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

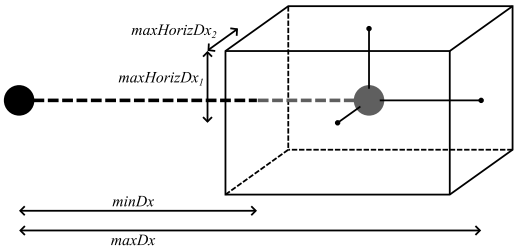

Figure 1: The valid region (box) that an element (black) imposes to a right neighbor (gray) with the original ChainMail algorithm is defined by the constraint parameters (*maxDx*, *minDx*, $maxHorizDx_1$ and $maxHorizDx_2$).

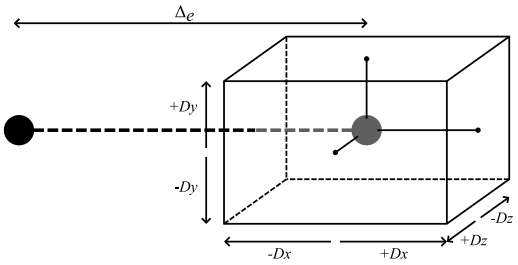

Figure 2: The valid region (box) that an element (black) imposes to a neighbor (gray), is defined by the constraint parameters (*Dx*, *Dy* and *Dz*) and by the initial separation of both elements Δ_e.

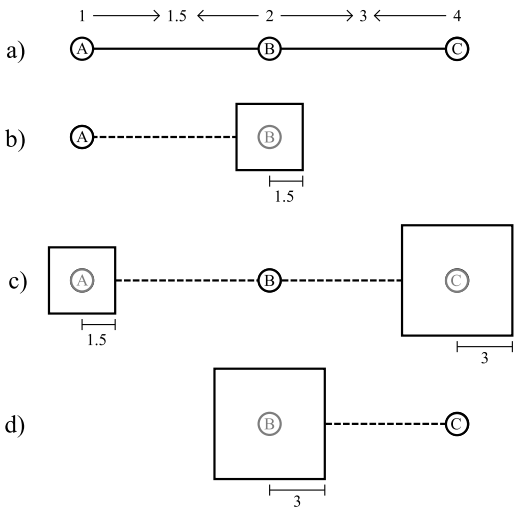

Figure 3: 2D example of heterogeneous constraints in a simple, heterogeneous model: the constraint values of each two linked elements, depicted on top of the elements, are averaged and stored in the link between them (*a*), and the constraints applied by each element to its neighbors are computed attending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

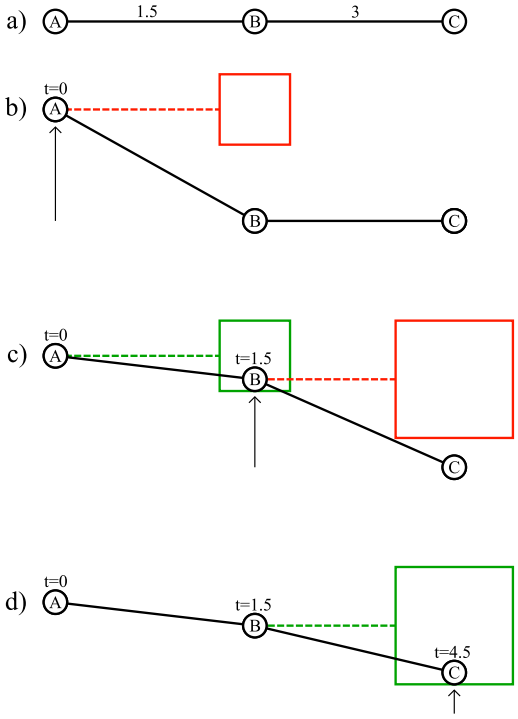

Figure 4: 2D example of the timestamp propagation system with heterogeneous materials (*a*). When a new deformation is applied to an element, the timestamp of the element is set to 0 (*b*), and the propagation starts. Again, the squares represent the valid regions. When a new element is reached through 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*).

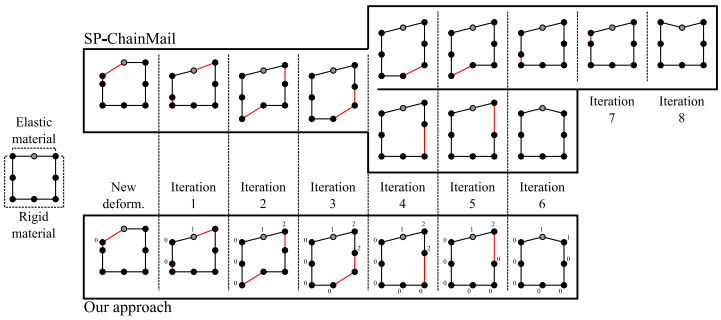

Figure 5: An "heterogeneous ring" model, composed by an elastic material at the top and a rigid material on the lateral and bottom parts, is deformed using both the SP-ChainMail algorithm and our algorithm. Even in this simple case, the SP-ChainMail algorithm has an unpredicted behavior, leading to two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

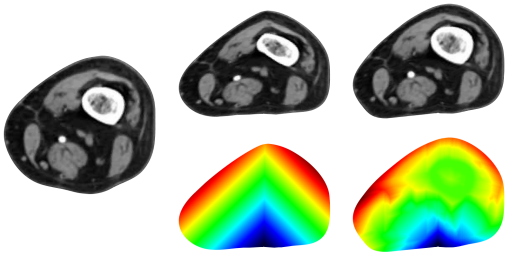

Figure 6: A 2D section of a leg (left) is compressed and deformed using a homogeneous material (middle) and a more realistic distribution of heterogeneous materials (right). The propagation time is shown below each case using a heatmap, ranging from dark blue at the source of the deformation, to a dark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

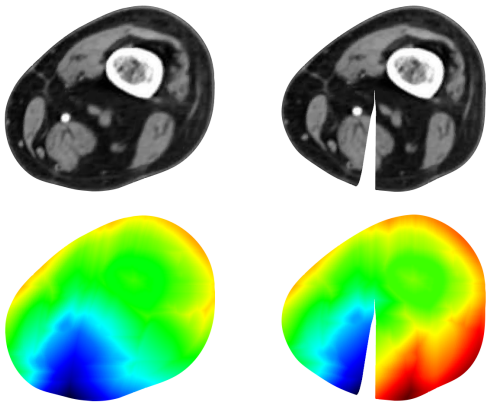

Figure 7: The same deformation is applied to a 2D section of a leg before and after applying a cut. The propagation time is shown below each case using a heatmap, ranging from dark blue at the source of the deformation, to a dark red at the last reached elements, showing that the topology modification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

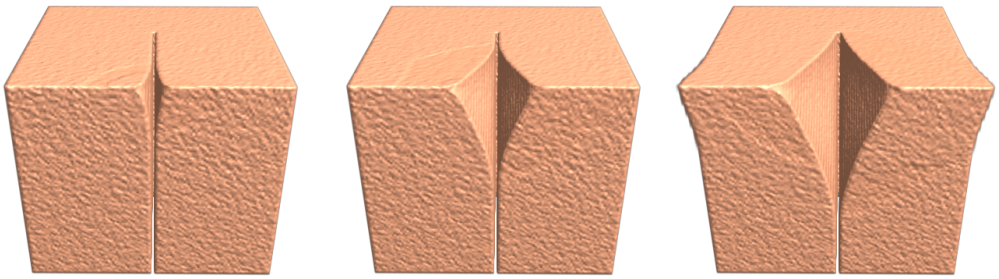

Figure 8: Two simultaneous deformations are applied to both sides of an incision in a tissue sample. Our propagation approach is able to handle and compute both deformations simultaneously.

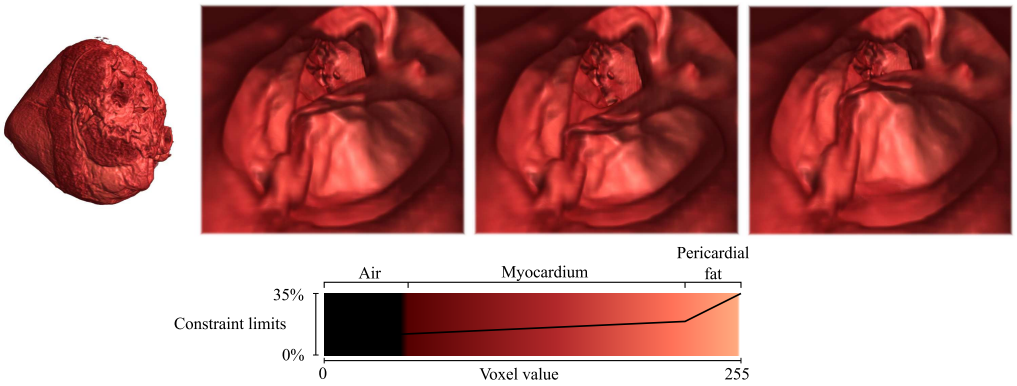

Figure 9: The inside of a sheep heart model with 12.4 million elements (top-left) is explored and several deformations are applied to an internal feature (top-right), which is correctly captured by the high resolution of our ChainMail model. The constraints of the materials assigned to the sheep heart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

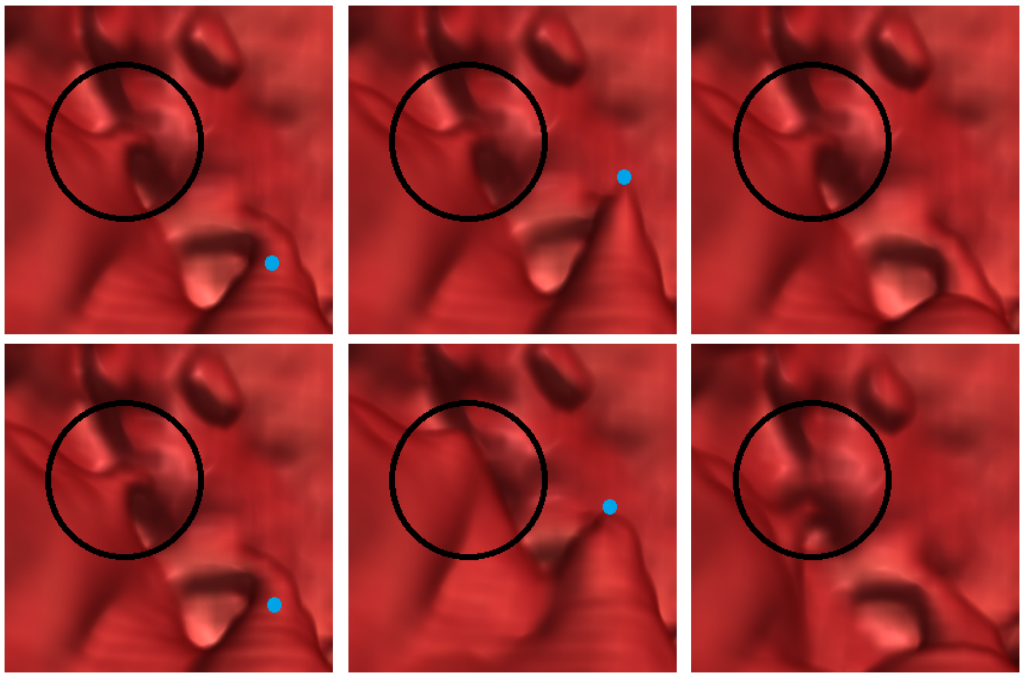

Figure 10: The same deformation is applied to a fine feature using our ChainMail approach (top row) and a mass-spring model (bottom row). The topology of the feature is incorrectly captured by the mass-spring model, leading to an incorrect transmission of the deformation from the right side of the tissue 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).

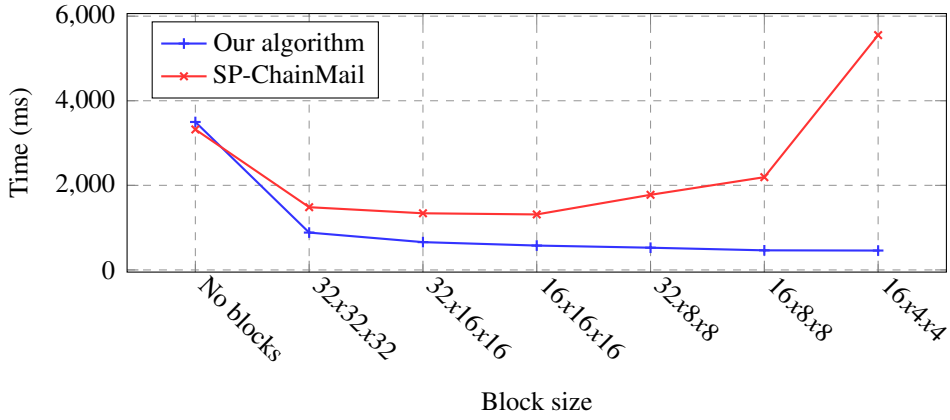

Figure 12: Performance comparison of our method and the SP-ChainMail algorithm using different block sizes for the blocking method. The SP-ChainMail algorithm fails to optimize the usage of small block sizes, leading to a misuse of the GPU resources. Our improved blocking scheme is able to efficiently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