Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2012)
J. Bender, A. Kuijper, D. W. Fellner, and É. Guérin (Editors)

# Mapping volumetric meshes to point-based motion models

T. Jund[1] and A. Allaoui[1] and E. Darles[1] and X. Skapin[1] and P. Meseure[1] and A. Luciani[2]

[1]University of Poitiers - CNRS UMR 7252- XLIM-SIC, France
[2]ICA Lab, University of Grenoble, France

**Abstract**

*Particle-based models produce various, flexible and optimized animations. Intrinsically, they are neither based on a boundary representation nor on volumetric objet meshed representations. Thus, they raise rendering issues since they do not contain enough geometrical information and do not even provide an underlying spatial topology. Consequently, various geometrical shapes can be used to render a motion produced by a meshless model, leading to different visual interpretations. To our knowledge, there is no generic methods to associate any set of points in motion with a topology-based geometric model.*
*In this paper, we propose a framework to map arbitrary volumetric meshes to arbitrary point-based motions and to control the topological changes. Therefore, from only one motion description, different visual results can be obtained. This framework breaks down into three distinct processes: a particles to vertices mapping, the definition of a motion function and the definition of topological modifications and events triggering them. We show how the manipulation of these parameters allows to experiment different mappings for a particular motion and that our framework includes most of previous known mappings.*

## 1. Introduction

In the past decades, physically-based animation took advantage of particle-based models such as proposed by [LC84, MP89] for animating viscous fluids or deformable colliding object. Unfortunately, since they are not based on a given topology, they raise rendering issues. Physically-based animation methods turned to meshed representations of animated objects such as Finite Element Methods [TPBF87] or mass-springs methods [Pro95] to alleviate this problem.

Although mesh-based methods produce impressive animations, particle-based models offer more freedom and versatility. For example, meshed models do not easily handle versatile and evolutive shapes. In these cases, the modularity and genericity of particle-based models have proven to be more suitable [KANB03].

Using a particle-based model does not require to define any given spatial topology: the initial organization of the particles can be used to define the initial and predetermined shape to visualize. This shape then follows the displacement of particles and is rendered through point based rendering techniques [MKN*04], implicit surfaces [DCG95], or meshes [MHTG05].

However, establishing a relationship between a set of particles of a mechanical model and a shape is not straightforward. For the approach to be flexible, the mechanical model has to be independant from the initial shape, the motion mapping and to the parameters leading to topological changes.

In this paper, we formalize a motion/geometry mapping (*MGM*) which defines the relationships between point-based motions and meshed models. Our main contribution is to provide some flexible means to represent and control the evolution of displayed shapes and their topology. More precisely, we define a set of parameters to easily control the impact of the particles motion on the associated mesh. We use a volumetric topological representation to describe and modify shapes with inner structures, which can be an issue without topological information.

Section 2 presents related works and discusses their advantages and limitations in this context. Section 3 describes our physical and geometrical models. Section 4 details motion/geometry mapping the *MGM*. Considering this definition, a classification of other works and some experimental results are given in section 5. Sections 6 and 7 present future works and a conclusion.

© The Eurographics Association 2012.

DOI: 10.2312/PE/vriphys/vriphys12/011-020

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

## 2. Related work

Many works link the physical elements and the representation of the mesh to simulate material behaviour [MG04] or handle topological modifications [OBH02]. Such methods provide both shape and topology of the displayed object so that the mechanical and visual parts are similar or closely related [DBB11, FGBP11]. In this context, topology control is ensured through direct modifications of the mesh by remeshing the physical domain [MBF04] or adapting mechanical parameters [FDA05].

To avoid the representation of physical elements, the usual solution consists in placing a small number of particles on the mesh, the vertices of this latter are then moved based on an interpolation of the displacement of the particles. Most methods following this principle use the Shape-Matching interpolation method. This idea was introduced by Müller et al. [MHTG05] to simulate elastic deformation and was extended by Rivers et al. [RJ07] to reduce computation times and obtain smoother and more complex deformations. The use of oriented particles has also been proposed to handle singular patterns of particles [MC11]. Creating an animation based on this principle still implies a strong link between the geometry and the movement. In our context, the motion of the particles is an entry point and may be undersampled compared to the needs of rendering.

If no initial shape is given, automatic coating methods can be used. Implicit surfaces were first used as they intrinsically deal with topological changes [WMW86]. Another approach to visualize meshless models is to provide a skinning which may result from a classical surface reconstruction approach. In [CG93], an isosurface is extracted from a scalar field which is constructed from a sum of radial basis functions defined at the center of each particle. With these kind of methods, the spatial proximity automatically controls the topology. If two particles are near, their potential functions and their associated surface merge. In [DCG95], Desbrun et al. extend this approach by using a blending graph to avoid unwanted blending and simulate precise contacts between different bodies. However, implicit surfaces do not represent well shapes containing sharp edges.

Using a volumetric representation allows to simulate inner structures or tearing effects. To our knowledge, no work treat the Motion/Geometry Mapping (*MGM*) for volumetric representation. This is usually due to the lack of a strong topological representation.

In this paper, we provide a thorough description of the mapping between a predefined motion description and a volumetric geometry. We identify the inherent parameters which provide control over the animation. These parameters allow us to generate a panel of animations from a unique motion. We thus provide tools for animators to create different visual results by working on parameters and shapes only.

## 3. Models and notation

In this section, we define the physical and geometrical models used for our *MGM*. They are both based on atomic elements and thus allows to follow a bottom-up approach.

### 3.1. Physical model

To generate motions, we use the generic CORDIS-ANIMA system [LJF*91] as it allows to model motions without any notion of shapes or specific geometric or visual representations. This system is a modular and discrete formalism, based on the concept of mass-interaction, which is a networked cellular implementation of Newtonian principles. The manipulated concept are only point masses, forces and action-reaction physical principles between any part of the network to another without any spatial contiguity or proximity criteria. That is why it allows to design models of dynamic phenomena, not exclusively spatial or visual but also acoustical and haptic. In computer animation, it allowed the simulation of phenomena such as plasticity, friction or fractures [LG97]. Animations are produced without any trace of underlying spatial structure.

Our method is not limited to the use of CORDIS-ANIMA. We only require a frame by frame description of positions evolving through time. Thus, other sources of kinematic information can be used, such as motion capture information.

### 3.2. Geometrical model

The geometrical model corresponds to the moving mesh that can be altered with respect to the motion defined in the physical model. Actually, the mesh can come from any source: it may be generated from the initial positions of the particles or come from any external source. Nevertheless, in order to get a rigorous object representation and to ensure topological consistency, we want the mesh to rely on a robust topological model. Therefore, we use generalized combinatorial maps defined in [Lie94] that model manifolds of dimension 2 or more. Note that similar topological models have been used in physical simulations, to treat specific operations such as gluing, cutting or more generally mesh evolution [LT07].

Our framework describes any object as a combinatorial map subdivided in topological cells (i.e. vertices, edges, faces and volumes). Each cell is made from grouping atomic topological elements. Relationships between those atomic elements are defined in such a way that retrieving adjacency relations between cells of different dimensions is fast and robust. Also, the definition of this model insure that topological modification always represent a consistent mesh. Using a volumetric representation of objects allows us to represent inner fractures or structures. Using a surface representation would forbid the identification of inner holes and would narrow the scope of our *MGM*.

We use the CGoGN library [CGo] that provides an efficient implementation of this model. It also provides tools to

© The Eurographics Association 2012.

manipulate the topology of meshes along with an attribute manager that allows to attach information (such as geometrical embedding or color) to each cell.

## 4. Motion/Geometry Mapping

The *MGM* defines how the geometrical model is mapped to the physical model. This mapping is used to determine the behaviour of the geometrical model according to the displacements of the particles. We decompose the *MGM* in three parts as shown in figure 1: a geometrical mapping, a motion mapping and the handling of topological changes. The geometrical mapping defines the part of the geometrical model influenced by each particle. The motion mapping defines how the motion of particles is transmitted to the geometrical model. Finally, the topological changes allow to manipulate the mesh structure and thus redefine the geometrical mapping.

The *MGM* initializes a geometrical mapping to perform motion mapping. In most cases, this operation is only performed at the start of the animation, or when a topological change occurs. Further details are given in the following.

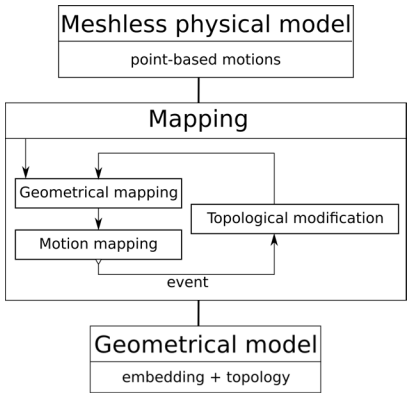


**Figure 1:** *Flowchart describing the mapping between physical and geometrical models.*

### 4.1. Geometrical mapping

The geometrical mapping associates each particle with a set of topological cells of the mesh. Usually those cells correspond to vertices, we propose a formalism allowing to use edges, faces and volumes as well. This association relies on a two-step algorithm. The first step defines a weight function such that for any pair $(C_i, P_j)$, with $C_i$ a topological cell and $P_j$ a particle, a real value is given. The second step consists in filtering the topological cells $C_i$ associated with the particles $P_j$, depending on this value. In other words, a first step allows to evaluate the influence of particles on vertices, the second one allows to limit this influence to given areas. This mapping is used in both motion mapping and topological mapping.

#### 4.1.1. Weighting

The weighting step can use different kinds of parameters, such as geometrical, topological or hybrid weightings. For instance, a geometrical weighting consists in associating each pair $(C_i, P_j)$ with a distance: the farther $C_i$ is from $P_j$, the higher (or the lower) the value becomes.

Different kinds of topological weightings can be defined and mainly follow the flood-fill algorithm principle, as described below.

We consider a topological representation as a graph with nodes (cells of the mesh) of dimension $i$ ($0 \leq i \leq 3$) and arcs (adjacency relationships representing a cell linking two nodes) of dimension $j \neq i$. If nodes are vertices and arcs are edges connecting vertices, we get the classical representation of meshes.

An initial relationship has to be done to determine which nodes of the graph are roots for the flood-fill algorithm. For example, we can choose the nearest vertex(ices) from a particle as root(s), or all vertices of a volume including a particle. After the initial relationship is done, a Dijkstra-type flood-fill algorithm is used and minimal weights are chosen whenever different paths lead to the same nodes. The algorithm stops as soon as all nodes have been processed.

Hybrid weighting is also worth considering. The principle is the same as above, but weights are associated with arcs and/or nodes depending on geometrical criteria, such as volumes or local curvatures. This allows the algorithm to favor some flooding area.

Figure 2 shows a topological weighting for a given particle on a graph representing the mesh. Once roots are chosen in the graph, all linked nodes are given a weight related to their distance to those roots.

#### 4.1.2. Filtering

Once all weights have been set for each pair $(C_i, P_j)$, they are multiplied by coefficients assigned to each particle $P_j$. They allow the user to modulate each particle influence and thus have more control on the geometrical mapping as shown in figure 3. Finally, we assign a subset of all the particles $P_j$ to each cell $C_i$.

This subset comes from a filtering operation based on the previously defined weights. For example, we can associate a cell with the particle having the minimum weight, particles within a given weight interval, or keep a specified amount of particles.

Depending on the kind of filter used, weighting operations do not have to be completely performed. The filtering may use criterions allowing to shorten the flooding operation.

At the end of the geometrical mapping, each vertex knows which particles guide its displacements.

© The Eurographics Association 2012.

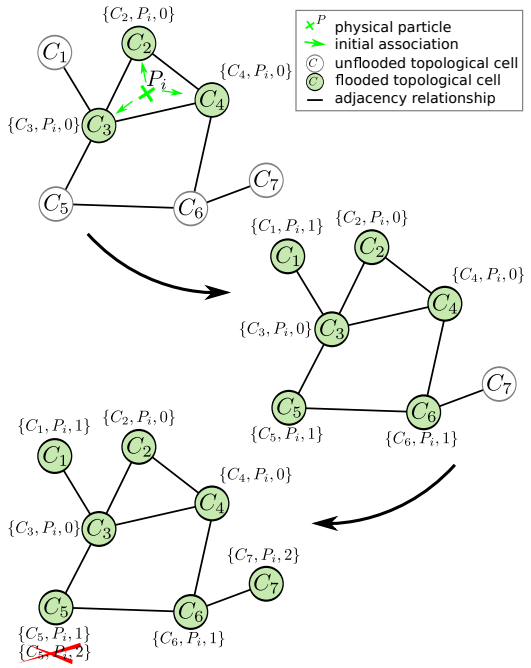


**Figure 2:** *A topological flood-fill algorithm: a weight is given to each cell of the topological graph according to its distance to the physical particle. A 3-uple $\{C_i, P_j, v\}$ represents the weight $v$ assigned to the pair $(C_i, P_j)$.*

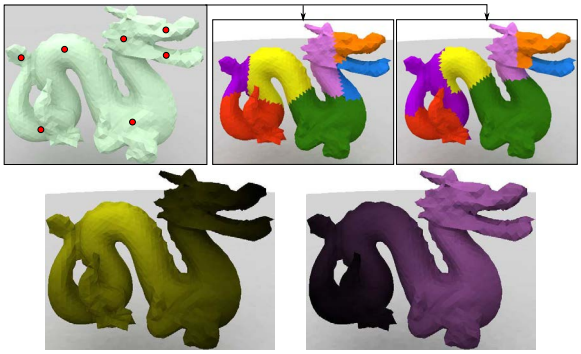

**Figure 3:** *A tetrahedral model containing seven particles, shown as red dots on the left. A topological flood is used, the filtering operation keeping at least one particle per vertex. Results of the filter are shown on the top right: different particles weights change the geometrical mapping. Below: the smoothly varying weights are shown for two particles before the filtering operation.*

### 4.2. Motion mapping

The motion mapping describes how those vertices are moved according to the displacements of their associated particles from one timestep to another. The idea is to have a frame of reference from which the displacement can be defined. This reference frame can be either the previous frame or the initial frame. This motion mapping is highly constrained by the cardinality of the sets of particles. Translation requires only one particle, whereas scaling and rotation require two particles at least.

Different effects can be created depending on the cardinality of particles/vertices associations. If only one particle describes the motion of a cluster of vertices, this cluster is animated as a translating rigid block without rotation. When several particles are used for displacements, we can use different approaches such as a mean displacement, a Shape-Matching algorithm [MHTG05], or other kinds of interpolation. Depending on the method, a transformation matrix or a displacement field is computed in order to move the related vertices.

We describe below different approaches to compute the new positions of vertices of the mesh at time $t+1$ depending on the previous positions of particles. Its versatility lets us imitate many methods found in the literature.

In the following, we note $\mathbf{x}^t$ and $\mathbf{p}_j^t$ the respective positions of a vertex of the mesh and of a particle $j$ at time $t$.

#### 4.2.1. Computing a transformation matrix

Methods such as Shape-Matching propose to compute a transformation matrix based on a reference frame. It consists in minimizing the moving least square error. The principle to compute the rigid transformation matrix is expressed in equation 1. The optimal rotation $R$ is computed using the covariance matrix and Jacobian rotations. The translation $T$ is considered as the vector between the barycenters of the reference shape and the current one.

$$v_i^t = \mathbf{p}_i^t - (\sum_{i=1}^{N} \mathbf{p}_i^t)/N$$

$$\varepsilon(R,T) = \sum_{i=1}^{N} \|\omega_i(v_i^t - (Rv_i^0 + T))\| \qquad (1)$$

In equation 1, $v_i^t$ is a vector from the particle $i$ to the barycenter of the set of particles in their current position, $N$ is the number of particles and $v_i^0$ is the same particle vector when the meshless model is in its reference shape. The parameter $\omega_i$ defines a parameter used to weight the particle significance in the motion.

A transformation matrix is computed for each cluster of particles induced by the geometrical mapping.

#### 4.2.2. Computing a displacement field

We explain how to compute displacement fields for the vertices according to the displacement of particles. Formally,

© The Eurographics Association 2012.

the displacement field is defined as follows:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + D(\mathbf{x}^t) \tag{2}$$

where $D$ is an interpolation function based on the displacement of the particles associated with $\mathbf{x}$ between step $t$ and step $t+1$ . An alternate version consists in computing this function using the initial position of the particles and $t+1$, as usually done in mesh deformations works [MHTG05, MQ07].

Various strategies can be adopted:

**Mean functions:** The simplest strategy consists in computing the mean of the particles displacements, independently from $\mathbf{x}^t$.

$$D(\mathbf{x}^t) = (\sum_{i=1}^{N} \mathbf{p}_i^{t+1} - \mathbf{p}_i^t)/N \tag{3}$$

When associated with a geometrical mapping selecting exactly one particle per vertex, we call this motion mapping "direct mapping".
This solution can be improved by weighting each displacement:

$$D(\mathbf{x}^t) = \sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{p}_i^t - \mathbf{x}^t\|)(\mathbf{p}_i^{t+1} - \mathbf{p}_i^t) \tag{4}$$

where $\phi$ is any decreasing function from $\mathbb{R}^+$ to $\mathbb{R}^+$, and each $\lambda_i$ is computed to satisfy the condition expressed in equation 5.

$$\sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{p}_i^t - \mathbf{x}^t\|) = 1 \tag{5}$$

**Barycentric functions:** This strategy is based on a geometric mapping selecting exactly 4 particles in dimension 3. It consists of the weighted mean of the displacement where each $\mathbf{x}^t$ has $\lambda_i$ as barycentric coordinates.

$$D(\mathbf{x}^t) = \sum_{i=1}^{N} \lambda_i (\mathbf{p}_i^{t+1} - \mathbf{p}_i^t) \tag{6}$$

**Radial basis functions:** Interpolation functions based on Radial Basis Functions (RBF), defined in [Mic86] for surface reconstruction, have been used for point-based mesh deformation in [BK05]. In our case, we make use of this method to compute a displacement field. Three scalar displacement fields $D_0$, $D_1$ and $D_2$ are defined for each axis of the 3D space. The equation for axis $i$ (with $\mathbf{p}_{i,j}^t$ being the $i$-th coordinate of $\mathbf{p}_j^t$) can be written as :

$$D_i(\mathbf{x}^t) = \sum_{j=1}^{N} \lambda_j \phi(\|\mathbf{p}_j^t - \mathbf{x}^t\|) + a_0 + \sum_{k=1}^{3} a_k \mathbf{x}_{j,i}^t \tag{7}$$

Gaussian, shifted log, quadratic and inverse quadratic functions are used as $\phi$ functions with various results as shown later in figure 7. To compute $\lambda_j$ and $a_i$, we refer the reader to the papers cited above.

Depending on the selected geometrical and motion mappings, different effects can already be achieved. The property of local or global influence change the way the geometry follows the motion. However, those parameters alone do not allow to handle topological changes of the meshes during an animation.

### 4.3. Topological mapping

During a simulation, evolution of particles may lead to some topological and geometrical changes. We measure compression and elongation on the physical and/or the geometrical models at each time steps to identify events inducing topology modifications.

As soon as an event is caught, specific topological modifications are triggered, which include adding or removing material, glueing or separating topological cells, local remeshing, subdivision, and so on.

When an elongation (or a compression) is measured on a cell of the mesh (e.g. an edge length is becoming to high/short), selecting the cells upon which topological operations must be applied is rather straightforward. On the contrary, when such a measure is performed on the physical model, the concerned cells must be deduced from the geometrical mapping. The remainder of this section considers this issue. We do not consider here any propagation mechanism and only rely on the particles positions and/or the cells configurations to perform topological operations.

After filtering is done as described in section 4.1.2, some vertices may be influenced by several particles. We have to determine which cells should be modified topologically. More precisely, we focus on the cells located on the frontier of each particle's influence area.

The frontier is either "well-defined" or "ill-defined". Let us consider two particles $P_i$ and $P_j$, and $G_i$ and $G_j$ their associated graphs defined following one of the methods described in section 4.1.1. The frontier between those graphs is said "well-defined" if: either $G_i \cap G_j$ is a subgraph for which there exists a single path between two vertices (figure 4.1.a); or $G_i \cap G_j = \emptyset$ and at least one pair of vertices $(C_i \in G_i, C_j \in G_j)$ are topologically connected (figure 4.2.a). The frontier is said "ill-defined" if: either $G_i$ and $G_j$ overlap, that is $G_i \cap G_j$ is a subgraph for which there are two different paths between at least one pair of vertices (figure 5.1); or $G_i \cap G_j = \emptyset$ and there are no direct mesh cells which connect $G_i$ and $G_j$ (figure 5.2).

Handling a new event when the frontier is well-defined is easy enough, as shown in figure 4 for some tearing operation. If there exists a single path between frontier vertices (figure 4.1.a), vertices duplications are performed (figure 4.1.b).

© The Eurographics Association 2012.

| Method | Mapping | | | Topological Events |
|---|---|---|---|---|
| | Geometrical | | Motion | |
| | Flood | Filtering | | |
| Point-based animation of elastic, plastic and melting objects [MKN*04] | Geometrical | Distance based | Shape-matching | No |
| Meshless animation of fracturing solids [PKA*05] | Topological | Distance based | Shape-matching | Yes |
| Meshless deformations based on shape-matching [MHTG05] | Topological | Keep particles within volumes | Shape-matching | No |
| Position based dynamics [MHHR07] | Geometrical | One particle per vertex | Direct | Yes |
| Fastlsm: fast lattice shape matching for robust real-time deformation [RJ07] | Geometrical | Distance based | Shape-matching | Yes |
| SOFA an Open Source Framework for Medical Simulation [ACF*07] | Geometrical | Keep 4 non-planar | Barycentric | Yes |
| Fast adaptive shape matching deformations [SOG08] | Geometrical | Distance based | Shape-matching | Yes |
| Meshless modeling of deformable shapes and their motion [AOW*08] | Topological | Keep at least 4 non-planar | Shape-matching | No |

**Table 1:** *Classification of existing methods considering the geometric mapping, motion mapping and handling of topological modifications.*

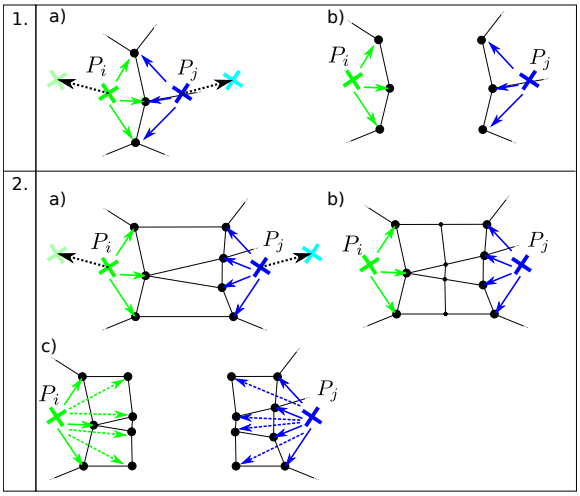


**Figure 4:** *Tearing a part of the mesh associated with two particles along a well-defined frontier. Two cases occur: 1. The frontier follows vertices; 2. The frontier is located between vertices. a) Before modification; 1.b) and 2.c) After tearing (2.b shows a required subdivision step).*

Otherwise (figure 4.2.a), new vertices are inserted on arcs influenced by both $P_i$ and $P_j$ (figure 4.2.b). Then, a local flood-fill algorithm is performed, and the frontier can be treated as in the first case.

Figure 6 shows different tearing operations on tetrahedrons between particles $P_i$ and $P_j$. The illustrations show all valid cases. The case where all vertices are associated with $P_i$ and at least one with $P_j$ does not define a clear frontier. We show here the typical cases on tetrahedrons, but as the topological representation is not limited to any volume shape, any kind of polyhedron is treated likewise. A plane is defined depending on particles configuration to create two distinct volumes. The separation can either go through edges only or cross vertices of the volume. This latter occurs when a vertex is shared by the two particles. The simple case, equivalent to figure 4.1, is not illustrated. It corresponds to a face between two volumes for which all vertices are associated with the two particles $P_i$ and $P_j$. This case only requires to unsew the faces of the volumes.

Figure 5.1 shows ill-defined frontiers between the graphs associated with particles $P_i$ and $P_j$. For particles raising a topological modification and presenting an ill-defined fron-

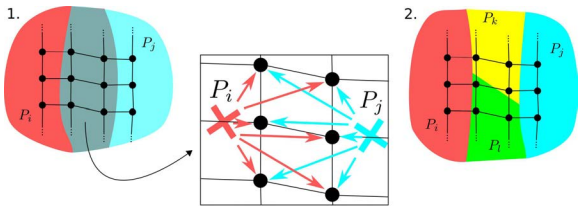


**Figure 5:** *Ill-defined frontiers between two particles $P_i$ and $P_j$ associated with vertices. 1) Overlapping graphs. 2) Although both graphs associated with $P_i$ and $P_j$ are distinct, they are connected by other vertices, which may be associated with other particles $P_k$ and $P_l$.*

© The Eurographics Association 2012.

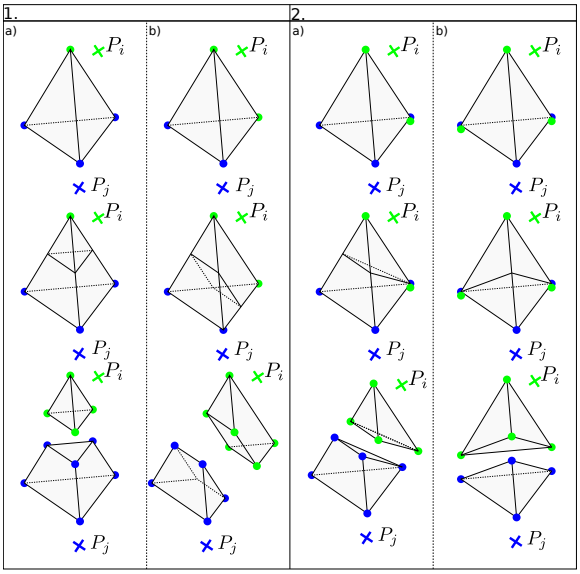


**Figure 6:** *1. Splitting a volume having a well-defined frontier separated by edges only. 2. Splitting a volume having a well-defined frontier separated by edges and vertices. Green and blue dots represent the association between particles and vertices.*

tier, a geometric mapping with different parameters have to be used in order to retrieve a well-defined frontier.

When using a topological flood-fill algorithm, the shortest paths between particles and vertices can change or become nil. A new configuration leads to the modification of vertices behaviour.

Other topological operations such as remeshing or glueing are treated in a similar manner.

## 5. Discussion

In this section, we present some results and compare our framework with existing mappings between meshless models and geometrical models. Most of those methods do not focus on the *MGM* and propose only one mapping. These mappings are often guided by constraints set by the motion mapping. Indeed, as expressed in section 4.2, some rotation, scaling or deformation properties can only be expressed with a certain amount of particles. We then propose an evaluation of the costs of the different steps of our framework.

Regarding our definition of the *MGM*, we distinguish various groups of methods depending on:

- the equivalent weighting and filtering algorithm using our definitions;
- the motion function;
- the ability to handle topological changes such as fractures.

Table 1 summarizes our classification of some existing methods using these parameters.

As shown in figure 7, changing parameters of the *MGM* for one unique meshless animation allows one to create clusters or, more generally, different deformation behaviours. Thus, the manipulation of the parameters of the *MGM* provide the possibility to create material effects.

We illustrate this principle on figure 8. These effects can either rely on the particle space-sampling or on the geometrical mapping. On the bottom right example, a gaussian distance function is used with an RBF motion mapping to create wrinkles on the mesh on sparsely sampled areas. On the middle right example, Shape-Matching with a localised geometrical mapping is used and create a clustering effect. Still on the same animation, other distance functions, such as splines, define a smoother interpolation as shown on the top right example.

Figure 9 shows different topological modifications on a hexahedral mesh undergoing tearing. The modification of the tearing threshold allows here to provide different animations based on a single movement. Figures 10 and 11 shows examples of fractures and internal fractures on a tetrahedral mesh. Furthermore, the use of volumetric representations allows to handle inner fracture during the animation.

In order to evaluate the performance of our framework to create an animation, we evaluate the cost of the components of the *MGM* distinctly. During the whole animation, the motion mapping process requires most computations. The need of a complete update of the geometrical mapping, due to some topological modification, is rather seldom. Most ge-

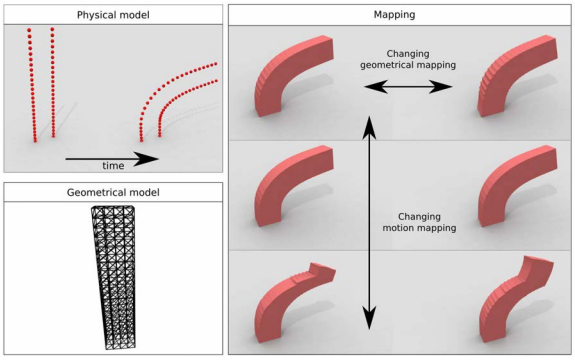


**Figure 7:** *Top-left: the physical model. Bottom-left: the geometrical model. Right: One timestep of a single animation based on 40 particles illustrating the* MGM. *Different motion mappings and geometrical mappings are used. Left column: topological flood, keeping 10 particules at least. Right column: topological flood, keeping 24 particles at least. First row: RBF using a gaussian function. Second row: RBF using a spline function. Third row: Mean function.*

© The Eurographics Association 2012.

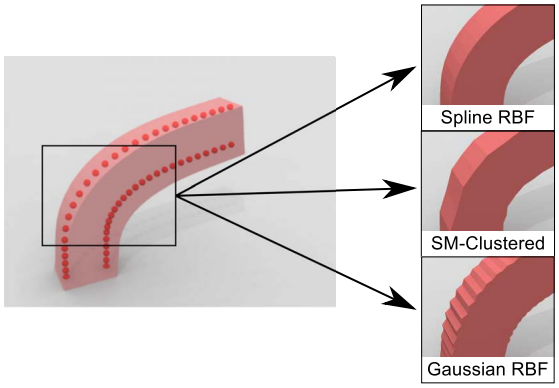


**Figure 8:** *Details of different motion mapping parameters on the same movement with one shape at the same frame.*

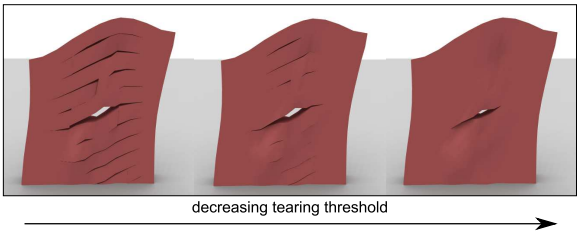


**Figure 9:** *Details of different topological modification effects on the same movement with one shape at the same frame.*

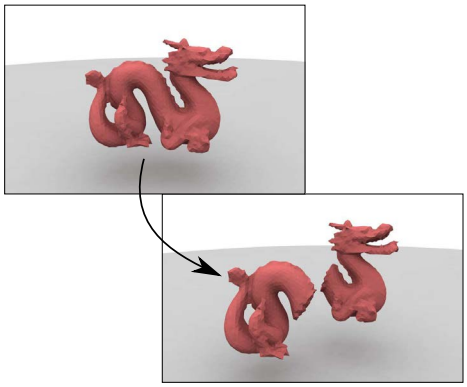

**Figure 10:** *A dragon fracture represented with a tetrahedral mesh.*

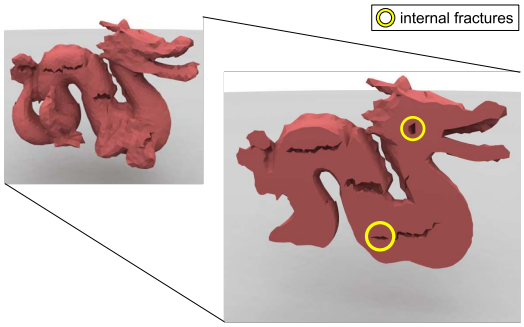


**Figure 11:** *Internal fractures on a volumetric mesh.*

ometrical updates can be restricted to a small area most of the time. The computation times are measured on a standard desktop computer using only one core.

Looking at the initialization of the geometrical mapping, a distinction has to be made between geometrical distance and topological distance. A geometrical flood-fill has a complexity in $\mathcal{O}(CP)$, with $C$ the number of topological cells in the mesh and $P$ the number of particles used to represent the movement. However, the use of a topological flood-fill depend on more variables. Indeed, depending upon the filtering, optimizations can be proposed. The worst case scenario could be in $\mathcal{O}(C^2P)$ due to the Dijkstra algorithm. Considering a mesh with 100,000 topological cells and a movement represented with 8 particles, the geometrical mapping takes approximately 0.2 seconds using a geometrical distance and 3.5 seconds for a topological flood. The cost of the topological flood is more important as it includes the retrieval of the adjacencies between cells. This retrieval represents up to 60% of the topological flood process and can be stored (and updated locally if required) during the animation for further updates.

The cost of the motion mapping depends in part on the geometrical mapping. Mean and barycentric motion mappings have a cost relative to the number of vertices of the mesh. However, even if Shape-Matching and RBF require more complex computation, they can be performed once per clusters of vertices. We call *cluster* a set of vertices of the mesh having the same geometrical mapping.

The topological mapping cost is negligible compared to the other computations. However, one should note that it may imply to perform a geometrical mapping operation to preserve a valid configuration.

## 6. Perspectives

We have presented a general method to control the relationships between motion and shape. Several aspects can still be improved to extend the scope of our method.

Our main structure is encoded by a combinatorial map. An extension of this model is given in [KCB09] and can be integrated in our framework. The multiresolution extension inherits the flexibility and efficiency of our topological structure. The multiresolution interest here lies in the ability to adapt the complexity of the shape depending on the viewing distance. A robust and generic definition of the *MGM* in this case has still to be provided.

© The Eurographics Association 2012.

Our framework is based on a volumetric representation which may imply a high memory cost. Most current volumetric representations are based on tetrahedrons or hexahedrons only. Our method is only constrained by the convexity properties of the polyhedrons to identify which ones contains particles. In this context, the degree of the polyhedrons is not important so allows us to consider simplification algorithms for volumetric representations.

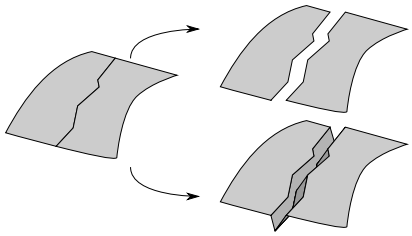

**Figure 12:** *Surface unsewing: a surface can either represent a thick mesh, or the border of a filled volume. In the second case, a remeshing has to be performed when the surface is torn or cut.*

In several works, such as [MG04], a surface mesh is used to represent filled volumes. When a topological operation is applied on the mesh, the interior, which has been implicitly represented, is then meshed as shown on the bottom of figure 12. This method is compatible with ours: indeed, this remeshing corresponds to the creation of an interior structure which is explicitly represented with our volumetric mesh. However, a surface-only representation would currently limit the definition of our flooding method and prevent the possibility to represent inner structure.

We define the association between the physical model and the geometrical one. However, if collisions occur on the geometrical model, no particular process can rule out interpenetration. Classical methods of collision detection are to be integrated to treat this problem to avoid this.

All our examples are currently based on physical models with few particles, we want to test the robustness of our framework with more complex particles patterns.

## 7. Conclusion

We have presented a generalisation of the *Motion/Geometry Mapping* defining the relationship between a physical meshless model and a topological model. This *MGM* allows to separate motion from shape and thus to create different visual effects with only one physical animation.

A first mapping assigns all vertices of the mesh to a set of physical particles. This mapping is then used to move the mesh and potentially change the topology underlying the geometrical model. A small set of parameters lets us produce a wide range of visual results.

These results can either be used for perception analysis of animations [BLAK12] or to guide creations by allowing to visualize with different shapes point-based kinetic information.

## Acknowledgement

Research supported by the ANR in the project DYNAMé (ANR-09-CORD-007). We are grateful to Benoit Crespin and Pierre Kraemer for their comments.

## References

[ACF*07] ALLARD J., COTIN S., FAURE F., BENSOUSSAN P., POYER F., DURIEZ C., DELINGETTE H., GRISONI L.: Sofa an open source framework for medical simulation. In *Medicine Meets VR* (2007), pp. 1–6. 6

[AOW*08] ADAMS B., OVSJANIKOV M., WAND M., SEIDEL H.-P., GUIBAS L. J.: Meshless modeling of deformable shapes and their motion. In *Symp. on Comput. Anim.* (2008), pp. 77–86. 6

[BK05] BOTSCH M., KOBBELT L.: Real-time shape editing using radial basis functions. In *Computer Graphics Forum* (2005), vol. 24, Citeseer, pp. 611–622. 5

[BLAK12] BENALLEGUE A. A., LUCIANI A., ALLAOUI A., KALANTARI S.: A cognitive analysis of the perception of shape and motion cooperation in virtual animations. In *Proceedings of the ACM Symposium on Applied Perception* (2012), pp. 130–130. 9

[CG93] CANI-GASCUEL M.-P.: An implicit formulation for precise contact modeling between flexible solids. In *SIGGRAPH* (1993), pp. 313–320. 2

[CGo] CGOGN: Combinatorial and Geometric modeling with Generic N-maps. http://cgogn.unistra.fr. 2

[DBB11] DIZIOL R., BENDER J., BAYER D.: Robust real-time deformation of incompressible surface meshes. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), pp. 237–246. 2

[DCG95] DESBRUN M., CANI-GASCUEL M.-P.: Animation of soft substances with implicit surfaces. In *SIGGRAPH* (1995), Comput. Graph., pp. 287–290. 1, 2

[FDA05] FOREST C., DELINGETTE H., AYACHE N.: Removing tetrahedra from manifold tetrahedralisation: application to real-time surgical simulation. *Med. Image Anal. 9*, 2 (2005), 113–122. 2

[FGBP11] FAURE F., GILLES B., BOUSQUET G., PAI D.: Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30*, 4 (July 2011). Special Issue: Proceedings of ACM SIGGRAPH 2011. 2

[KANB03] KACIC-ALESIC Z., NORDENSTAM M., BULLOCK D.: A Practical Dynamics System . *SCA* (2003), 7–16. 1

[KCB09] KRAEMER P., CAZIER D., BECHMANN D.: Extension of half-edges for the representation of multiresolution subdivision surfaces. *The Visual Comp. 25*, 2 (2009), 149–163. 8

[LC84] LUCIANI A., CADOZ C.: Modélisation et animation gestuelle d'objets - le système anima. *CESTA, 1er Colloque Image, Biarritz, France* (1984), 183–189. 1

[LG97] LUCIANI A., GODARD A.: Simulation of physical object construction featuring irreversible state changes. *Winter School of Comput. Graph. (WSCG)* (1997), 321–330. 2

© The Eurographics Association 2012.

[Lie94] LIENHARDT P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *Int. J. Comput. Geom. Appl. 4*, 3 (1994), 275–324. 2

[LJF*91] LUCIANI A., JIMENEZ S., FLORENS J., CADOZ C., RAOULT O.: Computational physics: a modeler simulator for animated physical objects. In *Eurographics* (1991), pp. 425–436. 2

[LT07] LINDBLAD A., TURKIYYAH G.: A physically-based framework for real-time haptic cutting and interaction with 3d continuum models. In *ACM Symp. on Solid and Phys. Model.* (2007), pp. 421–429. 2

[MBF04] MOLINO N., BAO Z., FEDKIW R.: A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. 23* (2004), 385–392. 2

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. In *SIGGRAPH* (2011), pp. 92:1–92:10. 2

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Graphics Interface* (2004), Canadian Human-Computer Commun. Soc., pp. 239–246. 2, 9

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Visual Commun. and Image Represent. 18* (2007), 109–118. 6

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005), 471–478. 1, 2, 4, 5, 6

[Mic86] MICCHELLI C. A.: Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation 2*, 1 (1986), 11–22. 5

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. *Symp. on Comput. Anim.* (2004), 141–151. 1, 6

[MP89] MILLER G. S. P., PEARCE A.: Globular dynamics: A connected particle system for animating viscous fluids. *Computers & Graphics 13*, 3 (1989), 305–309. 1

[MQ07] MCDONNELL K. T., QIN H.: Pb-ffd: A point-based technique for free-form deformation. *Journal of Graphics Tools 12*, 3 (2007), 25–41. 5

[OBH02] O'BRIEN J., BARGTEIL A., HODGINS J.: Graphical modeling and animation of ductile fracture. *SIGGRAPH* (2002), 291–294. 2

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. In *SIGGRAPH* (2005), pp. 957–964. 6

[Pro95] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graph. Interf.* (1995), pp. 147–154. 1

[RJ07] RIVERS A. R., JAMES D. L.: Fastlsm: fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26* (2007), 82–es. 2, 6

[SOG08] STEINEMANN D., OTADUY M. A., GROSS M.: Fast adaptive shape matching deformations. In *Symp. on Comput. Anim.* (2008), pp. 87–94. 6

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISHER K.: Elastically deformable models. *SIGGRAPH* (1987), 205–214. 1

[WMW86] WYVILL B., MCPHEETERS C., WYVILL G.: Animating soft objects. *The Visual Computer 2*, 4 (1986), 235–242. 2

© The Eurographics Association 2012.