INTERNATIONAL JOURNAL OF COMPUTERS
Issue 2, Volume 6, 2012

Multiresolution surface representation using combinatorial maps

M-H MOUSA M-K HUSSEIN

Abstract–Multiresolution surfaces are traditionally represented by data structures based on quadtrees which are derived directly from the subdivision operations. However, these data structures suffer from a number of drawbacks. First of all, they are restricted to triangular or quadrilateral grids and must be developed specifically for each mesh type separately. Moreover, the time complexity of adjacency query is not optimal. In this paper, we present a data structure for representing multiresolution meshes. This data structure extends the n dimensional generalized maps. Given a 3D mesh, the proposed data structure is defined as a hierarchy of nested 2 dimensional maps, i.e. each level of details of the mesh is defined as a separate 2 dimensional map. This inherits the generality and effectiveness of the original n map data structure. Additionally, we define, based on the multiresolution 2-map, the dual multiresolution representation which allows the topological description of the traditional mesh operations. The multiresolution 2-map enables the representation of arbitrary meshes and support the adaptivity and efficiency of the adjacency operators. We apply the proposed framework to the representation of progressive meshes.

Keywords– Hypermaps, 2-Maps, Progressive meshes.

I. INTRODUCTION

MULTIRESOLUTION representations of geometric objects have gained much of interest in many computer graphics areas, especially geometric modeling. The multiresolution representation has various applications such as mesh compression, progressive transmission, adaptive visualization and multiresolution mesh editing. In boundary representations, a surface is described by a polygonal mesh, i.e. a finite subdivision composed of vertices, edges and faces. In multiresolution representations, the surface is no longer defined by a single mesh, but by a sequence of nested meshes along with a number of rules for passing from one to another of these meshes. The sequence of meshes represents the different levels of details of the given object. The multiresolution representation is called adaptive if the highest resolution is not the same over all areas of the surface of the object.

Many techniques have been proposed for constructing multiresolution surfaces [20],[19], such as inserting sharp edges[3], applying boolean surface operations[2] or cutting-and-pasting surface elements[4]. In general, these techniques are focused on the obtained surface quality regardless of the structure of the underlying data. This leads to the use of non optimal structures for representing and manipulating 3D objects. The commonly used data structure is the quadtrees which is naturally generated from the faces hierarchy generated by the subdivision algorithms[9].

On the other hand, topological operations, which generates multiresolution surfaces, do not work uniformly on all types of meshes; some are applied to triangular meshes and others to polygonal meshes. The data structures based on quadtrees are limited to triangular and quadrilateral meshes and should be implemented separately for each type. Moreover, the adaptive subdivision generates holes in the mesh topology. In addition, the adjacency queries are executed in $O(log(n))$. Some solutions have been proposed to overcome the last issue [16],[13]. However, they have other limitations, such as the lack of adaptive representation.

To this end and many others, the main objective of this work is to define a unified and effective representation of multiresolution meshes while maintaining the topological consistency of the mesh in the adaptive case. In addition, we improve the effectiveness of the adjacency queries to be executed in a constant time. The combinatorial maps[8],[18],[17] are the start point for such representation. We propose a multiresolution 2-map as a hierarchy of nested combinatorial 2-maps. Each resolution level is described by a separate combinatorial 2-map. Indeed, the latter can represent arbitrary meshes, perform adjacency operators in constant time. Therefore, The proposed model inherits the generality and the effectiveness of the composing maps.

This paper is organized as follows. Section II. gives an overview of the basic definition of the combinatorial 2-maps as well as hypermaps. Section III. introduces how to extend the combinatorial 2-maps to multiresolution 2-maps and their multiresolution duals. An application of the proposed data structure to the progressive meshes is given in Section IV.. A simple C++ class interfaces as well as an analysis of the time and storage complexity of the proposed data structure is given in Section V.. Finally, we conclude in Section VI..

II. 2-MAPS

The 2-maps, the combinatorial maps of dimension 2, and their extensions[14],[15] are the base of many research studies[1],[7],[5],[12]. These models are defined in a more general form based on the concept of hypermaps[6]. This section presents a brief description for Hypermaps, 2-Maps and The dual 2-map.

Manuscript received January 25, 2012.

M-H MOUSA is with the Faculty of Computers & Informatics, Ismailia, 41522 Egypt. (phone: 201110673699; e-mail: mohamed_mousa@ ci.suez.edu.eg).

M-K HUSSEIN is with the Faculty of Computers & Informatics, Ismailia, 41522 Egypt. (e-mail: m_khamiss@ ci.suez.edu.eg).

A. *Hypermaps*

The hypermaps provide a general framework to define regular topological models. They can describe the topology of different types of geometric objects such as volumes and surfaces; either open, closed, oriented or non-oriented. This topology is described by subdividing the objects into elements of different dimensions (e.g. vertices, edges, faces, etc.) sharing incidence relations. We focus here on the representation of surface meshes.

A hypermap consists of a set of darts, connected together by relationships. Formally, given a finite set of darts β and permutations α_0 and α_1 over the set β, a hypermap is the triple:

$$H = (\beta, \alpha_0, \alpha_1) \tag{1}$$

The multiresolution elements are not explicitly represented in the model, but are implicitly defined by subsets of β. These subsets are achieved through the orbit notation. Given a permutation σ over β, the orbit of $x \in \beta$ is a subset of β denoted by $\langle\sigma\rangle(x)$ and is defined as:

$$\langle\sigma\rangle(x) = \{x, \sigma(x), \sigma^2(x), \ldots, \sigma^k(x)\}, \tag{2}$$

where k is the minimum positive integer such that $\sigma^{k+1}(x) = x$. It is clear that all the elements of $\langle\sigma\rangle(x)$ have the same orbit. Practically, starting by a dart $x \in \beta$, the orbit $\langle\sigma\rangle(x)$ is the set of reachable darts from x by a successive application of σ.

Such a model describes only the topology of the subdivision of the objects. To complete the object representation, we should therefore define a model embedding the geometric data to each cell of the mesh. The 0-embedding is the simplest geometric embedding, i.e. only the cells of dimension 0 are inserted. For example, we associate a 3D point to each vertex of the mesh. The embedding of other cells is obtained by linear interpolation of the embedded vertices. Another sophisticated embedding models can be used, e.g. associating curves to edges and surface patches to faces.

B. *2-Maps*

The 2-map is a hypermap with the following condition, α_0 is an involution, i.e. a permutation such that $\forall x \in \beta$: $\alpha_0(\alpha_0(x)) = x$ and $\alpha_0(x) \neq x$. This model enables the representation of the meshes of closed orientable 2-manifolds, i.e. meshes that enclosing volumes. Figure 1 shows a graphical representation

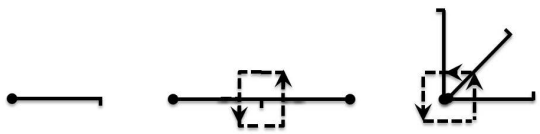


Fig. 1: From left to right: a graphical representation of a dart, involution α_0 and the permutation α_1 respectively.

of a dart and its relations according to the permutations α_0 and α_1. The vertices are defined by the orbit $\langle\alpha_1\rangle$, the edges by the orbit $\langle\alpha_0\rangle$ and the faces the orbit $\langle\alpha_0 \circ \alpha_1\rangle$.

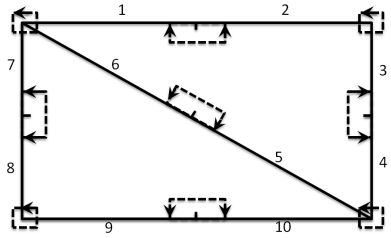


Fig. 2: An example of a 2-map consists of four vertices, five edges and three faces.

Figure 2 presents an example of a 2-map. This map consists of four vertices: $\{2,3\}$, $\{4,5,10\}$, $\{8,9\}$, $\{1,7,6\}$; five edges: $\{1,2\}$, $\{3,4\}$, $\{9,10\}$, $\{7,8\}$, $\{5,6\}$; and three faces: $\{1,3,5\}$, $\{6,10,8\}$, $\{7,9,4,2\}$.

C. *The dual 2-map*

Given a 2-map $M = (\beta, \alpha_0, \alpha_1)$, the triple $M' = (\beta, \varphi_1, \varphi_2)$, where $\varphi_1 = \alpha_0 \circ \alpha_1$ and $\varphi_2 = \alpha_0$, is also a 2-map and is called the dual map of M. The relation φ_1 is a permutation and φ_2 is an involution.

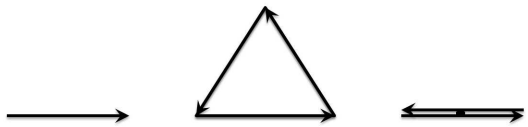


Fig. 3: From left to right: a dual 2-map dart, permutation φ_1 and involution φ_2 respectively.

Figure 3 shows the graphical representation of the dual 2-map. A dart is represented by an arrow. A sequence of darts linked by the orbit $\langle\varphi_1\rangle$ composes an oriented face. The relation φ_1 is used to link two oriented faces along the shared edge. The vertices are defined by the orbit $\langle\varphi_1 \circ \varphi_2\rangle$, the edges by the orbit $\langle\varphi_2\rangle$, and the faces by the orbit $\langle\varphi_1\rangle$. Figure 4 shows the dual

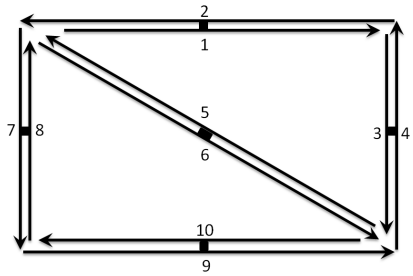


Fig. 4: The dual 2-map of the same example given in Figure 2.

of the 2-map presented in Figure 2.

III. THE MULTIRESOLUTION 2-MAP

We present here the multiresolution hypermaps as an extension of the traditional hypermaps presented earlier in Section II.. Next, we will present the multiresolution 2-maps and their dual.

A. *The multiresolution hypermaps*

The multiresolution hypermaps provide a framework allowing to define the topology of multiresolution objects. Each resolution level is presented as a hypermap. This forms a set of nested hypermaps and a set of darts describing the given level.

Formally, the multiresolution hypermap consists of a set of darts β and the relationships between these darts. The darts of β are indexed by the resolution level where they were firstly introduced. The darts describing the object at its coarsest level belong to the set β^0. Some darts are added to β^0, in order to describe the object in higher level (finer level), and form the set β^1. To describe each resolution level, we add some darts to the set of darts describing the coarser level.

Therefore, in a hypermap with a maximum resolution level k, the set of darts β is the sequence of subsets $\{\beta^i\}_{i\in[0,k]}$ such that :

$$\beta^0 \subset \beta^1 \subset \beta^2 \subset \cdots \subset \beta^k = \beta. \tag{3}$$

The subset β^i is a subset of β and contains the darts introduced at a level lower than or equal to i. The subset $\beta^i - \beta^{i-1}$ contains the darts introduced at the level i. The cardinality of β is the number of darts of the map describing the mesh at the finest level.

The topological relations between the darts are parameterized by the resolution level. For a permutation σ and a dart $x \in \beta^i$, $\sigma^i(x)$ is the dart linked to x by the relation σ at the resolution level i. Therefore, the multiresolution hypermap is the triplet:

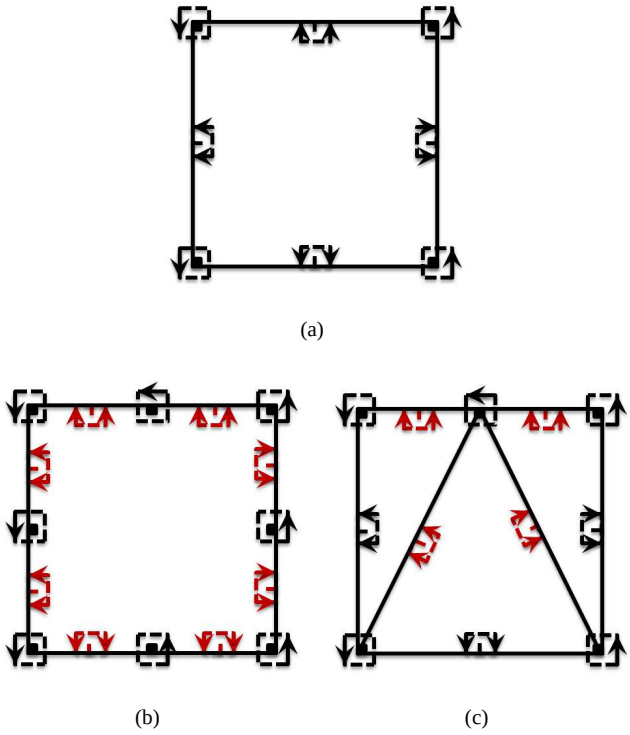
$$M = \left(\beta, \{\alpha_0^i\}_{i\in[0,k]}, \{\alpha_1^i\}_{i\in[0,k]}\right) \tag{4}$$

such that for all $i \in [0,k]$, the triplet $M^i = (\beta^i, \alpha_0^i, \alpha_1^i)$ is a hypermap describing the resolution level i.

B. *The multiresolution 2-map*

A multiresolution 2-map is a multiresolution hypermap with the following condition: α_0^i is an involution without fixed point, i.e., a permutation such that $\forall i \in [0,k]$, $\forall x \in \beta^i$, $\alpha_0^i(\alpha_0^i(x)) = x$ and $\alpha_0^i(x) \neq x$. This is the same condition that defines the 2-map. The multiresolution 2-map allows to represent the topology of the closed oriented 2-manifolds. Each resolution level i, $i \in [0,k]$, is represented by a 2-map $M^i = (\beta^i, \alpha_0^i, \alpha_1^i)$. The set of vertices, edges and faces at a level i is defined by the orbits $\langle\alpha_1^i\rangle$, $\langle\alpha_0^i\rangle$ and $\langle\alpha_0^i \circ \alpha_1^i\rangle$ respectively.

Figure 5(a) presents an example of a multiresolution 2-map at a level i. This 2-map consists of 4 vertices, 4 edges and 2 faces. Figures 5(b) and 5(c) show possible examples of the $i+1$ level. The red darts, in the two cases, are the darts introduced between the level i and the level $i+1$. These darts belongs to the subset $\beta^{i+1} - \beta^i$. According to the insertion of the darts, the relations linking the darts at level i are not necessarily different from the linking relations at level $i+1$. In Figure 5(b), the edges of the map at level i are cut. For the darts of β^i, the relations α_0^{i+1} are not equal to the relations α_0^i. However, the degree of the vertices are not changed, so for the darts of β^i the relations α_1^{i+1} are equal to the relations α_1^i. The 2-map at level $i+1$ consists of 8 vertices, 8 edges and 2 faces. In Figure 5(c), an edges at level i is cut and new edges are inserted between some



(a) (b) (c)

Fig. 5: A mutiresolution 2-map. (a) A 2-map consists of 4 vertices, 4 edges and 2 faces at level i. (b) and (c) possible examples of the $i+1$ level. The red darts represent the introduced darts in the two cases.

vertices at level i and the new vertex at level $i+1$. For some darts of β^i, the linking relations are different from the linking relations at level $i+1$. We have, here, 5 vertices, 7 edges and 4 faces.

C. *The dual multiresolution 2-map*

Given a multiresolution 2-map $M = \left(\beta, \{\alpha_0^i\}_{i\in[0,k]}, \{\alpha_1^i\}_{i\in[0,k]}\right)$, the dual of M is defined by the triplet:

$$M' = \left(\beta, \{\varphi_1^i\}_{i\in[0,k]}, \{\varphi_2^i\}_{i\in[0,k]}\right), \tag{5}$$

where $\varphi_1^i = \alpha_0^i \circ \alpha_1^i$ and $\varphi_2^i = \alpha_0^i$. M' is also a multiresolution 2-map since φ_1^i are permutations and φ_2 are involutions without fixed point. Each resolution level i, $i \in [0,k]$, is represented by the dual 2-map:

$$M^i = \left(\beta, \varphi_1^i, \varphi_2^i\right). \tag{6}$$

The vertices, edges and faces are defined by the orbits: $\langle\varphi_1^i \circ \varphi_2^i\rangle$, $\langle\varphi_2^i\rangle$ and $\langle\varphi_1^i\rangle$ respectively.

Figure 6 shows a two successive levels of a multiresolution 2-map dual. Figures 6(a), 6(b) and 6(c) are the dual of the multiresolution 2-maps presented in Figures 5(a), 5(b) and 5(c) respectively. Some darts, red arrows, are inserted between level i and level $i+1$. For certain darts of β^i, the linking relations at level $i+1$ are different from the corresponding relations at level i.

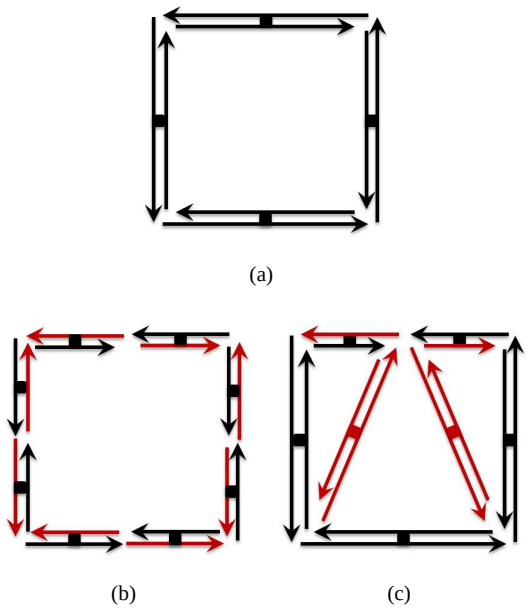


Fig. 6: The multiresolution 2-map duals that correspond to the multiresolution 2-maps presented in Figure 5.

IV. APPLICATION TO PROGRESSIVE MESHES

We will show here how to apply the presented multiresolution model to the construction of multiresolution meshes using the simplification technique based on the operators edge contraction and its inverse, vertex split, see Figure 7. The mesh simplification based on these operators have been popularly used since the introduction of the progressive meshes [10] which allow the access of different level of details of the mesh. These progressive meshes are used in many applications such as storage and progressive transmission of huge triangular meshes. These applications use the topological information of the mesh to access the neighborhood of the vertices. A model providing an optimal adjacency queries at all levels can provide a significant efficiency gain in the performance of these applications.

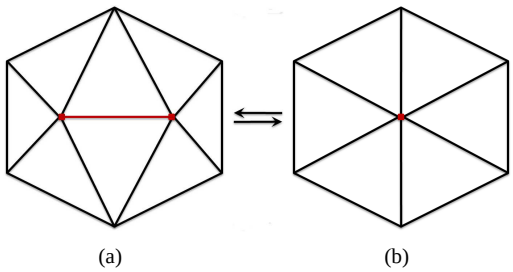


Fig. 7: A graphical representation of the edge contraction and vertex split operators. (a)→(b) is the application of the edge contraction operator and (b)→(a) is the application of the vertex split operator.

Figure 8 illustrates the multiresolution dual 2-map structure of the application of the edge contraction operation on a triangular mesh. The mesh, presented in Figure 8(a), depicts the dual 2-map of the mesh at level L_0. This dual 2-map contains

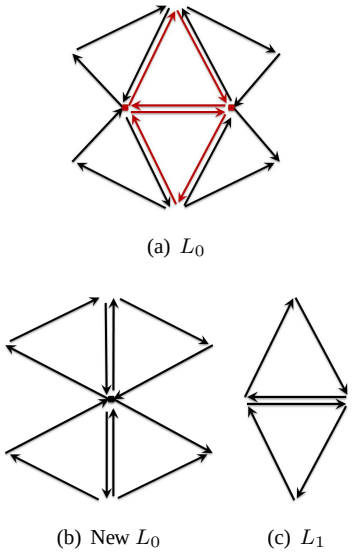


Fig. 8: The multiresolution dual map corresponding to the edge contraction operator. (a) The dual 2-map before the application of the operator. (b) and (c) are the newly created dual 2-maps after the application of the operator.

halfedges belonging to the set L_0. The edge contraction deletes two triangular faces from the mesh. The deleted halfedges, which correspond to the deleted faces, are removed from the set L_0 and added to the set of halfedges at the next finer level, L_1. The adjacent triangles are compacted together at level L_0 to form the mesh at the new simplified level L_0.

Any simplification process consists only of a single type of a sequence of these operations. This creates as many levels as the number of applied contractions. However, any set of independent topological contractions may also be performed at the same time while generating a coarser level containing about 20% of vertices.

Figure 9 shows an example of a prgressive mesh handled using the proposed multiresolution 2-map. Each resolution level corresponds to a union of a set of nested 2-maps. This set of 2-maps is constructed using the described edge contraction operators. In this example, the successive simplifications is applied while pushing the removed triangles as far as possible in the hierarchy. This results in a topological multiresolution structure having exactly the same properties as those of the subdivision surfaces. The different meshes corresponding to the intermediate levels can be navigated simply and more effective than of the conventional half-edges structures. The algorithms, which use the adjacency relations to calculate the geometric properties of the mesh, can be performed in an optimal time. In fact, the navigation along the geometric information can be performed without having to rebuild the intermediate topology of the meshes, since they are already available in the levels of the multiresolution dual 2-map structure.

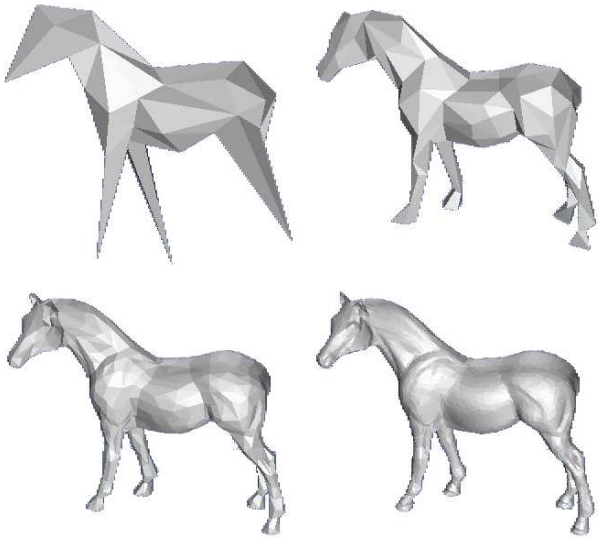


Fig. 9: A prgressive mesh example of the Horse model. Each resolution level corresponds to a union of a set of nested 2-maps.

V. IMPLEMENTATION AND DISCUSSION

In this section, we present an example of a data structure that implements the model of multiresolution 2-map that we have described later. We focus here on the primal representation, but the dual version is similarly defined. We then compare the time and storage complexities of this data structure with those of quad-trees which is conventionally used for representing the adaptive multiresolution subdivision surfaces.

A. *Data structure*

The multiresolution 2-map consists of a set of darts. The set of darts β is defined as a sequence $\{\beta^i\}_{i\in[0,k]}$ such that $\beta^0 \subset \beta^1 \subset \cdots \subset \beta^k = \beta$, see Section A.. Therefore, the darts can be stored in an array of lists. The i^{th} cell of the array contains the set of darts β^i/β^{i-1}, i.e. the darts inserted at the resolution level i. Each dart contains pointers to other darts realizing the topological relations. The link α_1 is represented directly by a pointer to another dart. The links α_0^i are stored as an array of pointers. Any dart has no topological relation at the resolution levels inferior to the proper insertion level of this dart. Let a is the insertion level of of the dart x ($x \in \beta^a/\beta^{a-1}$) and k is the maximum resolution level, the dart x does not store no more than $k-a$ relations α_0. Hence, the reference of the dart linked to x by the relation α_0^i is given by the pointer stored at the position $i-a$ in the array.

The geometric information is attached directly in the darts. We use here the 0-embedding, i.e. each dart contains a pointer to a 3D point. All darts of the same vertex are linked to the same 3D point. The embedding of the 3D point is parameterized by the resolution level since the vertex does not receive any embedding until their insertion level and this embedding can be changed from this resolution level to another. Therefore the embedding is stored in an array. Similar to the relation α_0, the darts store only a limited number of the necessary pointers to the 3D points. Let a is the insertion level of of the dart x and k is the maximum resolution level, the dart x stores only $k-a$ pointers to 3D points. Hence, the reference of the embedding of the vertex of the dart x in level i is given by the pointer at $(i-a)^{th}$ cell in the array. List 1 give an example of a C++ implementation of

Algorithm 1

```
class MultiresolutionMap{
  vector < list < Dart > > map;
};

class Dart{
  vector <Dart *> a0;
  Dart * a1;
  vector <Point3D *> em0;
};
```

the multiresolution 2-map and the dart.

As we mentioned above, the cells of the subdivision are represented implicitly at each resolution level i by the subsets β^i and constructed using the orbits. The navigation of an orbit can be simply implemented by iterators. The code for a face iterator can be as shown in Listing 2. This iterator is constructed by

Algorithm 2

```
class FaceIterator{
  Dart *first, *current;
  int level;
  FaceIterator (Dart *b, int n){
    first = current = b;
    level = n;
  }
  void next(){
    current =
current->alpha0(level)->alpha1();
  }
  bool end(){
    return current == first;
  }
}
```

two parameters, a given dart and the desired resolution level. It is obviously necessary that the given dart has already been inserted into the map at a lower level. Any two iterators, that are at the same level and initialized by two darts belonging to the same orbit, are considered to be equal. The iterators of the other cells of the subdivision are implemented in the same way.

Any cell iterator can be used in an object corresponding to this cell and have methods that act on that cell. Such an object can be implemented as shown in Listing 3. This type of object is created on-the-fly when processing an operation on that object.

B. *Time complexity*

In the framework of the subdivision surfaces, the adjacency query between the vertices are one of the most common op-

Algorithm 3

```
class Face{
  FaceIterator f;
  Face(Dart *b, int n):f(b,n){
  }
  void Display();
  Point3D Normal();
  void Subdivide();
}
```

erations. This operation is needed for the execution of both the synthesis and analysis operators.

In multiresolution hypermaps, the adjacency queries are executed in a constant time regardless of the considered resolution level. Indeed, the adjacent cells are retrieved directly by following a certain number of pointers. The multiresolution 2-maps inherits this property from the multiresolution hypermaps. For example, at the resolution level i, visiting all the vertices neighboring a vertex defined by a dart x can be performed as shown in Listing 4. We simply trace the orbit of the vertex of x using a

Algorithm 4

```
class VertexIterator {
  Dart *first, *current;
  int level;
  VertexIterator (Dart *b, int n) {
    first = current = b;
    level = n;
  }
  void next(){
    current = current->alpha1();
  }
  bool end(){
    return current == first;
  }
};

class Vertex {
  VertexIterator  it;
  Vertex (Dart *b, int n): it(b,n) {
  }
  void Visit_Neighboring_Vertices(){
    while(!it.end()){
      //processing the neighbor vertex using
      //Vertex(it.current->alpha0(it,n), it.n)
      it.next();
    }
  }
}
```

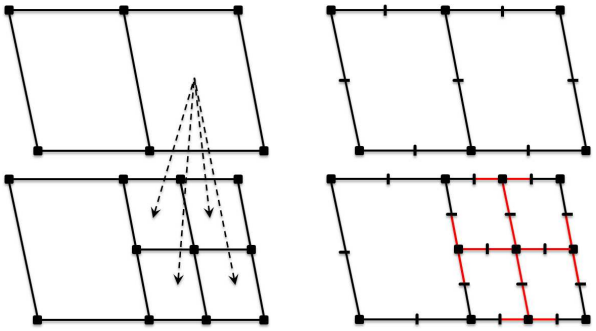
vertex iterator. For each dart inside this orbit, we reach a neighbor vertex by traversing the corresponding edge. This is done by following the involution α_0 at the corresponding level.

Similar procedures can be written, for example, to visit the adjacent faces to a certain face at a given resolution level i (e.g. in order to calculate the angles between the normal of the given face and those of the neighbor faces). This is done be traversing the orbit representing that face and for each dart in this orbit we reach a dart belonging to the corresponding neighbor face using the link α_0 at the given resolution level.

In quad-tree representations, neighborhood queries between two faces, sharing a commun edge, are performed by navigating the tree to reach a common parent of the two faces. These queries are executed in $O(\log(n))$, where n is the depth of the tree, i.e. the maximum resolution level. Although in practice $\log(n)$ does not reach a very high value, it is not a constant time. Additionally, these operations are commonly used for updating the object during mesh editing process, this improvement is of significant impact.

C. *Storage complexity*

In this subsection, we compare the memory cost of the multiresolution 2-map and the multiresolution triangular quad-trees. Since no general formulation is given for the adaptive subdivision and that the regular subdivision is the worst case for memory requirements, see Figure 10, we perform this comparison in the regular case.



(a) Quadtree (b) Multiresolution 2-map

Fig. 10: Regular Catmul-Clark subdivision using (a) quadtree and (b) multiresolution 2-map.

Let $|\beta|$ be the total number of darts a multiresolution 2-map. $|\beta|$ is equal to the number of necessary darts describing the object at the maximum resolution level. Let β_0 be the number of darts at the resolution level 0 and k be the maximum resolution level. Using the primal subdivision schemes, the number of darts is multiplied by 4 at each subdivision step, we get

$$\beta = \beta_0 \cdot 4^k. \tag{7}$$

To calculate the total cost of the topological information, we need to compute the number of pointers attached to each dart. Regarding the relation α_0, we have to add the size of the array of pointers contained in each dart. The size of this array is a function of the insertion level of the dart. Note that $3/4$ of darts are inserted at the maximum resolution level, and they have exactly a single link α_0. Moreover, $3/4$ of the rest of the darts, i.e. $3/16$, has two links α_0. More formally, for the level $i \in [1, k]$, there are $\beta \cdot \frac{3}{4^i}$ darts that each attached array has i elements. The darts describing the level 0 have $k+1$ elements in their arrays. The total number of elements contained in the array of links α_0 for all darts is given by:

$$\beta_0 \cdot (k+1) + \beta_0 \cdot 3 \cdot \sum_{i=1}^{k} i \cdot 4^{k-i}. \tag{8}$$

For the relation α_1, each dart has exactly a single link α_1. In other terms, there are $|\beta|$ or $\beta_0 \cdot 4^k$ stored pointers.

The geometric information is attached to the darts by an array of pointers to 3D points. Since the embedding of these points may be modified at each level, this embedding is parameterized by the resolution level. Therefore, the size of the array of these pointers has exactly the same size as that of α_0.

The total number of the pointers attached to the darts is therefore equal to:

$$\beta_0 \cdot 4^k + 2 \cdot \left(\beta_0 \cdot (k+1) + \beta_0 \cdot 3 \cdot \sum_{i=1}^{k} i \cdot 4^{k-i} \right). \quad (9)$$

The summation in Expression 9 can be identified by the following integer series: $\sum_{n \geq 0} n \cdot x^n = \frac{x}{(1-x)^2}$, where $|x| < 1$. Replacing n by i and x by $\frac{1}{4}$, we get (neglecting the terms $i > k$):

$$\sum_{i=1}^{k} i \cdot 4^{k-i} \approx 4^k \cdot \frac{\frac{1}{4}}{\left(1 - \frac{1}{4}\right)^2} \quad (10)$$

Expression 9 is then simplified to (neglecting the minor terms):

$$\frac{33}{9} \cdot \beta_0 \cdot 4^k \quad (11)$$

The quadtrees store 5 pointers per node for the topological information: 4 pointers to the children and one to the parent node. The roots of the quadtrees store seven pointers: 4 to their children and 3 for the adjacency information between these roots. For the geometrical information, 3 pointers to 3D points are stored in each node. This makes a total of 10 pointers per root, and 8 pointers for each of the other nodes. Let f_0 be the number of faces of the mesh at level 0 and k be the maximum resolution level. Since the number of faces is multiplied by 4 at each subdivision step, the total number of stored pointers is:

$$10 \cdot f_0 + 8 \cdot f_0 \sum_{i=1}^{k} 4^i \quad (12)$$

The summation term in Expression 12 can be identified by the following formula $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$. Therefore, we get

$$\sum_{i=1}^{k} 4^i = \frac{4^{k+1} - 1}{3} - 1. \quad (13)$$

Expression 12 is then simplified to (neglecting the minor terms):

$$\frac{32}{3} \cdot f_0 \cdot 4^k \quad (14)$$

Now, we can calculate the ratio between the memory cost of the multiresolution 2-maps, Expression 9, and the memory cost of the quadtrees, Expression 12. Given that for the triangular meshes, each triangle has 3 darts. Therefore, $f_0 = \frac{\beta_0}{3}$. The required ratio becomes

$$\frac{\frac{33}{9}\beta_0 4^k}{\frac{32}{9}\beta_0 4^k} = \frac{33}{32}. \quad (15)$$

We see that our data structure requires only about 3% more storage space than that of quad-trees. Taking into account the cost of storage of the 3D points, which is equal in both structures, this ratio is so weak.

Using an interpolating subdivision scheme makes this ratio an advantage point for the multiresolution 2-maps than for the quad-trees. Indeed, since the vertices are located on the surface boundary upon their insertion, the embedding information is no longer needed to be parametrized by the resolution level. The arrays of pointers to the 3D points contained in the darts are thus reduced to a single pointer. In this case, the ratio between the multiresolution 2-map and the quadtrees is $\frac{30}{32}$. Our model here requires about 6% less memory space.

VI. CONCLUSION

The multiresolution hypermaps defined in this paper provide a framework for the representation of multiresolution surface meshes. This framework is defined as an extension of the hypermaps which is a general notion based on combinatorial maps.

The multiresolution hypermap is used here to define the multiresolution 2-maps and their dual representation. We have applied these data structure here to the representation of the multiresolution subdivision surfaces. In this context, they provide a number of advantages over the usually used data structures based on quadtrees. The multiresolution maps allow, within the same model, the representation of surfaces generated by a large number of subdivision schemes. The ability to represent polygonal meshes is an additional advantage in the adaptive case, where the existence of different levels of resolution produces non-triangular or non-quadrilateral faces, which are not supported by conventional structures and then create topological holes. Adjacency queries are performed more efficiently, they are performed in constant time regardless of the resolution level. In addition, there no requirements to have a great memory space.

We have shown in this paper that the multiresolution 2-maps can be applied to the representation of multiresolution surfaces generated by topological operators such as edge contraction and vertex splitting [11]. This allows to answer adjacency queries at any resolution level without the need to reconstruct the mesh to the required resolution level.

REFERENCES

[1] Yves Bertrand and Jean-François Dufourd. Algebraic specification of a 3d-modeler based on hypermaps. *CVGIP: Graphical Models Image Processing*, 56:29–60, 1994.

[2] Henning Biermann, Daniel Kristjansson, and Denis Zorin. Approximate boolean operations on free-form solids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 185–194, 2001.

[3] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control.

In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 113–120, 2000.

[4] Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Transaction on Graphics*, 21:312–321, 2002.

[5] David Cazier and Jean-François Dufourd. A formal specification of geometric refinements. *The Visual Computer*, 15:279–301, 1999.

[6] Jean-François Dufourd. Formal specification of topological subdivisions using hypermaps. *Computer Aided Design*, 23:99–116, 1991.

[7] Jean-François Dufourd. Algebras and formal specifications in geometric modeling. *The Visual Computer*, 13:131–154, 1997.

[8] J. Edmonds. A combinatorial representation of polyhedral surfaces. *Notices of the American Mathematical Society*, 7, 1960.

[9] L.De Floriani, 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.

[10] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 99–108, 1996.

[11] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 105–114, 1998.

[12] Pierre Kraemer, David Cazier, and Dominique Bechmann. Multiresolution half-edges. In *Proceedings of Spring Conference on Computer Graphics*, SCCG '07, 2007.

[13] Michael Lee and Hanan Samet. Navigating through triangle meshes implemented as linear quadtrees. *ACM Trans. Graph.*, 19:79–121, 2000.

[14] Pascal Lienhardt. Subdivisions of n-dimensional spaces and n-dimensional generalized maps. In *Proceedings of the fifth annual symposium on Computational geometry*, SCG '89, pages 228–236, 1989.

[15] Pascal Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer Aided Design*, 23:59–82, 1991.

[16] Günther Schrack. Finding neighbors of equal size in linear quadtrees and octrees in constant time. *CVGIP: Image Understanding*, 55:221–230, 1991.

[17] W. Tutte. *Graph theory*. In Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1984.

[18] A. Vince. Combinatorial maps. *Journal of Combinatorial Theory, Series B*, pages 1–21, 1983.

[19] Denis Zorin. Modeling with multiresolution subdivision surfaces. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 30–50, 2006.

[20] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 259–268, 1997.