Interpolation of Triangle Hierarchies*

Axel Friedrich Konrad Polthier Markus Schmies

September 10, 1998

Abstract

We consider interpolation between keyframe hierarchies. We impose a set of weak constraints that allows smooth interpolation between two keyframe hierarchies in an animation or, more generally, allows the interpolation in an $n-$parameter family of hierarchies. We use hierarchical triangulations obtained by the Rivara element bisection algorithm and impose a weak compatibility constraint on the set of root elements of all keyframe hierarchies. We show that the introduced constraints are rather weak.

The strength of our approach is that the interpolation works in the class of conforming triangulations and simplifies the task of finding the intermediate hierarchy, which is the union of the two (, or more,) keyframe hierarchies involved in the interpolation process. This allows for an efficient generation of the intermediate connectivity and additionally ensures that the intermediate hierarchy is again a conforming hierarchy satisfying the same constraints.

1 Introduction

Surfaces in animation often change their shape in time. If there is no functional description of the deformation available, one often uses the keyframe technique to describe the animation. Here a surface is stored at a finite set of key-times, and in-between surfaces are computed by interpolating between adjacent keyframes. In the special case that all keyframe surfaces have the same underlying simplicial complex, the interpolation task reduces to the simple linear interpolation between corresponding pairs of vertices, or interpolation of higher polynomial order if more keyframes are considered.

The interpolation task becomes more delicate if the keyframe surfaces are allowed to have different underlying simplicial complexes. In our discussion we restrict all surfaces of an animation to be topologically equivalent, i.e. they have the same genus and boundary curves. Nevertheless, our task remains to interpolate between topologically equivalent – but differently discretized – surfaces. Such surfaces occur naturally, in numerics when an

*To appear in IEEE Visualization'98

initial surface evolves by minimizing an energy functional and it is adaptively refined and coarsened after each time step. These surfaces also arise in flow visualization, where an initial test ball is inserted in the flow and after some time-steps the ball strongly deforms, requiring an adaptive change of its mesh. In both cases one obtains a new keyframe object after each time-step, and interpolation between keys with different meshes is required for slow-motion playback.

We consider hierarchies of triangles obtained by an element bisection algorithm, rather than by vertex split/edge collapse methods, and impose the following two constraints: firstly, the bisection scheme in each hierarchy follows the rules of Rivara [14] and, secondly, a certain correspondence between the root elements of all hierarchies is required. These assumptions are rather weak, especially since it is only the second constraint which requires an adjustment between different hierarchies. After this initial adjustment, each keyframe hierarchy may be locally refined and coarsened depending only on its own error criteria without any reference to the other hierarchies of the family. The compatibility of different meshes follows from the Rivara bisection method.

Instead of using the Rivara bisection method, one might try to use a 4-1 split as subdivision rule. After a local refinement the 4-1 split must be accompanied by a process called conformal closure to remove hanging vertices, e.g. by introducing so-called green edges [1]. These green edges are responsible for case distinctions and require further subdivisions when interpolating between different hierarchies. These tasks can be handled, but the effort increases when interpolating in multi-parameter families.

2 Review

Triangle mesh as a simplicial complex In computer graphics and numerics a variety of different triangle meshes are used. We restrict ourselves to *conforming triangulations:* a triangle is not allowed to have a vertex of another triangle in the interior of one of its edges. This avoids discontinuity problems in the shape and so-called *hanging nodes*, vertices which are required to lie on an edge. It is not essential to constrain to conforming triangulations, but it avoids a number of unwieldy distinctions related to hanging nodes. Further, we restrict our discussion to piecewise linear meshes although piecewise "higher order" triangular meshes would also work.

In our concept the interpolation property between two different surfaces depends on their underlying topological simplicial complexes rather than on the actual geometric shapes. It is important to distinguish between the topology of the mesh, i.e. the combinatorics, and the geometric position of the vertices. Such a distinction is also essential in mesh optimization algorithms, see [10], where the same shape is equipped with different topological meshes.

Formally, a triangular mesh T has the topology of an *abstract simplicial complex* K combined with a geometric realization. The latter is uniquely determined by a set of *geometric vertices* $V = \{v_1, .., v_m\} \subset \mathbb{R}^3$, and we can identify T with the pair (K, V). The simplicial complex K formally represents the connectivity of the mesh. It is given by a finite set of abstract vertices $\mathfrak{V} = \{\mathfrak{v}_1, .., \mathfrak{v}_m\}$ and a finite set of subsets S, called simplices, representing the vertices, edges, and triangles of the mesh. Further it is required that if $\sigma \in S$ is a simplex, then every subset $\tau \subset \sigma$ is also a simplex $\tau \in S$.

Each abstract $n-$simplex $\sigma \in S$ containing $n+1$ abstract vertices has a *topological realization* $|\sigma|$ as the standard simplex $\Delta(e_1, .., e_{n+1}) \subset \mathbb{R}^{n+1}$, the convex hull of the unit vectors $e_1, .., e_{n+1}$ in $\mathbb{R}^{n+1}$.

A *geometric realization* of an abstract simplicial complex (K, V) is uniquely given by the set of geometric vertices V and a set ϕ_V of affine maps

$$\phi_\sigma : |\sigma| \to convexHull(v_{i_1}, .., v_{i_{n+1}}) \subset \mathbb{R}^3$$

for each abstract simplex $\sigma \in S$ with $\sigma = \{\mathfrak{v}_{i_1}, .., \mathfrak{v}_{i_{n+1}}\}$. In the same way as in [10] we denote the geometric realization by ϕ_V to emphasize that it is fully specified by the set of geometric vertices V.

If two, or more, abstract triangles share a common edge, they are called *adjacent* or *neighbours*. An edge belonging to only one triangle is part of the boundary. The above material can be found in any text book on algebraic topology, one source is the recent introduction by Bloch [3].

Bisection method of Rivara Refinement and coarsening algorithms have a long tradition in numerics and computer graphics, and some can be used to generate hierarchical data representations where each child triangle is combinatorially a subset of its parent. Since we consider curved surfaces in $\mathbb{R}^3$, it is essential to maintain the distinction between the simplicial combinatorics and the geometric realization: if we bisect a triangle to obtain two children, then combinatorially the two children are considered as subsets of its parent, but in the geometric realization the children need not be part of its parent triangle.

There exist different types of hierarchical triangulations, and a good overview and formal concept are given in DeFloriani and Puppo [6]. The concept of vertex-based hierarchies is described in detail in Hoppe's papers [8] and [9]. In numerics, it is essential to ensure stability of a sequence of triangulations or a hierarchical triangulation, i.e. to bound the angles inside all triangles uniformly from below. In visualization, small angles may also disturb the visual perception since they sometimes allow, the element normals to vary heavily in the neighbourhood of such a degenerate triangle.

The bisection algorithm of Rivara [14] addresses the problem of how to locally refine a conforming triangulation to a new conforming triangulation and, additionally, of how to ensure that all angles in subsequently refined triangulations are greater than or equal to half of the smallest angle in the original triangulation. The method leads to nested triangulations and allows smooth transition between different levels of detail. In his original formulation, Rivara bisects a triangle exactly at the longest edge. Bänsch [2] generalized the method by introducing a formal refinement edge. In each triangle a single edge is marked as *refinement edge,* i.e. if the triangle is refined, then it is refined by bisecting its refinement edge, and the two child triangles inherit a refinement edge in the manner shown in figure 1. In the simplicial complex an additional vertex is inserted at the midpoint of the refinement edge.

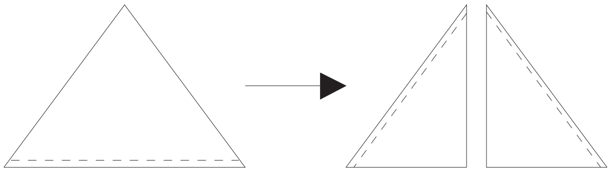

Figure 1: The Rivara bisection method refines a triangle exactly at its refinement edge. Each child inherits a refinement edge as shown.

Formally, the Rivara algorithm assumes that in a conforming triangulation T each triangle has an arbitrary edge marked as refinement edge. Let T_k be a conforming triangulation with a subset of triangles $S \subset T$ marked for refinement, usually according to some local error criteria, then the method consists of the following steps:

Rivara Bisection Method (A)

1. All marked triangles S are bisected according to the Rivara bisection rule. This produces a (possibly empty) new set of non-conforming triangles.

2. Mark all non-conforming triangles for refinement; the set is again denoted with S.

3. If S is not empty, then go to 1. Otherwise, there are no marked triangles and the algorithm stops. The new triangulation is T_{k+1}.

When the algorithm stops the new triangulation is conforming. As shown in [2] the algorithm stops after a finite number of steps since in one pass it inserts at most a single vertex on each edge. This is a fairly rough upper estimate for theoretical purposes – and one can construct such badly behaved examples – but, in practice, the subdivision has only local influence on the triangulation, see [2], [14]. The sequence $\{T_k\}$ is stable, i.e. all triangle angles are bounded from below by half the minimum triangle edges of the first triangulation T_1.

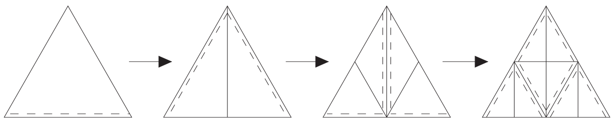

Figure 2: Successive applications of the Rivara algorithm introduce new vertices matching those retrieved by 4-1 splits.

The method using 4-1 splits, where each parent triangle is decomposed into four similar children, leads to non-conforming vertices if applied locally. Bank and Sherman [1] introduced so-called *green triangles* which join a non-conforming vertex with the opposite vertex of the non-conforming triangle, but this approach leads to non-nested triangulations over the green triangles. Rheinbold and Mesztenyi [13] work with non-conforming grids and, in order to maintain the continuity of the surface over the non-conforming points, they impose the condition that the geometric vertex over each non-conforming point is equal to the values interpolated from the nearby conforming points. However, the appearance of the non-conforming vertices complicates further geometrical or numerical computations because of the additional constraints.

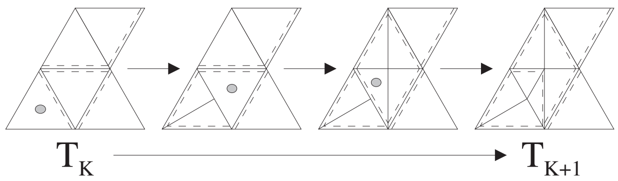

Figure 3: A single step of the Rivara bisection method with refinement edges and marked triangles.

The Rivara method (A) is a formalized version of the rule "bisect a triangle at its longest edge," where the method has its origin. We will later use the algorithm to ensure the interpolation property between hierarchies. Further, this algorithm bounds all triangle angles away from zero and therefore it guarantees numerical stability. We also note the following close connection: after applying the Rivara bisection method twice the same vertices as in a 4-1 split of triangles have been introduced (compare Figure 2). Therefore, the Rivara method does not differ too much from the well-used 4-1 rule, but merely eliminates the case distinctions occurring in connection with the conformal closure.

2.1 Triangle Hierarchy

A *triangle hierarchy* is a hierarchical structure of triangular elements where each element has a reference to one *parent* element, to one *child*, and to a *sibling*. The sibling is a child of the same parent and the children shall be produced by subdivision of the parent. Elements with no parents are called *root elements*, and elements with no children are *leaf elements*. We assume that the geometric vertices of a hierarchy are given in a global vertex array, and each triangle is determined by three vertex indices. Vertices and elements usually have color and material properties, or carry texture coordinates. Elements may have references to neighbour elements.

Similar to the situation with meshes, it is essential to maintain the distinction between the topological, i.e. combinatorial, structure of the hierarchy and the geometric shape. For example, when bisecting a parent triangle, from the topological viewpoint we identify the two children with subsets of the parent triangle. But in the geometric view, the additional geometric vertex introduced during bisection (possibly as the midpoint of an edge) may deviate from its original position on the parent edge after a further numerical process.

The main value of this distinction for interpolation different hierarchies is the existence of a unique relationship between each additional geometric vertex and a topological point in the parent triangle (as given explicitly by the level maps below).

For our interpolation property of a sequence of key hierarchies, it is essential that each hierarchy is generated using the Rivara bisection algorithm (A). In the implementation, we use a fixed numbering of the three vertex indices of each of the two child triangles (consider figure 4). If the parent Δ_p is determined by three indices $\{i, j, k\}$ which refer to the vertices $v[i]$, $v[j]$, and $v[k]$ in the global vertex array v, then we assume the two child triangles to reference vertices $\{l, k, i\}$ and $\{l, i, j\}$ in this specific order, where $v[l]$ is the new vertex inserted during bisection. The specific ordering of the vertices in the children simplifies the location of each child triangle within its parent in the topological mesh. Additionally, the refinement edge of the children is always opposite to the first vertex and, therefore, the information is implicitly given by the vertex ordering. The refinement rule is then reformulated to 'refine a leaf triangle at the edge opposite to its first vertex'. Further, the second and third vertex refer to parent vertices and are therefore implicitly given and do not need to be explicitly referenced. Both observations save memory, but the latter requires recursive calls to obtain the two vertex indices from one of its ancestral triangles. Only the root elements must have references to three vertices in the global vertex array.

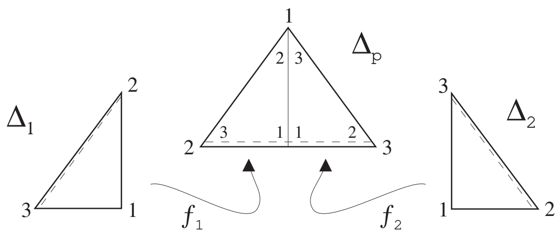

Figure 4: The refinement edge used in the Rivara bisection method can be implicitly stored to be opposite to vertex 1.

For smooth level-of-detail interpolation, it is important to observe that the new geometric vertex on a refined edge can be uniquely associated with the topological midpoint of the original edge in the simplicial complex. Let Δ_p be a parent triangle with two children, Δ_1 and Δ_2, generated by Rivara bisection. Let us work with barycentric coordinates and represent each point p of a triangle by its barycentric coordinate (b_1, b_2, b_3) with respect to the triangle vertices $\{v_1, v_2, v_3\}$. Explicitly, if $p = b_1v_1 + b_2v_2 + b_3v_3$, we use the notation $p_b = (b_1, b_2, b_3)$ for its barycentric representation. Then we can describe the parent-child relationship by two linear maps $f_i : \Delta_i \to \Delta_p$ given by

$$f_1(b_1, b_2, b_3) = \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & 1 \\ \frac{1}{2} & 0 & 0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{1}$$

and

$$f_2(b_1, b_2, b_3) = \begin{pmatrix} 0 & 0 & 1 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

which relate every point in a child triangle to a point in its parent. This relationship is only on the combinatorial level of the hierarchy connectivity, and it does not mean that, for a geometric point $p \in \Delta_i$, we have $f_i(p) = p$.

We observe that we can now relate each triangle Δ_n of the hierarchy on level n with a subset of one of the root elements Δ_1 by a recursive application of level functions f, namely $f^{(n-1)}(\Delta_n) := f \circ ... \circ f(\Delta_n) \subset \Delta_1$. The computational cost is reduced when using barycentric coordinates - since only division by two and addition are used in each composition. When the composite level function $f^{(i)}(\Delta_n)$ has been computed, it is applied to the geometric (x, y, z) coordinate representation of points. The level maps are also used in the computation of the local texture coordinate of a triangle, which are implicitly given via the texture coordinates of the root triangles.

3 Interpolating Different Hierarchies

Animation in computer graphics can be classified by image-based, see Stekettee and Badler [15], and geometry-based methods. Geometry-based methods split further into keyframe animations and into functional (resp. algorithmic) animations. For an overview we refer to the book [11] and its detailed bibliography.

We concentrate on the problem of interpolation between keyframe geometries and propose (in section 3.2) some constraints on the geometries. The constraints guarantee a smooth interpolation without the need to remesh during the interpolation process, and then further ensure the freedom for local grid modifications separately on each keyframe. This means that the refinement and coarsening process can be applied to a single keyframe without disturbing the interpolation property.

Finally, our constraints efficiently allow higher order spline interpolation and interpolation in a multi-parameter family of geometries shown in section 4.1 and 4.2.

3.1 Review of Keyframe Interpolation

The keyframe technique is a common and old technique in animation. The animator specifies, say n, key geometries G_i at certain time steps t_i, $i \in \{1, ..., n\}$. If there exists an interpolation method between each two successive pairs G_i and G_{i+1}, then a smooth animation is obtained by generating the geometry at time $t \in [t_i, t_{i+1}]$ on the fly by interpolating between key geometries G_i and G_{i+1}.

Any geometry mesh can be used in a keyframe animation if there exists an interpolation method, see Burtnyk and Wein [4], [5] for general shape interpolation techniques. In the simplest case, all key geometries have the same combinatorial mesh and differ only in their vertices. In this case the interpolation object at time t uses the same topological mesh and has vertices $v_j(t) \in \mathbb{R}^3$, $j \in \{1, ..., m\}$. They are given by linear combinations

$$v_j(t) = (1-t)v_j^i + tv_j^{i+1} \tag{2}$$

where v_j^i is the j-th vertex of G_i. Here, one can include more key geometries in the interpolation scheme and use polynomial interpolation of higher order in t.

In many applications the animated geometry varies heavily in time, and one would like to make local adaptions of the mesh on each keyframe geometry based on some local error criteria. But this spoils the simple interpolation technique above.

In [12], Polthier and Rumpf require at each time step t_i two topological meshes whose geometric realizations are of the same geometric shape. One connectivity is used to interpolate with the previous keyframe, and the other connectivity for interpolation with the next keyframe. In effect, they associate one connectivity per time interval $[t_i, t_{i+1}]$, and require two geometric realizations at each time step. Besides the additional storage requirement of two geometric realizations, this approach does not allow further modifications of keyframes since both geometric realizations must be modified in the identical way.

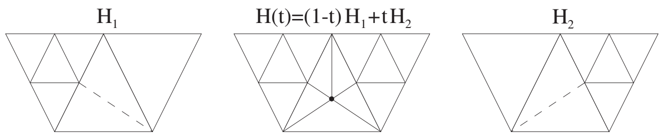

Figure 5: A simple example where the green elements in a 4-1 split require further subdivision if one interpolates between two key hierarchies H_1 and H_2.

3.2 Interpolation Constraints for Hierarchies

We now specify constraints on the key hierarchies that, firstly, guarantee a smooth interpolation without the need to remesh during the interpolation process, and, secondly, ensure the freedom for local grid modifications separately on each keyframe.

A family of triangle hierarchies F must fulfill the

Interpolation Constraints (B)

1. The simplicial complex of the root triangles of each hierarchy is the same for all hierarchies in F, i.e. for each pair of hierarchies $G, H \in F$ there exists a bijective simplicial map ϕ_{GH} between the set of root triangles.

2. Each root triangle has a refinement edge, and the simplicial map ϕ_{GH} maps each refinement edge to a refinement edge, i.e. the root triangles of all hierarchies are marked in the same way.

3. Each hierarchy is refined using the Rivara Algorithm (A).

The root triangles can be interpreted as charts of each hierarchy, and condition 1. requires a bijective correspondence between the charts of different hierarchies. Conditions 2. and 3. ensure that hierarchies are automatically refined in a synchronized way without further restricting the refinement process in each hierarchy. Each hierarchy can be refined according to its own error criteria without *a posteriori* synchronization with the other key hierarchies. Once one has agreed to use the Rivara bisection method, it only remains to ensure properties 1. and 2. for the family F in an initial synchronization step.

The Rivara bisection algorithm depends only on the initial choice of the refinement edges in the root elements. The subsequent position of the refinement edge in each child and further descendants is predetermined by the algorithm. Therefore we have

Theorem 1 *If two hierarchies G and H fulfill the interpolation constraints (B) then both of their topological simplicial complexes are a subcomplex of the same infinite complex obtained by infinitely refining the simplicial complex of the root triangles, see figure 6.*

Of course, the geometric representations of G and H are usually not identical since their geometric vertices differ.

In scientific computing a close connection between numerical computations and visualization is desirable. Since the Rivara method is a suitable tool in both fields, hierarchies generated with the Rivara bisection method allow smooth transition of data between numerical and visualization methods.

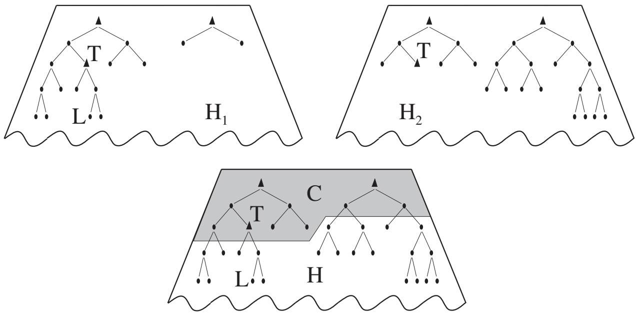

Figure 6: Two hierarchies fulfilling the interpolation constraints (B) are subsets of the same infinite hierarchy. The interpolation hierarchy H is the union of H_1 and H_2.

3.3 Interpolating between Hierarchies

We prove the interpolation property between hierarchies fulfilling the interpolation constraints (B) in a general form which, includes standard keyframe animation with a time parameter t described in section 3.1. Additionally, it covers higher order spline interpolation and interpolation in a multi-parameter family of hierarchies which we will apply in section 4.1 and 4.2.

Theorem 2 *Let $F = \{H_1, H_2, ..\}$ be a family of hierarchies which fulfill the interpolation constraints (B) and let $b = \{b_1, b_2, ..\}$, $b_i \in \mathbb{R}$, be a set of weights. Then there exists an interpolated hierarchy*

$$H(b) = \sum_i b_i H_i \tag{3}$$

which depends smoothly on b, and its underlying simplicial hierarchy is the union of the simplicial hierarchies of each H_i. Further, the interpolated hierarchy H depends smoothly on b and fulfills the same interpolation constraints as the elements of F.

Proof: For the proof we restrict ourselves to two hierarchies H_1 and H_2 and show how to interpolate between both. First we recall the existence of a bijective simplicial map ϕ between the two simplicial complexes formed by the root elements. Since ϕ extends to a bijective map between the refinement edges of the root triangles, the Rivara algorithm ensures that the different simplicial hierarchical complexes of H_1 and H_2 are subcomplexes of a theoretically infinite hierarchy which is obtained by infinitely refining the simplicial complex of the root triangles, see theorem 1 and figure 6.

It follows that interpolation between the common hierarchical subcomplex C of H_1 and H_2 can be done by simply interpolating corresponding geometric vertices.

Now assume H_1 is locally more refined than H_2. Then there exists a situation, as shown in figure 7, where a topological leaf triangle T of the common subhierarchy C is a leaf triangle of the complex of H_2 but not a leaf triangle of H_1 (since H_1 is more refined). Of course, T has different geometric realizations in H_1 and H_2.

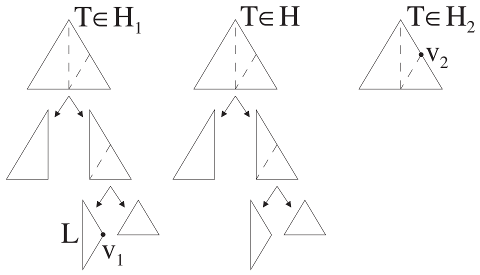

Figure 7: The central step when interpolating between two hierarchies.

All children and further descendants of T in H_1 are associated with a unique position in T via the level maps defined in equation 1. The topological subtree generated by T in H_1 can be projected via the level maps onto T and then associated with geometric positions of T in H_2. Here we make essential use of the distinction between topological and geometric realizations. The level maps operate on the topological realization and give for each geometric vertex v_1 in a leaf triangle L in H_1 the topological, i.e. barycentric, position b_2 in T. From the barycentric coordinates b_2, with respect to the vertices of T in H_2, one can immediately compute the geometric position v_2.$\square$

For the practical interpolation between a geometric leaf triangle L of H_1 and the corresponding subset of the geometric realization of T in H_2, we need to compute the barycentric coordinates of each vertex of L with respect to T. Let v_1 be one vertex of L with barycentric coordinates b_1 in L. Then we compose a level map f for the transition of L to T and use f to compute the barycentric coordinates b_2 of v_1 with respect to T:

$$b_2 = f(b_1).$$

By weighting the three vertices of the geometric realization of the triangle T in H_2 with b_2, we obtain the geometric position of v_2. Now we can interpolate between $v_1 \in H_1$ and $v_2 \in H_2$

$$v(t) = (1-t)\,v_1 + tv_2.$$

Applying the same procedure to the other two vertices of L gives the interpolation for L.

It is remarkable that the interpolation hierarchy is not more refined than the union of its keyframe hierarchies.

The interpolation hierarchy fulfills the same interpolation constraints as the keyframes. This is relevant in the following section for higher order interpolation and for further numerical or graphical processing, e.g. the interpolation hierarchy can immediately be used as a new keyframe. Additionally, each subhierarchy of a hierarchy in F fulfills the interpolation constraints (B) as soon as its leaf triangles form a conforming triangulation. This allows for smooth level-of-detail transitions within a single hierarchy and, by theorem 2, between different levels-of-detail of different hierarchies of F.

4 Applications

4.1 Higher Order Spline Interpolation

The interpolation property of theorem 2 immediately allows higher order interpolation in a keyframe animation. Let $H_i,..,H_{i+n}$ be $n+1$ successive keyframe hierarchies fulfilling the interpolation constraints (B), then

$$H(t) = \sum_{i=0}^{n} B_{i,n}(t) H_i \tag{4}$$

is a polynomial hierarchy interpolant of degree n where $B_{i,n}(t)$ are the Bernstein polynomials.

4.2 Multi-Parameter Families of Hierarchies

The idea of smooth interpolation in a set of hierarchies can be pursued to the interpolation in a two- or multi-parameter family of hierarchies (as shown in figures 8 and 9 in the appendix). The vertices of the discretized parameter domain, a square with four triangles and five vertices, represent five keyframe hierarchies at different resolutions and are shown in the lower part: an icosahedron, a bone, a cushion, a star, and a sphere in the midpoint of the domain. Each vertex of the domain represents a key hierarchy, and the set of keyframe hierarchies fulfills the interpolation property (B). This allows a barycentric interpolation in each domain triangle between the three key hierarchies at the vertices of each triangle. Let H_1, H_2, and H_3 denote the key hierarchies at the vertices of a domain triangle, and let $b = (b_1, b_2, b_3)$ be the barycentric coordinate of the point in the domain triangle. Then the interpolation hierarchy is given by

$$H(b) = b_1 H_1 + b_2 H_2 + b_3 H_3. \tag{5}$$

Similar to the 1-dimensional case, with time parameter t, the combinatorial structure of the interpolation hierarchy $H(b)$ is the topological union of all its three key hierarchies.

Multiparameter families of hierarchies occur frequently in geometrical and numerical problems depending on more than one parameter. But even when studying one-parameter families, the inclusion of a view-dependent rendering may be considered to be a 2-parameter family.

5 Hierarchy Generation

In practice, hierarchies are generated by two different approaches: One starts with a high resolution data set and iteratively coarsens the geometry to produce a hierarchy. This 'bottom up' approach is successfully used by Hoppe [8], [9] in the progressive mesh concept to obtain a vertex-based hierarchy by successive vertex-split and edge-collapse steps. As shown by Hoppe, different sections of one hierarchy can be interpolated, so-called geomorphs. The handling of arbitrary initial data sets is flexible, but this restricts the compatibility of two hierarchies obtained from similar initial geometries and reduces the possibility of interpolating between different hierarchies.

Generating a hierarchy 'top down' from a given coarse triangulation is ideal for element based approaches. One sets the triangles of the initial triangulation as root elements of a hierarchy, and successively refines according to some error criteria. For example, in the numerics of a boundary value problem for a partial differential equation one starts with a rough approximation of the solution and then refines/coarsens the geometry depending on a local numerical error.

Eck et al. [7] produce, from a given fine resolution mesh, a new element-based hierarchy for usage in multiresolution analysis. Their approach should also apply in generating hierarchies based on the Rivara bisection method since they already solved the major task of distributing vertices equidistantly on the surface.

6 Summary and Future Work

We have imposed some weak interpolation constraints (B) on a family of triangle hierarchies to allow interpolation while maintaining the freedom to locally refine each key hierarchy. The interpolation hierarchy, is combinatorially the union of its keyframe hierarchies and it therefore has the simplest structure possible without loosing information.

We introduced the Rivara bisection method as an alternative to the 4-1 split of triangles. Together with our constraints (B), the Rivara method ensures the interpolation property. Additionally, the Rivara method avoids a number of case distinctions occurring with hanging vertices in the 4-1 split approach. On the other hand, the Rivara method is similar to the 4-1 split since it inserts the same vertices after successive refinements.

For numerical purposes the interpretation of the parameter domain of a, say, two-parameter family as a triangulation has significant further implications. Similar to the approximation of a smooth surface by a triangulation, the triangulated domain may approximate a smooth family of surfaces. One may use an adaptive refinement of the parameter domain, i.e. an automatic process which inserts a new key hierarchy in the domain, if the interpolation hierarchy does not satisfy a given error threshold.

7 Acknowledgments

The Bunny geometry was produced by the Stanford Computer Graphics group. The second author thanks Martin Rumpf for fruitful discussions.

References

[1] R. E. Bank and A. H. Sherman. The use of adaptive grid refinement for badly behaved elliptic partial differential equations. *Advances in Computer Methods for Partial Differential Equations*, 3, 1979.

[2] E. Bänsch. Local mesh refinement in 2 and 3 dimensions. *IMPACT of Computing in Science and Engineering*, 3, 1991.

[3] E. D. Bloch. *A First Course in Geometric Topology and Differential Geometry.* Birkhäuser, 1997.

[4] N. Burtnyk and M. Wein. Computer generated key-frame animation. *J. Soc. Motion Picture and Television Engineers*, 80, 1971.

[5] N. Burtnyk and M. Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Communications ACM*, 19(10), 1976.

[6] L. DeFloriani and E. Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4), 1995.

[7] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, 1995.

[8] H. Hoppe. Progressive meshes. *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, 1996.

[9] H. Hoppe. View-dependent refinement of progressive meshes. *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, 1997.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, 1993.

[11] N. Magnenat-Thalmann and D. Thalmann. Computer Animation. *Computer Science Workbench*, Springer Verlag, second edition, 1985.

[12] K. Polthier and M. Rumpf. A concept for time-dependent processes. In M. Göbel, H. Müller, and B. Urban, editors, *Visualization in Scientific Computing*, 1995.

[13] W. C. Rheinboldt and C. K. Mesztenyi. On a data structure for adaptive finite element mesh refinements. *ACM Transactions on Mathematical Software*, 6, 1980.

[14] M. Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International J. for Numerical Methods in Engineering*, 20, 1984.

[15] S. N. Steketee and N. I. Badler. Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control. *Computer Graphics (SIGGRAPH 85 Conference Proceedings)*, 1985.

Appendix: Space Requirements

Model	# root nodes	# leaf nodes	# vertices	Mem. (MB)	Mem. (MB) opt.	size of non-hier. rep.
bunny1	400	127000	63520	17	13	5
bunny2	400	611340	305690	80	60	24
sphere	20	204800	102420	27	20	8
brezel	144	128280	64120	17	13	5

This table shows the memory requirements for some models using the principal hierarchical structures and a slightly optimized implementation in contrast to a non-hierarchical representation of the triangulation on the finest level. The non-hierarchical representation stores for each vertex the coordinates, the normals, and for each element the vertex indices, neighbour indices and the vertex texture coordinates. For the optimized hierarchical representation we used a data-structure which allowed us to store vertex information only once per refined root node.

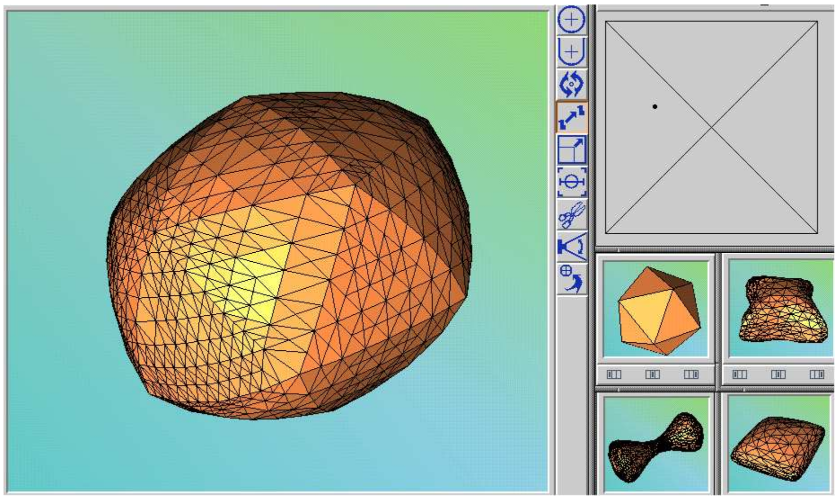

Figure 8: Interpolation between a two-dimensional set of keyframe hierarchies. The vertices of the parameter domain represent the four hierarchies on the bottom right and a sphere at the midpoint. All hierarchies have a different combinatorics, but fulfill the interpolation constraints mentioned in the paper. This allows a continuous interpolation when varying the position in the parameter domain.

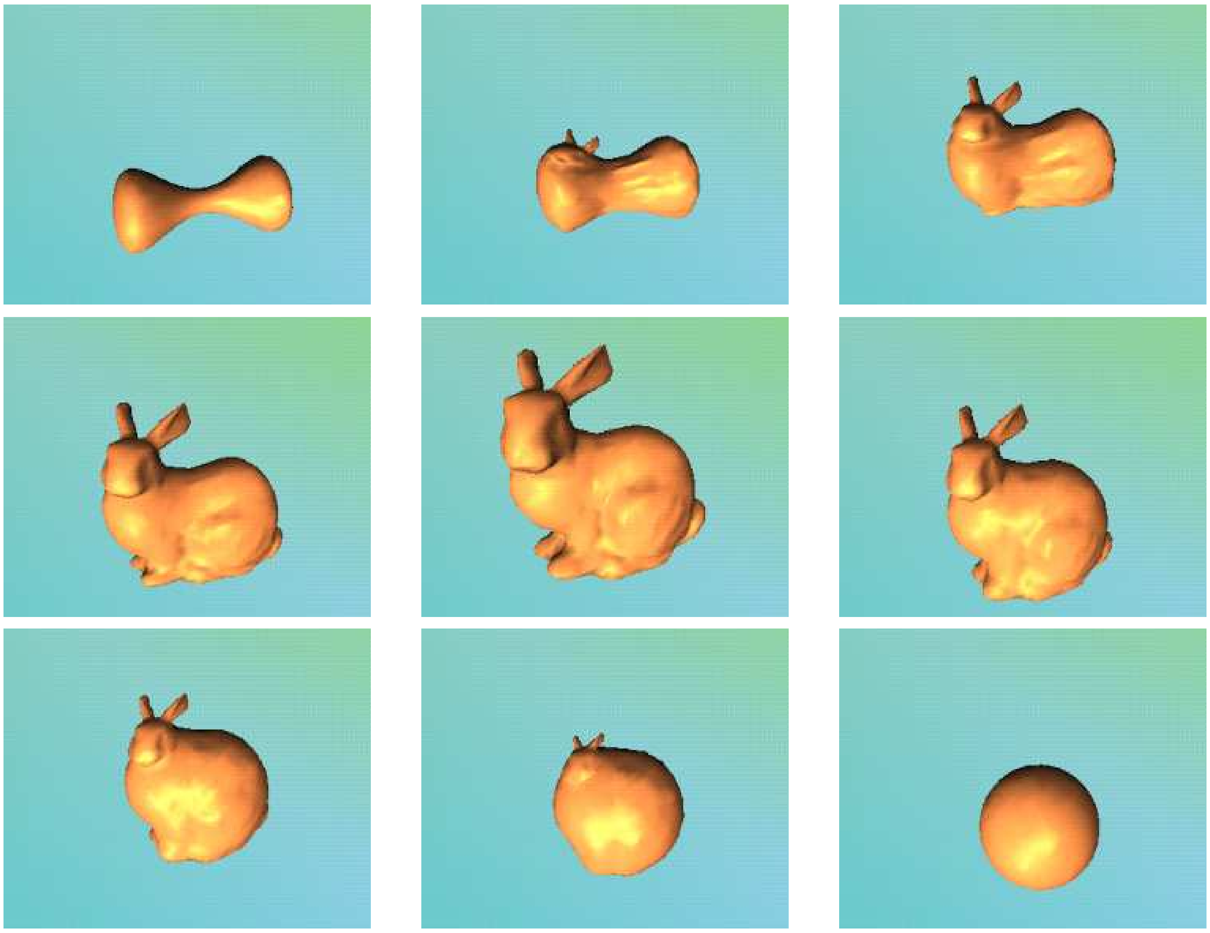

Figure 9: Interpolation between three keyframe hierarchies (a bone, the Stanford Bunny, and a sphere). The three models have different connectivities adjusted to their curvature.