Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2016) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

25th International Meshing Roundtable

An efficient approach for verifying manifold properties of simplicial complexes

Riccardo Fellegara[a,], Kenneth Weiss[b], Leila De Floriani[a,c]

[a] *University of Maryland, College Park, MD, USA*
[b] *Lawrence Livermore National Laboratory, Livermore, CA, USA*
[c] *University of Genova, Genova, Italy*

Abstract

While the vast majority of mesh processing tools assume a manifold mesh, many available meshes do not satisfy these constraints due to geometric defects and non-manifold singularities. We propose an efficient technique, based on a simple and compact data structure, for verifying topological properties of arbitrary simplicial complexes and experimentally demonstrate its effectiveness.

Keywords: spatio-topological data structure; compact and scalable representation; topological analysis

1. Introduction

Simplicial complexes are commonly used to model objects in a wide variety of applications including finite element analysis, solid modeling, animation, terrain modeling and visualization of scalar and vector fields. While the majority of these applications require that the input complex is manifold, many available datasets do not satisfy these constraints. Furthermore, on workflows that modify the mesh topology, such as mesh simplification, there is typically a need to preserve the manifoldness of the complex throughout the entire process.

We present an efficient strategy for verifying manifold properties on simplicial complexes of arbitrary dimensions. Our approach uses a simple and compact representation of the simplicial complex that couples an indexed representation of the complex with a hierarchical nested spatial index that decomposes the simplicial complex into blocks, enabling localized processing and analysis. Each such block b encodes a limited number of vertices from the mesh along with all simplices from the mesh necessary to locally reconstruct the *star* of each indexed vertex. The hierarchical spatial index can represent arbitrary complexes with a manifold or non-manifold domain in arbitrary dimensions, scales to large complexes and efficiently supports the generation of customized application-dependent localized data structures at runtime.

E-mail address: felle@umd.edu

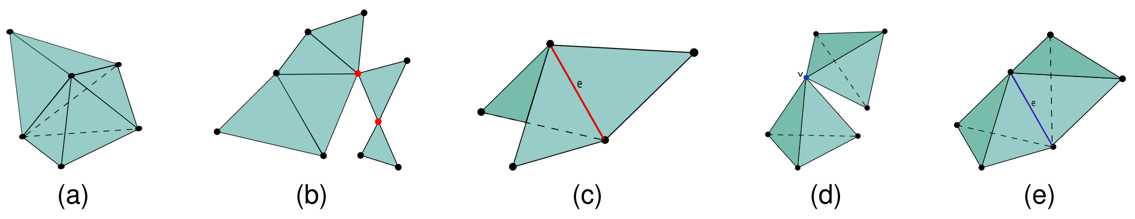

Figure 1: Example pure simplicial complexes. A manifold 3-complex (a), two non-manifold 2-complexes with vertex (b) and edge (c) singularities and two non-manifold 3-complexes with vertex (d) and edge (e) singularities. Meshes (b) and (d) are 0-connected, and (c) and (d) are 1-connected.

2. Background notions

A *k-simplex* σ is the convex hull of $k+1$ independent points in the Euclidean space $\mathbb{E}^n$ called *vertices*, where $k \leq n$ is the *dimension* of simplex σ. A d-dimensional *simplicial complex* in $\mathbb{E}^n$ is a finite set Σ of disjoint k-simplices of dimension at most d (with $0 \leq k \leq d \leq n$) such that the boundary of each k-simplex σ consists of the union of the other simplices of Σ with dimension less than k. The *star* of a simplex σ is the set of its *co-faces*, i.e. simplices in Σ that have σ as a face, and the *link* of a simplex σ consists of the faces in that star of σ which are not incident in σ. A simplex which does not belong to the boundary of any other cell is called a *top simplex*. If all top simplices have dimension equal to d, then Σ is a *pure* simplicial d-complex.

An *h-path* in Σ with $h \leq d$, is a sequence of simplices in Σ such that two successive cells σ_{i-1} and σ_i are h-adjacent, i.e. they are incident in a common h-simplex of Σ. The 1-skeleton of a simplicial complex Σ is a graph in which the nodes are the vertices of Σ, while the arcs are the 1-simplices of Σ. A simplicial complex Σ is said to be *connected* when its 1-skeleton is a connected graph, i.e. there is a 1-path between any two edges. A complex Σ is *k-connected*, with $k > 0$, when, for every pair of k-simplices f_1 and f_2, there is a (k-1)-path in Σ. A pure simplicial d-complex Σ is *pseudo-manifold* if Σ is d-connected, and each $(d-1)$-simplex has exactly one or two incident top simplices in Σ.

A simplicial complex Σ is a *combinatorial manifold* if and only if it is a pseudo-manifold and the link of every vertex v is combinatorially equivalent either to a $(d-1)$-sphere, if v is an internal vertex, or to a $(d-1)$-ball, if v is a boundary vertex. It has been shown in [3] that recognizing whether or not a $(d+1)$-complex is a combinatorial manifold $(d+1)$-complex is decidable for $d < 4$, is an open problem for $d = 4$, and is not decidable for $d \geq 5$ [5]. For each j-simplex (where j can be 0 or 1, for vertices and edges respectively) we check if the $(j+i)$-simplices in the link, with $j < i \leq k$, form a single connected component, and if the *Euler characteristic* of the link matches that of the $(d-1)$-ball or sphere.

3. Encoding a simplicial complex

We restrict our attention to pure simplicial complexes, since this property can be easily verified when the complex is loaded from file. That is, if any simplex does not have d+1 vertices, the complex is not pure.

We encode the simplicial complex using two data structures: (1) an indexed representation for the vertices and top simplices of the complex that encodes the boundary vertices of each top simplex in terms of an ordered list of the vertex [4] and (2) a spatio-topological data structure for reconstructing local mesh connectivity.

The indexed representation encodes the spatial position of each vertex as well as the indices of the vertices in the boundary of each top d-simplex. We represent the simplicial complex Σ using an array for the vertices Σ_V, and an array for the top d-simplices Σ_T. Since the arrays are stored contiguously, each vertex v has a unique position index i_v in the Σ_V array, and, similarly, each top simplex σ in Σ_T has a unique position index i_σ. Σ_T encodes the boundary connectivity from its top simplices to their vertices in terms of the indices i_v of the vertices within Σ_V.

The hierarchical data structure is a spatio-topological data structure that indexes the vertices and top simplices of Σ. For the spatial decomposition, we utilize a *bucket PR tree* [7] driven by a *bucketing threshold*, which we denote as k_V, that limits the the number of vertices indexed by a block b. A block b is considered *full* when it indexes more than k_V vertices and inserting a vertex into a full block causes the block to refine and to redistribute its indexed vertices among its children.

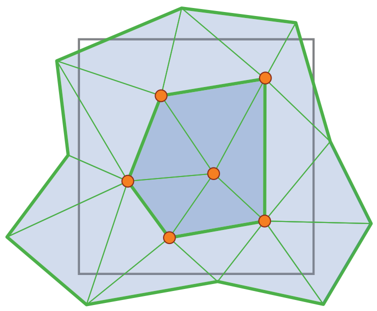

Figure 2: A leaf block in 2D with bucketing threshold $k_V = 6$ encodes a set of vertices and all the triangles incident in those vertices.

After inserting the vertices, the decomposition is fixed and each leaf block b of the tree uniquely indexes the set of vertices that it covers. We then insert into each block b the indices of all top simplices from Σ that are incident in at least one vertex v indexed by b. Thus, we have sufficient information to locally reconstruct the star of each vertex in b. Figure 2 shows an example of the geometry associated with a leaf block. Note that a simplex is indexed by each leaf block in the tree that contains one or more of its vertices. The storage requirements of the tree can be drastically reduced by exploiting the spatial locality of the complex to sort the indexed mesh representation and to compress the encoding of the lists associated with each leaf block of the tree, see [2] for details.

The simplicial complex can be processed in a streaming manner by iterating through the leaf blocks of the tree. For each leaf block b, we construct a local application-dependent topological data structure which we use to process the local complex. Since we are deferring the generation of these local topological data structures to runtime, we do not need to store a general global data structure and can customize the data structure to the application, e.g. using only the desired connectivity relations. The costs of generating these local data structures can be amortized over many local mesh processing operations.

4. Verifying global topological properties

Given a simplicial complex Σ, represented using the hierarchical representation described in Section 3, we verify the manifold conditions by checking that it is (1) connected (2) d-connected (3) pseudo-manifold and, where applicable, (4) a combinatorial manifold.

We use a recursive navigation of the tree to locally verify the connectivity of the complex. Within each leaf block b, we extract the *local* 1-skeleton of the complex in b and traverse the graph to label the edges of the complex by their connected component. The complex is connected if it contains only one connected component.

The procedure to check d-connectivity is similar. Within each leaf block b, we extract the *local* adjacency relations for the top d-simplices in b. These adjacency relations form a *local* adjacency graph, in which each node of the graph corresponds to a top simplex, while each arc corresponds to an adjacency relation between two top simplices. We then use the adjacencies to visit each top d-simplex σ in b, and label the connected components. If the procedure finds a non-manifold adjacency (e.g. more that two top simplices incident in a d-1 facet) or Σ has more than a single d-connected component, Σ is not pseudo-manifold.

The procedure for checking the combinatorial manifold property is completely local to the star of a vertex and can therefore be executed very efficiently in our representation. Since this test is only valid for $d < 4$, we test this property only on triangle and tetrahedral meshes through a recursive traversal of the tree. Within each leaf block b, we extract all the *local* link structures and then check the combinatorial manifold conditions (see Section 2), outputting the vertices and the edges that have invalid link conditions.

5. Experimental results

We evaluate the performance of our hierarchical data structure for verifying topological properties on simplicial complexes against the *Generalized Indexed Data Structure with Adjacencies (IA*)* representation [1], a low overhead

Table 1: Comparison of timings (in seconds) and auxiliary storage (in number of references) required to validate the models.

Data	**Type**		**connectedness**			***d*-connectedness**			**pseudo**
			time	**storage**		**time**	**storage**		**manifold**
NEPTUNE	TRIANGULAR	k_S	4.74	0.95K	✓	9.00	0.26K	✓	✓
		k_L	5.13	4.56K		8.27	1.15K		
		IA*	8.52	18.0M		0.48	4.01M		
LUCY		k_S	34.7	1.02K	✓	66.3	0.32K	✓	×
		k_L	38.8	4.64K		65.3	1.14K		
		IA*	44.5	126M		2.17	28.0M		
BONSAI	TETRAHEDRAL	k_S	46.9	7.98K	✓	108	2.80K	✓	✓
		k_L	45.2	16.0K		98.1	5.49K		
		IA*	55.1	86.3M		3.83	24.4M		
FOOT		k_S	54.8	7.94K	✓	141	2.81K	✓	✓
		k_L	55.0	16.1K		120	5.62K		
		IA*	66.5	104M		4.64	29.5M		
5D	PROBABILISTIC	k_S	98.0	3.46K	✓	520	15.2K	✓	✓
		k_L	89.7	37.4K		376	94.8K		
		IA*	98.7	28.5M		4.66	26.5M		
7D		k_S	1.78K	23.1K	✓	11.7K	1.05M	✓	✓
		k_L	1.69K	114K		9.17K	4.30M		
		IA*	OOM						
40D		k_S	8.29K	270K	✓	44.4K	1.36M	5.98K	×
		k_L	8.19K	2.25M		42.7K	5.04M	C.C.	
		IA*	OOM						

generalization of the *Indexed Data Structure with Adjacencies (IA)* [6] which supports pure, non-pure and non-manifold simplicial complexes of arbitrary dimension. The IA* is an array-based data structure that encodes all vertices and top simplices of an arbitrary simplicial complex Σ, plus a subset of its adjacent and boundary relations necessary to efficiently traverse the (possibly non-manifold) mesh connectivity. For each top p-simplex σ, the relation with its boundary vertices is encoded as well as the relations to all the top simplices sharing a $(p-1)$-face with σ. When the input mesh is pseudo-manifold, the IA* and IA representations are identical.

Since the adjacency relations are explicitly encoded in the IA* data structure, the algorithm for checking d-connectivity visits the *global* adjacency graph starting from any top k-simplices. Conversely for checking the connectivity, the IA* procedure first extracts and then visits the *global* 1-skeleton of Σ. Finally, for checking the combinatorial manifold property, the IA* first extracts the links from the star of each vertex, and then verifies the corresponding conditions using the same algorithm defined for our hierarchical representation.

Table 1 highlights our results on triangle and tetrahedral meshes in 3D as well as some higher-dimensional models generated through a recursive Sierpinski-like refinement procedure. The triangle and tetrahedral meshes are *native* models ranging from 4 to 28 million triangles and from 24 to 29 million tetrahedra. With the term *native*, we refer to models that we have found available in this format in the public domain, generated from meshing a volume or a surface discretizing an object in space. To experiment with models in higher dimensions we have generated some models based on a process that we call *probabilistic Sierpinski filtering*, where we uniformly apply regular refinement to all simplices in the mesh and randomly remove a subset of the generated simplices during each refinement step. For our experiments, we have created 5-, 7- and 40-dimensional models, with differing levels of refinement, ranging from 16.5 million (in 40 dimensions) to 258 million (in 7 dimensions) top simplices. All tests have been performed on a PC equipped with an Intel CPU i7-3930K, at 3.2 Gigahertz, and with 64 gigabytes of RAM.

For each tree, we generated two hierarchical indexes, parametrized by a smaller bucketing threshold, k_S, and a larger one, k_L. We first consider the effect of choosing different bucketing thresholds. On the native models, the k_S tree requires 10-20% less time than the k_L tree for connectedness, but 10-30% more time for d-connectedness. On the higher-dimensional models a k_L tree requires 10% less time for connectedness and up to 40% less time for

d-connectedness. Note that the k_S trees always require less auxiliary storage than its k_L variant: 40-50% less storage for tetrahedral meshes and 70-90% less storage for the other models.

We now compare our hierarchical representation against the IA* data structure. While the hierarchical trees are faster at validating connectedness, requiring from 10% to 60% less time, they are significantly slower at validating the d-connectedness. This is reasonable since the IA* data structure explicitly encodes the adjacency relations, while the hierarchical structure must compute adjacencies on the fly before running the connectedness algorithm. On the other hand, the hierarchical data structures requires less than half the overall storage of the IA* data structure, and have a significantly lower memory requirement while generating the structure. Thus, the hierarchical models can be easily generated on the *higher-dimensional* models, while the IA* data structure runs out of memory and can only be generated on the lower dimensional model. Furthermore, since all algorithms are local on the hierarchical structure, it requires a small fraction of the storage for encoding the auxiliary data structures in each processing kernel.

Finally, we compare the experiments executed for verifying the combinatorial manifold property (not shown in the table). In this case, we only consider the processing time for extracting the star of each vertex, as the procedures that extract the links and that verify the property are shared between the two data structure. For extracting the star of each vertex we have observed that our hierarchical representation requires from 80% to 95% less time than the IA* data structure.

Acknowledgements

This work has been supported by the National Science Foundation under grant number IIS-1116747. It has also been performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.

Datasets are courtesy of *Volvis* repository (BONSAI and FOOT) and *Aim@Shape* repository (NEPTUNE and LUCY).

References

[1] D. Canino, L. De Floriani, and K. Weiss. IA*: An adjacency-based representation for non-manifold simplicial shapes in arbitrary dimensions. *Computers & Graphics*, 35(3):747–753, 2011.

[2] R. Fellegara. *A spatio-topological approach to the representation of simplicial complexes and beyond.* PhD thesis, Department of Computer Science (DIBRIS), University of Genova, 2015.

[3] M. H Freedman and F. Quinn. *Topology of 4-manifolds*, volume 39. Princeton University Press Princeton, 1990.

[4] C. L. Lawson. Software for C^1 surface interpolation. In J.R. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, 1977.

[5] A Nabutovsky. Geometry of the space of triangulations of a compact manifold. *Communications in mathematical physics*, 181(2):303–330, 1996.

[6] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics (TOG)*, 12(1):56–102, 1993.

[7] H. Samet. *Foundations of Multidimensional and Metric Data Structures.* The Morgan Kaufmann series in computer graphics and geometric modeling. Morgan Kaufmann, 2006.