Interactive Editing of Multiresolution Meshes

Frutuoso G. M. Silva and Abel J. P. Gomes
IT - Networks and Multimedia Group
Department of Computer Science and Engineering
University of Beira Interior, Portugal
{fsilva,agomes}@di.ubi.pt

Abstract

This paper presents a new approach for interactive editing of multiresolution meshes. Editing operations (e.g. scaling and rotation) are confined to mesh regions of interest, called sub-meshes. Also, intuitive deformation operations (e.g. tapering and twisting) as required for animation systems can be used as well, allowing the user to have control on the local deformations of mesh objects. For that, each sub-mesh is defined interactively by picking up the cells of its frontier. Alternatively, a sub-mesh can be delimited by the shortest path of edges that approximates the intersection between a mesh and a given cutting plane. In order to speed up editing operations, the mesh is first simplified, the target sub-mesh is edited, and finally the whole mesh is refined back to its original resolution. This is particularly important for large meshes.

1. Introduction

Meshes are useful to represent surfaces in computer graphics and geometric modeling. Interactive editing and deformation of meshes is very important for a variety of applications, ranging from mechanical design to animation and virtual reality. For example, deforming a parametric surface can be done by moving at least one of its control points. In case of a mesh surface, we can get a more intuitive control on deformations by directly pulling a vertex. Alternatively, as proposed in this paper, editing operations can be confined to sub-meshes. These operations are geometric transformations such as, for example, scaling, rotation, tapering and twisting. When applied to a sub-mesh, they change the shape of a mesh locally without changing its overall shape, as required for many animation applications and game technologies.

Is is important to note that these editing operations are carried out in the context of multiresolution meshes. Editing a multiresolution mesh involves a three basics steps. First, we simplify a mesh to a desirable level of detail reducing the number of cells. Then, the mesh is subdivided into sub-meshes in such a way that editing operations can be applied to them. Finally, the resulting mesh is refined back to its original resolution using the multiresolution scheme.

The structure of the paper is as follows. Section 2 describes the related work. Section 3 presents the multiresolution scheme used. Section 4 describes the interactive editing of sub-meshes. Some results appear in Section 5. At last, Section 6 draws some conclusions and discusses future work.

2. Related Work

In the last few years much research has been done in polygonal meshes, particularly in the multiresolution area. Interactive mesh editing is a recent and interesting area of research due to the increasing use of meshes in computer graphics.

Lee [15] proposed a new method for interactive editing of arbitrary meshes. Basically, the user specifies a rectangular editing area that is mapped onto a plane where the user can edit mesh vertices. This method is constrained by the fact that the editing area must be homeomorphic to a 2D disk. However this method does not handle vertices directly. Instead, it manipulates vertex images in the mapping area in a way similar to control points of a parametric surface.

Kanai *et al.* [9] presented a new mesh modeling scheme, called mesh fusion. This scheme is based on the 3D metamorphosis that permits combining two parts of different meshes in a new mesh. Initially, a correspondence is established between two meshes by mapping each point of the source mesh to a point of the target mesh. Then, it generates a smooth transition by interpolating corresponding points.

Suzuki *et al.* [25] proposed a method for dragging a part of a triangular mesh, changing its position and orientation. This method is based on an adaptive re-meshing with topological changes.

Lee and Lee [16] developed a snake-based method for detecting features on a 3D triangular mesh. A geometric snake is an active contour model that slithers from its initial position specified by the user to a nearby feature while minimizing an energy function. Geometric snakes are similar to snakes used in image processing [3].

Madi and Walton [19] presented a new method for interactive selection and modification of polyhedral 3D objects. Their work takes advantage from a new data structure, called TLDS (Triangle Loop Data Structure), and a vertex pointer table to rapidly access the neighboring cells of a given vertex, allowing thus fast shape modification operations.

In the context of multiresolution, Zorin *et al.* [28] proposed the first editing tool for semi-regular meshes. This tool combines subdivision and smoothing algorithms to edit interactive multiresolution meshes.

Kobbelt *et al.* [12] generalized the previous multiresolution techniques to arbitrary triangle meshes. However, the multiresolution representation is created off-line. They introduced a fine-to-coarse geometric hierarchy by using the simple *Umbrella* algorithm.

Kobbelt *et al.* [13] extended the multiresolution hierarchies to unstructured triangle meshes, having presented a new method based on Phong-shading to calculate the detail coefficients. Similar approach was proposed by Guskov *et al.* [6], where they applied basic signal processing tools for irregular connectivity triangle meshes. Their subdivision and pyramid algorithms use well-known mesh simplification methods.

Kobbelt *et al.* [10] presented a new approach for multiresolution shape deformations, where the detail information is implicitly represented by the geometric difference between independent meshes. They also proposed a dynamic mesh representation that adapts the connectivity during the deformation operation in order to preserve a prescribed mesh quality. This operation causes distortion that can be reduced by dynamically restructuring the mesh.

Biermann *et al.* [2] proposed a set of intuitive cut-and-paste operations for multiresolution surfaces at interactive rates. However, it is difficult to generalize this approach to pasting regions that are not topologically equivalent to a subset of a plane.

More recently, Botsch and Kobbelt [4] proposed a new representation for multiresolution models which uses volume elements enclosed between the different resolution levels to encode the detail information. Keeping these volumes locally constant during an editing operation of the original surface leads to a natural behavior of the detail features.

In summary, several schemes for mesh editing and deformation have been proposed in the literature, some of which use multiresolution representations. However, editing operations are normally based on direct manipulation of vertices or control points [1, 8, 20]. Other editing techniques use axial deformation [14], deformation with lattices [18], and wire deformation [24]. For example, editing operations used by Kobbelt *et al.* [12, 13] are based on the definition of a region on which the user define a handle. When the user moves or modify the handle, the region surroundings of the mesh follow according to a constrained energy minimization principle.

3. Multiresolution Scheme

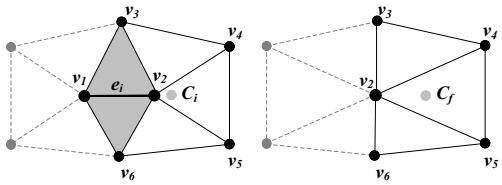
A multiresolution scheme usually provides a sequence of meshes representing a single object at different resolutions. It allows us to render a mesh at different levels of resolution, depending on its projective distance to a virtual viewer. However, unlike discrete LOD (Level-of-Detail) meshes, only a single mesh is in memory at a time. Simplifying or refining it gives rise to another mesh. Our multiresolution scheme was implemented on the AIF data structure [21], which has been re-designed to support sub-meshes [22].

Simplifying a polygonal mesh M_i consists of generating another mesh M_{i-1} with a less number of cells. The resulting mesh satisfies a given criterion, which is normally a measure of a maximum admissible error. This error may be implicitly given by a desired number of cells. There are some types of algorithms for simplifying a polygonal mesh, namely the algorithms based in the following operations: cell decimation, vertex clustering and edge collapse [17].

Collapsing an edge consists of contracting it and its bounding vertices into a single vertex. In our case, this vertex is the midpoint of the collapsing edge, but it is not mandatory to do so. Remarkably, this operation is invertible, which enables the construction of a multiresolution representation. The inverse operation, *vertex split*, is used for refining meshes.

Splitting a vertex into two vertices requires we provide the coordinates of the new vertex, as well as supplementary information. Normally, this information is stored during the edge collapse operation, allowing us to roll back to the original mesh.

In this paper, we use an edge collapsing operation for simplifying meshes. Given an edge bounded by the vertices v_1 and v_2, its collapse implies that all remaining edges incident at v_1 and v_2 are locked against further collapsing. This disables simplifying the mesh region around the collapsing edge again before completing the same simplification step over all the mesh. The next simplified mesh M_{i-1} is reached when no more collapses are possible, i.e. when all edges are locked or the remaining edges do not satisfy the simplification criterion. The simplification criterion is based on the planarity of the star of faces around the collapsing edge. Each simplification step removes about 45 %



a) Before collapsing edge. b) After collapsing edge.

Figure 1. Compute of the detail coefficients.

of faces. More details about our algorithm, including the simplification times, is available in [23].

To generate a fine-to-coarse hierarchy of meshes, we have to be able to find a coarser mesh M_{i-1} from a given a mesh M_i, as well as a set D of detail coefficients to allow us to roll back to M_i, i.e. M_i=DM_{i-1}. However, it is not clear how to use the detail coefficients in the refinement operation that outputs M_i, because we do not intend to come back to the original mesh, but an edited mesh that preserves the global shape of the object it represents.

Each step of the refinement operation (vertex split) consists of a topological operation (vertex insertion) and a geometric operation that places the re-inserted vertices at new positions. There is a need in repositioning the re-inserted vertices because editing operations change the position and shape of the mesh.

Note that the process of simplification-editing-refinement is more complicated that simplification-refinement as usual in multiresolution meshes (e.g. Progressive Meshes [7]), because we have to reconstruct the shape of a mesh after editing it.

The main difference between our approach and that one due to Kobbelt *et al.* [12, 13] lies in the method that computes the new position of the re-inserted vertices. They store the detail coefficients using an approximation for the removed points $\{p_i\}$. This approximation is based on a parameterization of the geometric constellation of each p_i. In fact, they need to solve a system to compute the coefficients of the parameterization that minimize some blending energy function. In contrast, our method uses a linear transformation (translation) to compute the new position of the re-inserted vertices. This linear function is based on the position of the vertices, before and after the editing process. The steps of our method to compute the new position of the re-inserted vertices are the following:

- For each vertex v bounding the collapsing edge, one computes the centroid of the polygon defined by the vertices of the star of faces incident at v, excluding the second vertex of the collapsing edge. For example, the centroid for v_2 in Figure 1 (a) is defined by the vertices v_3, v_4, v_5 and v_6. The centroid yields the geometric average of the position of a vertex. The centroid C_i of the polygon around v_2 is a detail coefficient.
- Conversely, in the vertex split operation of v_2, one computes the centroid C_f of the polygon defined by the vertices v_3, v_4, v_5, and v_6 (Figure 1 (b)). By taking the initial position centroid C_i and current or final centroid C_f associated to v_2, it is possible to define a linear transformation (translation). Note that, if the mesh has not been modified around v_2, C_i and C_f coincide, and the transformation is the identity.
- The linear transformation enables us to compute the position of the vertex to be re-inserted and also update the position of the splitting vertex.

The position of re-inserted vertices, in the refinement algorithm, is calculated from the current position of the neighboring vertices of the splitting vertex. Thus it is possible to edit a coarse mesh (or sub-mesh) by propagating the results of editing operations to other meshes in the hierarchy. So, with this strategy, we build an *interactive* editing tool for multiresolution meshes. On the contrary, the multiresolution representation used by Kobbelt *et al.* [12] needs an off-line pre-processing step, during which an incremental mesh decimation algorithm is applied and the detail coefficients are computed before editing the mesh.

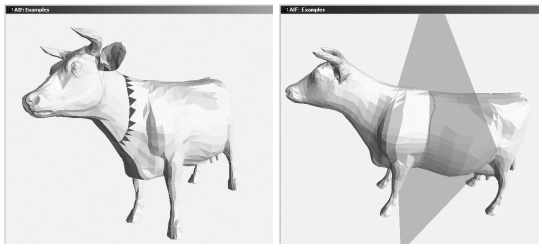
The next section presents some editing operations that we have been implemented for multiresolution meshes. Note that other editing techniques can be easily applied to sub-meshes using this multiresolution scheme.

4. Interactive Sub-mesh Editing

Kobbelt *et al.* [11] described the general requirements that a modeling tool should satisfy. They argued that a modeling tool should be intuitive, independent, and interactive. This means that sub-mesh editing should be easy to handle and independent of the mesh representation.

In this paper, editing a given multiresolution mesh can be described as sequence of the following steps:

- *Mesh simplification.* The mesh is simplified to a desirable level of detail to reduce the number of cells and ease the editing process.
- *Sub-mesh creation.* The mesh is subdivided into sub-meshes. This facilitates the editing of specific regions (sub-meshes) of a mesh without changing the rest of the mesh.
- *Sub-mesh editing.* Classical editing operations can be applied to sub-meshes satisfying the user expectations about shape deformation.
- *Mesh refinement.* The result of sub-mesh editing operations is propagated to a finer mesh using the multiresolution scheme.



a) Sub-mesh frontier. b) The cutting plane.

Figure 2. Definition of a sub-mesh.

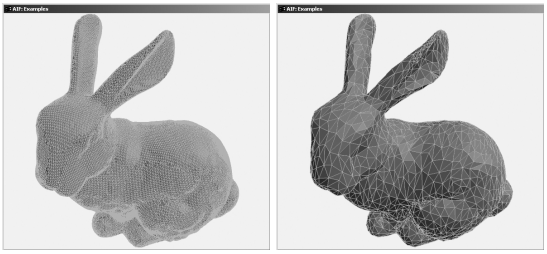
4.1. Sub-mesh creation

In geometric design, an object is normally built up by combining some sub-objects. But, the outcome is almost always an object with undistinguishable sub-objects from the data structure point of view. In fact, a geometric data structure has no means to distinguish between a cow head vertex and a cow leg vertex. This makes it difficult to edit and manipulate meshes. Fortunately, sub-meshes facilitate the manipulation of meshed objects by applying well-known editing techniques to some regions locally. Besides, it is always possible to re-edit them again and again.

The creation of a sub-mesh is done by picking up a collar of edges or faces (i.e. defining it frontier) as shown in Figure 2 (a). This process is acceptable for small meshes but it can be difficult for large meshes. Thus, using the multiresolution scheme we can create a coarse version of the mesh that enables us to define sub-meshes easily. For example, in the mesh of the Figure 3 (a), it is difficult to select a collar of faces to delimit a sub-mesh for a bunny ear. But, it is easy to do that by using a coarser mesh (Figure 3 (b)) because it has a smaller number of faces.

Alternatively, the shortest path of edges that approximates the intersection between a cutting plane and a mesh can be also used to define the sub-mesh frontier, as illustrated in Figure 2 (b).

We must be particulary careful about the consistency of each sub-mesh frontier during the refinement step of a mesh. In fact, when a mesh is refined as a whole, it may happen that the re-inserted vertex of a splitting vertex in the frontier of a coarse sub-mesh falls in or out the counterpart finer sub-mesh, or it remains on its frontier.



a) A bunny mesh. b) A simplified bunny mesh.

Figure 3. A finer and a coarser mesh.

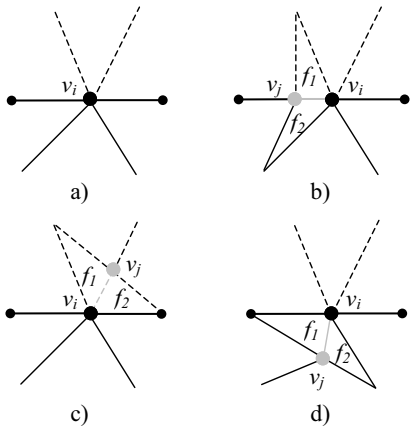
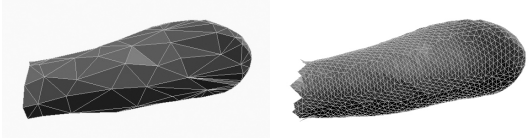


Figure 4. Vertex splitting operation for a vertex (v_i) in the frontier of a sub-mesh.

Let us consider the splitting vertex v_i in the Figure 4 (a). The frontier is represented by thick edges and vertices. It separates a sub-mesh (continuous edges) from another (dashed edges). The re-inserted vertex v_j will be placed in the frontier of both sub-meshes (Figure 4 (b)), or inside the first sub-mesh (Figure 4 (c)), or inside the second sub-mesh (Figure 4 (d)). Its classification depends on the classification of the splitting vertex as well on the classification of the neighbour vertices.

The new vertex v_j belongs to the frontier if it has neighbour vertices belonging to both sub-meshes. Otherwise, v_j belongs to one of the sub-meshes. The classification proceeds upwards to edges and faces by classifying their bounding vertices and edges, respectively, in relation to the frontier in between. Thus, it is possible to define sub-meshes for any multiresolution level. This fact allows us to define a sub-mesh in the coarse mesh, in particular when it is not easy to define it in the fine mesh. For example, Figure 5 shows a coarse sub-mesh and its finer counterpart for the bunny ear after the refinement process.



a) A coarse sub-mesh. b) and its finer counterpart.

Figure 5. Sub-mesh of the bunny ear.

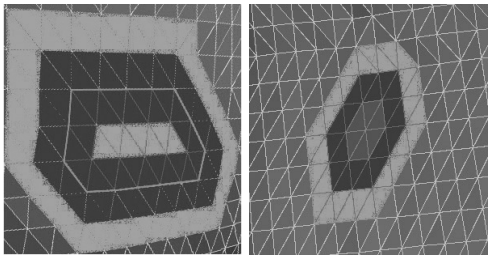


Figure 6. Propagation of the deformation.

4.2. Smoothing of the frontier collar

Editing operations can deform considerably the frontier region (i.e. the frontier collar) of a sub-mesh. Consequently, it is sometimes necessary to smooth this region. There are two ways to smooth it:

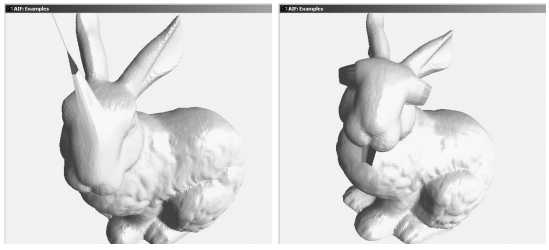
- by propagating the deformation to various levels of faces (or collar of faces) beyond the sub-mesh frontier; this is done during every editing operation.

- alternatively, we can use a smooth operator after every editing operation.

So, for every editing operation, we have to specify how many propagation levels (or collars of faces) will be affected by the deformation.

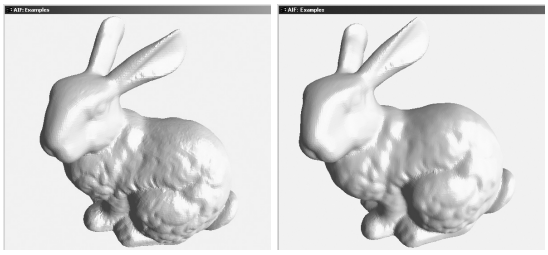
Let n be the number of propagation levels. If $n = 0$, no smoothing operation is carried out in deforming a vertex or a sub-mesh. Otherwise, the deformation is smoothed for n collars of faces (Figure 6). For example, the number of propagation levels of the vertex deformation of the bunny depicted in Figure 7 (a) is equal to 4, and it is 2 for the bunny eyes in Figure 7 (b). The bunny tongue in Figure 7 (b) was deformed with a propagation level equal to 0. That is why it looks so sharp. It is up to the user to control the propagation extension by specifying *a priori* the propagation level. Anyway, we can always confine the deformation to a sub-mesh in such a way that nothing happens beyond its boundary, independently of the pre-defined propagation level.

To smooth the frontier region of a sub-mesh after an editing operation, we use the Laplacian smooth operator proposed by Taubin in [26]. Taubin used this operator to smooth a mesh as a whole, but here it is also used to smooth sub-meshes and their frontier collars. This is illustrated in Figure 8 (a) for a sub-mesh and in Figure 8 (b) for all the mesh. This operation is also very useful for smoothing the mesh (or sub-mesh) when some irregularities appear in the mesh due to editing operations or the refinement process.



a) A single vertex. b) A set of vertices.

Figure 7. Deformation of vertices.



a) Smoothed bunny head. b) Smoothed bunny mesh.

Figure 8. Smoothing operation.

4.3. Sub-mesh editing

In this paper, editing operations can be applied to isolated vertices or to sub-meshes. This can be done by:

- selecting and pulling a set of vertices along a direction defined by a 3D vector;

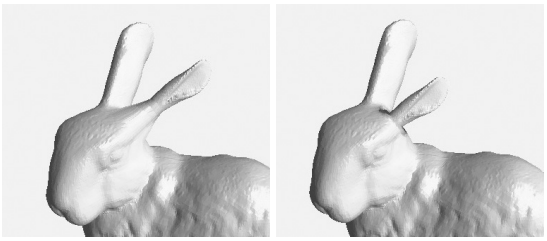
- applying geometric transformations to sub-meshes such as, scaling, rotation, taper and twisting.

Figure 7 (a) shows a local deformation of the bunny along a line or vector after picking up a single vertex, while Figure 7 (b) does the same for the set of vertices of its eyes. A local deformation depends on the distance between a vertex of the original mesh and its image in the deformed mesh, as well as the number of propagation levels imposed on the deformation.

Geometric transformations are normally applied to an object as a whole. But they can be also applied to sub-meshes; in particular, the scaling and rotation transformations [5, 27], as well as their progressive or vanishing counterparts introduced by Barr [1], i.e. tapering and twisting transformations. The idea behind Barr's approach is to change the transformation against the position at which it is applied. In this paper, the Barr's technique has been also adapted for editing sub-meshes.

The interactive editing of sub-meshes allows us to directly observe the outcome of a deformation. An geometric transformation can be undone by applying its inverse. For example, scaling a sub-mesh with a factor 2.0 can be undone by scaling with the factor 0.5.

Scaling. Scaling changes the size of a sub-mesh. It transforms a sub-mesh with reference to the origin. That is, it changes the position of all vertices in relation to the origin.



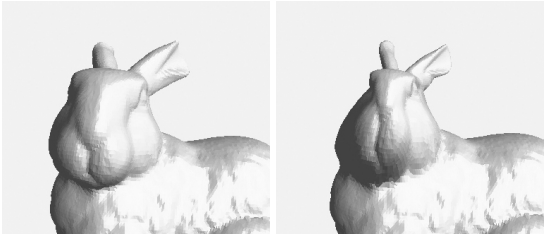
a) With propagation. b) Without propagation.

Figure 9. Sub-mesh scaling.

It can carried out with propagation or not. The propagation approach tends to smooth the transition collar between a sub-mesh and its neighboring faces. But, if the scaling does not involve propagation of the deformation, it has to automatically adjust the position of the sub-mesh to its boundary. This is so because the boundary vertices remain at the same position, while their neighboring vertices in the sub-mesh move away in relation to its boundary. Consequently, the scaled sub-mesh has to be adjusted by moving it back to the plane that approximates its original boundary in the mesh. This is illustrated in Figure 9, where we can see the result of scaling the left ear of bunny with (Figure 9 (a)) and without (Figure 9 (b)) propagation effects.

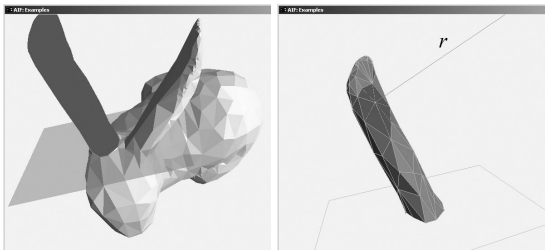
Tapering. The tapering operation is a kind of progressive scaling where the scale factor is not the same for all vertices levels. Even so, we can always apply a tapering operation only for some propagation levels and keeping the scaling factor constant for the remaining levels. If the number of propagation levels is maximum, the scaling factor is progressive for each level. This is the usual tapering (applied to the bunny head in Figure 10 (a)). Otherwise, the scaling factor is progressive for some propagation levels, after which it remains constant (applied to the bunny head in (Figure 10 (b) with propagation level of 20).

Rotation. To rotate a sub-mesh, we need to specify both an axis and an angle of rotation. A 3D vector defines the rotation axis. This vector is initially a normal vector to a vertex previously chosen by the user, and can be adjusted in-

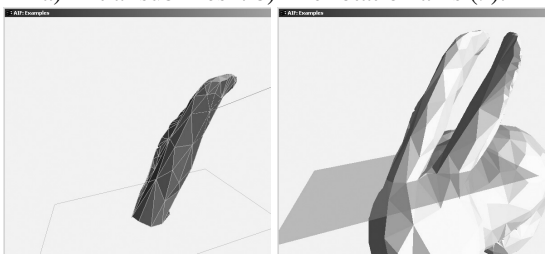


a) Usual tapering. b) Tapering for some levels.

Figure 10. Tapering with/without propagation.



a) Initial sub-mesh. b) The rotation axis (r).

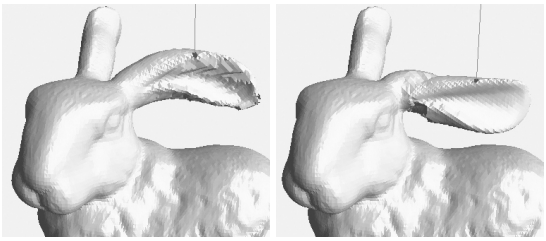


c) Sub-mesh after rotation. d) Final sub-mesh.

Figure 11. Rotating with a restriction plane.

teractively. The rotation may originate some undesirable results in the transition region of the sub-mesh; i.e. near the frontier of the sub-mesh. In fact, we must be careful about rotating a sub-mesh because that may lead to geometric overlapping and self-intersections of the transition region itself. There are two ways to overcome this problem. The first uses a number of propagation levels with varying rotation angles beyond the sub-mesh frontier in order to smooth the transition region. Alternatively, we can impose some restriction on the rotation in order to minimize undesirable deformations. For example, the result of rotating the right ear of the bunny in Figure 11 (b) about the axis r is shown in Figure 11 (c). The right bunny ear cells nearest to the left bunny ear tend to squeeze and self-intersect. For preventing this, we use a plane that approximates the frontier of the sub-mesh. This plane works as a barrier for undesirable deformations. A vertex positioned beyond or behind it cannot crosses it after rotating a sub-mesh. This mechanism can be also propagated to the other levels.

Twisting. Axial twisting is a kind of progressive rotation. In fact, what is done is scaling the rotation angle along various levels. This operation permits a smooth transition from a sub-mesh to its host mesh (Figure 12). Similar to tapering operation, if the propagation level is maximum, the rotation angle is different for all vertex levels (or propagation levels). This is illustrated in Figure 12 (a), where the left bunny ear has been twisted from bottom to top in relation to the original mesh. If the propagation level is not maximum, the rotation angle progressively changes along various propagation levels, after which it remains constant. This is illustrated in Figure 12 (b), where the left bunny ear has been partially twisted along 10 propagation levels.



a) Usual twisting. b) Twisting for some levels.

Figure 12. Twisting with/without propagation.

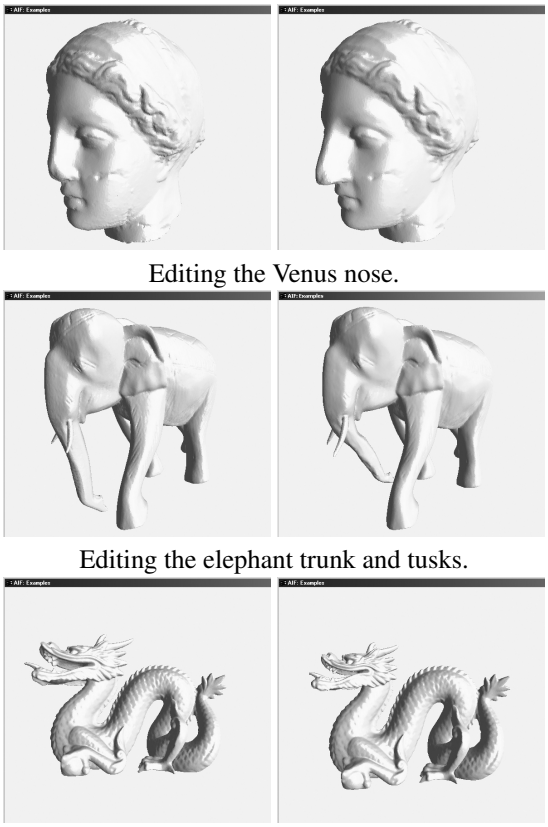
5. Results

All the mesh models used in this paper were edited on a PC equipped with 1.6GHz Intel Pentium 4, running Windows 2000 OS, 768MB of memory and a GeForce 4 graphics card with 64MB. They include those models shown in the Figures 13 and 14. In Figure 13, the Venus nose was changed by pulling away one of it vertices, the elephant trunk and tusks were scaled and rotated, and the dragon head was tapered. This suggests that it is not difficult to change the shape of an object locally, as required for many animation applications and game technologies, without changing its overall shape. Note that the dragon mesh in Figure 13 has more than 2.5 million of cells (i.e. vertices, edges and faces).

Figure 14 shows an editing process for a multiresolution dinosaur mesh. This sequence of pictures shows: (a) the original mesh; (b) the coarse mesh created by applying our simplification algorithm; (c)(d) the coarse mesh subdivided in sub-meshes; (e) the sub-meshes edited through some operations (head, arms and tail were scaled, tapered and rotated); and finally (f) the resulting refined mesh. This is the usual editing procedure of a mesh (or a sub-mesh) in our system. First the mesh is simplified to the desirable level of detail, after which the user defines the sub-meshes that he/she wants to edit. Then the sub-meshes can be edited by applying the necessary transformations, and finally the mesh is refined and smoothed if necessary.

6. Conclusions and Future Work

This paper has presented a new approach for interactive editing of multiresolution meshes using sub-meshes. In order to edit large meshes, initially the mesh is simplified to a desirable level of detail. Then, after subdividing a mesh into sub-meshes interactively, each sub-mesh can be edited through geometric transformations (e.g. scaling, rotation, tapering, twisting and smoothing). Finally, the overall mesh can be refined back. Note that these editing operations are confined to sub-meshes. So, we are able to deform



Editing the Venus nose.

Editing the elephant trunk and tusks.

Editing the dragon head.

Figure 13. Editing of several models.

a sub-mesh without changing the mesh cells beyond it. Besides, it is always possible edit them again and again.

This multiresolution scheme is supported by the modified AIF data structure, which is capable of coping with editing operations at interactive rates, because its performance in retrieving topological information holds independently of the mesh size.

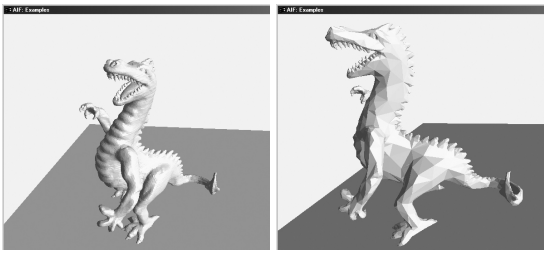
In the future we expect to implement more editing operations. For example, the deformation with lattices for sub-meshes. Other issue to approach in the future is to develop mechanisms for automatically detecting and preventing distortions during editing operations.

Acknowledgements

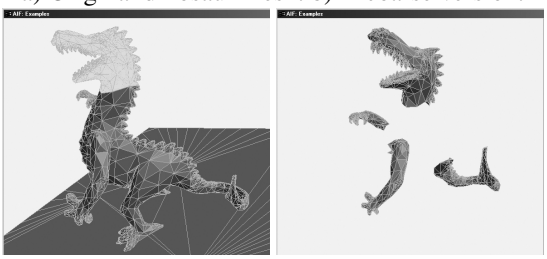
The mesh models are courtesy of Cyberware, Stanford University, Avalon, 3D Cafe, and Polygon Technology GmbH.

References

[1] A. H. Barr. Global and local deformations of solid primitives. In *Proceedings of Siggraph*, pages 21–30, 1984.



a) Original dinosaur mesh. b) A coarse version.



c) Subdivision of the mesh. d) Sub-meshes created.

e) Sub-meshes edited. f) Final refined mesh.

Figure 14. Editing of a multiresolution mesh.

[2] H. Biermann, I. M. Martin, D. Zorin, and F. Bernardini. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics*, 21(3):312–321, 2002.

[3] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, 1999.

[4] M. Botsch and L. Kobbelt. Multiresolution surface representation based on displacement volumes. In *Proceedings of Eurographics*, pages 483–491, 2003.

[5] J. Foley, A. V. Dam, and J. H. S. Feiner. *Computer Graphics: Principles and Pratices*. Addison-Wesley Publishing Company, 1996.

[6] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proceedings of Siggraph*, pages 325–334, 1999.

[7] H. Hoppe. Progressive meshes. In *Proceeding of Siggraph*, pages 99–108, 1996.

[8] W. Hsu, J. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, 1992.

[9] T. Kanai, H. Suzuki, J. Mitani, and F. Kimura. Interactive mesh fusion based on local 3d metamorphosis. In I. S. MacKenzie and J. Stewart, editors, *Proceedings of Graphics Interface*, pages 148–156, 1999.

[10] L. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. In *Proceeding of Eurographics*, volume 19(3), page C249, 2000.

[11] L. Kobbelt, S. Bischoff, M. Botsch, K. Kahler, C. Rossl, R. Schneider, and J. Vorsatz. Geometric modeling based on polygonal meshes. In *Eurographics Tutorial*, 2000.

[12] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of Siggraph*, volume 32, pages 105–114, 1998.

[13] L. Kobbelt, J. Vorsatz, and H.-P. Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry: Theory and Applications*, 14, 1999.

[14] F. Lazarus, S. Coquillart, and P. Jancene. Interactive axial deformations. Technical report, RR-1891, Genova, 1993.

[15] S. Lee. Interactive multiresolution editing of arbitrary meshes. In *Eurographics, Computer Graphics Forum*, volume 18(3), pages 73–92, 1999.

[16] Y. Lee and S. Lee. Geometric snakes for triangular meshes. In *Proceedings of Eurographics*, volume 21(3), 2002.

[17] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level Of Detail For 3D Graphics*. Morgan Kaufmann, 2002.

[18] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of Siggraph*, pages 181–188. ACM Press, 1996.

[19] M. Madi and D. Walton. Interactive selection and modification of polyhedral 3d objects. In *Proceedings of the International Conference on Computer Vision and Graphics*, volume 2, pages 510–517, Poland, 2002.

[20] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of Siggraph*, volume 20, pages 151–160, 1986.

[21] F. Silva and A. Gomes. AIF - a data structure for polygonal meshes. In *Proceedings of Computational Science and Its Applications (ICCSA 03)*, pages 478–487. LNCS Vol. 2669, part III, V. Kumar, M. Gravilova, C. Tan and P. L'Ecuyer (eds.), Springer-Verlag, 2003.

[22] F. Silva and A. Gomes. Interactive editing of arbitrary submeshes. In *Proceedings of Computer Graphice International (CGI 03)*, pages 218–221. IEEE Computer Society Press, 2003.

[23] F. Silva and A. Gomes. Normal-based simplification algorithm for meshes. In *Proceedings of Theory and Practice of Computer Graphics (TPCG 04)*, pages 211–218. IEEE Computer Society Press, 2004.

[24] K. Singh and E. Fiume. Wires: A geometric deformation technique. In *Proceedings of Siggraph*, pages 405–414, 1998.

[25] H. Suzuki, Y. Sakurai, T. Kanai, and F. Kimura. Interactive mesh dragging with an adaptive remeshing technique. *The Visual Computer*, 16(3/4):159–176, 2000.

[26] G. Taubin. A signal processing approach to fair surface design. *Computer Graphics*, 29(Annual Conference Series):351–358, 1995.

[27] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques - Theory and Practice*. Addison-Wesley, 1992.

[28] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proceedings of Siggraph*, pages 259–268, 1997.