A general and efficient representation for multiresolution meshes: application to quad/triangle subdivision

Pierre Kraemer* David Cazier† Dominique Bechmann‡

Abstract

Quad/triangle subdivision unifies triangular and quadrilateral subdivision schemes. Despite its interest, this scheme is rarely used in the multiresolution edition framework, mainly because of the lack of a sufficiently general data structure that allows a simple and efficient implementation of multiresolution meshes built with it. We show in this paper how multiresolution half-edges, defined as an extension to the half-edges data structure, can uniformly and efficiently manage this kind of multiresolution mesh.

1 Introduction

Modeling with multiresolution subdivision surfaces is very popular among the computer graphics community [9]. Indeed, multiresolution edition offers many advantages. It combines the simplicity and topological generality of subdivision surfaces, with the possibility of editing a mesh at different resolution levels: an edit at some fine level leads to a small scale deformation, while an edit at some coarse level leads to a large scale – details preserving – smooth modification of the surface.

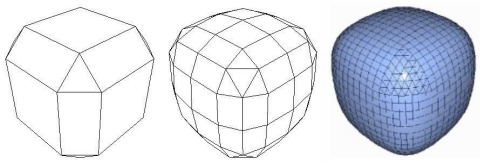
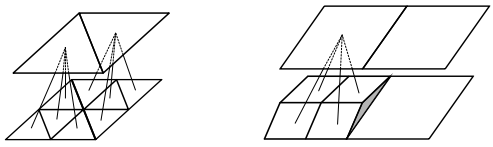


Figure 1: Quad/triangle subdivision

A majority of works on multiresolution subdivision surfaces are based on schemes that work on regular meshes. For example the Loop scheme works on triangular meshes and the Catmull-Clark scheme works on quadrilateral meshes. The quad/triangle subdivision scheme [7] (figure 1) unifies triangular and quadrilateral subdivision. The interest of this type of subdivision comes from two observations. First, most meshes are composed of both areas that are more naturally quadrilateral and areas that are more naturally triangular. Second, quadrilateral schemes behave poorly on triangular meshes, and vice-versa. This scheme has been studied these last years [4, 5] with the aim of improving the quality and smoothness of the limit surface. Our goal here is not to prove the relevance of this scheme, but to provide an easy way to use it in the multiresolution edition framework.

Previous works

The most widely used data structure for encoding multiresolution subdivision surfaces is the forest of quadtrees [1], which is naturally derived from the nested hierarchy of faces generated by the quadrisection-based subdivision schemes. To be able to support quad/triangle subdivision, this forest would have to mix two kinds of quadtrees (trianglular - figure 2a - and quadrilateral - figure 2b). In addition to the topological cracks generated by adaptive subdivision in this kind of structures (see figure 2b), this would lead to a tedious development of adjacency operators.



(a) Triangular (b) Quadrilateral

Figure 2: Quadtree structures

The Qreg data structure [6] proposes a two-layers representation that exploits the regularity of meshes. The mesh is separated in regions that must satisfy regularity constraints (so that neighborhood is implicitely encoded). Each region can be subdivided regularly. It can support quad/triangle subdivision but fine-grain adaptive subdivision is not easily supported, and topological cracks problems are not avoided.

The half-edges data structure [8] is a very general and efficient structure for the representation of arbitrary meshes. It has already been used in a multiresolution context, but in a decimation approach based on edge collapses on triangular meshes [3]. There is here no

*LSIIT, UMR 7005 CNRS, Université Louis Pasteur, Strasbourg, France, kraemer@lsiit.u-strasbg.fr

†cazier@lsiit.u-strasbg.fr

‡bechmann@lsiit.u-strasbg.fr

notion of resolution levels, and only the original mesh and the current decimated mesh are available.

We introduced in [2] a new data structure called multiresolution half-edges, which extends the classical half-edges with multiresolution capabilities. It has the advantage to be very general (any kind of mesh, any refinement operation between levels), while allowing as simple and efficient navigation as in a standard half-edge structure *on any resolution level*. It has already been used to encode multiresolution subdivision surfaces generated by schemes working on regular meshes (both triangular and quadrilateral). In this context, we compared it to quadtrees and showed that it is an efficient alternative, notably when subdividing the mesh adaptively (maintaining the topological consistency of the mesh). Moreover, memory requirements were shown to be equivalent.

Contribution

We show in this paper how the generality of the multiresolution half-edges data structure allows us to use it for the encoding of adaptive multiresolution subdivision surfaces generated by quad/triangle subdivision, bringing the advantages of this scheme to multiresolution edition. This can be achieved mainly because our topological model is completely independant of the type of mesh, and of the applied subdivision rules.

2 Multiresolution half-edges

Recalls

The standard half-edges data structure uses the adjacencies between the edges to represent the topology of the mesh of orientable 2-manifolds. Each edge of the mesh consists in two symmetric half-edges, each one having pointers to its *opposite*, *next* and *previous* half-edges. This structure comes in two definitions: each half-edge is associated with either a vertex-edge pair (primal), or a face-edge pair (dual). Even if our extension is defined for both, we will focus here on the primal version. The associated geometrical data, or embedding, consists here in 3D points associated to the vertices of the mesh. Each half-edge has a pointer to a 3D point, and all the half-edges attached to a same vertex share a pointer to the same point.

Definition

A multiresolution half-edge structure is a hierarchy of half-edge structures. As shown in figure 3 the half-edges are distributed in distinct sets corresponding to the resolution level in which they are introduced. The half-edges of the starting mesh (or level 0 mesh) belong to the set L^0. On each finer level i, a set of half-edges L^i is introduced and added to the existing ones.

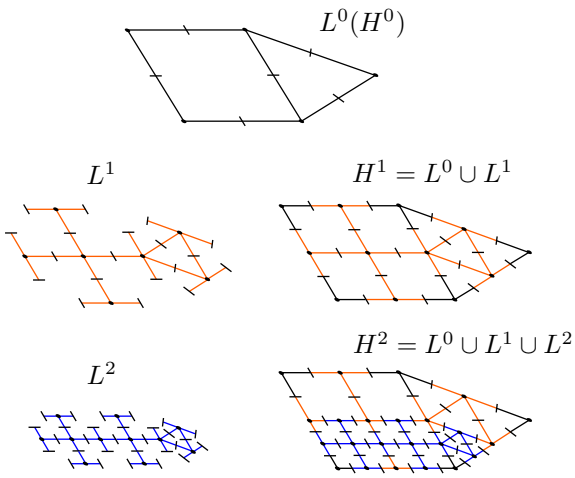
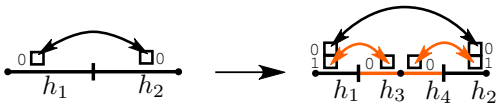


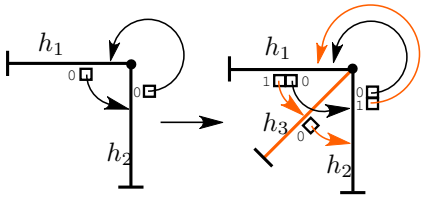
Figure 3: Successive sets of half-edges

Let call H^i the set of half-edges introduced in the levels inferior or equal to i. Obviously, we have $H^i = \bigcup_{j=0}^{i} L^j$. These sets form a sequence $\{H^i\}_{i\geq 0}$ such that: $H^0 \subset H^1 \subset H^2 \subset \ldots \subset H^i \subset \ldots$. The total number of half-edges is equal to the number of half-edges needed to describe the finest resolution level.

The *opposite* and *next* links are indexed by the resolution level. Each half-edge stores an array of *opposite* and an array of *next* pointers, corresponding to the links on different resolution levels. Let a be the introduction level of a half-edge h, and k be the maximum resolution level of the mesh. As h cannot be linked to other half-edges at a level which is inferior to a, its arrays only need to store $(k-a)$ links. Thus, the pointers to the half-edges linked to h on resolution level i can be retrieved in the cells of index $(i-a)$ of its arrays.



(a) *opposite* link



(b) *next* link

Figure 4: Indexation of topological links

Figure 4(a) shows an edge at resolution levels 0 and 1. Between these levels, a new vertex is inserted, introducing two new half-edges. h_1 and h_2 are linked to h_3 and

h_4 on level 1, preserving their relation on level 0. The array of *opposite* pointers of h_1 and h_2 contains now two elements. For these two half-edges whose introduction level is 0, the cell of index 1 contains the pointers of resolution level 1. For h_3 and h_4 whose introduction level is 1, it is the cell of index 0 that contains the pointers of resolution level 1. Figure 4(b) illustrates a vertex at resolution levels 0 and 1. Between these levels, a half-edge h_3 has been inserted in the vertex. Here again, we see that the half-edges introduced on level 0 (h_1 and h_2) store now two pointers in their array of *next* links.

The embedding information is also indexed by the resolution level. Each half-edge stores an array of pointers to 3D points corresponding to the embeddings of the vertices at different resolution levels. These arrays are managed exactly in the same way as the arrays of topological links.

The mesh of level l can be traversed as efficiently as a classic half-edges structure by considering the half-edges of H^l and their relations and embedding on level l.

3 Application to quad/triangle subdivision

As in every subdivision scheme, two steps can be distinguished in a quad/triangle subdivision step: refine the topology and compute the geometry on the new finer mesh. The topology refinement step consists in triangle quadrisection in the triangular areas, and square quadrisection in the quadrilateral areas of the mesh. This does not cause any topological problem as in both cases, face quadrisection is achieved by cutting the edges and inserting new edges. Different strategies can be applied for the geometry computation step and more details can be found in [7, 4, 5].

Figure 5 illustrates a detail of a mesh at resolution levels 0 and 1. The arrays of *opposite* pointers are illustrated for two edges. The half-edges in bold orange are those belonging to L^1. The topological refinement is executed in the same way as in a standard half-edge structure, except that the new half-edges are introduced and linked with the others in the new resolution level – thus keeping available the old one.

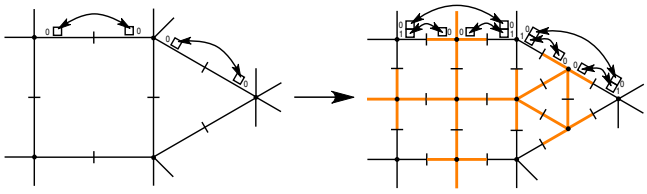


Figure 5: Quad/triangle with multiresolution half-edges

The valence of the vertices is not modified by such a subdivision step. This means that for a multiresolution half-edge h introduced on a resolution level l, its *next* link never changes in the finer resolution levels. Thus, it is unnecessary to redefine on each finer level that the *next* link of h is the same than on level l. In this particular case, we can store a simple pointer instead of the array of *next* pointers, and use this pointer on every resolution level finer than l. In a more general case, it can be solved in the same way as adaptivity is managed.

Adaptive subdivision

Adaptivity means that starting from a given resolution level, a face is no longer subdivided. In this kind of subdivision, it implies that some edges of the mesh are no longer cut. This means that for a multiresolution half-edge h whose corresponding edge stops being subdivided starting from a level l, its *opposite* link does not change in the finer levels. Thus, it is unnecessary to redefine this link on each finer level.

This is managed in the following way: if a relation does not change for a multiresolution half-edge between two resolution levels, then its corresponding array of pointers does not grow and no pointer is duplicated. In consequence, the way of accessing the topological links has to be changed: if nothing is defined on the queried resolution level (i.e. if the the array of pointers is too small), we pick the last pointer in the array. The same strategy is applied to access to the embedding.

In an adaptively generated mesh, cells belonging to different resolution depths coexist. The mesh of level l is composed of cells of level inferior or equal to l. The level of the different cells can be recovered without any supplementary storage cost: the level of an edge is equal to the maximum of the introduction levels of its two half-edges; the level of a face is the minimum of the levels of its edges; the level of a vertex is the maximum of the levels of its edges.

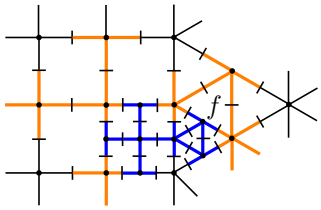
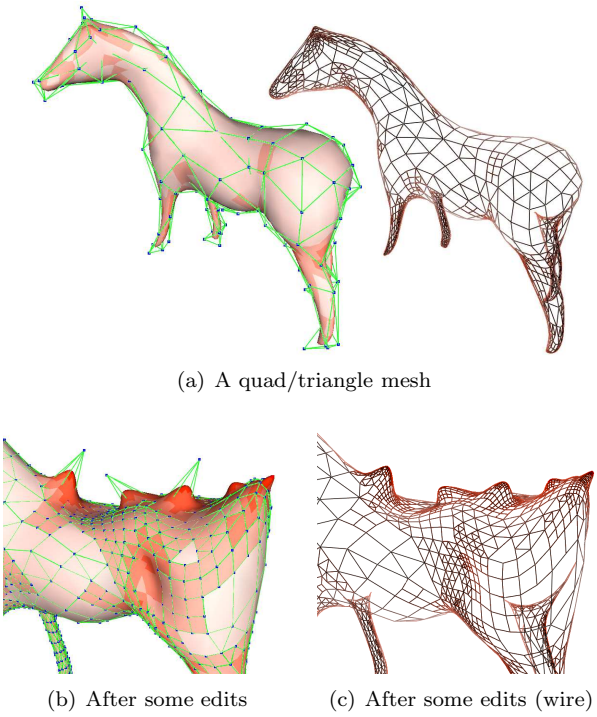


Figure 6: Adaptive quad/triangle mesh

Figure 6 illustrates an adaptively refined quad/triangle mesh. Thanks to the ability of multiresolution half-edges to represent arbitrary faces, topological consistency is maintained in the mesh: no topological crack is created at the frontier between resolution levels. In the same time, the number of edges of the faces that lie on this frontier is changed. For example the face f, which was a triangle on level 1, is now a 4-sided face on level 2. To be able to continue



(a) A quad/triangle mesh

(b) After some edits

(c) After some edits (wire)

Figure 7: Example of object

the subdivision properly, one must recover that this face was originally a triangle. This can be done simply and again without any supplementary storage cost: let l_f be the level of a face in the mesh of level l (following the above definition, f is a level 1 face in the mesh of level 2); to recover its original number of edges, one just have to count its edges in the mesh of level l_f.

Implementation

This data structure has been implemented in a multiresolution edition tool. Figure 7 shows examples of objects obtained by adaptive quad/triangle subdivision (colors are function of the resolution depth). Adaptivity is here automatically driven by a local curvature criterion. The mesh is dynamically updated during the editing to always satisfy the criterion. A restriction is introduced to forbid adjacent faces to have more than one level of difference: this allows to have always full subdivision masks, and thus to avoid the temporary computation of the eventually missing vertices.

Time complexity

In a multiresolution half-edges structure, whatever the considered resolution level, neighborhood queries are executed with exactly the same efficiency. Indeed, as said above, the mesh corresponding to every resolution level l can be traversed as a standard half-edge structure.

Adjacency queries within a given resolution level are thus executed as simply and efficiently as in a half-edge data structure, i.e. in constant time, compared to the logarithmic time needed to resolve such queries in a tree structure. In the multiresolution subdivision surfaces context, this is a relevant improvement, as the most widely used operation is the traversal of the neighborhood of vertices.

4 Conclusion

We have shown in this paper how the multiresolution half-edges data structure can be used for the representation of adaptive multiresolution quad/triangle subdivision surfaces. This is possible thanks to the generality of this structure which allows the representation of any kind of mesh and the application of any refinement operation between the resolution levels. Each resolution level is instantly available as a standard half-edge structure. Neighborhood queries are thus executed in optimal time on every resolution level. Moreover, thanks to the support of arbitrary faces, the topological consistency of the mesh is always maintained.

In our future works, we are going to consider the representation of multiresolution subdivision surfaces generated by other subdivision schemes. We will also investigate a volumetric version of the structure.

References

[1] L. DeFloriani, L. Kobbelt, and E. Puppo. A survey on data structures for level-of-detail models. *Advances in Multiresolution for Geometric Modelling, Series in Mathematics and Visualization*, pages 49–74, 2004.

[2] P. Kraemer, D. Cazier, and D. Bechmann. Multiresolution half-edges. In *Proceedings of SCCG'07 (to be published)*, 2007.

[3] R. Pajarola and C. DeCoro. Efficient implementation of real-time view-dependent multiresolution meshing. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):353–368, 2004.

[4] J. Peters and L.-J. Shiue. Combining 4- and 3-direction subdivision. *ACM Trans. Graph.*, 23(4):980–1003, 2004.

[5] S. Schaefer and J. Warren. On c^2 triangle/quad subdivision. *ACM Trans. Graph.*, 24(1):28–36, 2005.

[6] L.-J. Shiue and J. Peters. A pattern-based data structure for manipulating meshes with regular regions. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, pages 153–160, 2005.

[7] J. Stam and C. Loop. Quad/triangle subdivision. *Computer Graphics Forum*, 22(1):79–85, 2003.

[8] K. Weiler. Edge-based data structures for modeling in curved-surface environments. *Computer Graphics and Applications*, 5(1):21–40, 1985.

[9] D. Zorin. Modeling with multiresolution subdivision surfaces. *Tutorial Eurographics*, 2005.