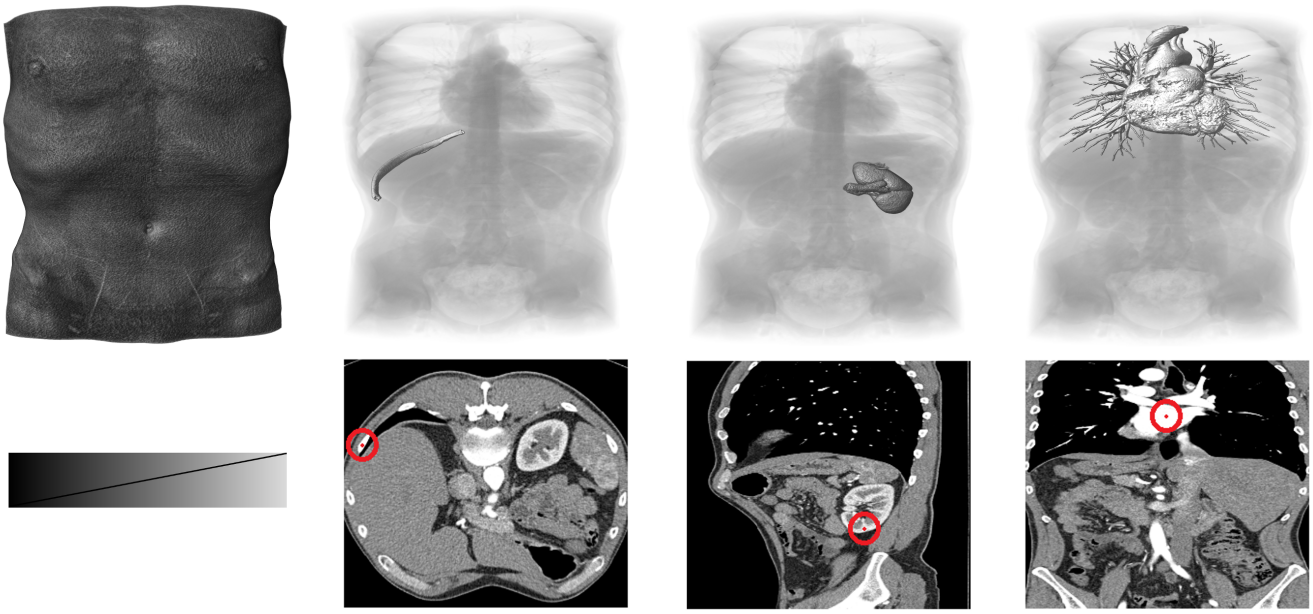

Figure 2: *The initial volume rendering state using the default transfer function (leftmost column) is combined with our opacity maps when the user selects and expands the desired feature. The final visualization is shown in the top row, and the selection of the data slices is shown below each case.*

4.1. Standalone visualization

The opacity maps make a clear distinction between ROI and contextual information, and thus a standalone visualization only accounting for this information should emphasize this difference. We test this by setting $o_{min} = 0.005$ (highly transparent) and $o_{min} = 1$ (opaque), so that the selected feature has the maximum visibility while preserving the contextual data, and we leave the default transfer function (Fig. 2, bottom left) untouched. Of course, a context-free visualization of the feature of interest is also easily obtained by setting $o_{min} = 0$.

Fig. 2 shows several examples of the achieved renderings, obtained simply by selecting the desired feature in the 2D slices. Several features can be highlighted at the same time by a simple combination of their opacity maps, as shown in Fig. 3.

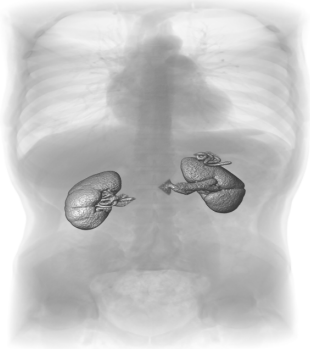

Figure 3: *Several opacity maps can be combined before performing the rendering stage. In this case, the two kidneys of a patient are highlighted.*

4.2. Enhanced contextual visualization

Although the standalone visualization allows a clear distinction of the ROI, all the contextual information is displayed homogeneously and thus makes it difficult to distinguish certain relations between features.

To provide a better visualization of the contextual information while maintaining the parameter-free interaction, an automatic transfer function generation method can be applied.

We exploit the data obtained from the seed voxel to generate a simple transfer function to distinguish contextual data with similar density values to those of the selected feature, referred as *primary* context information, from contextual data with density values far from those of the selected feature, referred as *secondary* context information:

We compute a Gaussian function with the form

$$g(x) = e^{-\frac{(x-\mu_s)^2}{2\sigma_s^2}},$$

with μ_s the mean density value of the 1-ring neighborhood of the seed voxel.

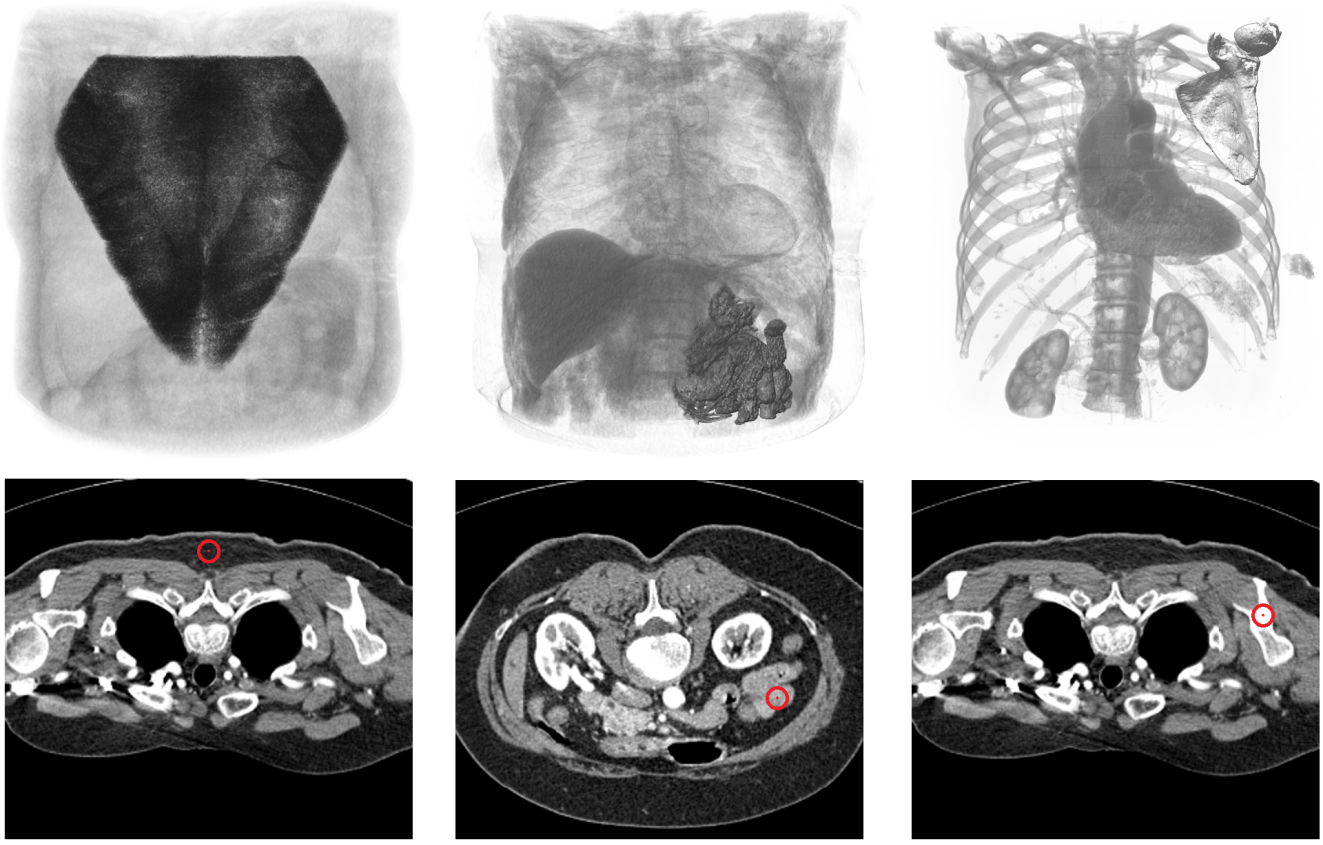

Figure 4: *From left to right, the fat layer of the back, the intestines and the scapula are selected. In each case, the transfer function is automatically updated to reveal related contextual structures.*

This function is then composed with the default transfer function $t(x)$ to produce a final transfer function with the form $f(x) = t(x) \cdot (a + b \cdot g(x))$, with a and b parameters that control the contrast between the *primary* and *secondary* context information. These parameters can be adjusted, ensuring $a \geq 0$, $b \geq 0$ and $a + b = 1$.

We combine our opacity maps with these automatic transfer functions by setting $a = 0.01$, $b = 0.99$, $o_{min} = 0.03$ and $o_{max} = 1$.

Fig. 4 shows the results using this combination on a CT scan. If the fat layer on the back is selected, the automatic transfer function hides the internal organs and the skeleton. If the intestine is selected instead, the transfer function automatically adapts to emphasize other internal organs while the intestine remains clearly highlighted by the opacity map. When the scapula is selected, it also becomes clearly highlighted, but in this case, the bones and other dense tissues stand out over the previous internal organs.

The proposed configuration also adapts to the different properties of the dataset, for instance, a CT scan with much lower contrast in Fig. 5. When the leg muscles, the kidney or the stent of the aorta are highlighted, the internal soft tissues or skeleton are properly provided as main contextual structures. Other results are shown in Fig. 1.

4.3. Interactive expansion

The expansion process of the opacity map can be performed automatically from the user-selected 2D point. However, enabling a direct control over the expansion process can be in fact regarded as an additional exploration tool that is specially useful when exploring features with a complex topology. Additionally, this active control allows the user to prevent the expansion from invading neighboring structures, which may happen if their density ranges overlap.

Therefore, we let the user to freely perform expansion steps at will, with the possibility to manipulate the 3D rendering at any intermediate point. This allows to better visualize the intricacies of complex topology, such as the vessels in a vascular network shown in Fig. 2 (rightmost column).

5. Results and discussion

We have implemented our method of parallel generation of the opacity maps using OpenCL and applied it using a variety of medical datasets. Along with the results shown throughout this work, we have applied our method to several datasets with different contrast and noise conditions to test the robustness of our algorithm.

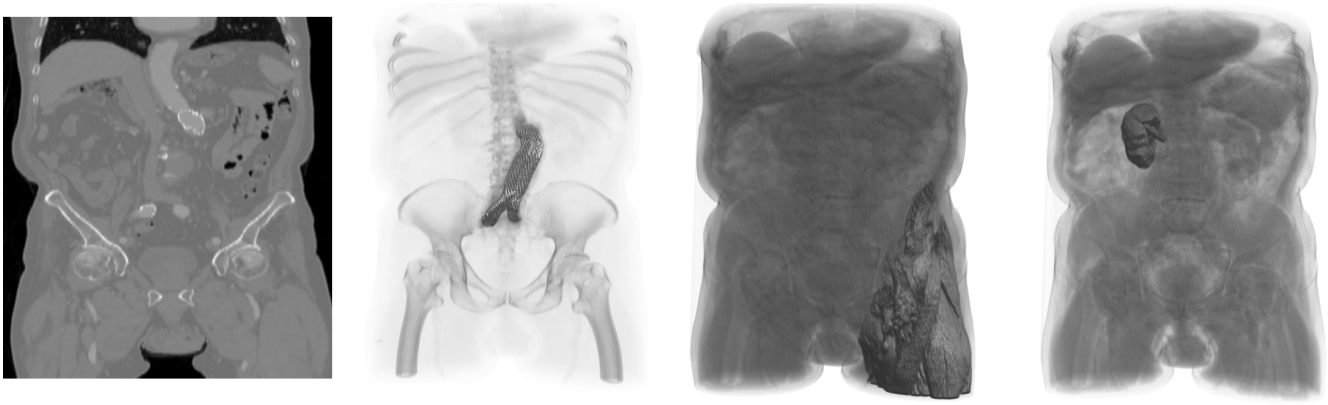

Figure 5: *In the presence of data with low contrast (leftmost), our method is also able to provide a clearly highlighted feature with contextual information. When the stent of the aorta is selected (left-center), the bone structures are revealed. If the leg muscles or the kidney are selected instead, the bones and stent are hidden, and other soft tissues are revealed.*

In this section we analyze and discuss the properties of our method to evaluate its behavior under different contrast and noise conditions. We have also conducted a performance test to measure the efficiency of our parallel implementation.

5.1. Method robustness

The main motivation for developing our method is to provide the user with an intuitive, parameter-free tool to explore medical datasets, and thus it should provide a proper response independently of the noise and contrast conditions of the data. For this reason, we have fixed the λ value for all the tested datasets (as explained in Sec. 3).

The datasets shown in Fig. 2 and Fig. 4 exhibit high contrast and a low amount of noise on the bone and vascular tissues. For these regions our method provides a very clear visualization of the selected features, which was expected since this is the ideal case for segmentation algorithms. The same outcome is observed for areas where low contrast and low amount of noise, such as the kidney and leg muscle areas shown in Fig. 5.

For datasets with a higher amount of noise and high contrast, such as the abdominal organs of the CT scan shown in Fig. 2, we also obtain a proper distinction of the desired features. For instance, we are able to clearly differentiate several parts of the kidney as shown in Fig. 6.

We find that our method fails for input datasets with low contrast and moderate or high noise. For instance we are unable to produce a clear view of the vascular network of the lungs in the dataset shown in Fig. 7. We believe that under these conditions, a region growing approach alone is not enough to capture features of interest, and other approaches should be explored.

We also observe malfunction of the method when the selected seed voxel belongs to a boundary between features, as seen in Fig. 8. We estimate the noise of the data through the computation of the standard deviation on the neighborhood around the seed voxel,

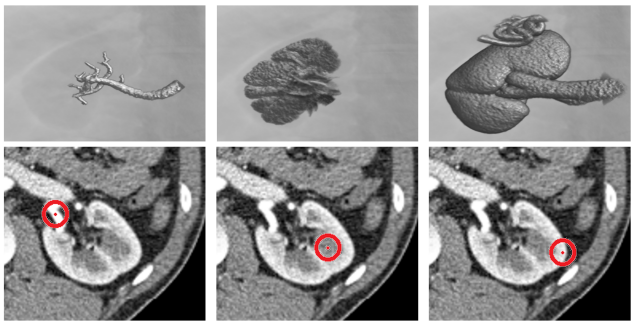

Figure 6: *From left to right, the renal vessels, the medulla and the cortex of the kidney can be highlighted by selecting the different structures in the 2D slices, shown below each case.*

thus we assume this neighborhood to belong to the same feature. This leads to a poor noise estimation for seed voxels located on boundary regions. A possible workaround for this issue was proposed by Huang and Ma [HM03], but we want to further explore solutions avoiding manual parameter tuning.

5.2. Performance

We have evaluated the performance of our algorithm using an Intel Core i5-3570 machine with 8 GB RAM equipped with an AMD Radeon R9-270X.

Due to the simple GPU scheduling mechanism proposed in Sec. 3.1, the performance of the algorithm is independent of the input dataset, and the computational burden of an expansion iteration depends only on the size of the reached cubic region.

Fig. 9 shows the computation time for an expansion iteration w.r.t. the cubic region size. Our implementation is able to provide

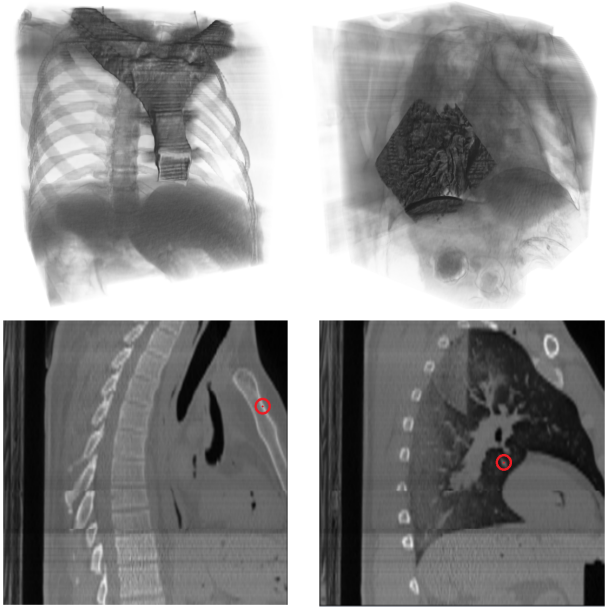

Figure 7: *In the presence of noise in data with low contrast, our algorithm is unable to capture the desired feature. The breastbone (left) is captured, but its boundary becomes blurred. In the case of the bronchi (right), neighboring structures are rapidly invaded during the expansion process.*

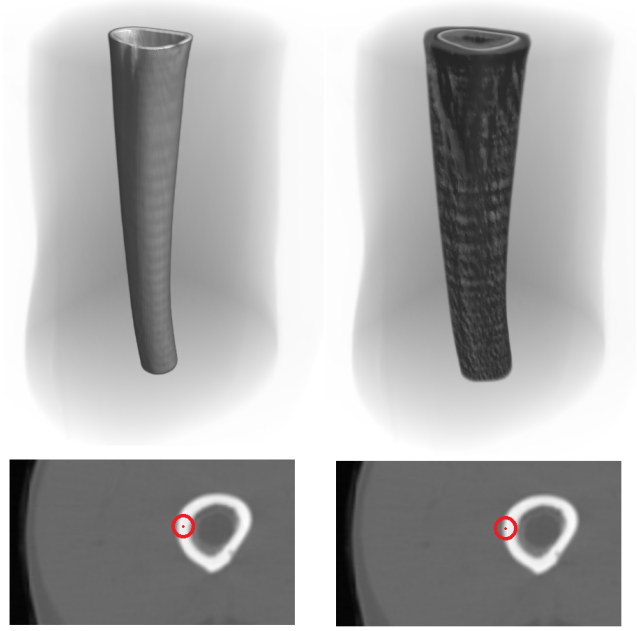

Figure 8: *Our method assumes an input point inside the feature of interest (left). If the selected point belongs to the boundary between structures, the change of structure is regarded as data noise and the method fails to capture the desired feature (right).*

real-time responses for small and medium size features and interactive responses for large features covering several million of voxels. For reference: the kidney in Fig. 2, bounded by a cubic region with 7.4 million voxels, required a total expansion time of 64 ms; the vascular network in Fig. 2, bounded by a cubic region with 89.3 million voxels, required a total expansion time of 714 ms; the stent in Fig. 5, bounded by a cubic region with 7.1 million voxels, required a total expansion time of 62 ms; the leg muscles in Fig. 5, bounded by a cubic region with 52.7 million voxels, required a total expansion time of 375 ms.

We have also measured the percentage of launched threads actually performing useful computation for all our examples and, although this amount varies depending on the topology of the selected feature, it ranges between a 0.7% and a 12%. This is due to the fact that the simplistic approach used for thread launching does not cope with the sparse nature of the expansion approach, since most of the launched threads perform computation on voxels that are not on the current frontier of the expanding feature and thus none of their neighbors had its opacity value updated.

Although our current implementation provides interactive responses, the performance would be greatly improved by using a sparse-aware scheduling approach ([RLAM15, Sæt13]).

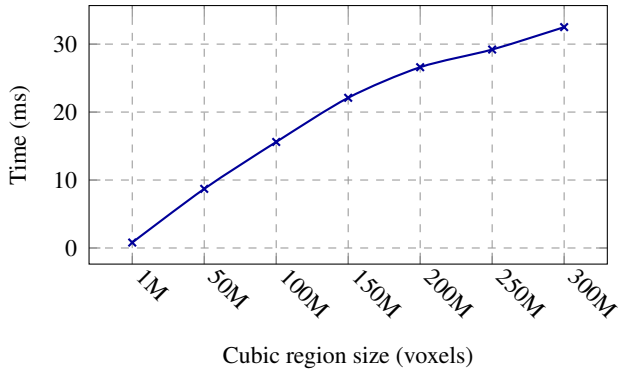

Figure 9: *Time required to perform an expansion iteration depending on the current size of the cubic region.*

6. Conclusion and future work

We have proposed the spatial opacity maps as a novel tool for interactive exploration of medical volume data. Our approach follows a parameter-free interaction paradigm and proves robust for many types of input datasets.

Although the spatial opacity maps as standalone tool are able to provide meaningful visualizations of the desired features, the display of the contextual information is rather poor, but we have shown

that improved visualizations can be achieved by combining our approach with other tools for medical image exploration. We have provided a method to seamlessly integrate the opacity maps into the standard volume rendering pipeline, and thus their combination with existing methods is achieved with almost no extra effort.

We have also proposed a simple automatic transfer function generation procedure, achieving a notable improvement in the visualization of contextual data, and we think that further combinations will lead to better visualizations while maintaining the parameter-free paradigm. For instance, we strongly believe that our method could be easily combined with the importance-driven volume rendering approach [VKG04] to serve as input for the latter and to improve the focus+context contrast, and with other automatic function generation methods, such as the one proposed by Zhou and Takatsuka [ZT09]. We would like to explore these and other combinations as future work.

The experiments carried out show that our implementation delivers interactive responses, although we will address the improvement of the GPU scheduling mechanism to account for the sparse nature of the computation in a way similar to the blocking method proposed in [RLAM15]. We believe that this will enable response times close to real-time even for large features.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the University of Granada, under the "Formación de Profesorado Universitario, Plan Propio de Investigación 2012" program. This work is also supported by the project TIN2014-60956-R of the Spanish Ministry of Economy and Competitiveness with FEDER funds. The datasets used in this work were obtained from The Volume Library (lgdv.cs.fau.de/External/vollib/) and the Osirix repository (osirix-viewer.com/datasets/).

References

[AB94] ADAMS R., BISCHOF L.: Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 16*, 6 (1994), 641–647. 2

[BGKG06] BRUCKNER S., GRIMM S., KANITSAR A., GROLLER M. E.: Illustrative context-preserving exploration of volume data. *Visualization and Computer Graphics, IEEE Transactions on 12*, 6 (2006), 1559–1569. 2

[BKKG08] BRUCKNER S., KOHLMANN P., KANITSAR A., GROLLER M. E.: Integrating volume visualization techniques into medical applications. In *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on* (2008), IEEE, pp. 820–823. 1

[BRGIG*14] BALSA RODRÍGUEZ M., GOBBETTI E., IGLESIAS GUITIÁN J., MAKHINYA M., MARTON F., PAJAROLA R., SUTER S. K.: State-of-the-art in compressed gpu-based direct volume rendering. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 77–100. 1

[CM11] CORREA C. D., MA K.-L.: Visibility histograms and visibility-driven transfer functions. *Visualization and Computer Graphics, IEEE Transactions on 17*, 2 (2011), 192–204. 2

[CSC06] CORREA C. D., SILVER D., CHEN M.: Feature aligned volume manipulation for illustration and visualization. *Visualization and Computer Graphics, IEEE Transactions on 12*, 5 (2006), 1069–1076. 2

[EHK*06] ENGEL K., HADWIGER M., KNISS J., REZK-SALAMA C., WEISKOPF D.: *Real-time volume graphics*. CRC Press, 2006. 1, 3

[HM03] HUANG R., MA K.-L.: Rgvis: Region growing based techniques for volume visualization. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on* (2003), IEEE, pp. 355–363. 2, 6

[KBKG08] KOHLMANN P., BRUCKNER S., KANITSAR A., GRÖLLER M. E.: Livesync++: Enhancements of an interaction metaphor. In *Proceedings of Graphics Interface 2008* (2008), Canadian Information Processing Society, pp. 81–88. 2

[KBKK07] KOHLMANN P., BRUCKNER S., KANITSAR A., KANITSAR A.: Livesync: Deformed viewing spheres for knowledge-based navigation. *Visualization and Computer Graphics, IEEE Transactions on 13*, 6 (2007), 1544–1551. 1, 2

[KKH01] KNISS J., KINDLMANN G., HANSEN C.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of the conference on Visualization'01* (2001), IEEE Computer Society, pp. 255–262. 2

[RBS05] ROETTGER S., BAUER M., STAMMINGER M.: Spatialized transfer functions. In *EuroVis* (2005), Citeseer, pp. 271–278. 2

[RLAM15] RODRÍGUEZ A., LEÓN A., ARROYO G., MANTAS J. M.: Sp-chainmail: a gpu-based sparse parallel chainmail algorithm for deforming medical volumes. *The Journal of Supercomputing 71*, 9 (2015), 3482–3499. 7, 8

[Sæt13] SÆTRA M. L.: Shallow water simulation on gpus for sparse domains. In *Numerical Mathematics and Advanced Applications 2011*. Springer, 2013, pp. 673–680. doi:10.1007/978-3-642-33134-3_71. 7

[SHN03] SHERBONDY A., HOUSTON M., NAPEL S.: Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *Visualization, 2003. VIS 2003. IEEE* (2003), IEEE, pp. 171–176. 2

[VKG04] VIOLA I., KANITSAR A., GROLLER M. E.: Importance-driven volume rendering. In *Proceedings of the conference on Visualization'04* (2004), IEEE Computer Society, pp. 139–146. 2, 8

[WVFH12] WIEBEL A., VOS F. M., FOERSTER D., HEGE H.-C.: Wysiwyp: what you see is what you pick. *Visualization and Computer Graphics, IEEE Transactions on 18*, 12 (2012), 2236–2244. 1

[ZT09] ZHOU J., TAKATSUKA M.: Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *Visualization and Computer Graphics, IEEE Transactions on 15*, 6 (2009), 1481–1488. 8

Chapter 5

Conclusions and Future Works

5.1 Conclusions

In this thesis we have presented a variety of contributions in the fields of deformable models, soft robotics, volume rendering and volume exploration. Our solutions are focused on interactive systems, both in medical and robot control environments. The limited time budget of these applications to provide feedback forces to sacrifice realism, in terms of the accuracy of the underlying simulation and/or the level of detail of the visual feedback, to meet the interactive performance requirements. Motivated by this limitation, we have focused our efforts on exploiting the parallel power of modern GPUs to improve the overall quality of the solutions and, along with our main contributions, we present several contributions on GPGPU computing.

The dissertation has been presented in the modality of "compendium" and has been structured in three blocks. Below we detail the specific conclusions and contributions related to each block.

5.2 Parallel ChainMail simulation of heterogeneous medical models

In Chapter 2, we have presented a novel parallel ChainMail algorithm by recasting the algorithm as a stencil computation problem. We have shown that this computing scheme allows to greatly increase the performance of the algorithm through a parallel computation of the propagation and relaxation stages on the GPU. We address the long lasting problem of efficient computation of heterogeneous ChainMail models by incorporating a timestamp-based mechanism to the propagation stage taking into account that the propagation speed of a deformation through a deformable body depends on the stiffness of the traversed material, including isotropic and anisotropic materials, and generalizing the energy minimization process in the relaxation stage, which models the elastic energy stored in the heterogeneous tissues based on the

Hooke's law. In terms of performance, our experiments show that our solution outperforms previous ChainMail algorithms by factors up to 20x, allowing interactive frame rates for models with up to several million elements.

To address the sparse nature of our stencil approach, we have proposed a blocking scheme for an efficient scheduling of the computation to the GPU, drastically reducing the unnecessary computation and increasing the performance of the algorithm. An experimental analysis showed that the blocking scheme exhibits good portability and scalability and leads to speedups of a factor of 7x with respect to a naive scheduling approach.

To sum up, the proposed algorithm addresses several limitations of the original algorithm by increasing its performance, efficiently handling heterogeneous materials and allowing for several deformations to propagate simultaneously through the model, together with other minor improvements, and we believe that solutions based on ChainMail models can build on these features to enhance and widen their applicability. The proposed blocking scheme is demonstrated to increase the performance and it is applicable to general stencil computation problems suffering from sparse computation.

5.2.1 Simulation-based control of soft robots

In the emerging field of motion planning of soft robots, the complex behavior of their compliant bodies, actuators and other involved components must be modeled in an efficient yet accurate fashion to allow for interactive simulation solutions to compute the required actuation in a few milliseconds. In Chapter 3 we have presented a novel GPU parallel method for an efficient estimation of fluid weight distribution inside dynamic cavities together with a novel modeling of the dynamic behavior of hydraulic constraints. We have integrated our solution within the SOFA soft robots framework [114], providing a system for online motion planning in the task-space for hydraulic actuated soft robots. An experimental analysis of our complete solution shows accurate estimations of both the fluid weight distribution and the dynamic model, and shows promising results for interactive control of fabricated soft robots.

We also present a novel general, simple, parallel leveraging algorithm that inexpensively alleviates the computation of irregular-parallel workload problems, and we have shown its application to our fluid weight estimation algorithm.

5.2.2 Interactive medical visualization

In Chapter 4 we have addressed two computationally demanding problems for interactive volume rendering in medical applications: the rendering of deformable volumetric data and the visualization of regions of interest for medical image exploration.

We have presented a parallel resampling algorithm for direct volume rendering of deformed volumetric data. The core of the method is an intermediate, GPU-friendly sampling mesh that decouples the resampling process from the underlying deformation method. A notable advantage of this decoupling is that any deformation method can be applied by mapping its deformation field to the sampling mesh, and we show the coupling mechanisms for tetrahedral mass-spring and finite element models, however, the major contribution of this method, as we show in our experiments is that this decoupling effectively eliminates the performance dependency on the resolution of the deformation existing on previous methods. This allows for interactive rendering of volumetric models under high resolution deformation schemes, such as our ChainMail algorithm, presented in Chapter 2, with no additional computational cost.

We have introduced the concept of spatial opacity maps as a novel tool for ROI exploration on medical volumetric data. With the objective of providing an easy integration of 3D rendering techniques with existing clinical workflows of medical volumetric data, which typically rely on 2D slice exploration, we follow a parameter-free paradigm for the interactive generation of the maps by following a GPU parallel *region growing*-based approach, with a single selection on a 2D slice as sole input to our method. We also show how to seamlessly integrate the opacity maps into the standard volume rendering pipeline, thus allowing their combination with existing methods for an enhanced focus+context rendering, and we demonstrate this feature by combining their application with an automatic transfer function generation procedure.

5.3 Future Works

5.3.1 Parallel ChainMail simulation of heterogeneous medical models

In [109, 103, 104] we have proposed a novel ChainMail algorithm that addresses several major limitations of the original method, however some drawbacks of the original algorithm still yield severe limitations which are yet to be solved. Primarily, developing a proper material characterization process to map properties of a real material to a ChainMail model remains a challenging problem. A novel constraint formulation to account for inter-element rotation has been recently addressed by Teske et al. [115], but a generalized constraint model is yet to be found.

5.3.2 Simulation-based control of soft robots

In [108] we have proposed a solution for online motion planning in the task-space for hydraulic actuated soft robots. We have demonstrated the accuracy of the fluid

weight distribution algorithm and the simulation model, and we successfully control a simulated hydraulic robot, however, we still need to apply our motion planning solution to a fabricated robot to evaluate and validate its accuracy. We would also like to explore the joint effect of hydraulic and other actuators on the same robot, since our motion planning solution is capable of handling different types of actuators and we believe that an increased capability can be reached by their joint efforts. Regarding the proposed parallel-leveraging algorithm, we believe that it could be applied to other algorithms such as GPU rasterization, and we would like to perform an experimental comparison with other current leveraging strategies to further explore performance and benefits of the different approaches.

5.3.3 Interactive medical visualization

Our resampling algorithm, described in [105], achieves great performance but it is currently limited by GPU memory, thus we would like to extend its usage by combining our sampling mesh with 3D texture mapping as explained in [101], as it would allow to capture additional detail without increasing the sampling mesh resolution once the memory limit is reached. Another interesting improvement is the development of a better solution to handle topology changes, as our current approach produces loss of material on the boundaries.

In [110], we have introduced the opacity maps for medical volume exploration of regions of interest. We have demonstrated that they can be seamlessly integrated in the volume rendering pipeline and can be thus integrated with other exploration strategies and rendering techniques. We have only combined their usage with a simple automatic transfer function generation procedure as proof of concept, however, we strongly believe that a combination with existing approaches such as the importance-driven volume rendering technique [85] could be easily achieved and would provide an improved *focus+context* visualization and we plan to further explore this and other combinations.

Bibliography

[1] Herb Sutter, "The free lunch is over: A fundamental turn toward concurrency in software," *Dr. Dobb's journal*, vol. 30, no. 3, pp. 202–210, 2005. (Cited on page 2.)

[2] Jason Sanders and Edward Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming, Portable Documents*, Addison-Wesley Professional, 2010. (Cited on page 2.)

[3] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W Sheaffer, and Kevin Skadron, "A performance study of general-purpose applications on graphics processors using cuda," *Journal of parallel and distributed computing*, vol. 68, no. 10, pp. 1370–1380, 2008. (Cited on page 2.)

[4] Aaftab Munshi et al., "The OpenCL specification," *Khronos OpenCL Working Group*, vol. 1, pp. l1–15, 2009. (Cited on page 2.)

[5] John E Stone, David Gohara, and Guochun Shi, "Opencl: A parallel programming standard for heterogeneous computing systems," *Computing in science & engineering*, vol. 12, no. 3, pp. 66–73, 2010. (Cited on page 2.)

[6] André R Brodtkorb, Trond R Hagen, and Martin L Sætra, "Graphics processing unit (gpu) programming strategies and trends in gpu computing," *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, pp. 4–13, 2013. (Cited on page 2.)

[7] Matt Pharr and Randima Fernando, *Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation*, Addison-Wesley Professional, 2005. (Cited on page 2.)

[8] David B Kirk and W Hwu Wen-mei, *Programming massively parallel processors: a hands-on approach*, Morgan Kaufmann, 2012. (Cited on page 2.)

[9] Justin Holewinski, Louis-Noël Pouchet, and Ponnuswamy Sadayappan, "High-performance code generation for stencil computations on gpu architectures," in *Proceedings of the 26th ACM international conference on Supercomputing*. ACM, 2012, pp. 311–320. (Cited on page 3.)

[10] Kaushik Datta, Mark Murphy, Vasily Volkov, Samuel Williams, Jonathan Carter, Leonid Oliker, David Patterson, John Shalf, and Katherine Yelick, "Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 4. (Cited on page 3.)

[11] Paulius Micikevicius, "3D finite difference computation on GPUs using CUDA," in *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*. ACM, 2009, pp. 79–84. (Cited on page 3.)

[12] Takashi Shimokawabe, Takayuki Aoki, Chiashi Muroi, Junichi Ishida, Kohei Kawano, Toshio Endo, Akira Nukada, Naoya Maruyama, and Satoshi Matsuoka, "An 80-fold speedup, 15.0 tflops full gpu acceleration of non-hydrostatic weather model asuca production code," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 2010, pp. 1–11. (Cited on page 3.)

[13] Anthony Nguyen, Nadathur Satish, Jatin Chhugani, Changkyu Kim, and Pradeep Dubey, "3.5-d blocking optimization for stencil computations on modern cpus and gpus," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. 2010, pp. 1–13, IEEE Computer Society. (Cited on page 3.)

[14] Yongpeng Zhang and Frank Mueller, "Auto-generation and auto-tuning of 3d stencil codes on gpu clusters," in *Proceedings of the Tenth International Symposium on Code Generation and Optimization*. ACM, 2012, pp. 155–164. (Cited on page 3.)

[15] André R Brodtkorb, Martin L Sætra, and Mustafa Altinakar, "Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation," *Computers & Fluids*, vol. 55, pp. 1–12, 2012. (Cited on page 3.)

[16] Martin L Sætra, "Shallow water simulation on gpus for sparse domains," in *Numerical Mathematics and Advanced Applications 2011*, pp. 673–680. Springer, 2013. (Cited on page 3.)

[17] Stanley Tzeng, Anjul Patney, and John D Owens, "Task management for irregular-parallel workloads on the gpu," in *Proceedings of the Conference on High Performance Graphics*. Eurographics Association, 2010, pp. 29–37. (Cited on page 3.)

[18] Kshitij Gupta, Jeff A Stuart, and John D Owens, "A study of persistent threads style gpu programming for gpgpu workloads," in *Innovative Parallel Computing (InPar), 2012*. IEEE, 2012, pp. 1–14. (Cited on page 3.)

[19] Fang Liu, Meng-Cheng Huang, Xue-Hui Liu, and En-Hua Wu, "Freepipe: a programmable parallel rendering architecture for efficient multi-fragment effects," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. ACM, 2010, pp. 75–82. (Cited on page 3.)

[20] Kayvon Fatahalian, Edward Luong, Solomon Boulos, Kurt Akeley, William R Mark, and Pat Hanrahan, "Data-parallel rasterization of micropolygons with defocus and motion blur," in *Proceedings of the Conference on High Performance Graphics 2009*. ACM, 2009, pp. 59–68. (Cited on page 3.)

[21] Christian Eisenacher and Charles Loop, "Data-parallel micropolygon rasterization," *Eurographics 2010 Short Papers*, pp. 53–56, 2010. (Cited on page 3.)

[22] Samuli Laine and Tero Karras, "High-performance software rasterization on gpus," in *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. ACM, 2011, pp. 79–88. (Cited on page 3.)

[23] Ullrich Meier, Oscar López, Carlos Monserrat, Mari C Juan, and M Alcaniz, "Real-time deformable models for surgery simulation: a survey," *Computer methods and programs in biomedicine*, vol. 77, no. 3, pp. 183–197, 2005. (Cited on page 3.)

[24] P. Moore and D. Molloy, "A survey of computer-based deformable models," in *Machine Vision and Image Processing Conference, 2007. IMVIP 2007. International*, 2007, pp. 55–66. (Cited on page 3.)

[25] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson, "Physically based deformable models in computer graphics," in *Computer graphics forum*. Wiley Online Library, 2006, vol. 25, pp. 809–836. (Cited on page 3.)

[26] Bernhard Eberhardt, Andreas Weber, and Wolfgang Strasser, "A fast, flexible, particle-system model for cloth draping," *IEEE Computer Graphics and Applications*, vol. 16, no. 5, pp. 52–59, 1996. (Cited on page 3.)

[27] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly, "Projective dynamics: fusing constraint projections for fast simulation," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 154, 2014. (Cited on page 3.)

[28] Maxime Tournier, Matthieu Nesme, Benjamin Gilles, and François Faure, "Stable constrained dynamics," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 132:1–132:10, 2015. (Cited on page 3.)

[29] Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan, "Fast simulation of mass-spring systems," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, pp. 214, 2013. (Cited on page 4.)

[30] Huamin Wang, "A chebyshev semi-iterative approach for accelerating projective and position-based dynamics," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 246, 2015. (Cited on page 4.)

[31] Morton E Gurtin, *An introduction to continuum mechanics*, vol. 158, Academic press, 1982. (Cited on page 4.)

[32] Eftychios Sifakis and Jernej Barbic, "Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction," in *ACM SIGGRAPH 2012 Courses*. ACM, 2012, p. 20. (Cited on page 4.)

[33] Sarah F Gibson, "3d chainmail: a fast algorithm for deforming volumetric objects," in *Proceedings of the 1997 symposium on Interactive 3D graphics*. ACM, 1997, pp. 149–154. (Cited on page 4.)

[34] Sarah F Frisken-Gibson, "Using linked volumes to model object collisions, deformation, cutting, carving, and joining," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 5, no. 4, pp. 333–348, 1999. (Cited on page 4.)

[35] Markus A Schill, Sarah FF Gibson, H-J Bender, and Reinhard Männer, "Biomechanical simulation of the vitreous humor in the eye using an enhanced chainmail algorithm," in *Medical Image Computing and Computer-Assisted Interventation*, pp. 679–687. Springer, 1998. (Cited on page 4.)

[36] Sarah Gibson, Joe Samosky, Andrew Mor, Christina Fyock, Eric Grimson, Takeo Kanade, Ron Kikinis, Hugh Lauer, Neil McKenzie, Shin Nakajima, Hide Ohkami, Randy Osborne, and Akira Sawada, "Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback," in *CVRMed-MRCAS'97*. Springer, 1997, pp. 367–378. (Cited on page 4.)

[37] Tanguy Le Fol, Oscar Acosta-Tamayo, Antoine Lucas, and Pascal Haigron, "Angioplasty simulation using ChainMail method," in *Medical Imaging 2007: Visualization and Image-Guided Procedures*, 2007. (Cited on page 4.)

[38] Florian Schulze, Katja Bühler, and Markus Hadwiger, "Interactive deformation and visualization of large volume datasets.," in *GRAPP (AS/IE)*. Citeseer, 2007, pp. 39–46. (Cited on pages 4 and 7.)

[39] Jörg Mensmann, Timo Ropinski, and Klaus Hinrichs, "Interactive cutting operations for generating anatomical illustrations from volumetric data sets,"

Journal of WSCG – 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, vol. 16, no. 1-3, pp. 89–96, 2008. (Cited on page 4.)

[40] Pierre-Frédéric Villard, Piers Boshier, Fernando Bello, and Derek Gould, *Virtual reality simulation of liver biopsy with a respiratory component*, InTech, 2011. (Cited on page 4.)

[41] P. F. Villard, F. P. Vidal, L. ap Cenydd, R. Holbrey, S. Pisharody, S. Johnson, A. Bulpitt, N. W. John, F. Bello, and D. Gould, "Interventional radiology virtual simulator for liver biopsy," *International Journal of Computer Assisted Radiology and Surgery*, vol. 9, no. 2, pp. 255–267, 2013. (Cited on page 4.)

[42] Carlos R Castro-Pareja, Barry Daly, and Raj Shekhar, "Elastic registration using 3d chainmail: application to virtual colonoscopy," in *Medical Imaging 2006: Image Processing*, 2006, vol. 6144, pp. 947–955. (Cited on page 4.)

[43] Raj Shekhar, Peng Lei, Carlos R Castro-Pareja, William L Plishker, and Warren D D'Souza, "Automatic segmentation of phase-correlated ct scans through nonrigid image registration using geometrically regularized free-form deformation," *Medical physics*, vol. 34, no. 7, pp. 3054–3066, 2007. (Cited on page 4.)

[44] Jinah Park, Sang-Youn Kim, Seung-Woo Son, and Dong-Soo Kwon, "Shape retaining chain linked model for real-time volume haptic rendering," in *Volume Visualization and Graphics, 2002. Proceedings. IEEE / ACM SIGGRAPH Symposium on*, 2002, pp. 65–72. (Cited on page 4.)

[45] Jinah Park, Sang-Youn Kim, and Dong-Soo Kwon, "Mechanical representation of shape-retaining chain linked model for real-time haptic rendering," in *Medical Simulation*, pp. 144–152. Springer, 2004. (Cited on page 4.)

[46] KIM Sang-Youn, PARK Jinah, and KWON Dong-Soo, "The real-time haptic simulation of a biomedical volumetric object with shape-retaining chain linked model," *IEICE transactions on information and systems*, vol. 88, no. 5, pp. 1012–1020, 2005. (Cited on page 4.)

[47] Ying Li and Ken Brodlie, "Soft object modelling with generalised chainmail — extending the boundaries of web-based graphics," *Computer Graphics Forum*, vol. 22, no. 4, pp. 717–727, 2003. (Cited on page 4.)

[48] Pierre-Frédéric Villard, Mathieu Jacob, Derek Gould, and Fernando Bello, "Haptic simulation of the liver with respiratory motion," in *Proceeding of Medicine Meets Virtual Reality 17 (MMVR17)*, 2009, vol. 142, pp. 401–406. (Cited on page 4.)

[49] Franck Patrick Vidal, P-F Villard, and Evelyne Lutton, "Tuning of patient-specific deformable models using an adaptive evolutionary optimization strategy," *Biomedical Engineering, IEEE Transactions on*, vol. 59, no. 10, pp. 2942–2949, 2012. (Cited on page 4.)

[50] Franck P Vidal and Pierre-Frédéric Villard, "Development and validation of real-time simulation of x-ray imaging with respiratory motion," *Computerized Medical Imaging and Graphics*, vol. 49, pp. 1–15, 2016. (Cited on page 4.)

[51] Dirk Fortmeier, Andre Mastmeyer, and Heinz Handels, "Image-based palpation simulation with soft tissue deformations using chainmail on the GPU," in *Bildverarbeitung für die Medizin 2013*, pp. 140–145. Springer, 2013. (Cited on page 4.)

[52] Dirk Fortmeier, Andre Mastmeyer, and Heinz Handels, "An image-based multiproxy palpation algorithm for patient-specific VR-simulation," *Medicine Meets Virtual Reality*, pp. 107–113, 2014. (Cited on page 4.)

[53] Rosell Torres, Jose M Espadero, Felipe A Calvo, and Miguel A Otaduy, "Interactive deformation of heterogeneous volume data," in *International Symposium on Biomedical Simulation*. Springer, 2014, pp. 131–140. (Cited on page 4.)

[54] Lily Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun, "Numerical coarsening of inhomogeneous elastic materials," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, pp. 51:1–51:8, 2009. (Cited on page 4.)

[55] Matthieu Nesme, Paul G Kry, Lenka Jeřábková, and François Faure, "Preserving topology and elasticity for embedded deformable models," in *ACM Transactions on Graphics (TOG)*. ACM, 2009, vol. 28, pp. 52:1–52:9. (Cited on page 4.)

[56] Joachim Georgii and Rüdiger Westermann, "Mass-spring systems on the gpu," *Simulation modelling practice and theory*, vol. 13, no. 8, pp. 693–702, 2005. (Cited on page 5.)

[57] CE Etheredge, "A parallel mass-spring model for soft tissue simulation with haptic rendering in cuda," in *15th Twente Student Conference*, 2011. (Cited on page 5.)

[58] Olivier Comas, Zeike A Taylor, Jérémie Allard, Sébastien Ourselin, Stéphane Cotin, and Josh Passenger, "Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA," in *Biomedical Simulation*, pp. 28–39. Springer, 2008. (Cited on page 5.)

[59] Jérémie Allard, Hadrien Courtecuisse, and François Faure, "Implicit fem solver on gpu for interactive deformation simulation," *GPU Computing Gems Jade Edition*, pp. 281–294, 2011. (Cited on page 5.)

[60] F Rößler, T Wolff, and T Ertl, "Direct GPU-based volume deformation," *Proceedings of Curac*, pp. 65–68, 2008. (Cited on page 5.)

[61] Deepak Trivedi, Christopher D Rahn, William M Kier, and Ian D Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, 2008. (Cited on page 5.)

[62] Carmel Majidi, "Soft robotics: a perspective—current trends and prospects for the future," *Soft Robotics*, vol. 1, no. 1, pp. 5–11, 2014. (Cited on page 5.)

[63] Gang Chen, Minh Tu Pham, and Tanneguy Redarce, "Development and kinematic analysis of a silicone-rubber bending tip for colonoscopy," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 168–173. (Cited on page 5.)

[64] Jessica Burgner-Kahrs, D Caleb Rucker, and Howie Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015. (Cited on page 5.)

[65] Rolf Pfeifer, Max Lungarella, and Fumiya Iida, "The challenges ahead for bio-inspired'soft'robotics," *Communications of the ACM*, vol. 55, no. 11, pp. 76–87, 2012. (Cited on page 5.)

[66] Andrew D Marchese, Konrad Komorowski, Cagdas D Onal, and Daniela Rus, "Design and control of a soft and continuously deformable 2d robotic manipulation system," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2189–2196. (Cited on page 5.)

[67] Andrew D Marchese, Robert K Katzschmann, and Daniela Rus, "Whole arm planning for a soft and highly compliant 2d robotic manipulator," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 554–560. (Cited on page 5.)

[68] Christian Duriez, "Control of elastic soft robots based on real-time finite element method," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3982–3987. (Cited on page 5.)

[69] Frederick Largilliere, Valerian Verona, Eulalie Coevoet, Mario Sanz-Lopez, Jeremie Dequidt, and Christian Duriez, "Real-time control of soft-robots using asynchronous finite element modeling," in *Robotics and Automation (ICRA),*

2015 IEEE International Conference on. IEEE, 2015, pp. 2550–2555. (Cited on page 5.)

[70] Frank Daerden and Dirk Lefeber, "Pneumatic artificial muscles: actuators for robotics and automation," vol. 47, no. 1, pp. 11–21, 2002. (Cited on page 5.)

[71] Deepak Trivedi, Amir Lotfi, and Christopher D Rahn, "Geometrically exact models for soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 773–780, 2008. (Cited on page 5.)

[72] Luis G Torres and Ron Alterovitz, "Motion planning for concentric tube robots using mechanics-based models," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 5153–5159. (Cited on page 5.)

[73] Klaus Engel, Markus Hadwiger, Joe M Kniss, Aaron E Lefohn, Christof Rezk Salama, and Daniel Weiskopf, "Real-time volume graphics," in *ACM Siggraph 2004 Course Notes*. ACM, 2004, p. 29. (Cited on page 5.)

[74] Klaus Engel, Markus Hadwiger, Joe Kniss, Christof Rezk-Salama, and Daniel Weiskopf, *Real-time volume graphics*, CRC Press, 2006. (Cited on pages 5 and 6.)

[75] T Todd Elvins, "A survey of algorithms for volume visualization," *ACM Siggraph Computer Graphics*, vol. 26, no. 3, pp. 194–201, 1992. (Cited on page 5.)

[76] Michael Meißner, H Pfister, Rüdiger Westermann, and CM Wittenbrink, "Volume visualization and volume rendering techniques," *Eurographics tutorial*, 2000. (Cited on page 5.)

[77] Markus Hadwiger, Patric Ljung, Christof Rezk Salama, and Timo Ropinski, "Advanced illumination techniques for gpu-based volume raycasting," in *ACM SIGGRAPH 2009 Courses*. 2009, SIGGRAPH '09, pp. 2:1–2:166, ACM. (Cited on page 6.)

[78] Klaus Engel, Martin Kraus, and Thomas Ertl, "High-quality pre-integrated volume rendering using hardware-accelerated pixel shading," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*. ACM, 2001, pp. 9–16. (Cited on page 6.)

[79] Jens Kruger and Rüdiger Westermann, "Acceleration techniques for gpu-based volume rendering," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. 2003, IEEE Computer Society. (Cited on page 6.)

[80] Stefan Bruckner, Peter Kohlmann, Armin Kanitsar, and M Eduard Groller, "Integrating volume visualization techniques into medical applications," in *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*. IEEE, 2008, pp. 820–823. (Cited on page 6.)

[81] Alexander Wiebel, Frans M Vos, David Foerster, and Hans-Christian Hege, "Wysiwyp: what you see is what you pick," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2236–2244, 2012. (Cited on page 6.)

[82] Gordon Kindlmann and James W Durkin, "Semi-automatic generation of transfer functions for direct volume rendering," in *Proceedings of the 1998 IEEE symposium on Volume visualization*. ACM, 1998, pp. 79–86. (Cited on page 6.)

[83] Hanspeter Pfister, Bill Lorensen, Chandrajit Bajaj, Gordon Kindlmann, Will Schroeder, Lisa Sobierajski Avila, KM Raghu, Raghu Machiraju, and Jinho Lee, "The transfer function bake-off," *IEEE Computer Graphics and Applications*, vol. 21, no. 3, pp. 16–22, 2001. (Cited on page 6.)

[84] Carlos D Correa, Deborah Silver, and Min Chen, "Feature aligned volume manipulation for illustration and visualization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 5, pp. 1069–1076, 2006. (Cited on page 6.)

[85] Ivan Viola, Armin Kanitsar, and Meister Eduard Groller, "Importance-driven volume rendering," in *Proceedings of the conference on Visualization'04*. IEEE Computer Society, 2004, pp. 139–146. (Cited on pages 6 and 134.)

[86] Stefan Bruckner, Soren Grimm, Armin Kanitsar, and M Eduard Groller, "Illustrative context-preserving exploration of volume data," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 6, pp. 1559–1569, 2006. (Cited on page 6.)

[87] Shiaofen Fang, Tom Biddlecome, and Mihran Tuceryan, "Image-based transfer function design for data exploration in volume visualization," in *Proceedings of the Conference on Visualization'98*. IEEE Computer Society Press, 1998, pp. 319–326. (Cited on page 6.)

[88] Stefan Roettger, Michael Bauer, and Marc Stamminger, "Spatialized transfer functions.," in *EuroVis*. Citeseer, 2005, pp. 271–278. (Cited on page 6.)

[89] Carlos D Correa and Kwan-Liu Ma, "Visibility histograms and visibility-driven transfer functions," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 2, pp. 192–204, 2011. (Cited on page 6.)

[90] Petr Sereda, Anna Vilanova, and Frans A Gerritsen, "Automating transfer function design for volume rendering using hierarchical clustering of material boundaries.," in *EuroVis*, 2006, pp. 243–250. (Cited on page 6.)

[91] Jianlong Zhou and Masahiro Takatsuka, "Automatic transfer function generation using contour tree controlled residue flow model and color harmonics," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 6, pp. 1481–1488, 2009. (Cited on page 6.)

[92] Marc Ruiz, Anton Bardera, Imma Boada, Ivan Viola, Miquel Feixas, and Mateu Sbert, "Automatic transfer functions based on informational divergence," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1932–1941, 2011. (Cited on page 6.)

[93] Lile Cai, Wei-Liang Tay, Binh P Nguyen, Chee-Kong Chui, and Sim-Heng Ong, "Automatic transfer function design for medical visualization using visibility distributions and projective color mapping," *Computerized Medical Imaging and Graphics*, vol. 37, no. 7, pp. 450–458, 2013. (Cited on page 6.)

[94] Runzhen Huang and Kwan-Liu Ma, "Rgvis: Region growing based techniques for volume visualization," in *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*. IEEE, 2003, pp. 355–363. (Cited on page 6.)

[95] Peter Kohlmann, Stefan Bruckner, Armin Kanitsar, and A Kanitsar, "Livesync: Deformed viewing spheres for knowledge-based navigation," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 6, pp. 1544–1551, 2007. (Cited on page 6.)

[96] Peter Kohlmann, Stefan Bruckner, Armin Kanitsar, and M Eduard Gröller, "Livesync++: Enhancements of an interaction metaphor," in *Proceedings of Graphics Interface 2008*. Canadian Information Processing Society, 2008, pp. 81–88. (Cited on page 6.)

[97] Megumi Nakao and Kotaro Minato, "Physics-based interactive volume manipulation for sharing surgical process," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 3, pp. 809–816, 2010. (Cited on page 6.)

[98] Kup-Sze Choi, Hanqiu Sun, and Pheng-Ann Heng, "Interactive deformation of soft tissues with haptic feedback for medical learning," *IEEE Transactions on Information Technology in Biomedicine*, vol. 7, no. 4, pp. 358–363, 2003. (Cited on page 6.)

[99] Joachim Georgii and Rüdiger Westermann, "A generic and scalable pipeline for gpu tetrahedral grid rendering," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 5, pp. 1345–1352, 2006. (Cited on page 7.)

[100] Carlos D Correa, Robert Hero, and Kwan-Liu Ma, "A comparison of gradient estimation methods for volume rendering on unstructured meshes," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 3, pp. 305–319, 2011. (Cited on page 7.)

[101] Jorge Gascon, Jose M Espadero, Alvaro G Perez, Rosell Torres, and Miguel A Otaduy, "Fast deformation of volume data using tetrahedral mesh rasterization," in *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2013, pp. 181–185. (Cited on pages 7 and 134.)

[102] Martin Meike, Dirk Fortmeier, Andre Mastmeyer, and Heinz Handels, "Real-time resampling of medical images based on deformed tetrahedral structures for needle insertion vr-simulation," in *Bildverarbeitung für die Medizin 2015*, pp. 443–448. Springer, 2015. (Cited on page 7.)

[103] Alejandro Rodríguez, Alejandro León, Germán Arroyo, and José Miguel Mantas, "Sp-chainmail: a gpu-based sparse parallel chainmail algorithm for deforming medical volumes," *The Journal of Supercomputing*, vol. 71, no. 9, pp. 3482–3499, 2015. (Cited on pages 9, 26, and 133.)

[104] Alejandro Rodríguez, Alejandro León, and Germán Arroyo, "Parallel deformation of heterogeneous chainmail models: Application to interactive deformation of large medical volumes," *Computers in Biology and Medicine*, vol. 79, pp. 222–232, 2016. (Cited on pages 9, 48, and 133.)

[105] Alejandro Rodríguez, Alejandro León Salas, Domingo Martín Perandrés, and Miguel A Otaduy, "A parallel resampling method for interactive deformation of volumetric models," *Computers & Graphics*, vol. 53, pp. 147–155, 2015. (Cited on pages 9, 93, and 134.)

[106] Rosell Torres, Alejandro Rodríguez, José M. Espadero, and Miguel A. Otaduy, "High-resolution interaction with corotational coarsening models," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 211:1–211:11, 2016. (Cited on page 9.)

[107] K A Mountris, J Bert, J Noailly, A Rodriguez Aguilera, A Valeri, O Pradier, U Schick, E Promayon, M A Gonzalez Ballester, J Troccaz, and D Visvikis, "Modeling the impact of prostate edema on ldr brachytherapy: a monte carlo dosimetry study based on a 3d biphasic finite element biomechanical model," *Physics in Medicine and Biology*, vol. 62, no. 6, pp. 2087–2102, 2017. (Cited on page 10.)

[108] A. Rodriguez, E. Coevoet, and C. Duriez, "Real-time simulation of hydraulic components for interactive control of soft robots," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017. (Cited on pages 10, 83, and 133.)

[109] Alejandro Rodríguez Aguilera, Alejandro Leon, Luis López Escudero, and Manuel García Sánchez, "A System Proposal for Interactive Deformation of Large Medical Volumes," in *Spanish Computer Graphics Conference (CEIG)*. 2014, The Eurographics Association. (Cited on pages 10, 13, and 133.)

[110] Alejandro Rodríguez-Aguilera and Alejandro León, "Spatial Opacity Maps for Direct Volume Rendering of Regions of Interest," in *Spanish Computer Graphics Conference (CEIG)*. 2016, The Eurographics Association. (Cited on pages 10, 119, and 134.)

[111] Domingo Martín, Germán Arroyo, Alejandro Rodríguez, and Tobias Isenberg, "A survey of digital stippling," *Computers & Graphics*, vol. 67, pp. 24–44, 2017. (Cited on page 10.)

[112] Alejandro Rodríguez and Alejandro León, "A framework for remote 3d interaction with handheld devices: Application to a 3d heritage gallery prototype," *SCIRES-IT-SCIentific RESearch and Information Technology*, vol. 6, no. 1, pp. 79–86, 2016. (Cited on page 10.)

[113] Alejandro Rodríguez and Alejandro León, "Smartphone-based remote 3d interaction for digital heritage applications," in *Digital Heritage, 2015*. IEEE, 2015, vol. 1, pp. 297–300. (Cited on page 11.)

[114] Christian Duriez, Eulalie Coevoet, Frédérick Largilliere, Thor Bieze, Zhongkai Zhang, Mario Sanz-Lopez, B Carrez, Damien Marchal, Olivier Goury, and Jérémie Dequidt, "Framework for online simulation of soft robots with optimization-based inverse model," in *SIMPAR: IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, 2016. (Cited on pages 132 and 150.)

[115] Hendrik Teske, Kathrin Bartelheimer, Jan Meis, Rolf Bendl, Eva M Stoiber, and Kristina Giske, "Construction of a biomechanical head and neck motion model as a guide to evaluation of deformable image registration," *Physics in Medicine and Biology*, vol. 62, no. 12, pp. N271, 2017. (Cited on page 133.)

Appendix A

Resumen y Conclusiones

Introducción

Modelar el comportamiento físico de los fenómenos del mundo real ha sido de interés científico a lo largo de la historia. Es una herramienta clave no sólo para mejorar nuestra comprensión del mundo, sino también para proporcionar un medio para reproducir y predecir el comportamiento de muchos tipos de procesos físicos, proporcionando ideas importantes que luego se pueden emplear para una mejor toma de decisiones, estudio detallado O entrenamiento previo a actuaciones reales, entre otras aplicaciones. El modelado computacional y la simulación están entre los desarrollos más significativos dentro de la investigación científica y de ingeniería en las últimas décadas y un amplio abanico de métodos de simulación física son utilizados en una amplia gama de disciplinas tan diversas como la astrofísica, el plegamiento de moléculas, la ingeniería civil o las industrias del cine y los videojuegos.

Un problema importante de los métodos de simulación computacional surge con la complejidad del modelo empleado para describir el fenómeno. Un modelo demasiado complejo de calcular o demasiado grande para almacenar puede requerir una cantidad inaceptable de tiempo para realizar la simulación deseada. Esta limitación se hace especialmente evidente en aplicaciones interactivas, en las que el presupuesto de tiempo para realizar un paso de simulación puede limitarse a unos pocos milisegundos. Muchas de las aplicaciones de simulación interactiva también requieren retroalimentación visual, por lo tanto la conexión entre el modelo de simulación y su representación visual debe realizarse de manera eficiente para permitir que la aplicación cumpla con el estricto límite de tiempo.

En el caso específico de las simulaciones médicas, como pueden ser casos la simulación de procesos fisiológicos o la simulación de procedimientos quirúrgicos (comúnmente conocida como cirugía virtual), se requiere un modelo complejo de la anatomía humana. La distribución tisular de las formas biológicas se captura típicamente a través de tecnologías de detección 3D tales como la tomografía computarizada o las técnicas de resonancia magnética. Como resultado, los datos

adquiridos incluyen hasta varios millones de vóxeles de información que representa la distribución de tejido dentro de los diferentes órganos y estructuras anatómicas. Esta enorme cantidad de datos puede usarse para obtener tanto los modelos biomecánicos como los visuales para aplicaciones interactivas, convirtiéndose en un problema computacionalmente muy exigente que ha recibido mucha atención de la comunidad científica.

La eficiencia en el cómputo de estas aplicaciones se ha incrementado a lo largo de los años, tanto por mejoras en el hardware como en el software. La introducción de las unidades de procesamiento gráfico (GPU) ya permitió acelerar la retroalimentación visual de estas aplicaciones, pero la aparición de las GPUs de cauce programable, y más importante, el desarrollo posterior de entornos de desarrollo para la computación de propósito general en gráficos (GPGPU), supuso un avance significativo que permitió extender su uso a muchos campos de la computación moderna, incluyendo el campo de simulación física.

En esta tesis exploramos las técnicas GPGPU con el objetivo de desarrollar nuevos métodos y algoritmos mejorados para la simulación y visualización de aplicaciones interactivas, con especial atención a aplicaciones médicas y de *soft robotics*, y específicamente en los métodos de simulación basados en ChainMail para simulación médica y modelado de elementos finitos para control de *soft robots*, y en las técnicas de visualización directa de datos volumétricos deformables y técnicas de exploración asistida de los mismos.

Estructura de la tesis doctoral

El principal objetivo de esta tesis doctoral es el estudio y aplicación de técnicas GPGPU para la aceleración de procesos de alto coste computacional y su aplicación a entornos de simulación interactiva. Las contribuciones de la tesis se han agrupado en tres bloques temáticos: simulación médica mediante algoritmo ChainMail, control de *soft robots* basado en simulación y visualización médica interactiva. La tesis se presenta en la modalidad de "compendio" y a continuación citamos las contribuciones en cada uno de los bloques.

Bloque I: Simulación ChainMail paralela de modelos médicos heterogéneos

- **A. Rodríguez**, A. León, L. López Escudero and M. García Sánchez (2014). "A System Proposal for Interactive Deformation of Large Medical Volumes". *Proceedings of Spanish Computer Graphics Conference* (CEIG 2014).

- **A. Rodríguez**, A. León, G. Arroyo, and J. M. Mantas (2015). "SP-ChainMail: a GPU-based sparse parallel ChainMail algorithm for deforming medical volumes". *The Journal of Supercomputing*, Volume 71, Issue 9, pp. 3482-3499.

- **A. Rodríguez**, A. León and G. Arroyo (2016). "Parallel deformation of heterogeneous ChainMail models: Application to interactive deformation of large medical volumes". *Computers in Biology and Medicine*, Volume 79, pp. 222-232.

Bloque II: Control de soft robots basado en simulación

- **A. Rodríguez**, E. Coevoet and C. Duriez. "Real-time simulation of hydraulic components for interactive control of soft robots". *IEEE International Conference on Robotics and Automation* (ICRA 2017).

Bloque III: Visualización médica interactiva

- **A. Rodríguez**, A. León Salas, D. Martín Perandrés and M. A. Otaduy (2015). "A parallel resampling method for interactive deformation of volumetric models". *Computers & Graphics*, Volume 53, pp. 147-155.

- **A. Rodríguez** and A. León (2016). "Spatial Opacity Maps for Direct Volume Rendering of Regions of Interest". *Proceedings of Spanish Computer Graphics Conference* (CEIG 2016).

Conclusiones

En esta tesis hemos presentado diferentes contribuciones en los campos de modelos deformables, *soft robots* y visualización y exploración de volúmenes. Nuestras soluciones están enfocadas hacia sistemas interactivos, tanto en entornos médicos como de control de robots. El presupuesto de tiempo limitado para proporcionar retroalimentación de estas aplicaciones conlleva un sacrificio en términos de precisión de la simulación subyacente y/o nivel de detalle de la retroalimentación visual, para satisfacer los requisitos de rendimiento interactivo. Motivados por esta limitación, hemos centrado nuestros esfuerzos en explotar el poder de cómputo paralelo de las GPU modernas para mejorar la calidad global de las soluciones y, junto con nuestras principales contribuciones, presentamos varias contribuciones relativas a computación GPGPU. A continuación detallamos las conclusiones y contribuciones específicas relacionadas con cada bloque.

Bloque I: Simulación ChainMail paralela de modelos médicos heterogéneos

Hemos presentado un nuevo algoritmo ChainMail paralelo replanteando el algoritmo como un problema de computación *stencil*. Hemos demostrado que este esquema de cómputo permite aumentar significativamente el rendimiento del algoritmo a

través de un cómputo paralelo de las etapas de propagación y relajación en la GPU. Abordamos el problema de la computación eficiente de modelos de ChainMail heterogéneos incorporando un mecanismo basado en marcas de tiempo a la etapa de propagación teniendo en cuenta que la velocidad de propagación de una deformación a través de un cuerpo deformable depende de la rigidez del material atravesado y generalizando el proceso de minimización de energía en la etapa de relajación, que modela la energía elástica almacenada en los tejidos heterogéneos basada en la ley de Hooke. En términos de rendimiento, nuestros experimentos demuestran que nuestra solución supera a los algoritmos ChainMail anteriores por factores de más de 20x, permitiendo simulaciones interactivas para modelos de varios millones de elementos.

Para abordar la naturaleza irregular de la computación de nuestro enfoque *stencil*, hemos propuesto un esquema de partición para una gestión eficiente de la computación en la GPU, reduciendo drásticamente la computación innecesaria y así aumentar el rendimiento del algoritmo. Un análisis experimental mostró que el esquema de partición exhibe buena portabilidad y escalabilidad y permite una aceleración del cómputo de un factor de 7x con respecto al esquema de gestión naif.

En conclusión, el algoritmo propuesto aborda varias limitaciones del algoritmo original aumentando su rendimiento, manejando materiales heterogéneos de forma eficiente y permitiendo que varias deformaciones se propaguen simultáneamente a través del modelo, junto con otras mejoras menores, y creemos que las soluciones basadas en los modelos ChainMail pueden aprovechar estas características para mejorar y ampliar su aplicabilidad. Se ha demostrado que el esquema de partición propuesto aumenta el rendimiento y es aplicable a problemas generales de computación *stencil* que sufren de computación irregular.

Bloque II: Control de soft robots basado en simulación

En el campo emergente de control de *soft robots*, el complejo comportamiento de sus cuerpos, actuadores y otros componentes involucrados debe ser modelado de una manera eficiente pero precisa para permitir soluciones de simulación interactivas para calcular la actuación requerida en unos pocos milisegundos. Hemos presentado un nuevo método paralelo usando la GPU para una estimación eficiente de la distribución del peso del fluido dentro de las cavidades hidráulicas junto con un nuevo modelo de su comportamiento dinámico. Hemos integrado nuestra solución dentro del entorno de simulación de *soft robots* de SOFA [114], proporcionando un sistema para la planificación interactiva del movimiento de *soft robots* con actuadores hidráulicos. Un análisis experimental de nuestra solución muestra la exactitud de las estimaciones de distribución de peso del fluido y el modelo dinámico.

También presentamos un nuevo algoritmo de distribución de trabajo general, simple y paralelo que mejora el cómputo de cargas de trabajo irregulares en la GPU y hemos mostrado su aplicación a nuestro algoritmo de estimación de distribución de peso.

Bloque III: Visualización médica interactiva

Hemos abordado dos problemas computacionalmente exigentes de procesamiento interactivo de volúmenes en aplicaciones médicas: la visualización de datos volumétricos deformables y la visualización de regiones de interés para la exploración de volúmenes médicos.

Hemos presentado un algoritmo de remuestreo paralelo para la visualización directa de datos volumétricos deformados. El núcleo del método es una malla de muestreo intermedia que desacopla el proceso de remuestreo del método de deformación subyacente. Una ventaja notable de este desacoplamiento es que cualquier método de deformación puede aplicarse mediante un mapeado de su campo de deformación a la malla de muestreo, tal y como mostramos en el caso de modelos de masa-muelle y elementos finitos con elementos tetraédricos, sin embargo, la principal contribución de este método, como se muestra en nuestros experimentos es que este desacoplamiento elimina de forma efectiva la dependencia del rendimiento del algoritmo respecto a la resolución del modelo de deformación, existente en los métodos anteriores. Esto permite la visualización interactiva de modelos volumétricos empleando esquemas de deformación de alta resolución, como nuestro algoritmo ChainMail, sin un coste computacional adicional.

Hemos introducido el concepto de mapas de opacidad espacial como una nueva herramienta para la exploración de regiones de interés en datos volumétricos médicos. Con el objetivo de proporcionar una integración sencilla de las técnicas de visualización 3D con los flujos de trabajo clínicos existentes, que normalmente se basan en la exploración de cortes 2D, seguimos un paradigma sin parámetros para la generación interactiva de los mapas de opacidad basado en una computación paralela de crecimiento de regiones, siendo la única entrada por parte del usuario la selección de un píxel en un corte 2D. También mostramos cómo integrar los mapas de opacidad en el cauce estándar de visualización de volúmenes, permitiendo así su combinación con los métodos existentes para una visualización contextual mejorada, y demostramos esta ventaja combinando su aplicación con un procedimiento de generación automático de la función de transferencia.