Haptic Rendering of FEM-based Tearing Simulation using Clusterized Collision Detection

Benoît Le Gouis[1] François Lehericey[1,3] Maud Marchal[1] Bruno Arnaldi[1] Valérie Gouranton[1] Anatole Lécuyer[2]

***Abstract*— Haptic rendering of deformable phenomena remains computationally-demanding, especially when topology modifications are simulated. Within this context, the haptic rendering of tearing phenomena is under-explored as of today. In this paper we propose a fully-functional interaction pipeline for physically-based simulation of deformable surfaces tearing allowing to reach a haptic interactive rate. It relies on a high efficiency collision detection algorithm for deformable surface meshes, combined with an efficient FEM-based simulation of deformable surfaces enabling tearing process. We especially introduce a novel formulation based on clusters for the collision detection to improve the computation time performances. Our approach is illustrated through interactive use-cases of tearing phenomena with haptic feedback, showing its ability to handle realistic rendering of deformable surface tearing on consumer-grade haptic devices.**

I. Introduction

Thin deformable surfaces are very common in everyday life, from cloth to paper, bags and food. When manipulated, these deformable objects often undergo topology modifications through tearing phenomena. The real-time simulation of such phenomena in virtual environments has therefore many applications, from surgery training to video games and animation. It remains however very computationally-demanding, preventing their use for many interactive applications. Adding haptic feedback to the simulation would significantly improve sensory feedback but would also increase the computational challenge. As the simulation handles thin deformable material, a fully functional interaction pipeline requires also an accurate and efficient collision detection method for the deformable components, which brings an additional computation strain to the system. For these reasons, the haptic rendering of tearing phenomena remains unexplored in the literature. Indeed, unlike cutting simulation, for which many methods include haptic interaction, tearing has received less attention, and there is to the best of our knowledge no method allowing haptic interaction with tearing simulation.

In this paper, we propose a fully-functional interaction pipeline for physically-based simulation of tearing with haptic rendering. It relies on the first hand on a novel and efficient collision detection method for thin deformable objects, including auto-collisions and collision with the environment, and on the other hand on a tearing simulation based on an efficient Finite Element Method (FEM) compatible with haptic rates. We also provide a haptic interaction scheme allowing the use of multiple haptic devices, such as for two-handed haptic tearing simulation.

The remainder of this paper is organized as follows. Section II presents the related work to our contribution. The general pipeline for haptic tearing simulation is then presented in section III. Section IV details the collision detection method used in our pipeline. The tearing method is then described in section V, and the haptic coupling is later presented in section VI. Several use-cases involving haptic tearing are proposed in section VII, as well as their corresponding computation time performances.

II. Related Work

Tearing simulation has been first introduced in computer graphics by Terzopoulos [TF88]: the tear creation and propagation was driven by the level of stress and strain of the objects. This basic idea of the approach has been further used for other fracture and tearing phenomena. For instance, the method of Souza et al. [SWC14] consists in splitting one vertex of the overstretched edges to create a tear that follows the existing edges. In order to avoid creating too many visual artifacts, this method requires a fine quality mesh that can impair performances, and is thus either not visually plausible, or unsuited for haptic interaction. Pfaff et al. [PNdJO14] based on the prior work on adaptive meshes by Narain [NSO12] proposed to refine the mesh where a crack is likely to propagate or be created and coarsen it elsewhere. The direction of a potential crack is computed by finding the potential splitting plane for which the most stress energy would be released. While more accurate than the previous methods, this method is computationally heavier, and does not seem suitable for haptic interaction. Gingold et al. [GSH*04] based their fracture handling on the strain tensor of each triangular face of the mesh: a face is split into two if the principal strain exceeds a material-specific threshold. The fracture splits the strained face along the direction orthogonal to the principal strain direction. This method has been extended by Allard et al. [AMC09] for anisotropic elements, simulating fibers. This anisotropy is taken into account for the tearing propagation. While both methods provide a simple – yet accurate – model for physically-based tearing simulation, neither method discusses haptics.

[1]INSA Rennes/IRISA, Campus de Beaulieu, Rennes, France.
[2]Inria Rennes, France.
[3]Vinci Construction France
{benoit.legouis, maud.marchal, bruno.arnaldi, valerie.gouranton, anatole.lecuyer}@irisa.fr

Besides the tearing simulation itself, the simulation pipeline for tearing phenomena should handle efficient collision detection of the torn components of the virtual environments. Collision detection remains one of the main bottlenecks of physics simulation, moreover in scenes containing highly dynamic objects such as deformable objects and topological changes where the cost tends to highly increase. In the context of haptic simulations where performance is more critical than others applications, this has led to specific methods. For instance, Gregory et al. [GLGT05] proposed a framework that uses spatial decomposition, bounding volumes hierarchy and temporal coherence to achieve collision detection on rigid objects at a frame-rate compatible with haptic simulations. In the case of deformable objects, Barbič [BJ08] proposed to perform collision detection by representing objects with a point-based representation and signed distance fields. To comply with the high frame-rate requirement of haptic rendering the authors proposed to use a graceful degradation of contacts: if there is not enough computation time to fully perform collision detection, the algorithm returns a reasonable answer instead of the complete one.

In this paper, we propose a method to optimize the collision detection by regrouping the vertices in clusters. Using a measurement of the relative displacements between the clusters, we can identify the vertices where a full collision detection needs to be performed. Based on this method, we are thus able to propose a full-functional interaction pipeline for simulating tearing phenomena on deformable surfaces with haptic rendering.

III. General description of our haptic tearing Approach

In our approach, the user can interact in real-time with a virtual environment composed of thin deformable objects and receive multi-sensory feedback through the use of screens and several haptic interfaces. Our physically-based simulator is composed of two main components: (1) a novel collision detection method to provide information of contact between the deformable surface and its environment, as well as auto-collisions, (2) a FEM simulation including a tearing model for thin deformable objects. During the simulation, collision response dictates the behavior of the surface in contact with its environment. Based on this contact information and on the user input, deformation of the surface is determined by the physically-based model, that can handle tearing phenomena. Feedback is then provided to the user using the information from the physically-based simulation. In our approach, a haptic coupling is implemented to provide a bimanual haptic feedback, depicted in Figure 1.

In order to achieve interactive simulation rate, each of the steps of the simulation pipeline must be performed with minimal computational cost. Collision detection and physically-based rendering of tearing are the main bottlenecks in terms of computation time, and are therefore hereafter described.

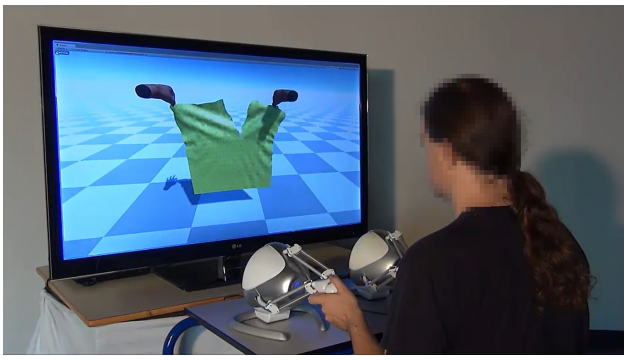

Fig. 1: The setup for our bimanual haptic tearing method: a user can tear a deformable surface using two haptic devices.

IV. Collision detection for surface meshes using a novel clustering formulation

In order to simulate a thin deformable surface, collision detection queries are required at each time step of the simulation. We chose to use an image-based method to perform collision detection since these methods are more adapted to highly dynamic objects such as a piece of cloth that can undergo deformations and topological changes. In particular, we used a ray-tracing based method proposed by [LGA15] since it is able to handle both surfacic objects (in the present case for the deformable surface) and volumetric objects (for the surrounding environment in our case).

For haptic rendering, the required frame-rate of the physically-based simulation is higher than for visual feedback only. To complete collision detection within efficient frame-rates, temporal coherency can first be used to accelerate the collision queries. Temporal coherency is a property that comes from the continuity of the motion of the objects in the simulation. It states that the positions and geometric configurations in space of the objects will be similar between successive time steps. To exploit this property, an incremental method is more profitable to the high frame-rates. Lehericey et al. [LGA13] proposed such an approach for rigid objects, in order to reuse the results of the collision detection from the last step and then update them incrementally instead of performing collision detection from scratch at each time-step.

In this paper, we propose to extend this approach to deformable objects, such as thin deformable material. To decide if an incremental method can be used, we measure temporal coherency between objects by measuring the quantity of relative displacement between pairs of objects. If objects have low relative displacement over time, e.g. the relative displacement stays under a threshold called *displacementThreshold*, then an incremental method is used to compute collision detection, otherwise a non-incremental method is used. The quantity of relative displacement is measured by studying the evolution of the frame of reference between pairs of objects. However, it does not take into account the internal deformation of the objects. We thus propose a novel method to enable us to measure the quantity of relative displacement between two deformable objects (or between a rigid and a deformable objects), that takes

into account the internal deformation of the objects. This new method does not perform a global measurement of relative displacements between the two objects but works on a decomposition of the objects called clusters. It is thus able to detect temporal coherency locally even if the two whole objects do not exhibit strong temporal coherency. In the following paragraphs, we detail the different steps of our approach.

A. Decomposition of objects in clusters

The first step of our method consists in decomposing the deformable objects in clusters. The cluster decomposition is motivated by two computation-intensive and memory-consuming issues: (1) contrary to rigid objects, the displacement of each individual vertex of a deformable object is independent and we therefore need to measure each displacement, (2) the history of the position of each vertex over time needs to be stored, in order to know when the relative displacement exceeds the threshold *displacementThreshold*. To reduce the time and memory complexity related to displacement measurement, we propose to measure the displacement of clusters of vertices instead of measuring the displacement of each vertex individually. The use of clusters is motivated by the observation that neighbouring vertices tend to have similar movements because of spatial coherency.

To decompose the object into clusters of vertices, we use the k-means clustering method. The number of vertices per cluster is empirically fixed in our case (100 vertices per cluster) as we have homogeneous deformable objects. We could however adapt it depending on the object properties. As a distance metric for the k-means, we do not use the Euclidean distance because it does not guarantee that the vertices will still be close in space if a deformation occurs. Instead, we use a metric based on the distance on the surface of the object which stays valid even after the objects have undergone some deformation. Our distance metric between two vertices is the length of the shortest path between the two vertices through the edges. Furthermore, this metric guarantees that no cluster will be disjointed. Figure 2 gives an example of the decomposition into clusters of cloth.

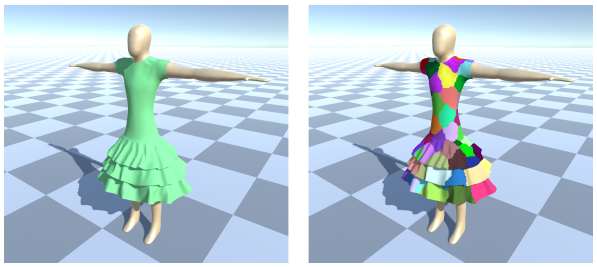

Fig. 2: Example of the decomposition of a dress (left) into clusters (right). The dress is composed of 11,000 vertices and is decomposed into 110 clusters.

B. Relative displacement measurement of clustered objects

The second step of our method is to compute the relative displacement measurement between clustered objects. The measurement is itself performed in three steps:

- Measurement of absolute displacement of clusters.
- Detection of neighbouring clusters.
- Measurement of relative displacement of neighboring clusters.

The first two steps are independent. The goal of the first step is to reduce the complexity of the movement of all the vertices in the cluster into a more easily manageable form. The second step is used to decide on which pairs of clusters we need to measure the relative displacement from (to avoid performing unnecessary measure on pairs that are far away). The third step combines the output of the two first steps to compute the relative displacement on the selected pairs of clusters.

1) Measurement of absolute displacement of clusters: To obtain a general information about the displacement within a cluster, we first determine the absolute displacement of each cluster. We measure the relative displacement only from the previous time-step and we do not need to store the values of the previous steps.

To reduce the complexity of measurement, we propose to measure the average displacement of the vertices and the standard deviation from the average displacement. For each cluster c, the average displacement $dis_{avg}(c) \in \mathbf{R}^3$ is a vector containing the main direction and the amplitude of the displacement, the standard deviation $dis_{dev}(c) \in \mathbf{R}$ is a scalar measuring the rotation and deformation of the cluster.

2) Detection of neighbouring clusters: In this step, we list the pairs of neighbouring clusters. This is done to perform culling on the next steps as we only need to perform a measurement of relative displacement and collision detection on very close clusters.

To list the pairs of close clusters, we propose to use a broad-phase collision detection algorithm. We propose to compute for each cluster an Axis-Aligned Bounding Box (AABB) and a pair of clusters is considered close if their AABBs are in collision. Figure 3 illustrates the detection of close clusters in the case of auto-collision.

When a new pair of clusters is detected (i.e. a pair that was not detected in the previous time-step), their relative displacement is initialized to zero. A full collision detection method is used when their relative displacement reaches the displacement threshold *displacementThreshold*. As two clusters should not enter in collision before a full collision detection algorithm is used on them, we extend the AABB of each cluster by a distance of *displacementThreshold*/2 in every direction to enforce this property.

3) Measurement of relative displacement of neighboring clusters: The measure of relative displacement is performed on each pair of close clusters. For each pair of clusters $(c_1;c_2)$, the relative displacement in the current time-step t is computed from the absolute displacement of each cluster:

$$CurrRelDis_{avg}(c_1,c_2)_t = dis_{avg}(c_1) - dis_{avg}(c_2) \quad (1)$$

$$CurrRelDis_{dev}(c_1,c_2)_t = dis_{dev}(c_1) + dis_{dev}(c_2) \quad (2)$$

This measure of relative displacement for the current time-step (*CurrRelDis*) is then added to the complete measure of

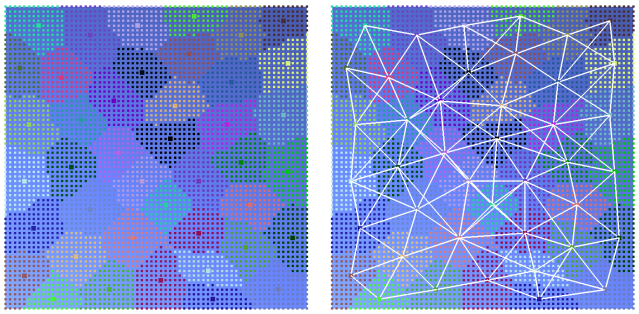

Fig. 3: Example of detection of close clusters. On the left, a squared piece of cloth (4,225 vertices) is decomposed into 42 clusters (the vertices are colored according to their affiliation to a cluster). On the right, the relation of closeness between pairs of clusters is displayed with white lines (which indicates collision between the cluster's AABBs in this specific geometric configuration). In this example, we have 121 pairs of close clusters.

relative displacement (*RelDis*) which stores the accumulated amount of relative displacement since the last use of a full collision detection algorithm (the formula is the same for both the average and the standard deviation):

$$RelDis(c_1,c_2)_t = RelDis(c_1,c_2)_{t-1} + CurrRelDis(c_1,c_2)_t \quad (3)$$

The amplitude of the relative displacement $\|RelDis(c_1,c_2)\|$ can then be computed:

$$\|RelDis(c_1,c_2)\| = \|RelDis_{avg}(c_1,c_2)\| + RelDis_{dev}(c_1,c_2) \quad (4)$$

When this amplitude exceeds $displacementThreshold$ a full collision detection algorithm is used and $RelDis(c_1,c_2)$ is reset to zero, otherwise an incremental collision detection algorithm is used. This distance has been empirically chosen as the best among several implemented metrics.

Our method for collision detection allows to measure more precisely the relative displacement between deformable objects, thus easing the use of incremental methods for collision detection. In particular our method can detect that two objects with similar velocities have low relative displacement, whereas an absolute displacement measurement would indicate that the two objects are undergoing high displacements.

In order to assert the efficiency of our method, we used a moving character wearing clothes (see Figure 5). If the character is immobile, a full collision detection takes on average 3.65ms (1.22ms standard deviation SD), with an absolute displacement measure, and 3.37ms ($SD = 0.75ms$) with a relative displacement measure, a rather similar amount. However, if the character is moving, the absolute displacement measure goes up to 11.76ms ($SD = 2.01ms$). As expected, our relative displacement measure brings that cost back to 3.18ms ($SD = 0.46ms$), because the relative movement between the body and the t-shirt is rather small, and allows a higher usage of incremental methods in the collision detection.

V. Physically-based simulation of tearing

In order to simulate deformable surfaces, such as cloth, FEM-based methods provide an accurate model for object deformation [NMK*06]. Our method relies on a co-rotational FEM for surface objects. For performance purposes, only the elastic deformation is taken into account, and no plasticity is computed. FEM-based methods compute stress based on the strain of elements. In order to simulate tearing, our method is inspired by the tearing simulation performed in [AMC09] for isotropic surfaces. Tearing occurs when the stress reaches a material-dependent threshold σ_E, as depicted in Figure 4. In the case of isotropic objects, the maximal stress is computed using an eigenvalue decomposition of the stress tensor. The maximal eigenvalue provides the maximal stress to be compared to the threshold. Several points can reach the threshold simultaneously, due to discrete time integration, in which case only the highest value is kept as candidate to start tearing. Once the threshold is reached, tearing starts orthogonal to the direction associated to the maximal stress, given by the eigenvector associated to the maximal eigenvalue. The orthogonal direction is chosen inside the plane around the tearing tip, and then projected on the mesh.

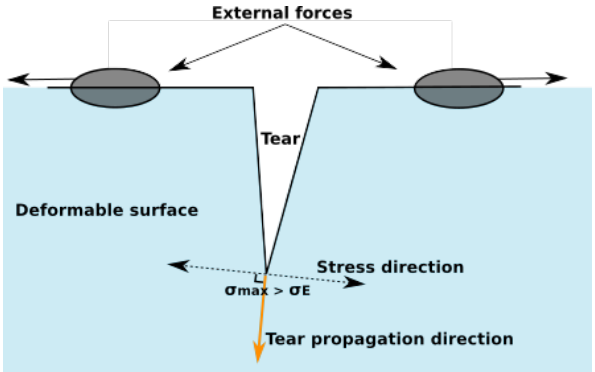

Fig. 4: Illustration of the tearing process: the surface is under stress generated by an external action (here two parts of the surface are pulled in opposite directions). Tearing starts when the local stress reaches the threshold σ_E, and the tear direction is orthogonal to the stress direction.

Computing the eigenvalues at each vertex of the mesh can prove costly in terms of computation. In order to address this, once started, we chose to propagate a tear only from the previous tearing tip. Stress eigenvalue decomposition is thus not performed on other vertices.

A loss of performance can also occur if too many elements are created during the tearing process. Several methods exist to address this issue. A first method would be to separate the mesh along existing edges, thus creating no new elements, similar to what is performed in [MMDJ01], but it creates visual artifacts, it is affected by the mesh resolution, and it does not provide an accurate tearing path. In our approach, a snapping method is used, such as described in [WWD14], which locally adapts the mesh to make it match the tearing path as best as possible. It thus creates a much less important

number of elements, allowing for rather stable performances during the simulation.

All these elements are gathered together to provide a tearing simulation, with performances compatible with haptic simulation. In particular, due to the snapping method, the performances are not significantly affected by the creation of new elements during the tearing process.

VI. Haptic rendering

On top of our physically-based simulator, we propose a coupling scheme to provide bimanual haptic interaction to the user. In our approach, haptics is handled in a separate thread with appropriate frequency. To each haptic device is associated a constrained part of the deformable object, representing the grasped handle of the object, and a virtual proxy to this handle. The proxy position is directly coupled to the device position, at haptic rate h_{rate}, as an affine function of the device position. The handle position is updated to the proxy position at simulation rate s_{rate}.

Force feedback is composed of several elements, given in equation (5). First, the force contribution of all the vertices in the handle, internal and external forces, are aggregated into a tearing force $\mathbf{F}_{tear}$, as shown in equation (6), updated at simulation rate. In order to provide a force at haptic rates, a proxy spring force is added, based on the relative positions of the proxy and the handle, and a damping factor is added on the velocity for stability purposes. This defines an impedance haptic scheme, such as described in [LO08].

$$\overbrace{\mathbf{F}_{tot}}^{h_{rate}} = \overbrace{\mathbf{F}_{tear}}^{s_{rate}} + \overbrace{\underbrace{k_{position} \times (\mathbf{x}_{proxy} - \mathbf{x}_{tool})}_{\text{Spring to the proxy position}} - \underbrace{k_{velocity} \times \mathbf{v}}_{\text{Damping}}}^{h_{rate}} \quad (5)$$

$$\mathbf{F}_{tear} = \sum_{i \text{ constrained}} (\mathbf{F}_{elastic}(\mathbf{x}_i) + \mathbf{F}_{external}(\mathbf{x}_i)) \quad (6)$$

This force is then scaled and provided as direct feedback. Such a scheme allows for a precise control of the deformation, and provides a direct force feedback. A tearing task usually involving two hands, our method allows for the use of multiple haptic devices simultaneously, in order to fully feel the forces involved in the tearing process.

VII. Use-cases and performance

A. Implementation setup

Our method has been implemented using SOFA framework [ACF*07] and runs on a PC (CPU: Intel Xeon E5-2620 2.1 GHz, GPU: AMD FirePro W9100, Memory: 32GB). Visualization is performed with Unity (www.unity3d.com). The haptic feedback is provided through a bimanual setup with two Novint Falcons, handled with chai3d framework (www.chai3d.org), that offers a device-transparent interface for haptic devices.

B. Illustrative use-cases

Our approach is illustrated through two use-cases using commonly torn material, paper and cloth. Both of our illustrative use-cases use a bimanual haptic setup. The first use-case illustrates a paper tearing scenario and involves a banknote being torn apart. Tearing starts almost immediately, since paper has a high Young Modulus and a relatively smaller stress threshold. This scenario involves only the autocollision computation. The second use-case features a superman-inspired t-shirt tearing scenario. The cloth tearing involves the entire simulation pipeline, namely collision detection with the environment, cloth simulation and tearing and bimanual haptic rendering, as well as a moving body. The full contact between the cloth and the body underneath is a really challenging scenario because it involves complex shapes and thus illustrates the efficiency of the collision detection algorithm. The use cases are depicted in Figure 5 and are also further illustrated in the accompanying video.

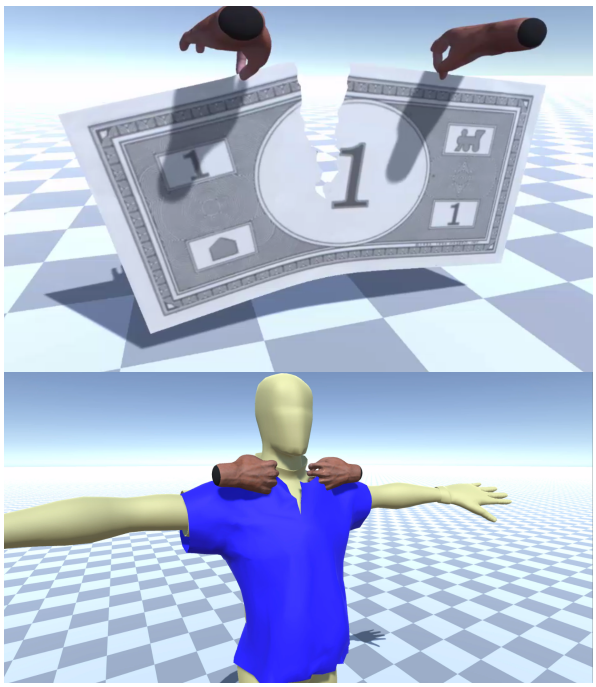

Fig. 5: Illustrative examples: (Top) a paper banknote is torn. (Bottom) a worn t-shirt where collision is activated for the full body is torn.

C. Performance

Table I shows the computation performance of our method, on several mesh sizes, corresponding to the square piece of cloth used as illustration of our setup in Figure 1, and the two use-cases displayed in Figure 5, the banknote and the t-shirt. Computation time is decomposed into three categories: collision detection, FEM deformation and tearing computation. Two time values are provided for the tearing contribution, since there is a difference between the computation before the tearing starts, with the computation of the principal stress for each vertex, and after the mesh starts tearing, with the

only computation of the tearing path. The objective total computation time corresponds to one time-step for the chosen illustrative use-cases. The performance table shows that the choice of propagating from an initial tear only induces a significant improvement in the performances as soon as the tearing starts, since computing the principal stress on one vertex and computing the tearing path requires significantly less computation than computing principal stress on every vertex. The T-shirt scenario is more complex in terms of collision, with full collision with the underlying body, which explains the more important computation time.

Scenario	# Triangles	Collision	FEM	Tearing	Total
Cloth	1.4K	1.8	8.7	3.6/2.2	14.1/12.7
Banknote	1.9K	2	11	4.5/2.3	17.5/15.3
T-shirt	3.3K	4.5	14	6/2.5	24.5/21

TABLE I: Computation time, in ms, decomposed in three parts: collision detection computation, FEM deformation computation, and tearing-exclusive computation. Two times are provided for the tearing contribution, before and after tearing has started, with and without the principal stress computation for each vertex.

Feedback force during the tearing simulation can be seen in Figure 6, with the force feedback intensity along the tearing axis for both hands. As expected, it can be decomposed in several clear phases, first an elastic deformation. The object then starts tearing, and the tearing propagates with an almost stable force at a fraction of the maximal force. Both hands have similar force profile (with opposite direction), the small difference coming from the not completely symmetric tearing scenario.

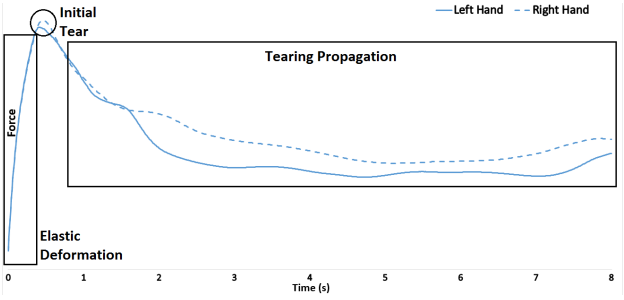

Fig. 6: Force feedback during the tearing process illustrating the 3 main phases for tearing simulation: initial elastic deformation, initial tearing, and tearing propagation. The force intensity along the horizontal axis is displayed for both hands.

VIII. Conclusion

In this paper, we presented a fully-functional interaction pipeline enabling haptic tearing of deformable surfaces. It relies on both a highly efficient collision detection algorithm and an efficient FEM-based tearing simulation method. We proposed an improved method for collision detection of moving objects with little relative movement, such as a worn cloth, relying on the clusterization of the collision models. The tearing method is compatible with the use of multiple haptic devices simultaneously, in order to provide a realistic tearing experience. It has been implemented on two different examples illustrating potential applications of our method.

Future work could include the tearing simulation of objects undergoing more complex phenomena such as plastic deformation or anisotropy. Concerning collision detection, a dynamic re-computation of the cluster distribution could be performed when the cluster is separated into several non-connected pieces, thus improving the computation time for collision detection. Finally, an extension of the method to volume objects could broaden the spectrum of possible applications.

References

[ACF*07] Allard J., Cotin S., Faure F., Bensoussan P.-J., Poyer F., Duriez C., Delingette H., Grisoni L.: Sofa-an open source framework for medical simulation. *Medicine Meets Virtual Reality 125* (2007), 13–18.

[AMC09] Allard J., Marchal M., Cotin S.: Fiber-based fracture model for simulating soft tissue tearing. *Studies in health technology and informatics 142* (2009), 13–18.

[BJ08] Barbič J., James D. L.: Six-dof haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Transactions on Haptics 1*, 1 (2008).

[GLGT05] Gregory A., Lin M. C., Gottschalk S., Taylor R.: A framework for fast and accurate collision detection for haptic interaction. In *ACM SIGGRAPH 2005 Courses* (2005), p. 34.

[GSH*04] Gingold Y., Secord A., Han J. Y., Grinspun E., Zorin D.: A discrete model for inelastic deformation of thin shells. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004).

[LGA13] Lehericey F., Gouranton V., Arnaldi B.: New iterative ray-traced collision detection algorithm for gpu architectures. In *Proc. of the ACM Symposium on Virtual Reality Software and Technology* (2013).

[LGA15] Lehericey F., Gouranton V., Arnaldi B.: Gpu ray-traced collision detection for cloth simulation. In *Proc. of the ACM Symposium on Virtual Reality Software and Technology* (2015).

[LO08] Lin M. C., Otaduy M. A.: *Haptic Rendering : Foundations, Algorithms, and Applications*. A K Peters, Ltd., 2008.

[MMDJ01] Müller M., McMillan L., Dorsey J., Jagnow R.: Real-time simulation of deformation and fracture of stiff materials. In *Proc. of Computer Animation and Simulation*. 2001.

[NMK*06] Nealen A., Müller M., Keiser R., Boxerman E., Carlson M.: Physically based deformable models in computer graphics. In *Computer Graphics Forum* (2006), vol. 25, pp. 809–836.

[NSO12] Narain R., Samii A., O'Brien J. F.: Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics* (2012).

[PNdJO14] Pfaff T., Narain R., de Joya J. M., OBrien J. F.: Adaptive tearing and cracking of thin sheets. *ACM Transactions on Graphics 33*, 4 (2014).

[SWC14] Souza M. S., Wangenheim A., Comunello E.: Fast simulation of cloth tearing. *SBC Journal on Interactive Systems* (2014).

[TF88] Terzopoulos D., Fleischer K.: Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proc. of ACM Siggraph Computer Graphics* (1988), vol. 22, pp. 269–278.

[WWD14] Wu J., Westermann R., Dick C.: Physically-based simulation of cuts in deformable bodies: A survey. In *Proc. of Eurographics (State of the Art Reports)* (2014), pp. 1–19.