An Easy Way of Detecting Subdivision Connectivity in a Triangle Mesh

Kai Hormann

Computer Graphics Group, University of Erlangen

Am Weichselgarten 9, 91058 Erlangen, Germany

hormann@cs.fau.de

Abstract

In this paper we discuss how a given triangle mesh can be analysed in order to decide whether it has subdivision connectivity or not. Our subdivision connectivity detection utilizes the fact that almost all vertices of a triangle mesh with subdivision connectivity are regular and that the remaining irregular vertices are vertices of the coarsest level. These vertices can therefore be used as seed points of a region growing algorithm that generates the entire base mesh from which the given mesh was refined. Once all the hierarchy levels of the given mesh are detected we reorder its vertices such that all the connectivity information of the given mesh is encoded in this vertex order and the connectivity of the base mesh only. This enables us to represent the given mesh as another triangle mesh with the same number of vertices but considerably fewer triangles and store it compactly using a standard file format, such as VRML.

1. Introduction and Related Work

Due to their simplicity, flexibility, and the fact that they are widely supported by modern graphics hardware, triangle meshes have become a standard 3D surface representation in many applications such as manufacturing, entertainment, architecture, and medicine.

A triangle mesh is usually represented by two components. Firstly, the geometric information, given as a set of vertices, and secondly, the topological or connectivity information, which tells how the vertices are connected. The latter is normally represented as a set of triangles whose corners are indices of vertices but other, e.g. edge-based data structures exist [1, 9].

A special class of triangle meshes are those with subdivision connectivity. These meshes have a special hierarchical structure that makes them a common tool in modelling and animation [11]. Their connectivity derives from the connectivity of a base mesh by iteratively refining the triangles by applying a 1-to-4-split, and their geometry, the vertex coordinates, can be determined by either a subdivision operator [6, 2], a remeshing procedure [3, 5, 4], or manually with a modelling package.

It is obvious that for triangle meshes of this special kind all the connectivity information can be encoded in the order of its vertices and the topological information of the base mesh. A subdivision connectivity mesh can therefore be represented as another triangle mesh with the same number of vertices but considerably fewer triangles and stored compactly using a standard file format, such as VRML's *IndexedFaceSet* rather than some special file format. This increases the interchangeability of the data and has the additional advantage that loading the mesh without decoding gives at least the base mesh.

Our method consists of two steps. First, we detect whether a given mesh has subdivision connectivity and of how many hierarchy levels it consists. Another algorithm for detecting subdivision connectivity has been proposed by Taubin in [7, 8] which is based on the concept of the *covering surface* in Algebraic Topology. He constructs the covering mesh of a given triangle mesh which is identical to the next coarser level if the mesh has subdivision connectivity. The method we propose here is somewhat simpler. It is based on the observation that most of the vertices of a triangle mesh with subdivision connectivity are regular and all other vertices are guaranteed to be vertices of the next coarser level. Taking the irregular vertices as seed points, we run a region growing algorithm that finds all vertices and triangles of the coarser level and also recognizes if the mesh does not have subdivision connectivity. In the second step we order the vertices of the given mesh in a generic way such that its connectivity can be reconstructed from the topological information of the base mesh only afterwards.

After introducing our notation and some simple facts about triangle meshes in Section 2 we discuss the two steps of our method in detail and give pseudocode for the detection as well as the reconstruction algorithm in Sections 3, 4 and 5. Section 6 summarizes the results and Section 7 suggests potential future work.

2. Triangle Meshes

2.1. Notation and Terminology

Let $V = \{v_0, \ldots, v_{|V|-1}\}$ be a set of $|V|$ vertices $v_i \in \mathbb{R}^3$ and $T = \{t_0, \ldots, t_{|T|-1}\}$ be a set of $|T|$ triangles, where each triangle is represented as a triplet of vertex indices, $t_i = (t_0^i, t_1^i, t_2^i)$ with $t_k^i \in \{0, \ldots, |V| - 1\}$. This *topological* representation of a triangle corresponds to the *geometric* realization as the convex hull of its corners, $\hat{t} = [v_{t_0}, v_{t_1}, v_{t_2}]$.

We call the pair $M = (V, T)$ a *triangle mesh* if the intersection $\hat{t}_i \cap \hat{t}_j$ of any two different triangles $t_i, t_j \in T$ is either empty, a common vertex, or a common edge and the union of all triangles, $\hat{M} = \bigcup_{t \in T} \hat{t}$, which is the *geometric realization* of M, is a two-manifold (with boundary).

The edges of a triangle mesh are defined by the edges of the triangles and we say that a triangle is *adjacent* to its three edges. We distinguish between *interior edges* with exactly two adjacent triangles and *boundary edges* with only one adjacent triangle and let E_I, E_B, and $E = E_I \cup E_B$ be the sets of interior, boundary, and all edges. Furthermore, we call the endpoints of the boundary edges *boundary vertices* and let V_B and $V_I = V \setminus V_B$ be the sets of *boundary* and *interior vertices*.

For any vertex v we call all those edges and triangles that contain this vertex *adjacent* to v. If two vertices v and w are the endpoints of some edge we say that v is a *neighbour* of w and vice versa. Similarly, a triangle t is a *neighbour* of a triangle s, if t and s are adjacent to a common edge. For any vertex v we call the collection of triangles that are adjacent to v the *neighbourhood* of v and the union of neighbourhoods of v's neighbours the *2-neighbourhood* of v.

The number $\eta(v)$ of neighbours of a vertex v is called the *valency* of v. Note that $\eta(v)$ equals the number of adjacent triangles if v is an interior vertex and exceeds this number by one for boundary vertices. Interior vertices v_I with $\eta(v_I) = 6$ and boundary vertices v_B with $\eta(v_B) = 4$ are called *regular*, all others *irregular*. For regular vertices we further say that two neighbours are *opposite* to each other if there are exactly two other neighbours between them. In other words, if we order the neighbouring vertices of v clock- or counterclockwise from 1 to $\eta(v)$, the indices of two opposite neighbours always differ by 3. Note that for regular boundary vertices only the two boundary neighbours are opposite to each other.

The union of all boundary edges is a set of closed simple *boundary polygons* and we denote the number of boundary polygons by $B \geq 0$.

The *genus* $G \geq 0$ of a triangle mesh is the number of *handles* of $\hat{M}$. E.g., if $\hat{M}$ is topologically equivalent to a sphere or a disk, then $G = 0$, if $\hat{M}$ is topologically equivalent to a torus, then $G = 1$.

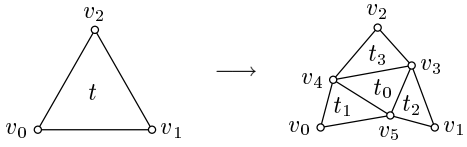


Figure 1. Refinement of a triangle.

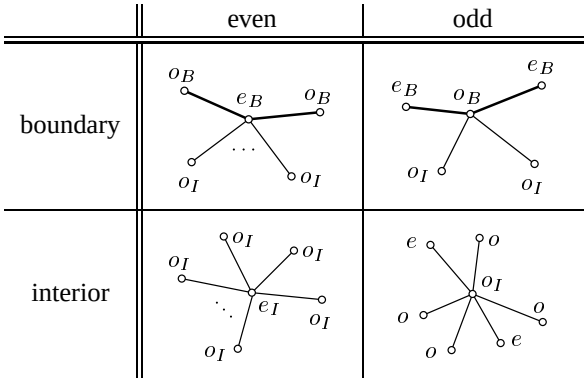


Figure 2. Neighbours of the vertices of a refined mesh. Even and odd vertices are denoted by e and o, the subscript B or I indicates boundary or interior vertices.

2.2. Subdivision Connectivity

Suppose we are given a triangle mesh $M = (V, T)$. If we add $|E|$ new vertices $W = \{v_{|V|}, \ldots, v_{|V|+|E|-1}\}$ to V, associate each edge of M with one of these new vertices, and split every triangle t into four new triangles t_0, t_1, t_2, t_3 as in Figure 1, then we obtain a new triangle mesh $M' = (V', T')$ with $V' = V \cup W$ and $T' = \bigcup_{t \in T} \{t_0, t_1, t_2, t_3\}$.

An operator that transforms a mesh M into a new mesh M' this way is called a *refinement operator* and M' is a *refinement* of M or simply a *refined mesh*. E.g., the subdivision schemes in [2, 10] are refinement operators of this kind.

Let $M' = (V', T')$ be a refinement of $M = (V, T)$, then we call $V \subset V'$ the *even* and $V' \setminus V$ the *odd* vertices of M'. Due to the special structure of the refinement operator we have the following propositions.

Proposition 2.1 *For the vertices of a refined mesh holds (cf. Figure 2):*

1. *all odd vertices are regular and have exactly two even neighbours that are opposite to each other,*
2. *all neighbours of even vertices are odd,*
3. *interior even vertices have interior neighbours only.*

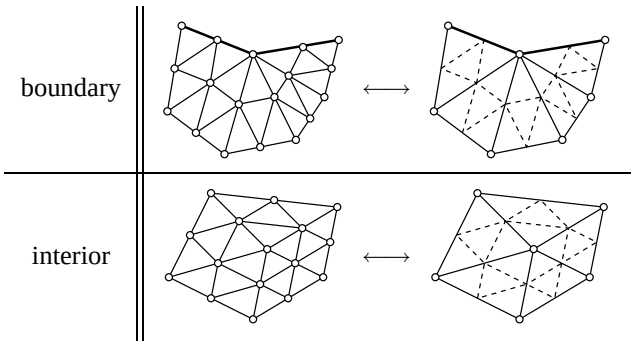


Figure 3. Connectivity of the 2-neighbourhood of a vertex with regular neighbours.

```
procedure detectSubdivisionConnectivity (M)
  M^0 = M,   m = 0
  do
    M' = coarsen (M^0)
    if (M' ≠ ∅)
      for j = m, ..., 0
        M^{j+1} = M^j
      M^0 = M'
      m = m + 1
  while (M' ≠ ∅)
  return M^0, ..., M^m
```

Figure 4. Pseudocode for detecting subdivision connectivity of a triangle mesh.

Proposition 2.2 *The 2-neighbourhood of a boundary vertex is a refinement if and only if all neighbours are regular and the 2-neighbourhood of an interior vertex is a refinement if and only if all neighbours are regular and interior vertices (cf. Figure 3).*

A triangle mesh M has *subdivision connectivity* of *level* $m \geq 1$, if it is the last element of a *hierarchical sequence* of meshes, $M^0, M^1, \ldots, M^m = M$, where each M^i is a refinement of M^{i-1} and the *base mesh* M^0 is not a refined mesh.

2.3. Simple Facts

The *Euler formula* for general polyhedra is

$$V - E + F = 2 - 2G - B, \tag{1}$$

where V, E, F, and B are the numbers of vertices, edges, faces, and boundary polygons and G is the genus of the polyhedron. For triangle meshes we observe

$$3|T| = 2|E_I| + |E_B| \tag{2}$$

by counting the edges of the triangles. Since every boundary edge is adjacent to two boundary vertices and every boundary vertex has exactly two adjacent boundary edges we further have

$$|E_B| = |V_B| \tag{3}$$

and rewrite (1) as

$$|V| - \tfrac{1}{2}(|V_B| + |T|) = 2 - 2G - B. \tag{4}$$

If all vertices of a triangle mesh are regular, we call it *totally regular* and conclude

$$6|V_I| + 3|V_B| = 3|T|$$

by counting the triangles adjacent to the vertices and (4) simplifies to

$$0 = 2 - 2G - B.$$

The only positive solutions to this *Diophantic equation* are $G = 1$, $B = 0$ and $G = 0$, $B = 2$, i.e. totally regular triangle meshes are either topologically equivalent to a torus without boundaries or a cylinder.

Let M' be a refinement of M. Then we know

$$\begin{aligned} |V'| &= |V| + |E|, \\ |E'| &= 2\,|E| + 3\,|T|, \\ |T'| &= 4\,|T|, \end{aligned}$$

and conclude for a mesh M with subdivision connectivity of level $m \geq 1$,

$$\begin{aligned} |V| &= |V^0| + (2p-1)|E^0| + (2p^2 - 3p + 1)|T^0|, \\ |E| &= 2p\,|E^0| + 3(2p^2 - p)|T^0|, \\ |T| &= 4p^2\,|T^0|, \end{aligned} \tag{5}$$

where $p = 2^{m-1}$. Using (2) and (3) we rewrite (5) as

$$|V| = |V^0| + \tfrac{1}{2}\big[(2p-1)|V_B^0| + (4p^2 - 1)|T^0|\big]. \tag{6}$$

3. Detecting Subdivision Connectivity

The situation we consider now is that we are given a triangle mesh M and want to decide whether M has subdivision connectivity of a certain level $m \geq 1$ or not. If the answer to this question is positive we would further like to know the hierarchical sequence $M^0, \ldots, M^m$.

We attack this problem by developing an algorithm for the process of inverse refinement, which we call *coarsening*. If a triangle mesh M is a refinement of a coarse mesh M', then M' is created and returned by this algorithm while an empty mesh $\emptyset$ is returned otherwise. Given this procedure for coarsening a mesh we solve the problem of detecting subdivision connectivity with the algorithm in Figure 4.

```
procedure coarsen (M)
    V' = ∅,   V'' = ∅,   T' = ∅
    # trivial reject
    if (|T| mod 4 ≠ 0) return ∅
    # set seed points
    let q be an empty queue
    for i = 0, ..., |V| − 1
        if (v_i is irregular)
            push v_i to q
            V' = V' ∪ v_i
    # region growing
    while q is not empty
        pop v from q
        let w_1, ..., w_η(v) be the ordered neighbours of v
        # check local refinement structure
        for i = 1, ..., η(v)
            if (w_i is irregular) return ∅
            if (w_i ∈ V') return ∅
            if (w_i ∈ V_B and v ∈ V_I) return ∅
            let v_i be the neighbour of w_i opposite to v
        # add new seed points and create coarse triangles
        for i = 1, ..., η(v)
            if (v_i ∉ V')
                push v_i to q
                V' = V' ∪ v_i
            if (i < η(v) or v ∈ V_I)
                j = i mod η(v) + 1
                if (v_i ∉ V'' and v_j ∉ V'')
                    T' = T' ∪ (v, v_i, v_j)
        V'' = V'' ∪ v
    # return coarse mesh
    return M' = (V', T')
```

Figure 5. Pseudocode for coarsening a triangle mesh.

Our method for coarsening triangle meshes is based on the observation that for refined meshes M the vertices V' of the coarse mesh M' are by definition the even vertices of M. Once these vertices are found, creating the triangles of M' is simple.

The algorithm starts by assuming that the given mesh M is a refined mesh. According to Proposition 2.1 all irregular vertices of M are even and we use them as seed points of a region growing algorithm that searches for even vertices until all of them are found or a configuration was detected that contradicts the assumption of M being a refined mesh.

The region growing works as follows. Suppose v is an even vertex and $w_1, \ldots, w_{\eta(v)}$ are its neighbours. From Proposition 2.1 we know that all w_i are regular and odd vertices and that the neighbours v_i of w_i that are opposite to v are even vertices. If the algorithm finds one of the neighbours w_i of v to contradict the properties of Proposition 2.1, i.e. w_i is either

- irregular,
- an even vertex, or
- a boundary vertex while v is an interior vertex,

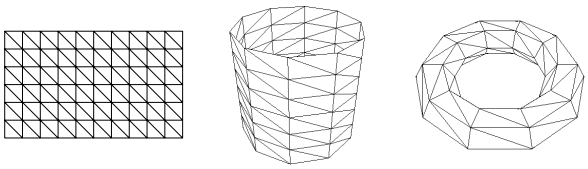
then we conclude from Proposition 2.2 that M is not a refined mesh. Otherwise we take the neighbouring even vertices v_i as new seed points unless they have already been seed points before and proceed with the region growing until no new seed points can be found.

Furthermore we know that (v, v_i, v_j) is a triangle of the coarse mesh M' if and only if (v, w_i, w_j) is a triangle of M and we can easily record all these coarse triangles. In order to prevent multiple occurrences of triangles we additionally have to check if one of the even vertices v_i or v_j has been processed before, because one of the triangles (v_i, v_j, v) or (v_j, v, v_i) would have already been collected in that case.

If the region growing terminates without having encountered a contradiction to the assumption of M being a refined mesh, then M is indeed a refined mesh and the algorithm returns the coarse mesh M' consisting of all even vertices and coarse triangles that were found during the region growing. Figure 5 shows the pseudocode for this algorithm and Figure 16 shows an example.

Although this algorithm is always guaranteed to work due to Propositions 2.1 and 2.2, we would like to add a few remarks on its limitations.

Firstly, the given mesh M could be totally regular, so that there is nothing to do for the algorithm. As we have discussed in Section 2.3, such a mesh is topologically equivalent to a torus or a cylinder. Moreover we state that such a mesh has a grid structure as shown in Figure 6 (a) with the left and right side of this grid glued together for a cylinder (b) and the top and bottom side glued together for a torus (c) in addition. It is easy to see that such a mesh has subdivision connectivity of level m if and only if the number of rows and columns of the grid are both divisable by 2^m, which can be checked directly.



(a) (b) (c)

Figure 6. Connectivity of totally regular triangle meshes.

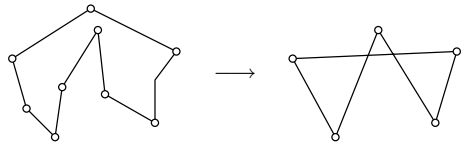


Figure 7. Coarsening may lead to intersections.

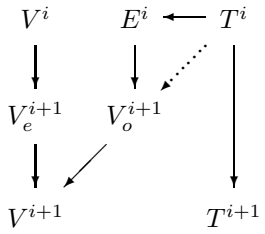


Figure 8. The order of vertices and triangles of level i defines an order of vertices and triangles of level $i+1$.

Secondly, we may encounter different kinds of degeneracies. If one of the boundary polygons of M has 4 vertices, the coarse mesh will contain two boundary edges that are glued together. It could also happen that the triangles of the coarse mesh intersect as shown in Figure 7.

4. Ordering the Vertices

Once we have analysed the given mesh M and found its level of subdivision connectivity as well as the hierarchical sequence $M^0, \ldots, M^m$ whose last element it is, we exploit this special structure to represent M in a very compact way. In fact, all we need to know in order to be able to reconstruct M is the geometric information of M, contained in the set of vertices V, and the topological information of the base mesh, represented by the triangles in T^0, provided that the vertices in V are in a special order.

We will now discuss what kind of order that is, give an algorithm for generating this order from the hierarchical sequence $M^0, \ldots, M^m$, and explain how to use it for the reconstruction of M. Our method is based on the observation that the order of vertices V^i and triangles T^i of level i implicitly defines an order of the vertices and triangles of the next level $i+1$ (cf. Figure 8), e.g. in the following way.

The even vertices V_e^{i+1} of M^{i+1} are by definition the vertices of M^i and inherit V^i's order. The odd vertices V_o^{i+1} are each associated with one of the edges of M^i and can thus be ordered in the same way. So far we have not defined an order of the edges E^i, but it is possible to derive one from the order of triangles T^i.

```
procedure orderEdges (T^i)
  E^i = ∅
  for j = 0, ..., |T^i| − 1
    let e_0, e_1, e_2 be the edges of T_j^i
    for k = 0, ..., 2
      if (e_k ∉ E^i)
        E^i = E^i ∪ e_k
  return E^i
```

Figure 9. Pseudocode for ordering edges.

```
procedure orderTriangles (M^0, ..., M^m)
  for i = 0, ..., m − 1
    T' = ∅
    for j = 0, ..., |T^i| − 1
      let v_0, v_1, v_2 be the vertex indices of T_j^i
      let w_0, w_1, w_2 be the indices of the odd vertices
        in M^{i+1} corresponding to the edges of T_j^i
      T^{i+1} = T^{i+1} ∪ (w_0, w_1, w_2) ∪ (v_0, w_2, w_1)
                        ∪ (v_1, w_0, w_2) ∪ (v_2, w_1, w_0)
    M^{i+1} = (V^{i+1}, T')
  return M^0, ..., M^m
```

Figure 10. Pseudocode for ordering triangles.

```
procedure orderVertices (M^0, ..., M^m)
  V' = ∅
  # visit all triangles of M^0
  for j = 0, ..., |T^0| − 1
    let v_0, v_1, v_2 be the vertices of T_j^0
      for k = 0, ..., 2
        if (v_k ∉ V')
          V' = V' ∪ v_k
  # visit only center triangles of M^1, ..., M^m
  for i = 1, ..., m
    for j = 0, ..., |T^{i−1}| − 1
      let v_0, v_1, v_2 be the vertices of T_{4j}^i
        for k = 0, ..., 2
          if (v_k ∉ V')
            V' = V' ∪ v_k
  return V'
```

Figure 11. Pseudocode for ordering vertices.

Suppose we loop over the ordered sequence of triangles $T_j^i \in T^i$ and label each edge of T_j^i with the value of a counter unless this edge has already been labelled before. If we keep incrementing the counter during this process, then the label values that were assigned to the edges define an order of the edges in E^i. Figure 9 gives the pseudocode for this little routine and shall familiarize the reader with our notation of ordered sequences which we will sub-

```
procedure updateTriangles (V', T)
    for i = 0, ..., |T| − 1
        let v0, v1, v2 be the vertices of Ti
        let w0, w1, w2 be the indices of v0, v1, v2 in V'
        Ti = (w0, w1, w2)
    return T
```

Figure 12. Pseudocode for updating the vertex indices of the triangles.

```
procedure convertTriangleMesh (M)
    M^0, ..., M^m = detectSubdivisionConnectivity (M)
    M^0, ..., M^m = orderTriangles (M^0, ..., M^m)
    V' = orderVertices (M^0, ..., M^m)
    T^0 = updateTriangles (V', T^0)
    return (V', T^0)
```

Figure 13. Pseudocode for converting a triangle mesh to its compact representation.

sequently use. The empty sequence is denoted by $\emptyset$ and we write $A = A \cup a$ for appending a to the sequence A.

The triangles T^{i+1} of M^{i+1} are finally ordered as Figure 1 suggests. For each triangle $T^i_j = (v_0, v_1, v_2)$ we order the corresponding refined triangles by first taking the *center* triangle that connects the odd vertices, followed by the other three that are adjacent to v_0, v_1, and v_2, and order these groups of refined triangles in the same way as the triangles in T^i.

We conclude that there exists an order of the vertices V and triangles T of the given mesh M that is implicitly defined by the order of the vertices V^0 and triangles T^0, and that we can reconstruct M from V and T^0 if the vertices of V are given in that order.

This vertex order is generated in two steps. Firstly, we rearrange the triangles on each level as explained above, starting with the triangles T^0 and propagating the induced order through the levels of the hierarchical sequence. Secondly, we take the vertices V^0 and iteratively append the odd vertices $V^1_o, \ldots, V^m_o$ of the refined meshes in the correct order. The pseudocode for these two algorithms is given in Figures 10 and 11. Note that we do not need to order the edges E^i on each level explicitly as we visit the odd vertices of M^{i+1} in the correct order by looping over the center triangles of M^{i+1}, i.e. we take the short cut in Figure 8 from T^i to V^{i+1}_o.

After the vertices have been ordered we have to update the vertex indices of the triangles in T^0 such that they match up with the new order. This is done by the procedure given in Figure 12. Figure 13 finally summarizes the entire process of converting a triangle mesh with subdivision connectivity to its compact representation.

```
procedure reconstruct (V, T^0)
    # reconstruct M^0 from V and T^0
    let n be the greatest vertex index of all T ∈ T^0
    V^0 = {v0, ..., vn} ⊂ V
    M^0 = (V^0, T^0)
    # determine level of subdivision connectivity
    m = 0,   n = 0,   p = 1
    while (n < |V|)
        n = |V^0| + 1/2 [(2p − 1)|V^0_B| + (4p^2 − 1)|T^0|]
        m = m + 1,   p = 2p
    if (n ≠ |V|) return M^0
    # reconstruct hierarchical sequence
    for i = 0, ..., m − 1
        T^{i+1} = ∅,   c = |V^i|
        for j = 0, ..., |T^i| − 1
            let v0, v1, v2 be the vertex indices of T^i_j
            let n0, n1, n2 be the indices of T^i_j's neighbour
                triangles
            for k = 0, ..., 2
                if (n_k > j) or (n_k = −1)
                    w_k = c,   c = c + 1
                else
                    get w_k from T^{i+1}_{4n_k}
            T^{i+1} = T^{i+1} ∪ (w0, w1, w2) ∪ (v0, w2, w1)
                              ∪ (v1, w0, w2) ∪ (v2, w1, w0)
        V^{i+1} = {v0, ..., v_{|V^i|+|E^i|−1}} ⊂ V
        M^{i+1} = (V^{i+1}, T^{i+1})
    return M = M^m
```

Figure 14. Pseudocode for reconstructing triangle meshes with subdivision connectivity.

5. Reconstructing Subdivision Connectivity

The reconstruction of M from V and T^0 proceeds in a similar way (cf. Figure 14). First, we reconstruct the base mesh M^0 from T^0 and the first $n+1$ vertices of V, where n is the greatest vertex index found in the triangles of T^0. Second, we use (6) to derive the level m of subdivision connectivity from the number $|V|$ of given vertices. Third, we build the hierarchical sequence $M^0, \ldots, M^m$ by iteratively constructing the refined triangles T^{i+1} from the coarse triangles T^i. The indices of the odd vertices that are required during this refinement step are determined by a counter that is initially set to the index of the first odd vertex of level $i+1$ and incremented each time an edge of T^i is visited for the first time.

Note that again we do not need to deal with edges explicitly but rather use the fact that an edge is visited for the first time if the other adjacent triangle has not been refined yet, i.e. its index is greater, or if it is a boundary edge. The algo-

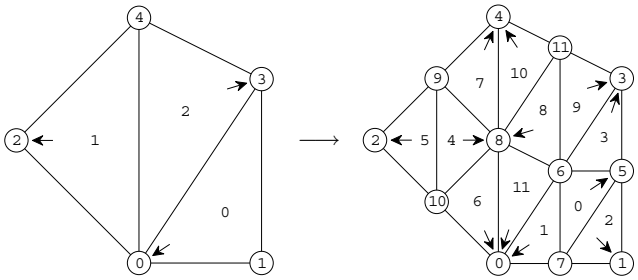


Figure 15. One refinement step.

rithm therefore requires to keep for each triangle the indices of the neighbouring triangles. At the boundary, the index of the non-existent adjacent triangle is -1 by definition.

In order to better understand this mechanism, let us consider a simple example. In Figure 15 the triangle mesh M^0 with 5 vertices and 3 triangles shall be refined to a mesh M^1. Since the 5 vertices of M^0 are the even vertices of M^1, the index of the first odd vertex of M^1 is 5. While refining the first coarse triangle $(0, 1, 3)$ its edges are visited for the first time and therefore the odd vertices with indices 5, 6, and 7 are used for creating the refined triangles $(5, 6, 7), (0, 7, 6), (1, 5, 7)$, and $(3, 6, 5)$. The same applies to the refinement of the second coarse triangle, but when the third coarse triangle $(3, 4, 0)$ is being processed, two of its edges are visited for the second time. The vertex indices of the odd vertices that correspond to these edges will then be derived from the center triangles 0 and 4 that were refined from the adjacent coarse triangles 0 and 1.

6. Implementation and Results

We have implemented and tested the described algorithms for detecting and reconstructing triangle meshes with subdivision connectivity using C++ as a programming language and an extended indexed triangle data structure. This data structure contains for each triangle not only the indices of its vertices but also the indices of the neighbouring triangles and for each vertex the index of one of its adjacent triangles.

Figure 16 shows the data set of a cat (j) that has been refined twice using Loop's subdivision operator [6] to give the subdivision connectivity mesh M^2 in (a). The irregular vertices of that mesh are visualized as green spheres in (b) and the sequence (c)–(e) shows how the region growing algorithm builds the coarser level. Note that in these pictures the black wireframe represents the edges of M^2 while the triangles are the triangles of the coarser mesh M^1 that is being generated. In (f)–(i) this process is repeated for M^1, thereby reconstructing the base mesh M^0 (j) which the algorithm detects as not being a refined mesh.

Figure 17 shows different levels of the hierarchical sequence of a triangle mesh with subdivision connectivity of level 7. This triangle mesh was generated with the remeshing method from [4] and our algorithm successfully recognized all hierarchy levels.

7. Conclusions and Future Work

We have presented in this paper a novel and simple method for detecting subdivision connectivity in a given triangle mesh and have discussed how these meshes are stored compactly using a standard file format and how to reconstruct them again. This compact representation of the given mesh consists of its reordered vertices and the connectivity information of the base mesh of the hierarchical sequence whose last element the given mesh is.

In future work we plan to develop similar techniques to detect and encode triangle meshes with adaptive subdivision connectivity.

References

[1] S. Campagna, L. Kobbelt, and H.-P. Seidel. Directed edges – a scalable representation for triangle meshes. *Journal of Graphics Tools*, 3:1–11, 1999.

[2] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9:160–169, 1990.

[3] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *ACM Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 173–182, 1995.

[4] K. Hormann, U. Labsik, and G. Greiner. Remeshing triangulated surfaces with optimal parameterizations. *CAD*, 33:779–788, 2001.

[5] A. Lee, W. Sweldens, P. Schröder, L. Coswar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *ACM Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 95–104, 1998.

[6] C. Loop. Smooth sudivsion surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[7] G. Taubin. Is this a quadrisected mesh? In *Proceedings of ACM Solid Modeling 2001*, pages 261–266, 2001.

[8] G. Taubin. Detecting and reconstructing subdivision connectivity. *Visual Computer*, 2002.

[9] K. Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics and Applications*, 5:12–40, 1985.

[10] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *ACM Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 189–192, 1996.

[11] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *ACM Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 259–268, 1997.

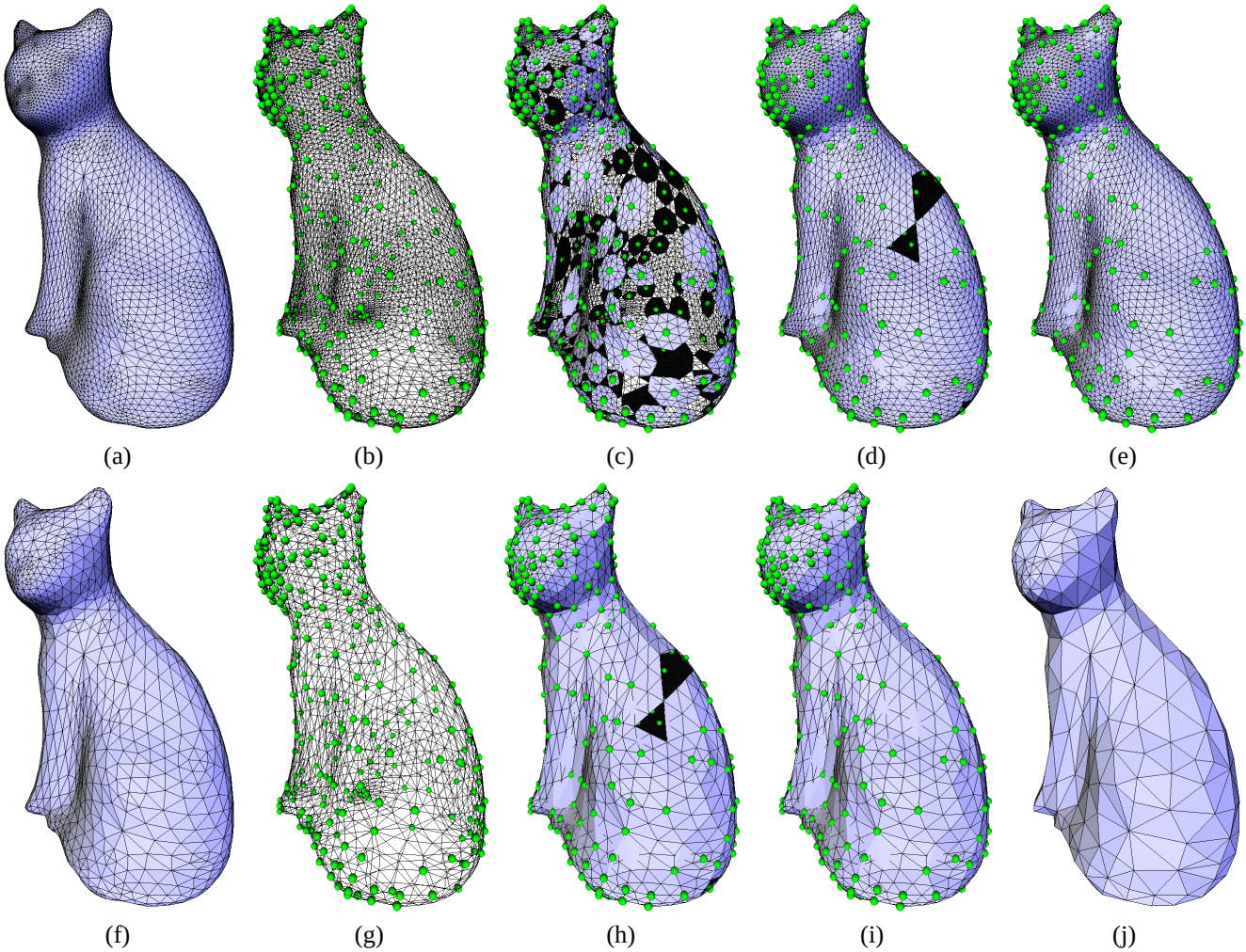


Figure 16. Detecting the subdivision connectivity of a given triangle mesh M^2 (a). The irregular vertices of M^2 are found (b) and taken as seed points of a region growing algorithm that determines the triangles of the coarser mesh M^1 (c)–(e). This process is repeated for M^1 (f)–(i) and finally gives the base mesh M^0 (j).

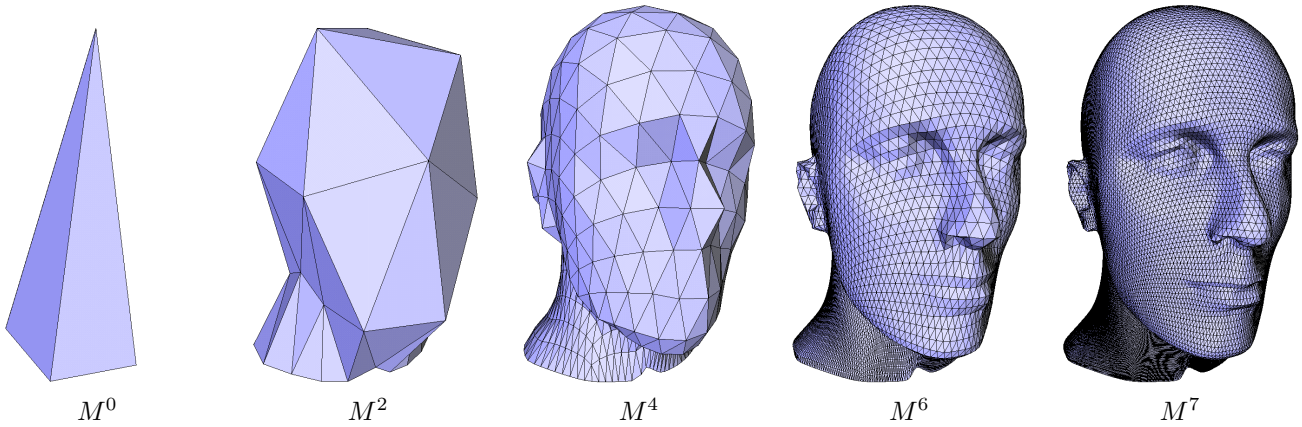


Figure 17. Different levels of the hierarchical sequence of a remesh of the mannequin head.