Considerations toward a Dynamic Mesh Data Structure

S. Pena Serna[1], A. Stork[1,2] and D. W. Fellner[1,2]

[1]Fraunhofer IGD, Germany
[2]TU Darmstadt, Germany

Abstract

The use of 3D shapes in different domains such as in engineering, entertainment, cultural heritage or medicine, is essential for representing 3D physical reality. Regardless of whether the 3D shapes are representing physically or digitally born objects, meshes are a versatile and common representation for the 3D reality. Nonetheless, the mesh generation process does not always produce qualitative results, thus incomplete, non-orientable or non-manifold meshes frequently are the input for the domain application. The domain application itself also demands special requirements, e.g. an engineering simulation requires a volumetric mesh either tetrahedral or hexahedral, while a cultural heritage color enhancement uses a triangular or quadrangular mesh, or in both cases even hybrid meshes. Moreover, the processes applied on the meshes (e.g. modeling, simulation, visualization) need to support some operations, such as querying neighboring information or enabling dynamic changes of geometry and topology. These operations need to be robust, hence the neighboring information can be consistently updated, during the dynamic changes. Dealing with this mesh diversity usually requires dedicated data structures for performing in the given domain application. This paper compiles the considerations toward designing a data structure for dynamic meshes in a generic and robust manner, despite the type and the quality of the input mesh. These aspects enable a flexible representation of 3D shapes toward general purpose geometry processing for dynamic meshes in 2D and 3D.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

The use of 3D shapes in different domains such as in engineering, entertainment, cultural heritage or medicine, is essential for representing 3D physical reality. This is true for several 3D shape representation schemes such as point clouds, isosurfaces, subdivision surfaces, meshes, parametric surfaces and B-reps, among others. These representation schemes have a tight inherited relationship with the production technique. In the case of physically born objects, for example a laser scanner can typically generate a point cloud suitable for triangulation, while a CT scanner can usually generate volume data suitable for isosurfacing. On the other hand and referring to digitally born objects, CAD systems ([Lee99]) classically model with B-reps or CSG schemes and free-form 3D modeling systems normally work with parametric, polygonal or subdivision surfaces. Although the representation schemes for digitally born objects can easily be transformed into the representation schemes of the physically born objects, the opposite case is not a straightforward process, e.g. producing a B-rep with its corresponding geometric and topological description from a polygonal mesh is a complex procedure. Furthermore, the application context also influences the utilization of specific shape representation schemes, for instance in engineering B-reps, parametric surfaces and meshes are commonly used; polygonal, subdivision or parametric surfaces are frequently employed in the entertainment industry; in the cultural heritage field subdivision or polygonal surfaces and meshes are widely established; while B-reps, parametric surfaces and meshes are extensively exploited in the medicine sector.

Meshes are commonly used in the above mentioned domains and within these domains; meshes commonly support visualization, analysis, animation, modeling and simulation processes or other dedicated domain applications. The performance of these domain applications greatly profits from the neighboring information of the mesh. For in-

stance, a curvature analysis computes the vertex curvature for the 1-ring neighborhood ([BKP*10]) and a non-diagonal entry in a linear system of a simulation problem is computed by means of the contribution of the elements around an edge ([PSSSM09]). Nonetheless, the mesh generation process ([FG00]) does not always produce qualitative results, thus incomplete, non-orientable or non-manifold meshes frequently are the input for the domain application. Representing such meshes require generic and robust methods, in order to reliably compute the neighboring information. Additionally, the representation of dynamic meshes demands efficient methods for consistently updating the corresponding changes in the topology of the mesh.

There is a trend in the computer graphics community and especially in the geometry processing research field, focusing on the analysis and processing of dynamic meshes. This trend is enabling the discovering and the development of sophisticated methods and algorithms, leading to a more realistic reproduction of the reality, by means of either faithful animations or novel procedures for analyzing the behavior of the reality. Several methods and algorithms have been developed, in order to deal with dynamic meshes. Nonetheless, the majority of these approaches only consider the geometry of the mesh as dynamic, while the topology of the mesh remains unchanged. This consideration is justified by the complexity and the performance overhead that operations with dynamic topology, such as meshing, remeshing and multiresolution generate. Nonetheless, the implementation of algorithms with only dynamic geometry causes several artifacts and limitations, which need to be compensated, in order to make them reliable. On the other hand, an integral processing of meshes, where geometry and topology can dynamically be handled, can unlock the potential of current algorithms and it will also enable the combination of many operations, which today are considered as divergent.

We present in this paper the considerations toward designing a data structure for dynamic meshes in a generic and robust manner, despite the type and the quality of the input mesh. The methodology is generic enough to deal with triangular and quadrilateral meshes or tetrahedral and hexahedral meshes, or even hybrid surface or volumetric meshes. The robustness of the methods enables the representation of incomplete, non-orientable, non-manifold meshes or meshes with a high genus, thus the neighboring information is always consistently computed regardless of the quality of the input mesh. We also propose recommendations on how to identify geometric degenerecies or topological inconsistencies. Additionally, we present a possible implementation of the methods for computing and querying the neighboring information. We believe that these methods allow for a flexible representation of 3D shapes toward general purpose geometry processing for dynamic meshes in 2D and 3D.

2. Related Work

There is a substantial work in the scientific community in developing mesh data structures, most of them dedicated to specific domain application. We classify the proposed data structures in data structures for static meshes and data structures for dynamic meshes, regardless the dimensionality of the mesh.

2.1. Data Structures for Static Meshes

One of the pioneers in this area is Baumgart [Bau72], [Bau75], who introduced the winged edge data structure. In a later work, Muuss and Butler [MB91] described the advantages of the radial-edge data structure for representing non-manifold geometry and the corresponding Boolean operations for modeling. Campagna et al. [CKS98] proposed a data structure for triangular meshes called directed edges. Kettner [Ket98], [Ket99] presented a design framework for combinatorial edge-based data structures of polyhedral surfaces and planar maps, where the half-edge data structured was chosen and implemented (in CGAL). Levy et al. [LCCC01] proposed the circular incident edge list (CIEL) data structure for generating iso-surfaces in an unstructured grid, based on the notion of oriented half-edge. Botsch et al. [BSBK02] developed OpenMesh, an implementation of the half-edge data structure for static polygonal meshes. Kallmann and Thalmann [KT02] implemented a data structure for representing planar meshes in a compact manner, by means of vertex adjacency. Allegre et al. [ABGA04] proposed a hybrid shape representation, by means of combining a skeletal implicit surfaces and a polygonal mesh, which are assembled with an extended CSG tree. Alumbaugh and Jiao [AJ05] developed a compact array-based representation for surface and volume meshes, as a generalization of the half-edge and half-face data structures respectively. Blandford et al. [BBCK03], [BBCK05] proposed a compact representation for simplicial meshes, based on storing the link for a set of (d – 2) simplices.

Weiler et al. [WMKE04] presented a tetrahedral strips encoded in texture maps for rendering purposes, by means of ray casting algorithms. Cignoni et al. [CGG*04] developed a technique for out-of-core construction and view-dependent visualization of large surface models, based on a tetrahedral spatial partition of the model. De Floriani and Hui [DFH03] presented a data structure for non-manifold three-dimensional simplicial complexes also able to represent one- and two-dimensional top simplexes (wire-edge and dangling faces respectively). De Floriani et al. [DFGH04] proposed a data structure for d-dimensional non-manifold simplicial complexes, based on encoding boundary and coboundary relations within an incident graph. De Floriani and Hui [DFH05] compared different data structures for simplicial complexes, concluding that the most compact data structures are adjacency-based, while almost all support optimal retrieval of topological information. De Floriani [DFKP05]

created a survey on data structures for Level-of-Detail models of freeform geometry, including point-based, triangle-based and tetrahedron-based data structures for regular and irregular meshes. Lage et al. [LLLV05] developed a compact half-face data structure for manifold tetrahedral meshes, which is able to cope with scalability issues according to the application. Hui et al. [HVDF06] proposed a decomposition approach for representing non-manifold simplicial complex 3D shapes as a collection of tetrahedra, dangling triangles and wire edges.

Sander et al. [SNCH08] presented a scheme for efficient traversal of mesh edges for rendering purposes, using adjacency primitives and ordering these primitives for vertex cache locality. Gurung and Rossignac [GR09], [GR10] developed a compact representation for tetrahedral meshes, based on the sorted opposite table concept, which requires 4 references and 9 bits per tetrahedron. They provide a set of wedge-based operators for querying and traversing the mesh. Sieger and Botsch [SB11] presented the design decisions for a polygonal mesh data structure, aiming to improve usability, performance and memory consumption. Their work is based on an array-based halfedge data structure. Gurung et al. [GLLR11a] introduced a compact triangle mesh representation, which only needs about 2 integer reference per triangle, by means of combining the use of a quad mesh to represent the connectivity of the triangle mesh and a special sorting algorithm (SOT), nonetheless they need to rebuild the S table, when the connectivity changes. Gurung et al. [GLLR11b] developed the LR (Laced Ring) data structure for representing the connectivity of manifold triangle meshes. Its supports constant-time adjacency queries and it is suited for meshes with fixed connectivity, as any changes to the connectivity require a rebuilt of the data structure. This is a characteristic, which is shared by the data structures for static meshes. These data structure requires a precomputation step, to encode the neighboring information and any change in the topology of the mesh requires a complete re-computation of the information.

Furthermore, the capabilities of GPU computing are enabling new opportunities for visualization, but most important for computing time consuming applications. In this context, Lefohn et al. [LSK*06] developed a template library (Glift), abstracting the random access to GPU data structures such as stack, quadtree, or octree. However, they did not investigate the implementation and abstraction of mesh connectivity structures. Lefebvre and Hoppe [LH06] proposed GPU-based hashing structure for efficient random access of sparse data. In order to handle 3D domains, they proposed a hashing function for mapping the 3D domain to a 2D texture, e.g. by storing 2D pointers of the connectivity of 3D-adjacent voxel elements, generated by the intersection of the voxels with the surface mesh. DeCoro and Tatarchuk [DT07] introduced the probabilistic octree data structure, similar to adaptive octree structures, for vertex clustering in the context of a mesh simplification algorithm. Hu et al. [HSH09] created a data structure on the GPU for progressive surface meshes, by building a static buffer for the vertices and a dynamic buffer for the topology, although to maintain the dynamic buffer, they also required additional static and dynamic data structures. GPU-based data structure are very promising, however many of the current implementations deal with pre-computed dynamic data for efficient visualization purposes. Dynamic changes (e.g. on the topology of the mesh) are still very expensive.

2.2. Data Structures for Dynamic Meshes

The small number of data structures dealing with dynamic meshes, allow for a limit set of topological operation on the mesh. Weiler [Wei85] evaluated 4 different edge-based data structures for solid modeling operations (e.g. Euler operations). Chen and Akleman [CA03] developed a set of topological validity algorithms for different 2-manifold data structures in the context of mesh modeling. Danovaro and De Floriani [DDF02] proposed a compact data structure for a half-edge multi-resolution of tetrahedral meshes, built through full-edge collapses. De Floriani and Hui [DFH04] presented edge collapse and vertex split update operations for non-manifold simplicial objects and their corresponding encoding for a non-manifold indexed data structure. De Floriani et al. [DFMPS02], [DFMPS04] developed a data structure and the algorithms for dealing with non-manifold multi-resolution simplicial meshes. Danovaro et al. [DDFM*05] proposed the half-edge tree data structure for compactly encoding Level-of-Detail tetrahedral meshes, built through the application of half-edge collapses. Tobler and Maierhofer [TM06] developed a surface mesh data structure for rendering and subdivision operations, by means of representing the rendering information as indexed face set and separately representing the topological information for rendering and subdivision. Attene et al. [AGFF09] proposed a set of algorithms for converting a non-manifold tetrahedral mesh to a combinatorial 3-manifold or a PL 3-manifold by means of local modifications on the mesh. The proposed data structures for dynamic meshes are mainly dealing with multi-resolution problems (e.g. Level-of-Detail), however there are limitations regarding either the type of the mesh or the quality of the input mesh. Thus, we present the considerations toward designing a data structure for dynamic meshes in a generic and robust manner.

3. Memory and Performance

Data structures for static meshes usually aim to compactly encode the topology (and neighboring information) of the mesh, exploiting sequential dependencies and enabling a minimal set of queries for maximizing the performance of the dedicated domain application. Any other query, which is not foreseen in the original design, is normally very expensive. Thus, these data structures achieve minimal memory consumption and maximal performance for the given

application. On the other hand, data structures for dynamic meshes cannot encode the topology (and neighboring information) with sequential dependency, since it will require a re-build of the encoding if modifications on the topology are performed. Hence, the memory consumption cannot always be optimized. This implies that the data structure needs a mechanism to rapidly update the neighboring information and additionally to increase or decrease the number of components without expensive memory handling (0D - vertices: V, 1D - edges: E, 2D - faces: F, 3D - cells: C). Memory buffers are a solution for avoiding the costly memory handling for removal or addition of components, nevertheless in order to be efficient, the accurate estimation of the number of components is crucial. In this context, the Euler Formula gives a very good estimation for surface meshes with genus 0:

$$F + V - E = 2.$$

Thus, for a given number of faces (F) and vertices (V), the number of edges is:

$$E = F + V - 2.$$

A similar formula holds for volume meshes with genus 0:

$$V + F - E = C + 1.$$

Table 1 presents the behavior of the formula for the tetrahedral (Figure 1) and hexahedral meshes (Figure 2). Hence, for a given number of cells (C) and vertices (V), the number of edges can by estimated by

$$E \approx (2 \times V) + C,$$

and in a similar way, the number of faces can be estimated by

$$F \approx V + (2 \times C).$$

This estimation is much more accurate than the typical estimation for tetrahedral meshes ($E \approx 6 \times C$ and $F \approx 4 \times C$) and for hexahedral meshes ($E \approx 12 \times C$ and $F \approx 6 \times C$). If the meshes (surface or volume) contain genera, shells and loops, an approximation can be made, based on the previous formulas.

Table 1: *Topological components of two tetrahedral (rows 1. Sphere and 2. Gargoyle) and two hexahedral (rows 3. Hex2 and 4. Rubber) volume meshes with genus 0.*

V	E	F	C	$V + F - E$
13	42	50	20	21
221039	1152799	1723886	792125	792126
12	20	11	2	3
9367	24354	20812	5824	5825

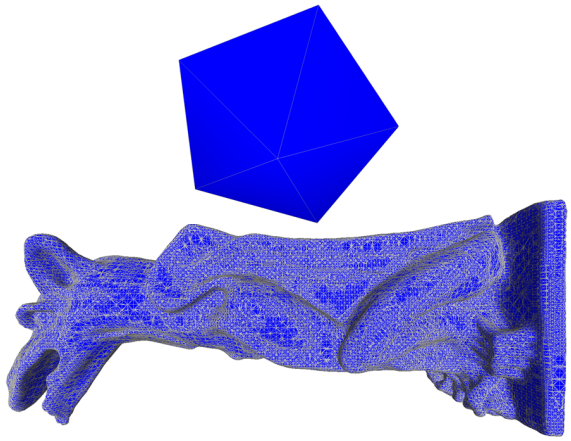

Figure 1: *Tetrahedral meshes with with genus 0: sphere and gargoyle.*

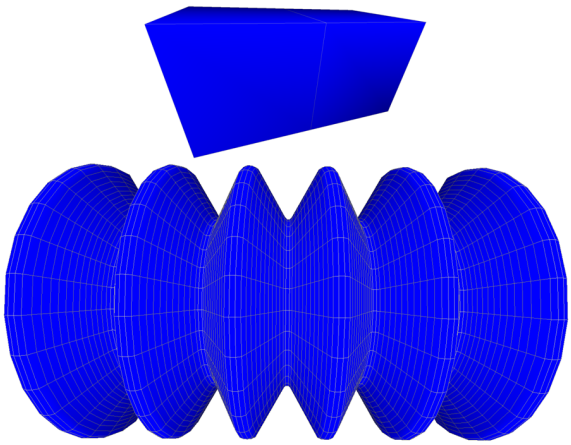

Figure 2: *Hexahedral meshes with genus 0: hex2 and rubber.*

4. Neighboring Information

After having the estimation regarding the number of components, it is important to define the needed neighboring information, in other words, the needed relationships for streamlining the application. For instance, a simulation process requires the computation of the contribution of the cells around an edge and the contributions of the cells around a vertex, therefore these relationships should efficiently be queried. These relationships can be pre-computed during the initialization phase of the mesh or these can be computed on demand during the querying process. The design decision is influenced by three factors: i) memory consumption, ii) querying performance, and iii) updating performance. If the relationships are pre-computed, more memory is required, the updating process is more time consuming, but the querying

process is very fast. On the other hand, if the relationships are computed on demand, less memory is required, the update process is less expensive, but the querying process is more time consuming.

An efficient querying process (considering that sequential dependencies cannot fully be exploited) can be achieved by means of creating *topological templates*. This is motivated by the hierarchical creation of the topological components of the mesh (*Cell*, *Face* and *Edge*). For instance, when creating a tetrahedron, the tetrahedron is reponsible for creating its triangles and at the same time, each triangle is reposible for creating its edges. Hence, given this hierachical process, the edges of the tetrahedron can be inferred from the edges of the triangle by means of the *topological template*. Table 2 presents an example of a topological template for a tetrahedron, based on the schematic indications from Figure 3.

The table presents the 4 triangles (F_i) of the tetrahedron and the generated edges (Fe_i) for each triangle. The arrows indicate the correspondence between the edges of the triangles with the edges of the tetrahedron (Ce_i). For this particular case of the topological template of the tetrahedron, a simple exercise was performed with the gargoyle of Figure 1 with about 800K tetrahedra, in order to compare the performance difference between the pre-computing and computing on demand strategies. The task aimed to traverse the 6 edges of the tetrahedra in a for loop: the pre-computation case required about 40ms, while the computation on demand required about 240ms, 6 times more, nonetheless the initialization process and the updating process were faster. In both cases, the topological template was used for pre-computing and for computing on demand.

The topological templates do not only improve the querying process, they also improve the initialization and updating process, because of the built hierarchal information. Referring back to the schematic example of Figure 3, a concept of *smart edges* is represented by two opposite edges, in this case e_i and e_{i+1}. Therefore when adding a new tetrahedron to the mesh, instead of looking for existing faces (triangles) in the mesh, by means of querying faces around the vertices of the tetrahedron, the existence of the smart edge is beforehand evaluated. If the edge exists, the query is reduced to the faces around the edge, which are much less than the faces around the three vertices of the needed face.

In the case that the edge does not exist, the face cannot exist either, therefore the face can directly be created. A similar process is applied, when creating a face, but in this case, *smart vertices* are the reference for querying for the edges of the face. It is worth mentioning that depending on the domain application, the querying process can also be streamlined by means of using spatial data structures (e.g. k-dimensional tree - k-d tree, binary space partitioning - BSP, octree, bounding volume hierarchy - BVH, etc.), which assist in finding subparts of the mesh in 3D space, avoiding querying and evaluating every single component of the mesh, for instance for efficient collision detection or mesh Boolean operation applications.

Table 2: *Topological templates for a tetrahedron, showing the correspondence between the edges of the triangles and the edges of the tetrahedron, given the hierarchical creation process.*

F_i	Fe_i	Fe_{i+1}	Fe_{i+2}
V_i, V_{i+1}, V_{i+3}	V_i–V_{i+1} $\rightarrow Ce_i$	V_{i+1}–V_{i+3} $\rightarrow Ce_{i+3}$	V_{i+3}–V_i $\rightarrow Ce_{i+4}$
V_{i+1}, V_i, V_{i+2}	V_{i+1}–V_i	V_i–V_{i+2}	V_{i+2}–V_{i+1}
$V_{i+2}, V_{i+3}, V_{i+1}$	V_{i+2}–V_{i+3} $\rightarrow Ce_{i+1}$	V_{i+3}–V_{i+1}	V_{i+1}–V_{i+2} $\rightarrow Ce_{i+2}$
V_{i+3}, V_{i+2}, V_i	V_{i+3}–V_{i+2}	V_{i+2}–V_i $\rightarrow Ce_{i+5}$	V_i–V_{i+3}

5. Mesh Modifications

The access to the neighboring information of the mesh, either by pre-computing or by computing on demand, enables efficient processes (e.g. modeling, simulation, visualization) of the domain application. In the case of dynamic meshes, these processes generally change the geometry and the topology of the mesh. There are two different approaches, either the geometry remains constant and the topology is changed (e.g. progressive meshes) or the geometry is changed and those changes invoke modification in the topology (e.g. mesh modeling). In either case, the typical actions on the mesh are called *topological operations*. The basic operations are: i) edge-split, ii) edge-collapse, and iii) edge-swap, and they aim to correct degeneracies on the mesh (e.g. inverted elements – faces or cells) or to improve the quality of the mesh (e.g. condition number for numerical simulations).

Although the operations require several steps, the mesh data structure only needs to support vertex or element removal and vertex or element addition. For instance, edge-split involves removing all the elements around an edge. Edge-collapse requires removing all the elements around one of the vertices of the edge to be collapsed (sometimes also merging the two vertices), and therefore removing the vertex (with its correspinding references). Edge-swap implies removing the elements around the edge to be swapped. The addition of new vertices or elements is already provided; since these operations are used during the initialization of data structure. Any other operation should be created on a top layer, in order to avoid overloading the mesh data structure with multiple application-motivated operations.

If the modifications of the mesh are not carefully applied,

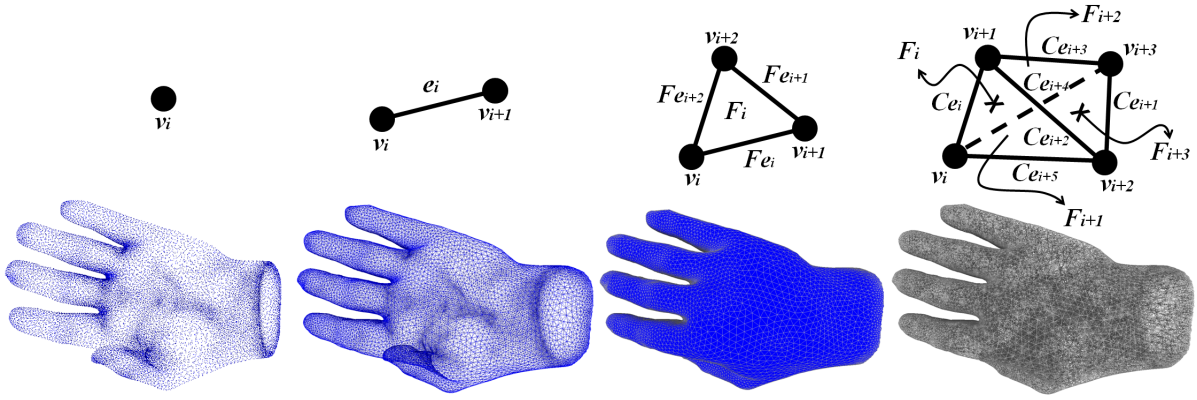

Figure 3: *Topological components of a mesh: vertices V, edges E, faces F, and cells C (model from the AIM@SHAPE repository).*

the mesh can easily become inconsistent. Thus, the data structure should also support basic geometric and topological tests, to warn the user if needed. For instance, to check the internal angles of faces and dihedral angles of cells, or to check the manifoldness of the mesh (for 2D: two faces per edge, for 3D: two cells per face or in both cases orientation of components). Nevertheless, the user should be able to enable or disable those checks, according to the needs of the application. An additional important aspect given the continuous change in the mesh is the ability to know the boundary of the mesh at any time without the need to traverse the mesh, especially for vertices and edges.

In the case of volume meshes, the faces on the boundary are easily recognized by having only one associated cell and this fact can be used for transferring information to its associated edges and vertices during the initilization and update processes. If a reference counter is associated to each vertex and edge, this needs to increase when an incident face is identified as being on the boundary and this needs to decrease, when an incident face is identified as not being on the boundary (e.g. when the neighboring cell is created). By the end of the initialization process or after the update process, a counter of 0 means that the edge or vertex is not on the boundary, any number above 0 indicated that the edge or vertex is on the boundary. This information is relevant for rapidly visualizing the changes in the mesh without performance drawbacks. Figure 4 illustrates on the top the modification of different surface and volume meshes and on the bottom the deformation (drag of a hole) of a tetrahedral mesh within a simulation environment.

6. Conclusions

There are several mesh data structures for triangular meshes and some for tetrahedral meshes. The exploration of quadrangular and hexahedral data structures is very limited in the community. Many of the data structures are designed for minimizing the memory consumption for specific domain applications. Nevertheless, there are not enough data structures, which robustly represent 3D shapes regardless of the quality of the given input mesh and there are very few data structures supporting dynamic meshes. These two aspects are an essential requirement toward general purpose geometry processing in 2D and 3D. We presented in this paper considerations in terms of memory and performance, neighboring information, and mesh modifications, toward a data structure for representing dynamic meshes in a generic and robust manner. We will further develop our ideas, in order to find the most appropriate tradeoff between the different exposed aspects, as well as new strategies for improving the performance without affecting the memory consumption and keeping the flexibility.

7. Acknowledgements

This work is partially supported by the European projects VISTRA (FP7-FoF-ICT-2011.7.4-285176) and 3D-COFORM (FP7-ICT-2007.4.3-231809).

References

[ABGA04] ALLEGRE R., BARBIER A., GALIN E., AKKOUCHE S.: A hybrid shape representation for free-form modelling. In *Proceedings of the Shape Modeling International 2004* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 7–18. 2

[AGFF09] ATTENE M., GIORGI D., FERRI M., FALCIDIENO B.: On converting sets of tetrahedra to combinatorial and pl manifolds. *Computer Aided Geometric Design 26*, 8 (November 2009), 850–864. 3

[AJ05] ALUMBAUGH T. J., JIAO X.: Compact array-based mesh data structures. In *Proceedings of the 14th International Meshing Roundtable* (September 2005), Springer-Verlag, pp. 485–504. 2

[Bau72] BAUMGART B. G.: *Winged edge polyhedron representation*. Tech. rep., Stanford University, Stanford, CA, USA, October 1972. 2

[Bau75] BAUMGART B. G.: A polyhedron representation for

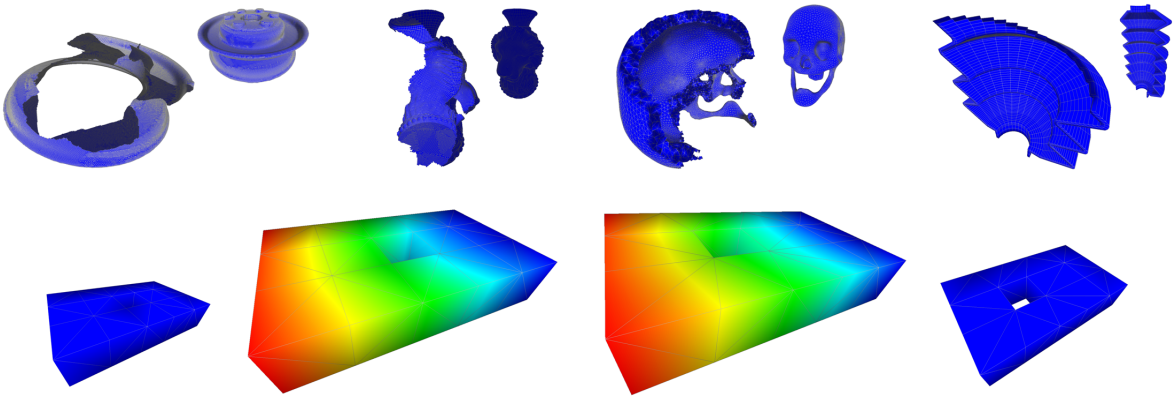

Figure 4: *Modification of an original triangular, quadrangular, tetrahedral and hexahedral mesh (some models from the AIM@SHAPE repository) on the top and a deformation of a tetrahedral mesh with a coupled simulation feedback on the bottom.*

computer vision. In *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition* (New York, NY, USA, 1975), AFIPS '75, ACM, pp. 589–596. 2

[BBCK03] BLANDFORD D. K., BLELLOCH G. E., CARDOZE D. E., KADOW C.: Compact representations of simplicial meshes in two and three dimensions. In *Proceedings of the 12th International Meshing Roundtable* (September 2003), pp. 135–146. 2

[BBCK05] BLANDFORD D. K., BLELLOCH G. E., CARDOZE D. E., KADOW C.: Compact representations of simplicial meshes in two and three dimensions. *International Journal of Computational Geometry and Applications 15*, 1 (2005), 3–24. 2

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LEVY B.: *Polygon Mesh Processing*. AK Peters, 2010. 2

[BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: Openmesh - a generic and efficient polygon mesh data structure. OpenSG Symposium, 2002. 2

[CA03] CHEN J., AKLEMAN E.: *Topologically Robust Mesh Modeling: Concepts, Data Structures and Operations*. Tech. rep., Texas A&M University, 2003. 3

[CGG*04] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Adaptive tetrapuzzles: efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. *ACM Transactions on Graphics 23*, 3 (August 2004), 796–803. 2

[CKS98] CAMPAGNA S., KOBBELT L., SEIDEL H.-P.: Directed edges - a scalable representation for triangle meshes. *Journal of Graphics Tools 3*, 4 (December 1998), 1–11. 2

[DDF02] DANOVARO E., DE FLORIANI L.: Half-edge multitessellation: A compact representation for multi-resolution tetrahedral meshes. In *Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission* (Washington, DC, USA, 2002), 3DPVT '02, IEEE Computer Society, pp. 494–499. 3

[DDFM*05] DANOVARO E., DE FLORIANI L., MAGILLO P., PUPPO E., SOBRERO D., SOKOLOVSKY N.: The half-edge tree: A compact data structure for level-of-detail tetrahedral meshes. In *Proceedings of the International Conference on Shape Modeling and Applications* (Washington, DC, USA, 2005), SMI Š05, IEEE Computer Society, pp. 332–337. 3

[DFGH04] DE FLORIANI L., GREENFIELDBOYCE D., HUI A.: A data structure for non-manifold simplicial d-complexes. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (New York, NY, USA, 2004), SGP '04, ACM, pp. 83–92. 2

[DFH03] DE FLORIANI L., HUI A.: A scalable data structure for three-dimensional non-manifold objects. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), SGP '03, Eurographics Association, pp. 72–82. 2

[DFH04] DE FLORIANI L., HUI A.: Update operations on 3d simplicial decompositions of non-manifold objects. In *Proceedings of the ninth ACM Symposium on Solid Modeling and Applications* (Aire-la-Ville, Switzerland, Switzerland, 2004), SM '04, Eurographics Association, pp. 169–180. 3

[DFH05] DE FLORIANI L., HUI A.: Data structures for simplicial complexes: an analysis and a comparison. In *Proceedings of the third Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), SGP '05, Eurographics Association, pp. 119–128. 2

[DFKP05] DE FLORIANI L., KOBBELT L., PUPPO E.: A survey on data structures for level-of-detail models. *Advances in Multiresolution for Geometric Modelling 1* (2005), 57–82. 2

[DFMPS02] DE FLORIANI L., MAGILLO P., PUPPO E., SOBRERO D.: A multi-resolution topological representation for non-manifold meshes. In *Proceedings of the seventh ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 2002), SMA '02, ACM, pp. 159–170. 3

[DFMPS04] DE FLORIANI L., MAGILLO P., PUPPO E., SOBRERO D.: A multi-resolution topological representation for non-manifold meshes. *Computer-Aided Design 36*, 2 (2004), 141–159. 3

[DT07] DECORO C., TATARCHUK N.: Real-time mesh simplification using the gpu. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), I3D '07, ACM, pp. 161–166. 3

[FG00] FREY P. J., GEORGE P.-L.: *Mesh Generation: Application to Finite Elements*. HERMES Science, 2000. 2

[GLLR11a] GURUNG T., LANEY D., LINDSTROM P.,

ROSSIGNAC J.: Squad: Compact representation for triangle meshes. *Computer Graphics Forum 30*, 2 (2011), 355–364. 3

[GLLR11b] GURUNG T., LUFFEL M., LINDSTROM P., ROSSIGNAC J.: Lr: compact connectivity representation for triangle meshes. *ACM Trans. Graph. 30*, 4 (July 2011), 67. 3

[GR09] GURUNG T., ROSSIGNAC J.: Sot: compact representation for tetrahedral meshes. In *Proceedings of the 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling* (New York, NY, USA, 2009), SPM '09, ACM, pp. 79–88. 3

[GR10] GURUNG T., ROSSIGNAC J.: *SOT: compact representation for tetrahedral meshes*. Technical Report GT-IC-10-01, Georgia Institute of Technology, 2010. 3

[HSH09] HU L., SANDER P. V., HOPPE H.: Parallel view-dependent refinement of progressive meshes. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 169–176. 3

[HVDF06] HUI A., VACZLAVIK L., DE FLORIANI L.: A decomposition-based representation for 3d simplicial complexes. In *Proceedings of the fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 101–110. 3

[Ket98] KETTNER L.: Designing a data structure for polyhedral surfaces. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry* (New York, NY, USA, 1998), SCG '98, ACM, pp. 146–154. 2

[Ket99] KETTNER L.: Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry: Theory and Applications 13*, 1 (May 1999), 65–90. 2

[KT02] KALLMANN M., THALMANN D.: Star-vertices: a compact representation for planar meshes with adjacency information. *Journal of Graphics Tools 6*, 1 (January 2002), 7–18. 2

[LCCC01] LÉVY B., CAUMON G., CONREAUX S., CAVIN X.: Circular incident edge lists: a data structure for rendering complex unstructured grids. In *Proceedings of the Conference on Visualization '01* (Washington, DC, USA, 2001), VIS '01, IEEE Computer Society, pp. 191–198. 2

[Lee99] LEE K.: *Principles of CAD / CAM / CAE Systems*. Addison-Wesley, 1999. 1

[LH06] LEFEBVRE S., HOPPE H.: Perfect spatial hashing. *ACM Trans. Graph. 25*, 3 (July 2006), 579–588. 3

[LLLV05] LAGE M., LEWINER T., LOPES H., VELHO L.: Chf: A scalable topological data structure for tetrahedral meshes. In *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 349–356. 3

[LSK*06] LEFOHN A. E., SENGUPTA S., KNISS J., STRZODKA R., OWENS J. D.: Glift: Generic, efficient, random-access gpu data structures. *ACM Trans. Graph. 25*, 1 (January 2006), 60–99. 3

[MB91] MUUSS M. J., BUTLER L. A.: Combinatorial solid geometry, boundary representations, and non-manifold geometry. *State of the Art in Computer Graphics: Visualization and Modeling 1* (June 1991), 185–223. 2

[PSSSM09] PENA SERNA S., SILVA J., STORK A., MARCOS A.: Neighboring-based linear system for dynamic meshes. In *Proceedings of the 6th Workshop in Virtual Reality Interactions and Physical Simulations* (Aire-la-Ville, Switzerland, Switzerland, 2009), VRIPHYS '09, Eurographics Association, pp. 95–103. 2

[SB11] SIEGER D., BOTSCH M.: Design, implementation, and evaluation of the surface_mesh data structure. In *Proceedings of the 20th International Meshing Roundtable* (2011), IMR, pp. 533–550. 3

[SNCH08] SANDER P. V., NEHAB D., CHLAMTAC E., HOPPE H.: Efficient traversal of mesh edges using adjacency primitives. *ACM Trans. Graph. 27*, 5 (December 2008), 144:1–144:9. 3

[TM06] TOBLER R. F., MAIERHOFER S.: A mesh data structure for rendering and subdivision. In *Proceedings of WSCG: International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2006), pp. 157–162. 3

[Wei85] WEILER K.: Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics and Applications 5*, 1 (January 1985), 21–40. 3

[WMKE04] WEILER M., MALLON P. N., KRAUS M., ERTL T.: Texture-encoded tetrahedral strips. In *Proceedings of the 2004 IEEE Symposium on Volume Visualization and Graphics* (Washington, DC, USA, 2004), VV '04, IEEE Computer Society, pp. 71–78. 2