# A Unified Topological-Physical Model for Adaptive Refinement

Elsa Fléchon, Florence Zara, Guillaume Damiand, Fabrice Jaillet

# A Unified Topological-Physical Model for Adaptive Refinement

Elsa Fléchon[a], Florence Zara[a], Guillaume Damiand[a], Fabrice Jaillet[a,b]
[a]Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, F-69100, Villeurbanne, France
[b]Université de Lyon, IUT Lyon 1, Computer Science Department, F-01000, Bourg-en-Bresse, France
firstname.name@liris.cnrs.fr

**Abstract**

*In Computer Graphics, physically-based simulation of deformable objects is a current challenge, and many efficient models have been developed to reach real-time performance. However, these models are often limited when complex interactions involving topological modifications are required. To overcome this, the key issue is to manage concurrently, and at minimal cost, both the topology and physical properties.*
*Thus, this paper presents a unified topological-physical model for soft body simulation. The complete embedding of physical and topological models will facilitate operations like piercing, fracture or cutting, as well as adaptive refinement. Indeed, the difficulty is to treat topological changes during the simulation, requiring combined geometric and physics considerations. Rigorous topological operations guarantee the validity of the mesh, while direct access to the adjacent and incident relations will ease the update of physical properties of new elements created during these operations.*
*These features are illustrated on an embedded mass-spring system undergoing topological modifications performed during simulation. Different levels of subdivision are also presented.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically-based modeling I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and geometric transformations

## 1. Introduction

A current challenge in Computer Graphics consists in achieving interactive physically-based simulations. In many areas, such as entertainment application, manufacturing or medical training, the goal is to simulate the realistic behavior of deformable objects in interaction with their environment in real time. Additionally, topological modifications have to be handled efficiently, to enable operations like tearing, cutting, fracture, etc.

For this, two physical models are classically used [NMK*06, Pay12], both underlying on a mesh. The finite elements method (FEM) is based on continuum mechanics. This allows to solve the partial differential equations of the problem on irregular grids by spatially discretizing the objects into elements (typically tetrahedra or hexahedra in 3D). The resulting solutions are quite accurate, but this method is known to be computationally expensive especially for interactive event management, despite numerous optimizations recently proposed. The mass-spring system (MSS) is a valuable alternative approach in this case. It consists in discretizing the object into a set of particles connected by springs. Forces representation is very simple, and can be computed for each node (instead of element), allowing for very fast computation. The spring parameters can be linked to the mechanical properties of the object to improve the accuracy of the discrete model [LSH07, BBJ*09]. In this paper, to illustrate our approach, we focus on this second method that allows for better control in the integration of external interactions or topology alterations.

Furthermore, to improve the tradeoff between accuracy and computation time for these models, a typical solution considers relatively coarse initial meshes that will be locally refined when and where required. For example, the refinement can be restricted to the elements undergoing external actions, such as contacts with a surgical tool, or along a fine cutting path inside elements. The refinement can also be driven by some criteria, such as error estimators that may

Authors version of paper published in Proc. of 11th Workshop on Virtual Reality Interaction and Physical Simulation, Eurographics Digital Library, September 2014. Original version is available http://dx.doi.org/10.2312/vriphys.20141222

be based on geometry or energy, or on distance to camera for Level Of Detail strategies.

This paper proposes an extension of the work of [FZDJ13] which introduced a unified topological-physical model (called LCC+MSS) adapted for a mass-spring simulation that handles cutting or piercing simulated objects. An interesting feature of this work is the use of the same model to define the topology of the mesh (*i.e.*, its subdivision in cells plus all the incidence and adjacency relations), its geometry (the coordinates of the vertices) and the physical properties of the system (mass of particles, stiffness of springs, etc.) altogether. Moreover, on the fly topological modifications are also simply allowed by updating the model.

The main contribution of our work is the definition of adaptive refinement operations during 2D or 3D simulation. Our paper addresses the following problems (i) the local modification of the topological description of the mesh, (ii) the assignment of the physical properties of elements newly created during the refinement according to those of initial elements and (iii) the updating of physical properties of existing elements. Moreover, the method handles T-junctions in order to support local refinement.

The rest of this paper is organized as follows. Section 2 presents previous works and Section 3 describes the initial LCC+MSS model which serves as preliminary work. Section 4 is the core of our contribution, and presents the proposed modifications to tackle the problem of refinement during the simulation. Then, Sections 5 and 6 present in details the refinement process for 2D and 3D embedded MSS, with emphasis on adaptive refinement in Section 7. Finally, results are given in Section 8, and Section 9 concludes this paper with future work.

## 2. Related work

**Adaptive refinement of surfaces.** As stated in [AUGA05], surface remeshing is a key component in many applications involving geometric manipulations. The simulation of deformable objects is not an exception, but in this case the refinement is usually error-driven to optimize the trade-off between accuracy and computation time. In this sense, Hutchinson *et al.* [HPH96] proposed a method that both improves the accuracy and reduces the computational cost for cloth simulations. During the animation, a MSS is adaptively refined by adding masses and springs according to a geometric criterion based on angles between springs. In addition, a hierarchical structure will ensure the correct behavior of the refined meshes. The challenge here is to preserve the initial mass and stiffness, and to define the time step according to the subdivision level. In the same context, Villard *et al.* [VB05] proposed a method to adaptively refine a mesh, but this time without multi-resolution grids. Non conformal meshes (including T-junctions) are handled by defining active and virtual particles which are processed differently. Lee *et al.* [LYO*10] presented a multi-resolution method for cloth simulation with a triangle-based energy model. The simulation begins with a high-resolution mesh. The size of the linear system is then reduced using an adaptive mesh. Regions to be simplified are then detected according to criteria depending on stretching, shearing, bending or particles status, or according to collision. More recently, Busaryev *et al.* [BDW13] simulated fractures on multi-layered thin plate objects (such as paper), based on FEM. However, a fracture-aware remeshing scheme based on constrained Delaunay triangulation is employed beforehand, that cannot be applied during the simulation.

**Adaptive refinement of volumes.** Burkhart *et al.* [BHU10] presented a method for FE analysis using Catmull-Clark solids but restricted to three-manifolds meshes of hexahedra. The advantage of this approach is the use of a unified representation for both the geometry and the physical simulation. However, the use of a surface or solid subdivision scheme cannot mix several kind of elements, nor elements with different levels of refinement in the same mesh. Wicke *et al.* [WRK*10] used a physical model based on a linear co-rotational finite element formulation. To avoid degradation of tetrahedral elements quality, they remeshed small parts of the object during the simulation. This is based on a smoothing method which consists in moving vertices to improve the quality of incident elements without topological changes. Thus, they cannot subdivide individual tetrahedra into smaller ones.

**Cutting.** Refinement takes part in the problematic of cutting simulation, for example, by allowing for the following of a fine cutting path. The recent state of the art of Wu *et al.* [WWD14] summarizes works about virtual cutting on deformable objects. In this sense, Dick *et al.* [DGW11] proposed a model for simulating cuts in objects. The model is composed of an adaptive octree grid (to represent cuts at a very fine scale), a multigrid (to run the simulation according to topological changes) and a simulation grid based on FEM. Jeřábková *et al.* [JBB*10] presented a model using two level hierarchies of hexahedral elements: coarse level for simulation purpose and fine level of detail for visualization and collision. Cutting is performed by removing hexahedra at the finer level.

**Topological model.** In 2013, Untereiner *et al.* [UCB13] demonstrated the benefit of using combinatorial maps for the representation of 3D multi-resolution meshes. This model is capable of handling different levels of subdivision with arbitrary elements and supports adaptive subdivision, but at high memory cost. They applied this approach on an electro-thermal simulator, but neither the geometry, nor the topology is modified during the simulation.

**Topological model for physical simulation.** Amongst the few attempts to explore the combination between topological modeling and physical simulation, Meseure *et al.* [MDS10a, MDS10b, DKS*11] presented an approach

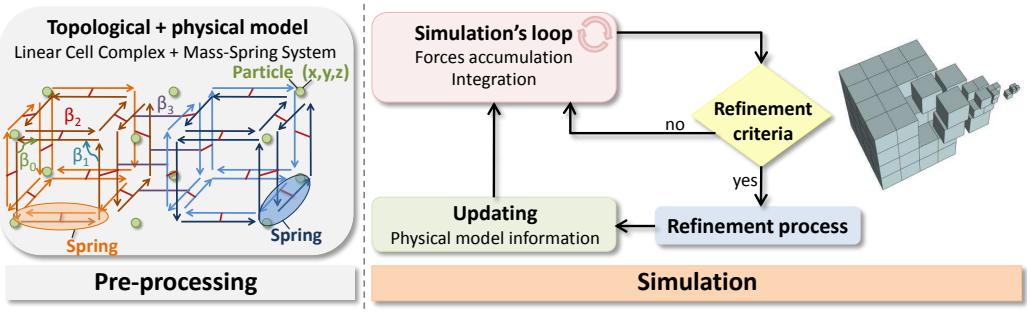



Figure 1: Overview of our method. Our topological data structure stores the geometry and physical properties of the deformable object modeled with a MSS. During the simulation, this data structure is updated to take into account topological modifications such as refinement of elements.

based on mass-interaction and generalized maps (a variant of combinatorial maps). The model allows topological changes such as element removing, or even cutting. Jund *et al.* [JAD*12] also used generalized maps with a meshless physical model. By mapping a set of particles to the cells of the topological model, their framework allows for the association of any volume shape to any point based motion. This way, they are able to handle topological changes, shown on tearing experiments. However, even if the topological and physical models are strongly connected, in the end two different structures still remain to be maintained, at least for external interactions.

Fléchon *et al.* [FZDJ13] presented a hybrid model to simulate soft body deformations. A mass-spring system (MSS) is fully embedded into a topological model based on Linear Cell Complex (LCC). The great advantage of this model is to propose a unified framework for topology and physically-based simulation, that will simplify many topological operations and will ensure to always preserve the model integrity. So far, local or adaptive refinement is not possible as appropriate subdivision of elements is not proposed.

**Adaptive refinement of topological models for physical simulation.** Choi *et al.* [CHCK02] proposed the only adaptive refinement method with a topological model we are aware of. A topological model (based on winged-edge) and an MSS physical model are combined to deal with adaptive refinement during the simulation. A multi-resolution mass-spring model is locally refined. Moreover, they used surface wavelets to switch between the different levels of subdivision of the model. However, this method was developed for 2D triangular meshes, and its edge-based design cannot be easily extended to 3D.

**T-junctions.** One issue that may eventually appear in refinement algorithms that treat elements in isolation, is the presence of *T-junctions*. It is possible to address this problem either by avoiding *non-conformal* elements like the red-green method of Molino et al. [MBTF03] or by mixing different types of elements [Lob13]. Others treat this case by snapping *T-junction* particles on edges or faces of adjacent elements or by using other models that do not necessitate any special treatment [GKS02, NFP06, SSH10].

In this paper, to overcome the different limitations encountered in previous work, we proposed a unified model avoiding redundancy and facilitating the update of information. In addition, local adaptation is possible both for 2D and 3D meshes during the physical simulation.

## 3. LCC+MSS model

Mass-spring systems have largely been used in animation [TW90, TPF91, ZGF07] because of their intuitive implementation. Objects are discretized into a set of masses (also called particles) inter-connected by springs. The dynamics of the model are computed with the classic simulation loop [NMK*06]: (1) computation of the forces applied on the particles (due to springs and external forces); (2) computation of the acceleration of each particle according to Newton's second law: for each particle $v$, we have at time $t$:

$$m_v \frac{d^2}{dt^2}\mathbf{P}_v(t) = \mathbf{F}_v(t) \quad (1)$$

where $m_v$, $\mathbf{P}_v$ and $\mathbf{F}_v$ are respectively the mass, position and the sum of the forces applied to the particle $v$; (3) computation of the velocity and position of each particle using numerical integration schemes (for example, Euler semi-implicit or implicit method).

The LCC+MSS model proposed in [FZDJ13] uses, as underlying data structure, the 3D linear cell complexes [Dam12] from the CGAL Open Source geometric algorithms library [The12]. This topological structure is able to represent an orientable 3D object thanks to 3D combinatorial maps, called 3-maps [Lie94]. A 3-map describes the subdivision of a 3D object as a set of $i$-cells, $i \in \{0,1,2,3\}$. The 0-, 1-, 2-, 3-cells correspond respectively to vertices, edges, faces and volumes. They are described by darts which generalize half-edges [Män87] to $n$-dimension, plus pointers between these darts named $\beta_i$ with $i \in \{0,1,2,3\}$. Two

$i$-cells are adjacent if they share a common $(i-1)$-cell (for example, two volumes are adjacent if they share a common face); and two cells are incident if one belongs to the boundary of the other (for example a vertex is incident to a face if it belongs to its boundary).

The 3-map shown in Fig. 1 represents two adjacent cubes. Darts are displayed as arrows. $\beta_1(d)$ is the next dart following dart $d$ in the same face and the same volume. $\beta_0(d)$ is the previous dart in the same face and same volume. $\beta_2(d)$ gives the other dart from $d$ belonging to the same edge, same volume but not the same face (eventually represented by small segments in red in Figures 1, 5 and 6). Lastly, $\beta_3(d)$ gives the other dart from $d$ belonging to the same edge, same face but not the same volume. Also, some properties are defined on these pointers to guarantee the topological validity of the described objects [Lie94].

Thanks to these darts and pointers, the main benefit of 3-maps is to describe the cells of 3D objects as well as the incidence and adjacency relationships. This information is very useful in algorithms to compute and update physical properties while allowing for topological modifications. Moreover, the use of a topological model ensures the permanent topological validity of the mesh after modification.

The construction of a LCC+MSS starts with the description of a mesh. A 3-map is built according to this input data. Then, the coordinates of the vertices of the mesh are added to the combinatorial map to obtain a Linear Cell Complex (LCC). The construction of the LCC+MSS is finalized to fit MSS requirements: the data structure `Particle` is associated to each vertex and the data structure `Spring` is associated to each edge, as seen in [FZDJ13].

The `Particle` structure contains information related to the MSS such as its mass, acceleration, velocity or the sum of the forces applied to this particle. Note that in 2D (resp. 3D), each quad (resp. hexahedron) has 2 (resp. 4) internal diagonal springs (as shown in Fig. 2 for 3D case). Each particle stores a list of all the diagonal springs attached to it. Note that the geometric coordinates of the particle are directly associated to the corresponding 0-cell in the 3D LCC.

The `Spring` structure contains the physical properties of the spring: its initial length, stiffness and its two extremities (*i.e.*, pointers to the two `Particles` connected by the considered spring). The formulation of the spring stiffness constant is based on Baudet *et al.* [BBJ*09] to take into account the mechanical properties (Young's modulus and Poisson's ratio) of the simulated object. In this paper, surfaces (*resp.* volumes) are represented by quads meshes (*resp.* hexahedral meshes) and the cells of highest dimension are called *elements*, *i.e.*, quads in 2D or hexahedra in 3D.

The different information associated with particles and springs can be initialized thanks to geometrical and topological information. To compute the mass $m_v$ of a particle $v$, the contribution of each element incident to $v$ is accumulated:

$$m_v = \frac{1}{c} \sum_{e \in E_v} \rho_e \, \mathcal{D}_e$$

with $E_v$ the set of elements incident to $v$, $\rho_e$ the density and $\mathcal{D}_e$ the area of element $e$ in 2D (*resp.* volume in 3D), and $c$ a constant equal to 4 in 2D (*resp.* 8 in 3D).

For the stiffness of each spring $s$, we accumulate the contribution of each element incident to $s$. The stiffness constant associated to the length $i$, for a 2D rectangular element $E$ of length $i$ and $j$, is given by [BBJ*09]:

$$k_E(i,j) = \frac{\mathsf{E}\left(j^2\,(3\,\nu+2) - i^2\right)}{4\;i\;j\,(1+\nu)}$$

with $\mathsf{E}$, $\nu$ respectively the Young's modulus and the Poisson's ratio of the object (or element for heterogeneous object). Thus, the two stiffness constants of a 2D element $E$ of length $l_X$ and $l_Y$ are defined by $k_E(l_X,l_Y)$ and $k_E(l_Y,l_X)$ with $\{i,j\} \in \{l_X,l_Y\}$ and $\{i \neq j\}$. For a 3D element $E$ of sizes $i$, $j$ and $k$, the stiffness constant associated to the length $i$ is:

$$k_E(i,j,k) = \frac{\mathsf{E}\left(6\;j^2k^2\,(1+\nu) + (\nu(j^2+k^2) - i^2)\left(i^2+j^2+k^2\right)\right)}{24\;(1+\nu)\;i\;j\;k}$$

Thus, the three stiffness constants of a 3D element $E$ of sizes $l_X$, $l_Y$ and $l_Z$ are $k_E(l_X,l_Y,l_Z)$, $k_E(l_Y,l_X,l_Z)$ and $k_E(l_Z,l_X,l_Y)$ with $\{i,j,k\} \in \{l_X,l_Y,l_Z\}$ and $\{i \neq j \neq k\}$.

In 2D, 4 internal diagonal springs $k_d$ are created. Their stiffness constant is computed with the following formula:

$$k_d = \frac{\mathsf{E}\left({l_X}^2 + {l_Y}^2\right)}{4\;l_X\;l_Y\,(1+\nu)}$$

The same goes in 3D, with 8 internal diagonal springs:

$$k_d = \frac{\mathsf{E}\left({l_X}^2 + {l_Y}^2 + {l_Z}^2\right)^2}{24\;(1+\nu)\;l_X\;l_Y\;l_Z}$$

## 4. LCC+MSS extension for adaptive refinement

In [FZDJ13], the LCC+MSS structure is used for a MSS simulation allowing topological changes. This paper proposes an extension of this model to handle complex topological operations, like local mesh refinement, while keeping the possibility to cut or pierce objects. The principle of a single data structure is preserved to store the physical properties and geometric and topological information of the mesh. The basic tool of the adaptive refinement process during the simulation is the *local subdivision* of an element (*local* means that only one element is modified, not its neighbors).

Without adaptive refinement, each quad is always composed of 4 edges, and each hexahedron is always composed of 6 quads. This is not true anymore after local subdivisions, since the side of an element can be composed of several edges in 2D, and several quads in 3D (see Figs. 3 and 4).

To define our local refinement algorithm, all the edges (*resp.* quads) belonging to the same side of an element must be retrieved. For that, the following notions are introduced:

- The *subdivision level* of an element $E$ (*resp.* an edge $e$), noted $l_E$ (*resp.* $l_e$), is related to the number of times $E$ (resp. $e$) is subdivided. These levels are initialized to 1 and updated during the refinement process (multiplied by 2 at each new refinement).
- A *hierarchical face* (*resp. hierarchical edge*) is the set of faces (*resp.* edges) generated by the subdivisions of the same initial face (*resp.* edge). For a non-subdivided face (*resp.* edge), it corresponds to the face (*resp.* edge) itself. Hierarchical faces only exist in 3D, while hierarchical edges exist in 2D and 3D.
- A *corner particle* is one extremity of a hierarchical edge. In 2D, each face (*resp.* hierarchical face in 3D) has always 4 hierarchical edges and 4 corner particles.
- A *border T-junction* is a particle which is inside a hierarchical edge (*i.e.* it is not an extremity of a hierarchical edge). Each border T-junction stores the two corner particles of its hierarchical edge.
- A *middle T-junction* (which only exists in 3D) is a particle which is inside a hierarchical face (*i.e.* not on the border of a hierarchical face). Each middle T-junction stores the four corner particles of its hierarchical face.
- A face (*resp.* volume) is said *non-conformal* if it has at least one T-junction particle (border or middle) in addition to its corner particles. An element is *non-conformal related to* a T-junction if the dart belonging to the element and incident to the T-junction is not hierarchical.
- A *hierarchical dart* is the first dart of a *hierarchical edge*. All the darts are initialized as hierarchical, and updated during split operations. These darts allow for the retrieval of all the corner particles.

This additional information allows the local refinement process without any geometric computation, avoiding complex computation and possible rounding error. Indeed, we make no angle calculation to retrieve the different sides of an element, nor any length computation between the extremities of an edge to find, for example, the middle dart of an element side. Fig. 2 summarizes the information contained in the `Particle` and `Spring` structures associated to each vertex, edge, face or volume of a 3D LCC.

In Fig. 3, the 2D model was initially composed of 4 quads $E_0$, $E_1$, $E_2$ and $E_3$. $E_1$ was subdivided into $E_{10}$, $E_{11}$, $E_{12}$ and $E_{13}$, followed by the subdivision of $E_{10}$. Different subdivision levels of elements coexist, for example $l_E(E_0) = 1$, $l_E(E_{11}) = 2$ and $l_E(E_{100}) = 4$. For edges, $l_e(e_0) = 1$, $l_e(e_3) = 2$ and $l_e(e_2) = 4$. The left hierarchical edge $e_0$ of $E_0$ is only composed of one edge, while the right hierarchical edge is composed of 3 edges $(e_1, e_2, e_3)$. $E_0$, $E_{12}$ and $E_{23}$ are non-conformal, while $E_{20}$ and $E_{300}$ are conformal.

In Fig. 4, the 3D model was initially composed of 6 hexahedra. The central hexahedron is not subdivided (thus its subdivision level is equal to 1), while some of its neighbors are. For this reason, the hierarchical faces of the central hexahedron are eventually composed of several faces, and the central hexahedron is non-conformal. The following subdivision levels of edges coexist, $l_e(e_0) = 1$, $l_e(e_1) = 2$ and $l_e(e_2) = 4$. The hierarchical face $F_0$ is non-conformal, even if it contains only one face. Indeed, $F_0$ has no middle T-junction, while it has 8 border T-junctions. The hierarchical face $F_1$ is non-conformal as well, contains 7 faces, has 4 middle T-junctions and 8 border T-junctions.

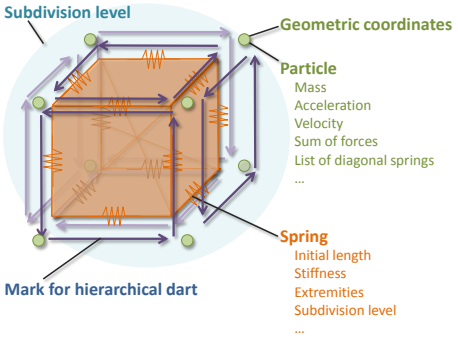



Figure 2: A 3D LCC with additional information for an hexahedral MSS. `Particle` and `Geometric coordinates` are associated to all 0-cells. `Spring` is associated to all 1-cells, and a mark identifies all the hierarchical darts. The MSS is composed of springs on the edges of the mesh, and internal diagonal springs in the elements.

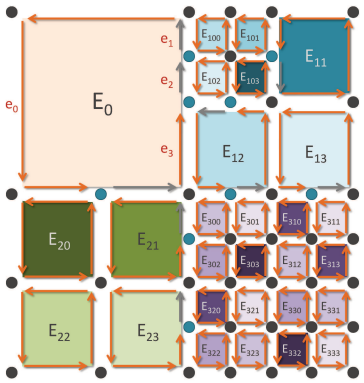
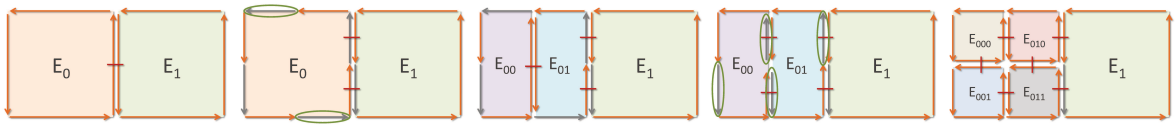



Figure 3: 2D model showing a set of quads with different subdivision levels. Corner particles are drawn in black, border T-junctions in blue and hierarchical darts in orange.

## 5. 2D regular refinement 1:4

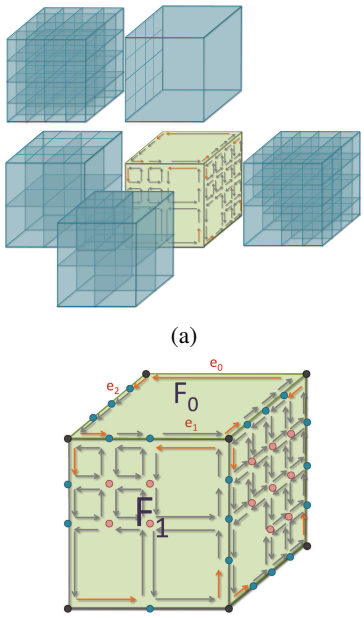
In this section, we present our approach to locally refine a 2D LCC+MSS model. Firstly, we explain the topological part of our refinement process, and secondly its physical one.

**Topological refinement.** Fig. 5 illustrates the refinement steps on a 2D object composed of two adjacent quads $E_0$




(a) (b) (c) (d) (d)

Figure 5: $E_0$ and $E_1$ are two adjacent quads having the same subdivision level. (b-d) The different steps for the subdivision of $E_0$ into four sub-quads: $E_{000}$, $E_{001}$, $E_{010}$ and $E_{011}$.




(a)

(b)

Figure 4: (a) 3D model showing a hexahedron (in green) adjacent to more subdivided hexahedra (in blue). (b) Zoom on the central green element. Corner particles are drawn in black, middle T-junctions in pink and border T-junctions in blue, hierarchical darts are drawn in orange.

and $E_1$ at the same initial subdivision level. All edges have the same subdivision level (equal to 1 if never subdivided) and all darts are marked as hierarchical (see Fig. 5($a$)). Element $E_0$ is refined in a regular way in 4 sub-quads (called regular refinement 1:4), see Fig. 5(e). The subdivision process, based on the insertion operations of the combinatorial maps [BADSM08], is done in four steps:

1. A new vertex is inserted in the middle of each edge of $E_0$. The insertion of a vertex splits the initial edge in two. The subdivision level of the resulting darts is twice that of the initial darts. New particles are associated to the new vertices, and new springs are associated to the new edges. In the current example, this operation creates 4 new darts drawn in gray in Fig. 5($b$).
2. A new edge is inserted between the two darts associated with the two vertices in the middle of two opposite hierarchical edges of $E_0$ (the two darts circled in Fig. 5($b$)). The insertion of the edge creates two darts (see Fig. 5($c$)). The subdivision level of the new edge is initialized to $2l_{E_0}$, which corresponds to the value obtained after the subdivision process. The two darts of the edge are marked hierarchical. After this step, $E_0$ is now subdivided in two 2-cells: $E_{00}$ and $E_{01}$.
3. A new vertex is inserted in the middle of the edge previously created, similarly to step 1 (see Fig. 5(d)), associated with a new particle. Two new springs are created and associated with the two new edges.
4. The two new elements created, $E_{00}$ and $E_{01}$, are then subdivided as in step 2: two edges are inserted by using two opposite darts belonging to the same face. The subdivision levels of the two new edges are initialized to $2l_{E_0}$. A spring is associated to each new edge, and the new darts created are all marked as hierarchical. This operation induces the creation of four new elements: $E_{000}$, $E_{001}$, $E_{010}$ and $E_{011}$ (see Fig. 5(e)).

This refinement process involves that the adjacent elements to the element to be refined have a smaller or equal subdivision level. This is not necessarily true and thus this must considered during the subdivision process.

Let us reconsider the example given in Fig. 3. The first step to subdivide $E_0$ consists in inserting a vertex in the middle of its hierarchical edges. But since $E_0$ is adjacent to elements already refined (the refinement of $E_1$ and $E_2$ followed by the refinement of $E_{10}$), its right and bottom hierarchical edges are already subdivided. Consequently, the vertices in the middle of these two hierarchical edges already exist, and hence no new vertex has to be inserted.

To test if an edge $e$ of an element $E$ has to be split in two or not, its subdivision level $l_e$ is compared to the level of the element $l_E$. If they are equal, the edge is not subdivided and thus a vertex must be inserted. Otherwise, the edge is already subdivided and there is no need to insert a new vertex. A second modification of the subdivision process appears when we recover the darts associated with the vertices in the middle of a hierarchical edge. The darts circled in Fig. 5($b$) or Fig. 5(d) are obtained by $\beta_1$ from a hierarchical dart. But, this is not true anymore if the hierarchical edge has already been subdivided. To obtain this middle dart, $S$ is computed as the sum of the inverse of the subdivision level $1/l_e$ of all the edges $e$ of the hierarchical edge. Then, the method starts

from the corresponding hierarchical dart and iterates $\beta_1$, accumulating $1/l_e$, until $S/2$ is reached.

**Physical properties updating.** During the subdivision process, the insertion of vertices and edges involves the addition of particles and springs in the MSS. Their properties are initialized or updated to preserve the initial physical properties of the object.

- ***Mass***. The mass of all the particles incident to the newly subdivided face must be initialized or updated. We proceed as seen in Section 3. However, a specific treatment must be done for T-junction particles (explained below).
- ***Diagonal spring***. When a quad is subdivided into four, its two initial diagonal springs are removed, and two diagonal springs are created for each new sub-quad (while updating the lists of all the diagonal springs linked to the particles at their extremities). The stiffness constant of new springs are computed again as previously described (Sec. 3). Their initial lengths are computed with the initial lengths of their element, relatively to the level of subdivision of the considered element.
- ***Spring***. For a face newly refined, the set of springs associated to the edges are updated and new springs are initialized. It is done as in the initialization process. The initial length of each spring is updated by division by $l_e$. Therefore, the stiffness constants are computed relatively to the hierarchical edges and not to the considered edges. The stiffness constant $k_{E|e}(i,j)$ of a spring associated to edge $e$ (of subdivision level $l_e$) of the element $E$ (of subdivision level $l_E$) is computed with the following formulation:

$$k_{E|e}(i,j) = k_E(i,j)\frac{l_e}{l_E}$$

  where $k_E(i,j)$ is the stiffness constant of the hierarchical edge containing edge $e$. The dimensions of $E$ are $l_X$ and $l_Y$ corresponding to the initial lengths of the springs associated to two successive hierarchical edges of $E$. $\{i,j\} \in \{l_X, l_Y\}$ and $\{i \neq j\}$.
- ***Velocity and acceleration***. To avoid slowing down the system, the velocity and acceleration of new particles are initialized from their neighborhood. When a particle is inserted in an edge (step 2 of the refinement process), its velocity (*resp.* acceleration) is the average of the one of the extremities of the considered edge. For the particle inserted in the middle of the face to be refined (step 4 of the refinement process), its velocity (*resp.* acceleration) is the average of the four particles corresponding to the first extremities of the hierarchical darts of the initial face.
- ***Border T-junction***. During the refinement process, if a hierarchical edge incident to two faces is subdivided in two, the new particle becomes a border T-junction. In Fig. 6, $v_{01}$ is a T-junction which does not exist for the *non-conformal* element $E_0$. Consequently, $v_{01}$ is constrained to remain on the edge ($v_1$, $v_4$) to avoid inconsistencies in the physical simulation (as shown in Fig. 6(*b*) where $v_{01}$ enters inside $E_0$). The T-junction is projected orthogonally on its corresponding hierarchical edge at each iteration (illustrated by Fig. 6(*c*)). Its mass is computed according to its incident faces ($E_{10}$ and $E_{12}$) which are *conformal* related to it. Moreover, to conserve the global force of a conformal element, the force of each T-junction is distributed to the particles of the incident conformal elements. In our example, the force of $v_{01}$ is distributed to $v_1$, $v_4$, $v_{04}$, $v_{00}$ and a double weight is used for $v_{02}$ by construction.

## 6. 3D regular refinement 1:8

The 2D refinement (sec. 5) is now extended to the 3D case.

**Topological refinement.** Fig. 7 illustrates the six refinement steps of a 3D model composed of 1 hexahedron $E$ which is split into eight (called regular refinement 1:8).

1. The six faces of $E$ are subdivided in four, creating new points/particles and darts/springs (the new 0-cells are represented in red in (*b*)). The 2D refinement process is used.
2. A 2-cell (see (*c*)) is inserted in the middle of the 3-cell, splitting $E$ into two parts (blue and green hexahedra). This face is inserted along a list of darts forming a cycle in the middle of the four lateral faces.
3. The new face is split into four (the new 0-cells and 1-cells are represented in red in (*d*)) using again the 2D refinement process.
4. A 2-cell is inserted in the two new hexahedra (see (e)).
5. The two new faces are split into two (the new 1-cells are red in (f))).
6. A 2-cell is inserted in the four new hexahedra (see (g)).

**Physical properties updating.** In addition to particles and springs corresponding to the edges, four internal diagonal springs have to be updated for each element. For a hexahedron, the four initial diagonal springs are removed, and four new diagonal springs are created for each new hexahedron. The computation of the stiffness constant of the new springs is done similarly than for the initialization (Sec. 3).

As in 2D, T-junctions are projected orthogonally on their associated cell to avoid physical inconsistences: border T-junctions are projected on their associated hierarchical edge, while middle T-junctions are projected on their associated hierarchical face. The projections are done using the corner particles associated to T-junctions.

## 7. Adaptive refinement

Local refinement can be used, both for 2D and 3D meshes, to propose an adaptive refinement method. Thus, during the simulation, each element can be tested and subdivided (by using the local refinement presented in previous sections) if a given criterion is satisfied. The main benefit of this adaptive refinement method is to automatically adapt the mesh depending on the need of the simulation.

We chose a geometric criteria based on the deformation of internal diagonal springs. An element is subdivided if

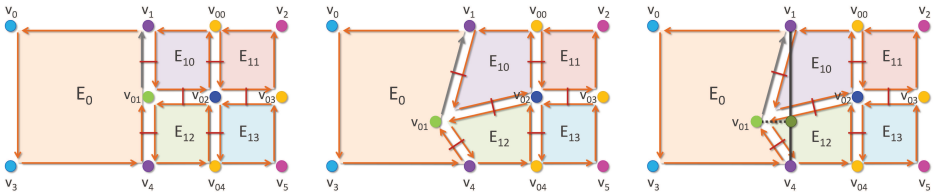



Figure 6: We handle *T-junctions* by orthogonal projection on their hierarchical edge. Particles of the same color have the same mass.

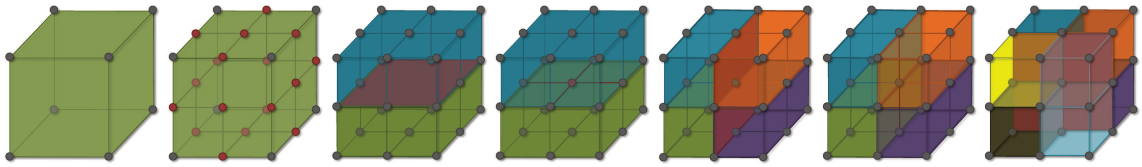



Figure 7: The different steps of the subdivision of a hexahedron into eight sub-hexahedra.

one of its diagonal springs satisfies the following formula:

$$\frac{|l - l_0|}{l_0} * 100.0 > threshold$$

where $l$ and $l_0$ are respectively the length at time $t$ and the initial length of the diagonal spring.

Other criteria can be defined using, and possibly mixing, the geometry of the elements or their physical properties.

## 8. Results

Our adaptive refinement method was implemented in the generic topological framework of Fléchon *et al.* [FZDJ13] by improving the LCC+MSS model.

The results shown in Fig. 8 illustrate that our method handles adaptive refinement during the simulation while allowing cutting operations. Another experiment also exhibiting adaptive 3*D* refinement is shown Fig. 9. The automatic refinement of a 3D mesh can be observed during its fall on a spike. Fig. 10 illustrates the capability of our method to describe objects with hexahedra of different subdivision levels.

A comparison between our adaptive method and the results obtained without adaptive refinement is given in Fig. 11 These results show that the adaptive refinement method is in between the coarse and refined results which validates the precision of our method.

The "Adaptive 2-4" version, whose curve is presented in Fig. 11(*c*), is composed of 18 elements refined twice and 880 elements refined three times, as seen in Table 1. This version requires an average computation time of 13.12ms by simulation step. Moreover, the average computation time for the mere subdivision process is 28.66ms, and 5.86ms for the distribution of the T-junction forces to their neighbors (120 T-junctions). The low tradeoff in time between "Adaptive 2-4" and "Refined $\times$3" is explained by the small difference between the number of elements (898 vs 1024) and the extra cost due to T-junctions. The first point can be solved by choosing a better subdivision criteria, and the second by improving how T-junctions are handled.

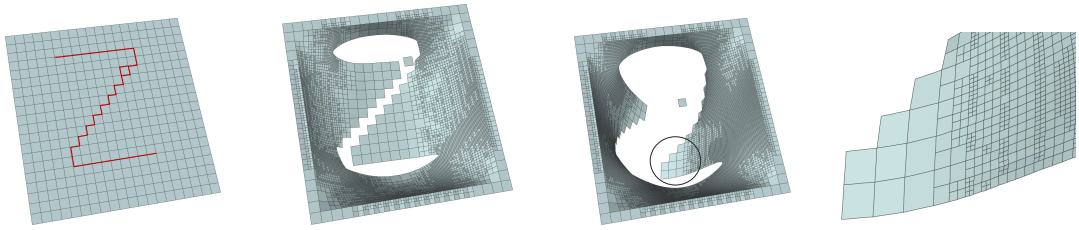
| | Time (ms) | Number of elements |
|---|---|---|
| Coarse | 0.37 | 2 elts lvl. 1 |
| Adaptive 2-3 | 2.49 | 2 elts lvl. 2 and 112 elts lvl. 3 |
| Refined $\times$3 | 13.47 | 1024 elts lvl. 4 |
| Adaptive 2-4 | 13.12 | 18 elts lvl. 3 and 880 elts lvl. 4 |
| Refined $\times$4 | 97.53 | 8192 elts lvl. 5 |

Table 1: For each curve shown in Fig. 11(c), average computation time for one simulation step, and number of elements with their subdivision level.

## 9. Conclusion

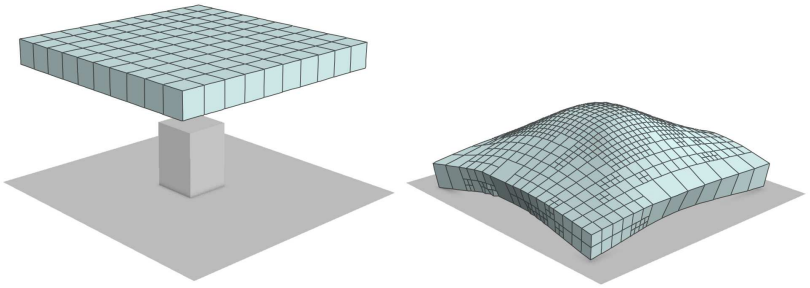
In this paper, the LCC+MSS model presented by Fléchon in [FZDJ13] has been extended. Our main contribution is the use of a 2D or 3D unified model, and the definition of an adaptative refinement operation during the simulation. The physical description is updated while the validity of the mesh is guarantied thanks to the underlying topological model. Our refinement method is implemented based on CGAL Linear Cell Complex.

Besides, the coarsening operation will be investigated as future work. Note that we already have implemented the coarsening of edges (or faces in 3D) which is used after a cut involving different subdivision levels. Finally, other refinement patterns may be considered (tetrahedra, prisms), as well as other physical models such as the Mass-Tensor model [FZJM12].



(a) (b) (c) (d)

Figure 8: 2D experiment. The elements on the boundary of the mesh are constrained and the other ones are subject to gravity. (a) Initial mesh composed of $20 \times 20$ quads with the cutting path in red. (b) A step of the simulation where some elements are automatically subdivided. (c) Another step that exhibits four different subdivision levels. (d) Zoom on the circled part in (c).



(a) (b)

Figure 9: 3D experiment. (a) The initial mesh composed of $10 \times 10$ hexahedra with a spike (a cube centered on a plane). (b) The mesh after collapsing on the spike, with three different subdivision levels. $\rho = 400$ Kg/$m^3$ $\mathsf{E} = 30$ KPa, $\nu = 0.4$.

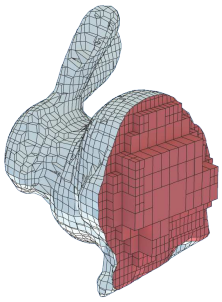
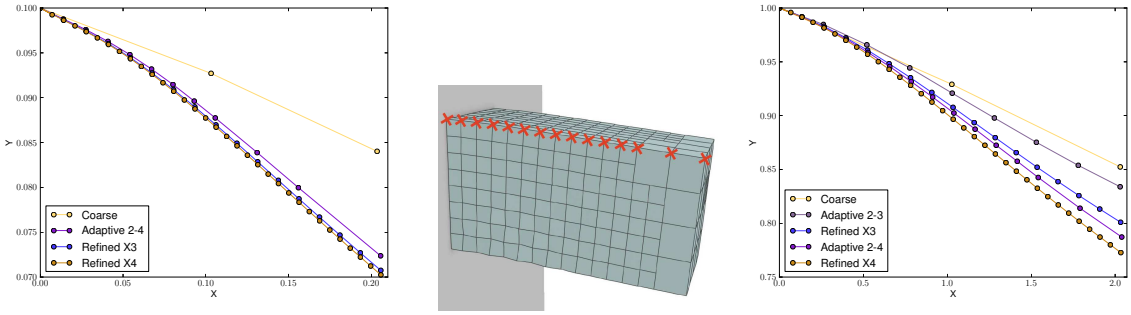


Figure 10: A rabbit composed of hexahedra having different subdivision levels (the 3D mesh is visually cut in order to see the internal hexahedra).




(a) (b) (c)

Figure 11: Each graph shows plots representing a simulated beam at different subdivision levels: 3 fixed subdivisions levels ("Coarse", "Refined $\times$3" and "Refined $\times$4") and one (*resp.* two for (c)) adaptive refinement "Adaptive 2-4" (*resp.* "Adaptive 2-3" and "Adaptive 2-4"). The coarse model is adaptively refined during the simulation. The subdivision level of its elements range from 2 to 3 for "Adaptive 2-3" and from 2 to 4 for "Adaptive 2-4". As illustrated in (b), the curves give the position of all the upper particles of the beam (red crosses) after 10,000 simulation steps, $\mathsf{E} = 100$ KPa, $\nu = 0.3$. (a) 2D model, initially composed of two quads of size 0.1 $m \times 0.1$ $m$. $\rho = 1000$ Kg/$m^3$ with a *threshold* $= 2\%$ for the geometric criteria. (c) 3D model, initially composed of two hexahedra of size 1 $m \times 1$ $m \times 1$ $m$, $\rho = 100$ Kg/$m^3$ with a *threshold* $= 1.2\%$ for the geometric criteria.

Work partially supported by the French ANR-12-MONU-0006 *SAGA* project, and the LIRIS *TopoSim* project. Thanks to Nicolas BONNEEL for its valuable review of the draft.

## References


[AUGA05] ALLIEZ P., UCELLI G., GOTSMAN C., ATTENE M.: *Recent Advances in Remeshing of Surfaces*. State-of-the-art report, AIM@SHAPE EU network of excellence, 2005. 2

[BADSM08] BABA-ALI M., DAMIAND G., SKAPIN X., MARCHEIX D.: Insertion and expansion operations for n -dimensional generalized maps. In *Proc. of DGCI 2008* (Lyon, France, April 2008), vol. 4992 of *LNCS*, Springer Berlin/Heidelberg, pp. 141–152. 6

[BBJ*09] BAUDET V., BEUVE M., JAILLET F., SHARIAT B., ZARA F.: Integrating Tensile Parameters in Hexahedral Mass-Spring System for Simulation. In *Proc. of WSCG 2009* (Feb. 2009). 1, 4

[BDW13] BUSARYEV O., DEY T. K., WANG H.: Adaptive fracture simulation of multi-layered thin plates. *ACM Trans. Graph. 32*, 4 (July 2013), 52:1–52:6. 2

[BHU10] BURKHART D., HAMANN B., UMLAUF G.: Isogeometric finite element analysis based on catmull-clark subdivision solids. *Comp. Graphics Forum 29*, 5 (2010), 1575–1584. 2

[CHCK02] CHOI Y.-J., HONG M., CHOI M.-H., KIM M.-H.: Adaptive mass-spring simulation using surface wavelet. In *Proc. of VSMM 2002* (Sep 2002). 3

[Dam12] DAMIAND G.: Linear cell complex. In *CGAL User and Reference Manual*, 4.0 ed. CGAL Editorial Board, 2012. 3

[DGW11] DICK C., GEORGII J., WESTERMANN R.: A hexahedral multigrid approach for simulating cuts in deformable objects. *IEEE Transactions on Visualization and Computer Graphics 17*, 11 (2011), 1663–1675. 2

[DKS*11] DARLES E., KALANTARI S., SKAPIN X., CRESPIN B., LUCIANI A.: Hybrid Physical – Topological Modeling of Physical Shapes Transformations. In *Proc. of DMDCM 2011* (Washington, DC, USA, 2011), pp. 154–157. 2

[FZDJ13] FLÉCHON E., ZARA F., DAMIAND G., JAILLET F.: A generic topological framework for physical simulation. In *Proc. of WSCG 2013* (Plzen, CZ, June 2013), pp. 104–113. 2, 3, 4, 8

[FZJM12] FAURE X., ZARA F., JAILLET F., MOREAU J.-M.: An Implicit Tensor-Mass solver on the GPU for soft bodies simulation. In *Proc. of VRIPHYS* (Dec. 2012), pp. 1–10. 8

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: Charms: A simple framework for adaptive simulation. *ACM Trans. Graph. 21*, 3 (July 2002), 281–290. 3

[HPH96] HUTCHINSON D., PRESTON M., HEWITT T.: Adaptive refinement for mass/spring simulations. In *Proc. of In 7th Eurographics Workshop on Animation and Simulation* (1996), Springer-Verlag, pp. 31–45. 2

[JAD*12] JUND T., ALLAOUI A., DARLES E., SKAPIN X., MESEURE P., LUCIANI A.: Mapping Volumetric Meshes to Point-based Motion Models. In *Proc. of Vriphys* (2012), pp. 11–20. 3

[JBB*10] JERABKOVA L., BOUSQUET G., BARBIER S., FAURE F., ALLARD J.: Volumetric modeling and interactive cutting of deformable bodies. *Progress in Biophysics and Molecular Biology 103*, 2-3 (Dec. 2010), 217–224. 2

[Lie94] LIENHARDT P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *Int. J. Comput. Geometry Appl. 4*, 3 (1994), 275–324. 3, 4

[Lob13] LOBOS C.: A set of mixed-elements patterns for domain boundary approximation in hexahedral meshes. *Studies in health technology and informatics 184* (2013), 268–272. Proceedings of MMVR20. 3

[LSH07] LLOYD B., SZÉKELY G., HARDERS M.: Identification of spring parameters for deformable object simulation. *IEEE Trans. on Visualization and Computer Graphics 13*, 5 (Sept-Oct 2007), 1081–1094. 1

[LYO*10] LEE Y., YOON S.-E., OH S., KIM D., CHOI S.: Multi-resolution cloth simulation. *Computer Graphics Forum 29*, 7 (2010), 2225–2232. 2

[Män87] MÄNTYLÄ M.: *An Introduction to Solid Modeling*. Computer Science Press, Inc., New York, NY, USA, 1987. 3

[MBTF03] MOLINO N., BRIDSON R., TERAN J., FEDKIW R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *In 12th Int. Meshing Roundtable* (2003), pp. 103–114. 3

[MDS10a] MESEURE P., DARLES E., SKAPIN X.: A Topology-Based Mass/Spring System. In *Proc. of CASA2010* (St Malo, France, June 2010). 2

[MDS10b] MESEURE P., DARLES E., SKAPIN X.: Topology-based Physical Simulation. In *Proc. of VRIPHYS 2010* (Copenhague, Danemark, Nov. 2010), pp. 1–10. 2

[NFP06] NESME M., FAURE F., PAYAN Y.: Hierarchical Multi-Resolution Finite Element Model for Soft Body Simulation. In *2nd Workshop on Computer Assisted Diagnosis and Surgery* (Santiago de Chile, Chili, 2006). 3

[NMK*06] NEALEN A., MULLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Comp. Graphics Forum 25*, 4 (Dec. 2006), 809–836(28). 1, 3

[Pay12] PAYAN Y.: *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*. SpringerLink Bücher. Springer, 2012. 1

[SSH10] SEILER M., SPILLMANN J., HARDERS M.: A three-fold representation for the adaptive simulation of embedded deformable objects in contact. *Journal of WSCG 18*, 1-3 (2010), 89–96. 3

[The12] THE CGAL PROJECT: *CGAL User and Reference Manual*, 4.0 ed. CGAL Editorial Board, 2012. 3

[TPF91] TERZOPOULOS D., PLATT J., FLEISCHER K.: Heating and melting deformable models. *The Journal of Visualization and Computer Animation 2*, 2 (1991), 68–73. 3

[TW90] TERZOPOULOS D., WATERS K.: Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation 1* (1990), 73–80. 3

[UCB13] UNTEREINER L., CAZIER D., BECHMANN D.: n-dimensional multiresolution representation of subdivision meshes with arbitrary topology. *Graph. Models 75*, 5 (Sept. 2013), 231–246. 2

[VB05] VILLARD J., BOROUCHAKI H.: Adaptive meshing for cloth animation. *Eng. with Comput. 20*, 4 (2005), 333–341. 2

[WRK*10] WICKE M., RITCHIE D., KLINGNER B. M., BURKE S., SHEWCHUK J. R., O'BRIEN J. F.: Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on Graphics 29*, 4 (July 2010), 49:1–11. 2

[WWD14] WU J., WESTERMANN R., DICK C.: Physically-based simulation of cuts in deformable bodies: A survey. In *Eurographics 2014 State-of-the-Art Report* (Strasbourg, France, 2014), Eurographics Association, pp. 1–19. 2

[ZGF07] ZERBATO D., GALVAN S., FIORINI P.: Calibration of mass spring models for organ simulations. In *Proc. of IROS 2007* (2007), IEEE, pp. 370–375. 3