Atlas of Connectivity Maps

Ali Mahdavi Amiri[a], Faramarz Samavati[a]

[a] *University of Calgary*

Abstract

Semiregular models are now ubiquitous in computer graphics. These models are constructed by refining a model with an arbitrary initial connectivity. Due to the regularity enforced by the refinement, the vertices of semiregular models are mostly regular. To benefit from this regularity, it is desirable to have a data structure specifically designed for such models. We discuss how to design such a data structure, which we call the atlas of connectivity maps (ACM) for semiregular models. In an ACM, semiregular models are divided into regular patches. The connectivity between is captured at the coarsest resolution. In this paper, we discuss how to find these patches in a given semiregular model and how to set up the ACM. We also show some of the benefits of this data structure in applications such as the multiresolution framework. ACM can support a variety of different multiresolution frameworks including compact and smooth reverse subdivision methods. The efficiency of ACM is also compared with a standard implementation of half-edge.

1. Introduction

Semiregular models are very common in computer graphics [1]. These models are obtained by applying repetitive refinement on an arbitrary initial mesh or they may be constructed by a parametrization method (Figure 1). Applying refinement on a mesh produces a large large number of vertices. However, these vertices are mostly regular, with irregular vertices corresponding to extra-ordinary vertices of the initial unrefined model. This regular structure should be taken into consideration in order to efficiently capture the connectivity information of the model. Additionally, the geometry of the vertices (coming from sources such as subdivision schemes, projections, parametrization) should also be recorded.

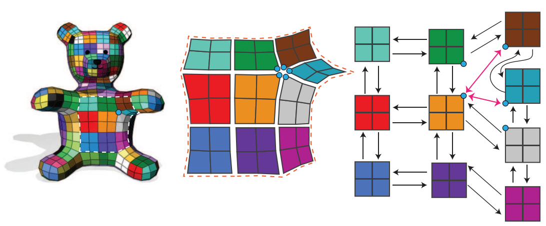

Figure 1: A semiregular mesh with mostly regular vertices.

In [2], we introduce ACM: Atlas of Connectivity Maps to efficiently capture the connectivity between the regular patches of a semiregular model. These patches, which can be obtained from an arbitrary refinement, are mapped onto a set of quadrilateral 2D domains. The connections between vertices and faces are captured by these 2D domains (connectivity maps) and their interconnections. The ACM can be used as an efficient data structure for semiregular meshes to handle connectivity queries.

In ACM, a coordinate system is assigned to each connectivity map such that each vertex has integer coordinates. These integer coordinates are used to index the faces and vertices and handle neighborhood queries. A hierarchical relationship exist between connectivity of vertices and faces at various resolutions. To establish this hierarchical relationship, we apply rotation, translation, and/or scaling to transform the coordinate system of one resolution to another. The vertices' 3D coordinates are stored in 2D arrays associated with each connectivity map and indexed by the vertices' connectivity map coordinates.

We also categorize regular refinements for quad meshes, and for each category, we propose methods to handle adjacency and hierarchical queries using our data structure. We then describe how to support triangle meshes and discuss applications such as subdivision and multiresolution.

In this paper, we extend our previous work [2] in several different ways. In [2], we describe how to set up an ACM for an initial coarse mesh with arbitrary connectivity. We then make the semiregular model by applying regular refinements on the initial coarse mesh. Here, we present how to set up an ACM for a given semiregular mesh and associate a connectivity map with each regular patch of the mesh. We also describe how to handle sharp features such as creases and corners.

One of the immediate applications of the ACM is the support of meshes resulting from different subdivision methods. In [2], we note that ACM is efficient both in terms of space and time at supporting connectivity queries of subdivision surfaces. A multiresolution framework, which allows one to transition between the high and low resolution versions of a model without losing details, can be developed by pairing subdivision with its reverse subdivision scheme. The ACM can also be efficiently used to support multiresolution frameworks.

In [2], we describe how to support Catmull-Clark, Loop, and $\sqrt{3}$ reverse subdivision. Here, we extend it to support $\sqrt{2}$ reverse subdivision. In addition, supporting the recently developed "smooth reverse subdivision" multiresolution framework is also presented. We also provide the filters for reverse $\sqrt{2}$ and $\sqrt{3}$ reverse and smooth reverse subdivision.

We have compared the time and space efficiency of our data structure with alternatives in [2]. The half-edge data structure used for our comparison in [2] was implemented by ourselves and was not based on a standard implementation. Here, we report the speed of the half-edge data structure implemented in CGAL to make a comparison with a standard implementation of the half-edge data structure [3].

We organize the paper as follows: in Section 2 some related work is presented. We provide an overview and a detailed description of the ACM in Section 3. A method is provided in Section 4 to adapt the ACM to a given high resolution semiregular model. A representation for sharp features in the ACM is described in Section 5. In Section 6, an efficient technique for multiresolution representations for Catmull-Clark, Loop, $\sqrt{2}$, and $\sqrt{3}$ is proposed. We also describe how to handle smooth reverse subdivision using the ACM. We compare our work with CGAL as an efficient implementation of half-edge in Section 7. Future work and limitations are presented in Section 8 and we conclude in Section 9. We also provide the filters of $\sqrt{2}$, and $\sqrt{3}$ compact reverse subdivision in Appendix A, and Appendix B and the filters of smooth reverse $\sqrt{2}$ and $\sqrt{3}$ subdivision in Appendix C.

2. Related Work

Data structures for semiregular models can be found primarily in literature related to subdivision and multiresolution. We present work related to our proposed method, divided into two categories: subdivision and multiresolution.

Subdivision: Subdivision is a well-studied subject in computer graphics. There are many subdivision schemes, such as Loop, Catmull-Clark, Doo-Sabin, $\sqrt{2}$ and $\sqrt{3}$ subdivision [4, 5, 6, 7, 8]. Subdivision is typically a two-step process: one step of refinement followed by an averaging step. The relationship between lattices at different resolutions resulting from different types of refinement has been previously classified in [9, 10]. Our categorization of refinements is similar to their work. However, we have classified subdivision to assist in designing an efficient data structure to address connectivity queries on an arbitrary connectivity model.

The half-edge data structure and its variations are commonly used to model subdivision surfaces [11]. These data structures are designed for general topological objects' adjacency queries. However, the half-edge data structure cannot be directly used for hierarchical access. Furthermore, it does not benefit from the regularity of subdivision and, therefore, for objects with a large number of vertices it becomes inefficient.

An alternative data structure that supports hierarchical operations is the quadtree [12]. Quadtrees are commonly used for hierarchical meshes, particularly for hierarchical editing applications [13]. Although quadtrees are quite effective at supporting hierarchy between resolutions, they need to store many pointers to maintain their nodes' conductivities and hierarchical dependencies. To overcome this inefficiency, indexing methods exist which assign a unique index to every node and discard the tree structure. [12]. However, these indexing methods are primarily designed to support hierarchy and ignore adjacency relationships. Moreover, since quadtrees are designed to support 1-to-4 refinement, they cannot be directly used to support other refinements.

Patch-based refinement methods rely on data structures that are specifically designed for subdivision methods [14, 15, 16, 17, 18]. Here, meshes are divided into patches and subdivision is separately applied to each patch. Each patch is stored in an array and the connectivity between the patches' boundaries is handled using repetitive points at the boundary edges or a first resolution edge based data structure. These methods are mostly designed for a specific type of refinement or primitive shapes [15, 18]. Some of these data structures use spiral 1D indexing for vertices [16, 17]. Spiral indexing complicates neighborhood access, especially for non-immediate neighbors that are essential for applications like multiresolution. We instead use simple 2D domains to maintain connectivity information and extend patch-based methods to support all types of refinement.

Multiresolution: While subdivision generates high resolution objects, multiresolution provides a means to transition from high to low resolution and vice versa [19]. Some multiresolution frameworks, though not all, maintain the semiregularity of objects. This can be achieved by reversing the process of subdivision (i.e. via a reverse subdivision process) [20, 21, 22] or by considering a property of the coarse vertices, such as smoothness (computed via the Laplacian) [13]. Since both the Laplacian and reverse subdivision use local operators to coarsely sample the fine model, our proposed method can handle these operations.

Olsen et al. [20, 21] provide a compact multiresolution framework using the concept of even/odd vertices. At different resolutions, the even/odd labeling distinguishes multiresolution details from coarse vertices. They use an edge based data structure to handle connectivity queries and a hashing method to map vertices to details or coarse vertices [23]. To show that our ACM can efficiently support multiresolution frameworks, we describe how to support the compact multiresolution proposed in [20, 21] and compare the speed of our data structure with [23].

To adapt the half-edge structure to multiresolution frameworks, Kraemer et al. [24] modify this data structure by defining sequences of half-edges. Using this *multiresolution half-edge* structure, it is possible to support primal and dual schemes. However, this data structure requires a large amount of memory for high resolution models due to the storing of all edges and an extensive amount of time is needed to update the structure after each refinement. By comparison, our method, saves a significant amount of memory and time.

3. ACM Description

In this section, we first provide an overview of ACM and describe the basic ideas behind this method. Afterwards, we give a detailed description of ACM as well as the essential elements of each connectivity map. In the detailed description, a formal mathematical notation is used to describe the method.

3.1. Overview of ACM

A semiregular model is made of a number of regular patches connected to each other. The connectivity of each patch can be captured easily by a simple 2D domain with a 2D indexing method and then the geometry of vertices can be recorded in a 2D array. The indices of the vertices can be based on a simple Cartesian coordinate system assigned to each 2D domain. While connectivity queries between the vertices internal to each patch are handled by simple neighborhood vectors, a transformation is used to traverse from one patch to another. These simple 2D domains and their interconnections can be maintained through the resolutions (for all types of refinements) by applying a transformation (imposed by the refinement) to the coordinate system of each 2D domain. In the following section, we formally describe how to set up these 2D domains and the essential transformations for a variety of refinements.

3.2. Detailed Description

As mentioned earlier, semiregular meshes consist of/are made of connected regular patches (Figure 2). Each regular patch in an ACM is assigned a simple 2D domain, this 2D domain and its connectivity information in a mesh is called *connectivity map* ($CM(i)$). For $CM(i)$, a 2D coordinate system is used to index vertices. An index $(i, j)_r$ refers to a 2D location array that stores the 3D positions of vertices at resolution r. Connectivity queries for vertices falling in $CM(i)$ are handled using vectors called neighborhood vectors. By adding simple vectors to the index of a vertex, its neighbors are found. Figure 2 (c) illustrates the indexing method for vertices and the use of neighborhood vectors $(1, 0)$, $(-1, 0)$, $(0, 1)$, and $(0, -1)$ to connect a vertex to its neighbors.

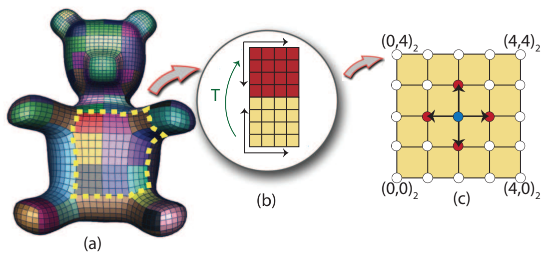

Figure 2: (a) A semiregular mesh. (b) For each patch a 2D domain (connectivity map) with a coordinate system is assigned. It is possible to move from one patch to another by mapping the coordinate systems of two adjacent connectivity maps. (c) The coordinate system of each connectivity map is used to index vertices. Neighbors of vertices are found using neighborhood vectors.

To record the connectivity of the entire model, $CM(i)$ records the adjacency information of its neighboring connectivity maps $CM(N(i))$. To access $CM(N(i))$ from $CM(i)$, we use a transformation mapping the coordinate

Table 1: T_{ref} and T_{int} for different refinements and subdivision methods. S, T, and R denote scaling, translation, and rotation. Subscripts $e \to o$ and $e \to o$ denote transitions from even to odd and odd to even resolutions.

Refinement	Subdivision	T_{ref}	T_{int}
1-to-4	Catmull-Clark	$S(\frac{1}{2})$	$S(2)$
1-to-4	Doo-Sabin	$S(\frac{1}{2})T(\frac{1}{2},\frac{1}{2})$	$S(2)T(\frac{-1}{2},\frac{-1}{2})$
1-to-2	$\sqrt{2}_{e\to o}$	$S(\frac{1}{\sqrt{2}})R(\frac{\pi}{4})$	$S(2)$
	$\sqrt{2}_{o\to e}$	$S(\frac{1}{\sqrt{2}})R(\frac{-\pi}{4})$	I
1-to-2	$\text{Simplest}_{o\to e}$	$S(\frac{1}{\sqrt{2}})R(\frac{\pi}{4})T(\frac{1}{2},0)$	$S(2)$
	$\text{Simplest}_{o\to e}$	$S(\frac{1}{\sqrt{2}})R(\frac{-\pi}{4})T(\frac{-1}{2},\frac{1}{2})$	$T(\frac{-1}{2},\frac{-1}{2})$

system of $CM(i)$ to the coordinate system of its neighbor. These transformations are encoded as integer numbers, as shown in Figure 3 (b), to make the implementation and storage easier. As a result, each connectivity map $CM(i)$ has a 2D array of 3D points storing all vertices' locations, two 1D arrays recording the neighbors $CM(N(i))$ of a connectivity map $CM(i)$ and a transformation mapping the coordinate system of $CM(i)$ to its neighbors $CM(N(i))$. For a semiregular mesh with N patches, an array of connectivity maps CM is stored. Therefore, $CM(i)$ in which $0 \leq i < N$ refers to ith patch of the model. Since the corners of $CM(i)$ may be irregular, we also store a list of integers for each corner to capture its connectivity information (Figure 3 (c)). This list stores the index of connectivity maps in CM attached to each corner of $CM(i)$. Figure 3 (a) illustrates the essential elements of each connectivity map.

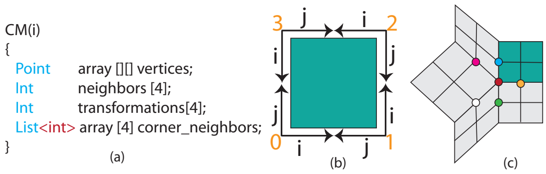

Figure 3: (a) The elements of each connectivity map. (b) Transformations mapping the coordinate system of one connectivity map to another are encoded as integer numbers. (c) The red vertex is a corner. Quads attached to corners are saved as well. Corners can be extraordinary.

Each patch of a semiregular model can be treated as a bounded lattice. As studied in [10], when a refinement is applied on a lattice, the resulting lattice is transformed by the refinement.

For example, the 1-to-4 refinement used in Catmull-Clark subdivision imposes a scaling by $\frac{1}{2}$ on the lattice at the next resolution (Figure 4) . We call this transformation T_{ref} (Figure 5 (b)). To convert these scaled coordinates to integer coordinates, which are necessary to refer to the 2D location array, we apply another transformation called T_{int} (Figure 5(c)).

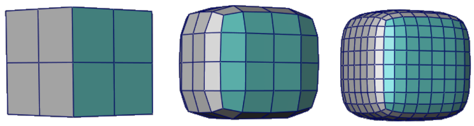

Figure 4: Applying Catmull-Clark subdivision on a refined cube.

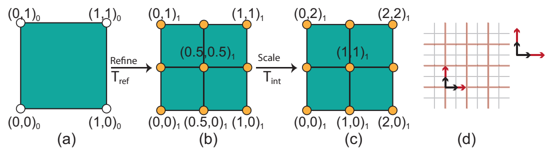

Figure 5: (a) a quad before refinement. (b) 1-to-4 refinement imposes a scaling by $\frac{1}{2}$ called T_{ref}. (c) T_{int} which is an scaling by two is applied to get integer coordinates. (d) Red and black lattices are the connectivity lattices before and after 1-to-4 refinement, respectively.

As we discussed earlier, the neighbors of a vertex in a connectivity map $CM(i)$ are found using vectors called neighborhood vectors defined in the coordinate system of $CM(i)$. Since, after refinement, the connectivity of the vertices are affected by both transformations T_{ref} and T_{int}, the neighborhood vectors are changed by $T_{ref} \circ T_{int}$ in a transition from one resolution to another. For example, $T_{ref} \circ T_{int} = I$ in 1-to-4 refinement. As a result, the neighborhood vectors are the same after applying 1-to-4 refinement (Figure 6).

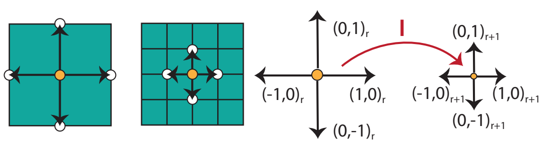

Figure 6: Neighborhood vectors at two successive resolutions of 1-to-4 refinement. Neighborhood vectors are the same due to the identity transformation.

In [2], we use the transformation imposed by each refinement and provide a categorization of refinements. For each category, we discuss how to maintain the struc-

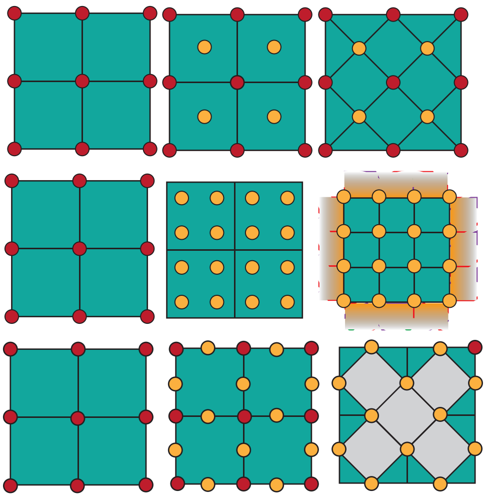

Figure 7: Top: 1-to-refinement of $\sqrt{2}$ subdivision [7]. Middle: 1-to-4 refinement of Doo-Sabin subdivision [6]. Orange faces are shared with other connectivity maps. Bottom: 1-to-2 refinement in Simplest subdivision [25].

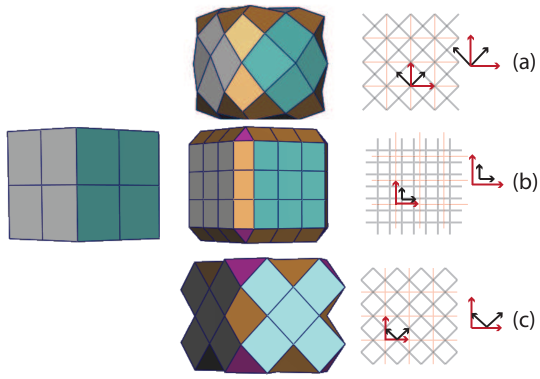

Figure 8: The lattices of different refinements and subdivision methods. (a) Scaling, rotation, no translation appearing in $\sqrt{2}$ subdivision [7]. (b) Scaling, translation, no rotation appearing in Doo-Sabin subdivision [6]. (c) Scaling, translation, rotation appearing in simplest subdivision [25].

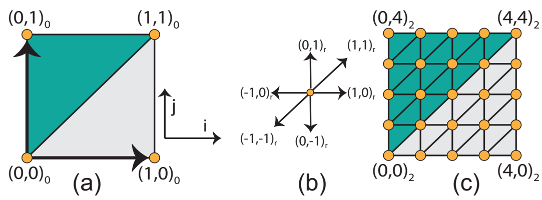

Figure 9: (a) A connectivity map assigned to a triangle pair. (b) Neighborhood vectors of vertices in triangular meshes. (c) Applying 1-to-4 refinement twice on a triangular connectivity map.

ture of the ACM. As illustrated in Figure 5(d), since the lattice is only scaled by the refinement, 1-to-4 refinement falls in the category of "Scaling, No Rotation, No Translation". In [2], we have discussed three more categories: "Scaling, Rotation, No Translation", "Scaling, No Rotation, Translation", and "Scaling, Rotation, Translation". For each category, we first find T_{ref} by looking at the lattices at two successive resolutions and then define T_{int} to obtain integer indices. Table 1 presents the transformations T_{ref} and T_{int} for different subdivision schemes. The 1-to-2 refinements used for $\sqrt{2}$ and simplest subdivision and the 1-to-4 refinement used by Doo-Sabin subdivision are presented in Figure 7. The result of these subdivision schemes on a refined cube and lattices of their refinements at two successive resolutions are also illustrated in Figure 8. Further discussion of each category is presented in [2].

The ACM can be easily extended to triangular meshes by pairing connected triangles. We can assign a connectivity map to each triangle pair and treat it as a quad, allowing us to continue using quadrilateral connectivity map domains (Figure 9 (a)).

In order to create such a domain, we can pair adjacent triangles to form a quad, creating a single connectivity map. Suppose that we have a set of faces $F = \{f_1, f_2, ..., f_M\}$, we can pair f_i with f_j if f_i and f_j are adjacent. Afterwards, both f_i and f_j are removed from F and the process repeats until no adjacent faces exist in F.

A complete pairing of triangles is possible and is computable in $O(Mlog^4 M)$, where M is the number of triangles [26, 27]. Triangles in F that remain unassigned to a pair (isolated triangles) may each be assigned to a half-empty connectivity map. As a result, isolated triangles can be handled without compromising the data structure (Figure 10). However, for the purposes of efficiency, it is better to reduce the number of isolated triangles by using methods that can make a pure quad mesh from a given triangular one [27, 28].

Similar to quad meshes, various refinement methods can be supported by ACM for triangular meshes. For example, Figure 11 illustrates the result of Loop subdivision, which uses a 1-to-4 refinement, on Venus and Figure 10 illustrates the result of $\sqrt{3}$ subdivision with 1-to-3 refinement on a pawn . Each connectivity map is colored differently.

4. Connectivity Maps Identification

In [2], we describe how to build an atlas of connectivity maps by assigning a connectivity map to each quad or triangle pair at the coarse resolution, from which finer

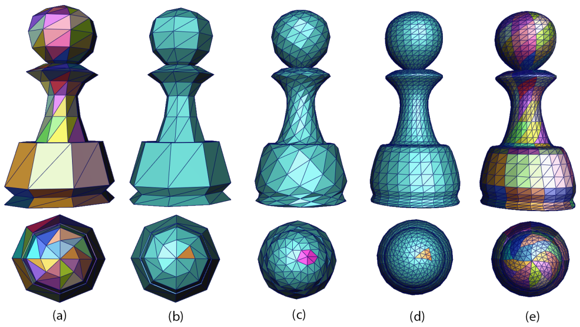

Figure 10: Applying $\sqrt{3}$ subdivision on the pawn model. Bottom: zoomed screen shots of the head of the pawn. The isolated triangle is shown in orange. The pink portion indicates the region formed by subdividing the isolated triangle. Triangle pairs are shown with different colors in (a) and (e).

Figure 11: Applying Loop subdivision on Venus. Each connectivity map is in a different color.

resolutions may be found. However, there are many applications in which the semiregular model is at a high resolution and we wish to assign an ACM structure to it. For example, the models resulting from the parametrization methods proposed in [29, 30, 31] are semiregular. This means that the mesh can be considered as the result of repetitive refinement on a coarse mesh. The ACM may be a good data structure to handle adjacency queries for these resulting models. As a result, it is desirable to assign an ACM to a given high resolution semiregular model.

For a given semiregular mesh M, we want to find patches that cover all faces and vertices of M and assign a connectivity map to each patch. The basic idea of this method is to start from an irregular vertex and move along two edges incident to this vertex and find a patch connecting two irregular vertices. M can be the result of any type of refinement. Here, we describe 1-to-4 refinement. Other cases are fairly similar.

Let v_0 be an extraordinary vertex. v_0 is considered be the corner of a connectivity map. Given a face f, two directions i and j are defined based on the two edges incident to v_0 in f. We consider two pointers moving along i and j called $count_i$ and $count_j$, illustrated with green and pink arrows in Figure 12. $count_i$ and $count_j$ start from v_0 and move along both directions simultaneously until another irregular vertex v_1 is met. The quadrilateral patch along i and j including v_0 and v_1 is stored as a connectivity map and removed from M. The process is repeated until no face remains in M.

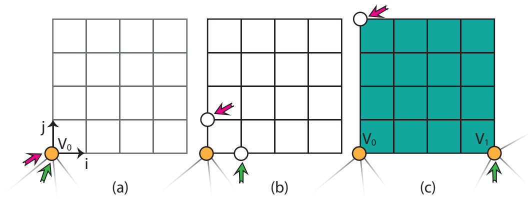

Figure 12: (a) v_0 is an irregular vertex. $count_i$ and $count_j$, colored in green and pink respectively, starting at v_0. (b) $count_i$ and $count_j$ move along i and j simultaneously. (c) $count_i$ and $count_j$ move until they reach to an extraordinary vertex.

Consider face f including vertices w_0, w_1, w_2, and w_3 in which w_0 is extraordinary (Figure 13 (a)). In f, e_0 connects w_0 to w_1 and e_1 connects w_1 to w_2. We consider the direction of e_0 to be direction i in f and $count_i$ needs to move along i starting at w_0. Let traverse $(f, w_1, e_1) = (f_i, e_i)$ where f_i shares e_1 with f and $e_i \neq e_1$ is an edge in f_i incident to v_1. We determine traverse $(f, w_1, e_1) = (f_i, e_i)$ and take the direction of e_i to be direction i in face f_i. The process repeats for $count_i$ starting at w_1 and face f_i in the direction of e_i.

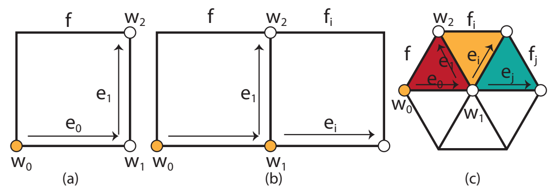

Figure 13: (a) Face f, its edges, and vertices. v_0 is extraordinary. (b) To move along e_0, face f_i sharing e_1 with f is found. (c) To move along e_0, in a triangular face f, first, f_i sharing e_1 with f is found and then f_j sharing e_i with f_j is found.

Given a triangular mesh, we should slightly modify the algorithm. Consider w_0 to be an extraordinary vertex in face f including vertices $\{w_0, w_1, w_2\}$. The edges e_0 and e_1 incident to w_0 in f are found. Then, we determine traverse $(f, w_1, e_1) = (f_i, e_i)$ and traverse $(f_i, w_1, e_i) = (f_j, e_j)$. The direction of e_j is taken to be direction i in f_j, $count_i$ is assigned to w_1 in f_j, and the process repeats.

$count_i$ and $count_j$ simultaneously move k steps until another extraordinary vertex w_k is met. The $k \times k$

quadrilateral section including w_0 and w_k along $count_i$ and $count_j$ is considered to be a patch. After finding a patch and assigning a connectivity map to it, the i and j directions together define the coordinate system of the connectivity map. The vertex locations, the transformations mapping the coordinate system of a connectivity map to its neighbors and the connectivity maps neighboring each corner are stored in separate arrays.

It is possible for a patch to be identified that includes more than one connectivity map. To be consistent with the structure of the ACM, we find the smallest connectivity map and split the bigger connectivity maps into connectivity maps with equal dimensions. For instance, if there exists a 16×16 connectivity map but mesh M has a 4×4 connectivity map, the 16×16 connectivity map is split into four 4×4 connectivity maps.

This algorithm, as described, is for the 1-to-4 refinement used in Catmull-Clark or Loop subdivision. Other refinements such as the 1-to-2 and 1-to-3 refinements used in $\sqrt{2}$ and $\sqrt{3}$ subdivision, respectively, are handled exactly the same. For some other refinements, we may slightly modify the process. For example, for the 1-to-4 refinement used in Doo-Sabin subdivision, we start from the vertices of a non-quad face and move along the quad faces until another non-quad face is met. As a result, we need to know which refinement method the given mesh has had applied.

The proposed algorithm is not only very simple but it is also very efficient in terms of speed. Table 2 reports the time needed to set up an ACM for high resolution semiregular models with different numbers of faces. These models are also illustrated in Figure 14. We have only used face-list vertex-list of a model to adapt ACM to a given semiregular mesh since we only need to traverse from one face to another and find extraordinary vertices.

Table 2: Time (in seconds) required to set up an ACM for different models.

Model	#of Faces	Time
Teddy	1028	1.81
Bullet	1536	0.45
Dog	2592	3.43
Seat	1152	0.32
Pawn	912	0.26

5. Sharp Features

Many realistic objects have sharp features such as creases and corners. Subdivision methods handle sharp features by treating some vertices and edges differently along the creases or corners [32, 33].

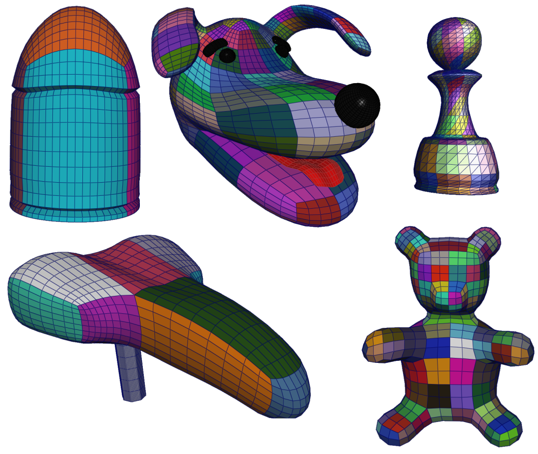

Figure 14: Patches of semiregular models are found in the time reported in Table 2. A connectivity map with a distinct color is assigned to each patch.

To support creases, some edges are initially tagged as sharp edges. Then, at each subdivision iteration, the edges resulting from the initial sharp edges are tagged as sharp and they are subdivided using curve subdivision masks [33].

Here, the main task is to find the edges resulting from iterative subdivision on a sharp edge. This hierarchical task can be addressed efficiently in the ACM. The main questions are how to represent an edge and how to find the resulting edges after subdivision. In the ACM, an edge can be represented by a pair of vertices. Since these two vertices have indices in the ACM, they can be used to handle the task of hierarchically determining all edges produced by the refinement of a sharp edge. For example, if an edge connecting vertices $v_r = (a, b)_r$ and $w_r = (a + 1, b)_r$ is considered to be sharp, all vertices with index $(c, 2^n b)_{r+n}$,$a \leq c \leq a + 2^n$, are considered to be sharp at the n the resolution after 1-to-4 refinement. As a result, iteratively tagging sharp edges at intermediate resolutions is unnecessary and a tagging at the coarsest resolution is enough. We can also use this method to apply semi-smooth masks on boundary edges. A semi-smooth mask (boundary mask) must be applied on the boundary edge of any connectivity map with a null neighbor.

This is also extendible to other types of refinement. For example, the boundary and sharp edges are stationary at odd resolutions of $\sqrt{3}$ subdivision, hence we apply a curve subdivision scheme to the corresponding vertices at every other resolution. That is, vertices with index $(c, 3^n b)_{r+2n}$, $a \le c \le a + 3^n$ are modified when an edge connecting vertices $v_r = (a, b)_r$ and $w_r = (a+1, b)_r$ is considered to be sharp. Figure 15 illustrates some results with sharp features and boundary edges.

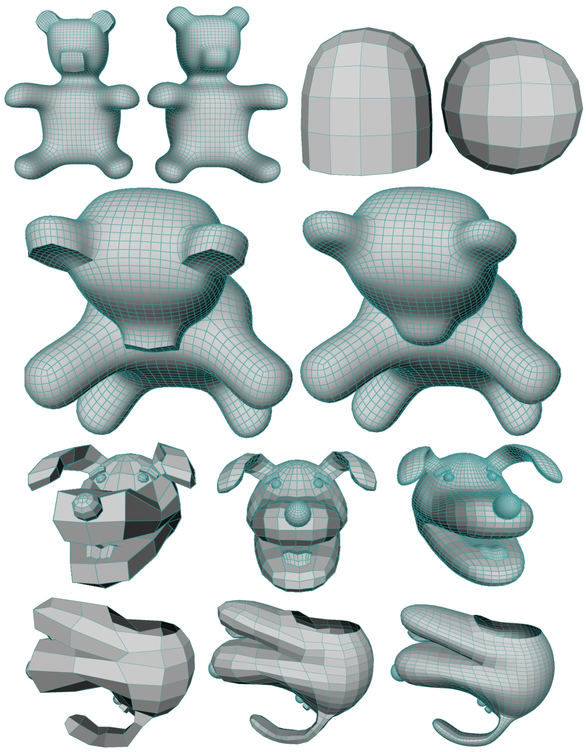

Figure 15: Sharp features on the nose and ears of Teddy are notable in comparison to the one which is subdivided smoothly. The edges at the bottom of the cube are also sharp. The dog's head also has boundary edges that are subdivided using boundary masks.

6. Multiresolution

The multiresolution representation of a mesh consists of a simple base mesh and a sequence of wavelet coefficients called detail vectors at various resolutions [34]. This representation provides a framework in which it is possible to traverse between the levels of detail/resolutions of a mesh. While subdivision methods are used to create high resolution objects, reverse subdivision can be used to decompose the high resolution model to a low resolution version along with the details lost in the process (wavelet coefficients) [35, 36]. In this multiresolution framework, it is possible to define a compact representation in which the storage requirements for the details and coarse vertices together equal the storage requirement of the fine vertices. As noted earlier, Olsen et al. [20] provides such a compact multiresolution. They categorize vertices into even and odd vertices, with even vertices storing coarse vertex locations and odd vertices storing multiresolution details after reverse subdivision. For example, the vertex-vertices and edge-vertices of Loop reverse subdivision are respectively labeled as even and odd (see Figure 16). The details of an even vertex, therefore, are found using a linear combination of odd details in its neighborhood.

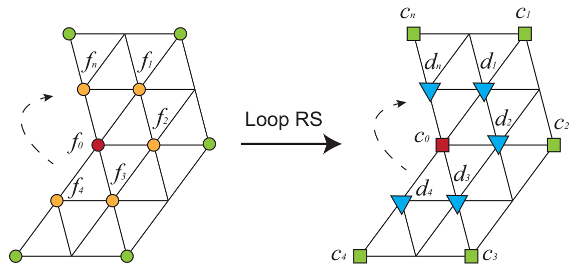

Figure 16: Vertex and face-vertices (●,● and ●) are replaced by coarse vertices and details (■ ,■ and ▲).

Taking advantage of of this even/odd partitioning of vertices could potentially help in creating a data structure. One possibility is to use an edge-based data structure with an additional structure such as a hash table to encode even and odd vertices [23]. However, the ACM can be used as an efficient data structure for this type of multiresolution. In ACM, since each vertex has a 2D index, this index can be used to distinguish between vertices and support a partitioning of the vertices. There exists a variety of multiresolution frameworks that impose a similar partitioning of vertices with different geometrical properties [37, 38]. Due to the possibility of categorization of vertices using their indices, it is also possible to efficiently support these methods.

In this section, we first describe our proposed data structure for compact multiresolution frameworks based on Catmull-Clark, Loop, $\sqrt{3}$, and $\sqrt{2}$ reverse subdivision. We then describe smooth reverse subdivision in this section, which has been proposed to reduce the coarsening effects of compact multiresolution frameworks. Each of these multiresolution frameworks are expressed by geometric masks indicating the contributions of neighboring vertices to the final position of the coarse vertices or details. The masks of smooth reverse subdivision for Loop and Catmull-Clark have been provided in [37]. We also provide masks for $\sqrt{3}$ and $\sqrt{2}$ smooth reverse subdivision in Appendix C.

Catmull-Clark: The reverse schemes of Catmull-Clark and Loop subdivision are respectively introduced in [21] and [20]. Here, we discuss how to access the neighbors of a coarse vertex and its corresponding details, which are essential operations in the reconstruction process.

When Catmull-Clark reverse subdivision is applied on a semiregular mesh, vertex-vertices are replaced by coarse approximations and face-vertices and edge-vertices are replaced by details. In the even/odd categorization, vertex-vertices are tagged as even and face-vertices and edge-vertices are tagged as odd. This categorization can be easily supported in ACM. A vertex with index $(a, b)_r$ is replaced by a coarse approximation at resolution $r - 1$ if a and b are both even, otherwise it is replaced by a detail.

This representation of vertices can be extended to several levels of reverse subdivision by using the mentioned hierarchical relationships. As a result, vertices with indices $(a, b)_r$ after n levels of reverse subdivision are coarse vertices if $\left\lfloor \frac{a}{2^n} \right\rfloor = \frac{a}{2^n}$ and their corresponding details are located at $(c, d)_r$ if $\left\lfloor \frac{c}{2^{n-1}} \right\rfloor = \frac{d}{2^{n-1}}$. Figure 17 illustrates the application of Catmull-Clark reverse subdivision to a connectivity map. To access the neighbors of a coarse vertex and its corresponding details, scaled neighborhood vectors are used. The structure of the neighborhood vectors remains the same as in Figure 2 but they are scaled by 2^n (after n levels of reverse subdivision) to access the neighbors of a coarse vertex and 2^{n-1} to access the corresponding details (Figure 17).

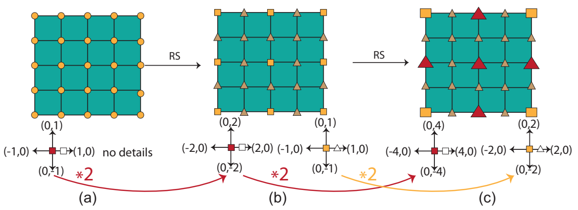

Figure 17: A subdivided connectivity map after two levels of reverse subdivision. □: coarse vertices. △ and ▲: details after one and two levels of RS. and : access to coarse vertices and details.

Loop: In Loop reverse subdivision, edge-vertices are replaced by details and vertex-vertices are replaced by coarse approximations [20]. Since Loop subdivision uses a 1-to-4 refinement similar to Catmull-Clark subdivision, the categorization of vertices is very similar to Catmull-Clark reverse subdivision. However, different neighborhood vectors, as illustrated in Figure 9, are used. Figure 18 illustrates the application of Loop reverse subdivision to a connectivity map and Figure 19 illustrates a semiregular Venus and its reverse subdivision at three resolutions.

$\sqrt{3}$ **subdivision:** In $\sqrt{3}$ subdivision, the connectivity of the vertices is affected by the 1-to-3 refinement. At each step, a face-vertex is inserted in each face and

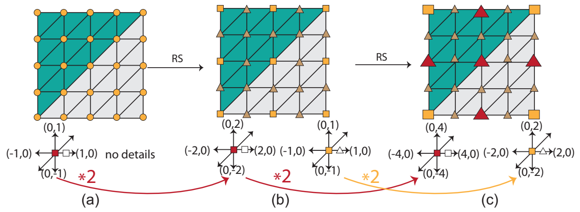

Figure 18: A subdivided connectivity map after two levels of reverse subdivision. □: coarse vertices. △ and ▲: details after one and two levels of RS. and : access to coarse vertices and details.

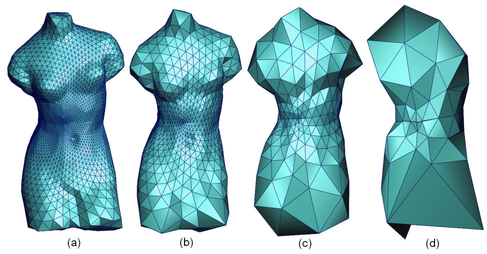

Figure 19: A semiregular venus after three applications of Loop reverse subdivision.

edges are flipped (see Figure 20).

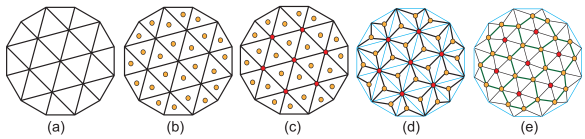

Figure 20: (a) A triangular patch. (b) Face-vertices are inserted. (c) Vertex-vertices are drawn in red. (d) Face-vertices are connected to vertex-vertices. (e) Edges are flipped.

$\sqrt{3}$ reverse subdivision can be also determined using a similar method to that proposed in [20]. In this case, face-vertices are replaced by details and vertex-vertices by coarse approximations (see Figure 21). More discussion and the masks of $\sqrt{3}$ reverse subdivision in a compact multiresolution framework have been provided in Appendix A.

Handling adjacency queries in $\sqrt{3}$ reverse subdivision is again very straightforward using neighborhood vectors with a scaling factor of three. To access details and coarse vertices, the transformations $T_{e \to o}$ and $T_{o \to e}$ discussed in [2] are used to change the neighborhood vectors as illustrated in Figure 22.

$\sqrt{2}$ **subdivision:** In the 1-to-2 refinement of $\sqrt{2}$ subdivision, face-vertices are inserted in each face and connected to vertex-vertices and the previous edges are removed (see Figure 7). In $\sqrt{2}$ reverse subdivision, face-vertices are replaced by details and vertex-vertices are

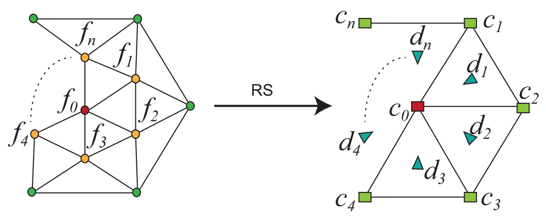

Figure 21: Vertex and face-vertices (●,● and ●) are replaced by coarse vertices and details (■ ,■ and ▲)

.

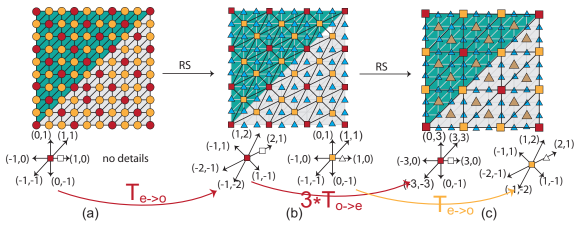

Figure 22: (a) ● and ● are face and vertex vertices that are replaced by details and coarse vertices. (b) □ and ▲ : coarse vertices and details. (c) ▲:details after two levels of RS. and : access to coarse vertices and details. Transformations of essential neighborhood vectors are shown.

replaced by coarse approximations. We have discussed $\sqrt{2}$ reverse subdivision and its masks in Appendix B. Adjacency queries are again handled by neighborhood vectors that may be scaled by a factor of two. To access details and coarse vertices, transformations $T_{e\rightarrow o}$ and $T_{o\rightarrow e}$ (discussed in [2]) for $\sqrt{2}$ subdivision are used to change the neighborhood vectors as illustrated in Figure 23.

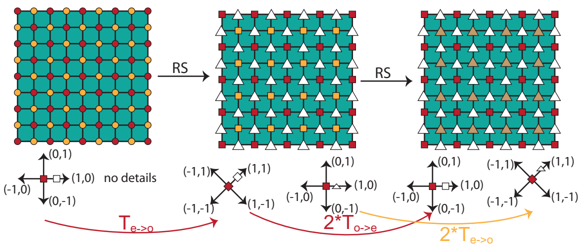

Figure 23: (a) ● and ● are face and vertex vertices that are replaced by details and coarse vertices. (b) □ and △ : coarse vertices and details. (c) ▲:details after two levels of RS. and : access to coarse vertices and details. Transformations of essential neighborhood vectors are shown.

Smooth Reverse Subdivision: Note that in compact reverse subdivision, in which a local refinement is applied on the resulting coarse approximations, unwanted exaggerations of the mesh often appear, which we wish to reduce (see Figure 24). Sadeghi and Samavati have proposed a method in which they consider the smoothness of the coarse mesh by perturbing coarse vertices using a Laplacian constraint [37]. Using a modified Laplacian [38], a compact representation can be found for the multiresolution framework presented in [37]. However, here, we would like to explore how to handle multiresolution frameworks with over-representation such as those found in [37] or [13].

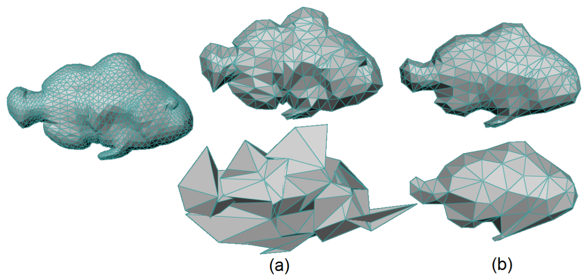

Figure 24: A fish is reverse subdivided by (a) compact reverse Loop subdivision (b) smooth reverse Loop subdivision.

For example, to support the method in [37] using the ACM, in addition to a connectivity map that stores the coarse approximations and details for odd vertices we also store a connectivity map called the over-representation connectivity map which stores an additional detail for each coarse approximation (Figure 25). The reconstruction process is very similar to compact multiresolution, except the indices of the details connectivity maps are also scaled and added to the coarse approximations. Although Sadeghi and Samavati proposed the method only for the Catmull-Clark and Loop subdivision methods, it is also extendible to $\sqrt{2}$, and $\sqrt{3}$ subdivision methods. We have provided the masks in Appendix C.

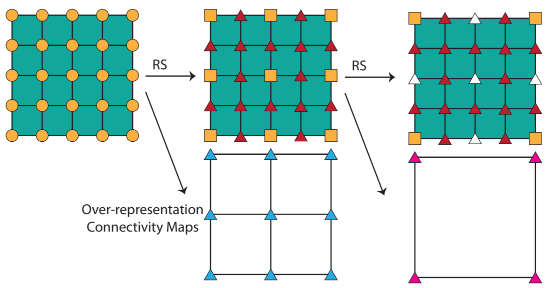

Figure 25: A connectivity map which is reverse subdivided using the Catmull-Clark method. A details connectivity map with dimensions equal to the number of coarse approximations is also needed to completely reconstruct the model.

This method is also usable to support the multiresolution framework proposed in [13]. The connectivity structure of this framework is the same as [37]. The only difference is that in [13], coarse approximations are found by down-sampling the fine points instead of applying reverse subdivision masks.

Sadeghi and Samavati recently modified the smooth reverse subdivision framework by introducing a fairing technique with banded inverse [38]. They replaced the discrete Laplacian operator in the smooth reverse subdivision framework with this new fairing technique. This technique has two steps. First, to smooth vertex v, it is perturbed along its Laplacian vector towards the average M of its neighbor vertices. To do so, vertices in the neighborhood of v are fixed and v alone is moved to w in which $w = \alpha v + (1 - \alpha)M$ (see Figure 26(a)). In the next step, all vertices connected to v are moved similarly. As a result, the vertices of a mesh faired in this manner must be partitioned into two disjoint sets, *red* and *green*, in which no vertex in either the red or green set has a neighbor in its own set.

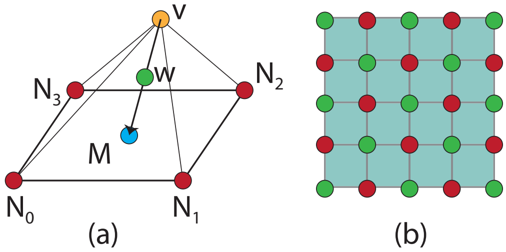

Figure 26: (a) v is moved along the vector connecting v to M. M is the average of vertices N_i in the neighborhood of v. (b) Green and red vertices in a connectivity map.

As mentioned in [38], the meshes resulting from Catmull-Clark subdivision have this property since the vertices can be partitioned into edge-vertices (red) and face- and vertex-vertices (green) (see Figure 26). The ACM can provide a simple distinction for the red and green sets in meshes resulting from Catmull-Clark subdivision. Any vertex with index $(a, b)_r$ in which either a or b is even (not both simultaneously) is an edge vertex and falls in the red set and all other vertices are in the green set. Figure 26(b) shows the coloring of a connectivity map and Figure 27 illustrates the result of fairing on a mesh resulting from Catmull-Clark subdivision. As a result, using the ACM, we can partition Catmull-Clark meshes into two disjoint sets without any additional tagging and apply the fairing method proposed in [38].

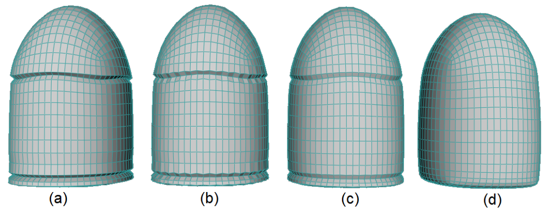

Figure 27: (a) A bullet resulting from Catmull-Clark subdivision. (b) Bullet after fairing red vertices. (c) Bullet after fairing green vertices. (d) bullet after fairing vertices several times. Reduction of the details is apparent.

7. Comparison with CGAL

We have compared the ACM with our lab implementation of half-edge data structure in [2]. Here, for implementing subdivision surfaces, we compare the performance of ACM with CGAL [3]. We report the CPU time usage of half-edge in comparison with the ACM. The ACM remains far more efficient than half-edge for subdividing meshes even against the CGAL implementation, which is one of the most efficient half-edge implementations available. In Table 3, it is apparent that the ACM CPU time usage is much more efficient than CGAL. Tests were run on an intel i7 quad core processor under Windows 7.

Table 3: $\sqrt{3}$ subdivision time (in seconds) required by the ACM and CGAL for a tetrahedron to reach resolution i from resolution $i - 1$.

Resolution	#of Faces	ACM	CGAL
7	8748	0.009	0.092
8	26244	0.015	0.181
9	78732	0.063	0.701
10	236196	0.109	2.20
11	708588	0.975	6.810

In Table 4, we again compare the ACM with CGAL to subdivide a cube using Doo-Sabin subdivision to show the efficiency of the ACM for quadrilateral meshes. To reach resolution seven, the ACM needs only 0.024 seconds while CGAL needs about two minutes. The reason for this difference is that the half-edge structure needs to maintain many pointers to capture the connectivity information of vertices. However, in the ACM, the connectivity of faces, vertices and edges is implicit in each connectivity map and handled using simple algebraic operations.

8. Future work and limitations

The ACM provides an efficient data structure for semiregular models. However, there exist meshes

Table 4: Doo-Sabin subdivision time (in seconds) required by the ACM and CGAL for a cube to reach resolution i from resolution $i-1$.

Resolution	#of Faces	ACM	CGAL
3	116	0.011	0.015
4	404	0.015	0.109
5	1556	0.016	0.656
6	6164	0.016	8.128
7	24576	0.024	117.899

whose vertices are mostly regular but do not have subdivision connectivity, such as meshes resulting from adaptive subdivision [39, 40]. Although it is possible to subdivide one connectivity map more than others, irregular usage of adaptive subdivision makes meshes which are not compatible with ACM. As potential future work, it would be interesting to consider modifications to the ACM to support such meshes. We have described how to handle connectivity queries for triangular and quadrilateral meshes. Hexagonal meshes and their refinements may also be supported by the ACM [41]. Supporting hexagonal meshes presents another direction for future work for our proposed method.

9. Conclusion

We have described the ACM (Atlas of Connectivity Maps) and have shown that it can be efficiently used for a variety of semiregular meshes and adjacency queries on them, including quadrilateral or triangular models refined with different methods. Establishing the ACM at a coarse resolution is described as well as a method to adapt the ACM for a given high resolution semiregular model. Supporting sharp features (creases and corners) in subdivision surfaces is also described. We have emphasized the applications of the ACM to multiresolution frameworks by discussing how to support various multiresolution frameworks. We also compared our proposed ACM with half-edge implemented in CGAL to show the speed efficiency of the ACM in handling connectivity queries in subdivision surfaces.

Acknowledgments

We would like to thank anonymous reviewers for their thoughtful comments in the review process. We also thank Troy Alderson for insightful discussions and editorial comments. This research was supported in part by the National Science and Engineering Research Council of Canada and GRAND Net-work of Centre of Excellence of Canada.

Appendix A. $\sqrt{3}$ Reverse Subdivision

Using the method proposed in [20], the following equations are obtained for $\sqrt{3}$ reverse subdivision. In this method, the details of vertex-vertices (d_0) are determined by a linear combination of the details of face vertices (d_i) in their neighborhood (Figure 21). Equation A.1 demonstrates this relationship. Equation A.2 also indicates how the coarse vertex of a face vertex can be obtained using the vertices in its neighborhood. To avoid magnifying the results from reverse subdivision, coarse vertices are refined by vector δ_0, which is obtained by an optimization to reduce the magnitude of the details d_i [20]. Note that $\alpha = \frac{4-2cos(\frac{2\pi}{n})}{9}$ is the parameter to find the position of vertex-vertices in $\sqrt{3}$ subdivision [8].

$$d_0 = \frac{3}{2n}\alpha \sum_{i=1}^{n} d_i \tag{A.1}$$

$$c_0 = \frac{1}{1-\frac{3}{2}\alpha} f_0 - \frac{\alpha}{n(\frac{2}{3}-\alpha)} \sum_{i=1}^{n} f_i \tag{A.2}$$

$$\delta_0 = \frac{\frac{3}{2n}(1-\alpha)+\frac{1}{3}}{(1-\alpha)^2+\frac{n}{9}} \sum_{i=1}^{n} d_i \tag{A.3}$$

Appendix B. $\sqrt{2}$ Reverse Subdivision

$\sqrt{2}$ subdivision is a scheme designed for quadrilateral meshes [7]. Similar to the $\sqrt{3}$ subdivision scheme, there are two types of masks for face-vertices and vertex-vertices. Face-vertices are simply found by averaging the four vertices making a face, and vertex-vertices are obtained by moving a vertex towards the average of the vertices in its neighborhood. Equations B.1 and B.2 provide the masks for face and vertex vertices respectively. Figure B.28 illustrates the vertices involved in Equations B.1 and B.2. α and β can receive different values. For example, in [7], $\alpha = \frac{1}{2}(1 - cos(\frac{2\pi}{n}))$ and $\beta = 0$. Figure B.29 illustrates the result of this subdivision on Teddy.

$$v_f = \frac{1}{4}(v_0 + v_1 + v_2 + v_3) \tag{B.1}$$

$$v_v = (1-\alpha-\beta)v + \frac{1}{n}\sum_{i=0}^{n-1} \alpha v_{2i} + \beta v_{2i+1} \tag{B.2}$$

Using the method proposed in [20], we can define a compact multiresolution for a class of $\sqrt{2}$ subdivision in which $\alpha = 2\beta$. In this method, the details of vertex

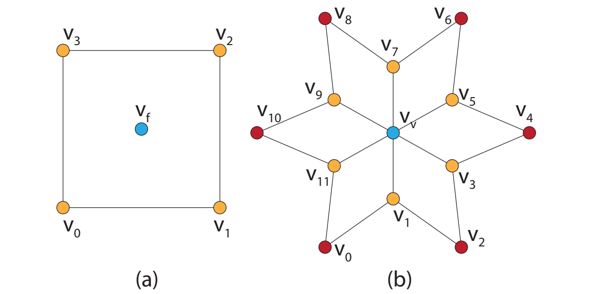

Figure B.28: (a) The vertices (orange) involved in determining the position of face-vertices (blue vertex). (b) The vertices involved in determining the position of blue vertex-vertices. v_{2i} is red and v_{2i+1} is orange.

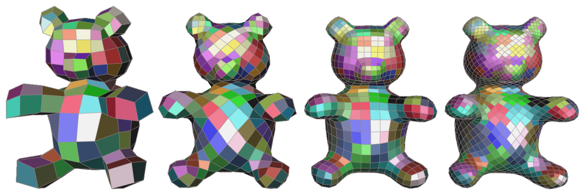

Figure B.29: $\sqrt{2}$ subdivision on Teddy.

.

vertices (d_0) are again found by a linear combination of the details of face vertices (d_i) in their neighborhood (Figure 21). Equation B.3 demonstrates this relationship. Equation B.4 also indicates how coarse vertices are determined. We also give δ_0 to reduce the magnification effect of reverse subdivision for $\sqrt{2}$ subdivision in Equation B.5.

$$d_0 = \frac{4\alpha}{n}\sum_{i=1}^{n} d_i \tag{B.3}$$

$$c_0 = \frac{1}{1-4\alpha} f_0 - \frac{4\alpha}{n-4\alpha n}\sum_{i=1}^{n} f_i \tag{B.4}$$

$$\delta_0 = \frac{\frac{4\alpha}{n}(2-3\alpha)+\frac{1}{2}}{2(1-\frac{3\alpha}{2})^2+\frac{n}{16}}\sum_{i=1}^{n} d_i \tag{B.5}$$

Appendix C. $\sqrt{3}$ and $\sqrt{2}$ Smooth Reverse Subdivision

The basic idea of smooth reverse subdivision is to perturb coarse approximations by vector δ to minimize an energy function $E_{total}(\Delta) = \omega E_{sub}(\Delta) + (1-\omega)E_{smooth}(\Delta)$ in which E_{sub} is the euclidean distance between fine points and subdivided coarse approximations and E_{smooth} is the energy of coarse approximations in the local neighborhood. Using the method proposed in [37], we can solve this weighted optimization problem for $\sqrt{3}$ and $\sqrt{2}$ subdivision methods and find perturbation vector δ. After finding the coarse approximations using compact reverse subdivision methods, we can perturb the coarse approximations using δ. Equations C.1 and C.2 provide δ for the $\sqrt{3}$ and $\sqrt{2}$ smooth reverse subdivision respectively.

$$\delta_{\sqrt{3}} = \frac{(\omega(1-\alpha)\frac{3\alpha}{2n}+\frac{1}{3}\omega)\sum_{i=1}^{n} d_i}{\omega(\frac{n}{9}+(1-\alpha)^2)+(1-\omega)} + \frac{\frac{1-\omega}{n}\sum_{i=1}^{n} c_i-(1-\omega)c_0}{\omega(\frac{n}{9}+(1-\alpha)^2)+(1-\omega)} \tag{C.1}$$

$$\delta_{\sqrt{2}} = \frac{(\omega(1-\alpha)\frac{2\alpha}{n}+\frac{\omega}{4})\sum_{i=1}^{n} d_i}{\omega(\frac{n}{16}+(1-\alpha)^2)+(1-\omega)} + \frac{\frac{1-\omega}{n}\sum_{i=1}^{n} c_i-(1-\omega)c_0}{\omega(\frac{n}{16}+(1-\alpha)^2)+(1-\omega)} \tag{C.2}$$

References

[1] D. Bommes, B. Levy, N. Pietroni, E. Puppo, C. S. a, M. Tarini, D. Zorin, State of the art in quad meshing, in: Eurographics STARS, 2012.

[2] A. Mahdavi-Amiri, F. Samavati, ACM: Atlas of connectivity maps for semiregular models, in: Proceedings of Graphics Interface 2013, 2013, pp. 99–107.

[3] L. Kettner, Halfedge data structures, in: CGAL User and Reference Manual, 4.2 Edition, CGAL Editorial Board, 2013.

[4] C. Loop, Smooth subdivision surfaces based on triangles, Department of mathematics, Masters Thesis, University of Utah (1987).

[5] E. Catmull, J. Clark, Recursively generated b-spline surfaces on arbitrary topological meshes, Computer-Aided Design 10 (6) (1978) 350 – 355. doi:10.1016/0010-4485(78)90110-0.

[6] D. Doo, M. Sabin, Seminal graphics, ACM, 1998, Ch. Behaviour of recursive division surfaces near extraordinary points, pp. 177–181.

[7] G. Li, W. Ma, H. Bao, $\sqrt{2}$ subdivision for quadrilateral meshes, Vis. Comput. 20 (2) (2004) 180–198.

[8] L. Kobbelt, $\sqrt{3}$ subdivision, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, SIGGRAPH 2000, ACM, 2000, pp. 103–112.

[9] M. Alexa, Refinement operators for triangle meshes, Comput. Aided Geom. Des. 19 (3) (2002) 169–172.

[10] I. P. Ivrissimtzis, N. A. Dodgson, M. A. Sabin, A generative classification of mesh refinement rules with lattice transformations, Comput. Aided Geom. Des. 21 (1) (2004) 99–109.

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

[12] H. Samet, Foundations of Multidimensional and Metric Data Structures, Morgan Kaufmann Publishers Inc., 2005.

[13] D. Zorin, P. Schröder, W. Sweldens, Interactive multiresolution mesh editing, in: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH ’97,

ACM Press/Addison-Wesley Publishing Co., 1997, pp. 259–268.

[14] J. Bolz, P. Schröder, Rapid evaluation of catmull-clark subdivision surfaces, in: Proceedings of the seventh international conference on 3D Web technology, Web3D '02, 2002, pp. 11–17.

[15] M. Bunnell, Adaptive tessellation of subdivision surfaces with displacement mapping, In: GPU Gems 2, Vol. 2, Addison-Wesley, 2005.

[16] L.-J. Shiue, I. Jones, J. Peters, A realtime gpu subdivision kernel, ACM Trans. Graph. 24 (2005) 1010–1015.

[17] L.-J. Shiue, J. Peters, A pattern-based data structure for manipulating meshes with regular regions, in: Proceedings of Graphics Interface 2005, GI '05, 2005, pp. 153–160.

[18] A. Mahdavi-Amiri, F. Samavati, Connectivity maps for subdivision surfaces, in: GRAPP/IVAPP, 2012, pp. 26–37.

[19] E. J. Stollnitz, T. D. Derose, D. H. Salesin, Wavelets for computer graphics: theory and applications, Morgan Kaufmann Publishers Inc., 1996.

[20] L. Olsen, F. Samavati, R. Bartels, Multiresolution for curves and surfaces based on constraining wavelets, Computers and Graphics 31 (3) (2007) 449 – 462.

[21] L. J. Olsen, F. F. Samavati, A discrete approach to multiresolution curves and surfaces, in: Proceedings of the 2008 International Conference on Computational Sciences and Its Applications, IEEE Computer Society, 2008, pp. 468–477.

[22] M. Bertram, Biorthogonal loop-subdivision wavelets, Computing 72 (1-2) (2004) 29–39.

[23] L. Olsen, Constraining Wavelets for Multiresolution, Canadian theses, University of Calgary (Canada), 2006.

[24] P. Kraemer, D. Cazier, D. Bechmann, Extension of half-edges for the representation of multiresolution subdivision surfaces, The Visual Computer 25 (2009) 149–163.

[25] J. Peters, U. Reif, The simplest subdivision scheme for smoothing polyhedra, ACM Trans. Graph. 16 (4) (1997) 420–431.

[26] J. Petersen, Die theorie der regulren graphs, Acta Mathematica 15 (1891) 193–220.

[27] T. C. Biedl, P. Bose, E. D. Demaine, A. Lubiw, Efficient algorithms for petersen's matching theorem, Journal of Algorithms 38 (1) (2001) 110 – 134.

[28] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, Practical quad mesh simplification, CG Forum (Eurographics (2010) 407–418.

[29] M. Tarini, E. Puppo, D. Panozzo, N. Pietroni, P. Cignoni, Simple quad domains for field aligned mesh parametrization, in: Proceedings of the 2011 SIGGRAPH Asia Conference, SA '11, ACM, New York, NY, USA, 2011, pp. 142:1–142:12.

[30] N. Pietroni, M. Tarini, P. Cignoni, Almost isometric mesh parameterization through abstract domains, IEEE Transactions on Visualization and Computer Graphics 16 (4) (2010) 621–635.

[31] E. Praun, W. Sweldens, P. Schröder, Consistent mesh parameterizations, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01, 2001, pp. 179–184.

[32] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, W. Stuetzle, Piecewise smooth surface reconstruction, in: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1994, pp. 295–302.

[33] T. DeRose, M. Kass, T. Truong, Subdivision surfaces in character animation, in: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, SIGGRAPH '98, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998, pp. 85–94.

[34] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle, Multiresolution analysis of arbitrary meshes, in: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, SIGGRAPH '95, ACM, New York, NY, USA, 1995, pp. 173–182.

[35] F. F. Samavati, R. H. Bartels, Multiresolution curve and surface representation: reversing subdivision rules by least-squares data fitting, Computer Graphics Forum 18 (1999) 97–119.

[36] R. H. Bartels, F. F. Samavati, Reversing subdivision rules: Local linear conditions and observations on inner products, Journal of Computational and Applied Mathematics 119 (1999) 29–67.

[37] J. Sadeghi, F. F. Samavati, Smooth reverse loop and catmull-clark subdivision, Graphical Models 73 (5) (2011) 202 – 217.

[38] J. Sadeghii, F. Samavati, Local fairing with local inverese, in: Proceedings of Graphics Interface 2013, 2013, pp. 117–124.

[39] D. Panozzo, E. Puppo, Implicit hierarchical quad-dominant meshes, Comput. Graph. Forum 30 (6) (2011) 1617–1629.

[40] H.-R. Pakdel, F. F. Samavati, Incremental adaptive loop subdivision, in: Proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2004), Vol. 3045 of Lecture Notes in Computer Science, Springer, 2004, pp. 237–246.

[41] J. Claes, K. Beets, F. Van Reeth, A corner-cutting scheme for hexagonal subdivision surfaces, in: Proceedings of the Shape Modeling International 2002 (SMI'02), SMI '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 13–.