GVU Tech Report. May 2003. To appear in the ACM Symposium on Geometry Processing, Aachen, 2003.

Edge-Sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces

Marco Attene[†], Bianca Falcidieno[†], Jarek Rossignac[‡], Michela Spagnuolo[†]

[†] Istituto per la Matematica Applicata e Tecnologie Informatiche, CNR, Genova, ITALY.
[‡] College of Computing, Georgia Institute of Technology, Georgia, USA.

Abstract

3D scanners, iso-surface extraction procedures, and several recent geometric compression schemes sample surfaces of 3D shapes in a regular fashion, without any attempt to align the samples with the sharp edges and corners of the original shape. Consequently, the interpolating triangle meshes chamfer these sharp features and thus exhibit significant errors. The new Edge-Sharpener filter introduced here identifies the chamfer edges and subdivides them and their incident triangles by inserting new vertices and by forcing these vertices to lie on intersections of planes that locally approximate the smooth surfaces that meet at these sharp features. This post-processing significantly reduces the error produced by the initial sampling process. For example, we have observed that the L^2 error introduced by the SwingWrapper[9] remeshing-based compressor can be reduced down to a fifth by executing Edge-Sharpener after decompression, with no additional information.

1. Introduction

Due to the focus of popular graphic accelerators, triangle meshes remain the primary representation for 3D surfaces. They are the simplest form of interpolation between surface samples, which may have been acquired with a laser scanner [1,2,3], computed from a 3D scalar field resolved on a regular grid [4,5], or identified on slices of medical data [6,7]. Most acquisition techniques restrict each sample to lie on a specific curve whose position is completely defined by a pre-established pattern. For example, a laser-scanner measures distances along a family of parallel or concentric rays that form a regular pattern or grid. One may argue that an iso-surface extraction uses three such patterns, aligned with the principal directions. Because the pattern of these rays or stabbing curves is not adjusted to hit the sharp edges and corners of the model, almost none of the samples lie on such sharp features. Therefore, the sharp edges and corners of the original shape are lost by the sampling process and replaced in the resulting triangulation by irregular chamfers. The size of these chamfers may decrease if a finer sampling step is used, but, as observed by Kobbelt et al. [8], the associated aliasing problem will not be solved by over-sampling, since the surface normals in the reconstructed model will not converge to the normal field of the original object.

Similar aliasing artifacts can be observed on models produced by remeshing, which forms the basis of three of the most effective compression techniques published recently [9,10,11]. Basically these methods create a new mesh that approximates the original one. Vertices of the new mesh are placed on the original surface or at quantized locations near the surface, so that their position can be predicted more accurately and encoded with fewer bits. Unfortunately, almost none of the new vertices fall on sharp edges or corners. As a consequence, the sharp features are not captured in the new mesh. An exception in this class of algorithms is described in [12], where the remeshing process aligns samples on sharp edges.

In this paper we present a novel algorithm that automatically identifies these aliasing artifacts and replaces them with refined portions ot the mesh that more accuratley approximate the original shape. This edge-sharpening process works well for meshes generated by various kinds of uniform samplings and does not introduce undesirable sideffects away from sharp features, as shown in Figure 1.

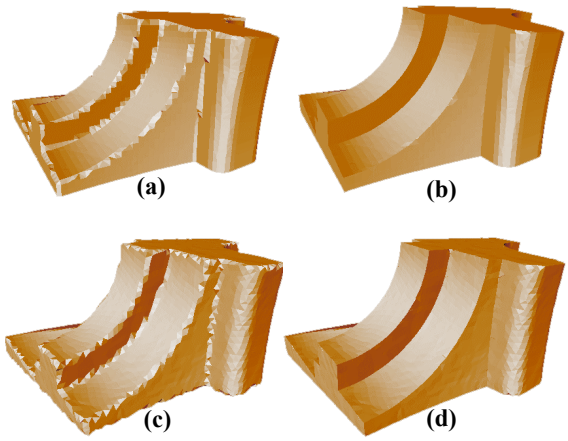

Figure 1. *An aliased model (a) generated with Marching Cubes is improved automatically (b) by EdgeSharpener. A version (c) of the original model was generated through the lossy SwingWrapper compression [9]. An improved version (d) was obtained with no additional information by running EdgeSharpener after decompression.*

2. Previous Work

When a point cloud is dense enough, sharp features may be inferred by analyzing the neighborhood of each point [13,14]. However, in most situations, surface samples are sparse and their interpolation defined by a triangle/vertex incidence graph. The loss of sharp features during the triangulation of implicit surfaces has been addressed in [15,16], where the standard marching-cubes algorithm is improved by moving the sample points to optimized locations. In [17], the identification of perceptually salient curvature extrema was used to guide mesh simplification [18]. This method can be coupled with a proper skeletonization procedure to extract actual creases and sharp feature lines approximating blends in the model. During the remeshing of a surface, some of

the evenly distributed vertices may be attracted to sharp feature lines, as described in [19]. During the creation of an iso-surface, an *extended* marching cubes (EMC) algorithm [8] derives vertex normals from the original scalar field and uses them to decide whether a voxel contains a sharp feature and, if so, to estimate the location of additional vertices on these features. One year later, a new solution to the iso-surface polygonization problem was presented in [20], where sharp features are preserved using fewer samples.

All the methods discussed above use some information about the original surface to reconstruct or to preserve the features. In contrast, the Edge-Sharpener filter introduced in this paper recovers sharp features when no information about the original surface is available, except for the regularly sampled triangle mesh that approximates it.

3. The Edge-Sharpener algorithm

The errors produced by a feature-insensitive sampling are concentrated in what we call **chamfer triangles**, which cut through the solid near sharp convex edges or through the solid's complement near sharp concave edges. Our objective is to identify these chamfer triangles and to replace them with a finer triangle mesh that better approximates the sharp features of the solid.

In order to preserve the integrity of the triangle mesh, we will subdivide the chamfer triangles by inserting new vertices inside chamfer triangles or on edges between two chamfer triangles, but not on edges between chamfer and non-chamfer triangles. Hence our task involves three parts:

1. Identify the chamfer triangles that must be processed,
2. Decide how to subdivide them,
3. For each newly inserted vertex, estimate the sharp edge or corner that we are trying to restore and move the vertex onto that edge or corner.

We have explored three approaches (global, local and filter) for identifying the chamfer triangles. They produce nearly equivalent results, but offer different compromises between elegance of the formulation, code simplicity, and running time efficiency. We outline all three before providing the details of the filter approach that we have chosen. All three approaches identify what we call **chamfer edges**, which are shown in blue in Figure 2.

In the remainder of the paper, the term **smooth edge** will denote an edge whose dihedral angle approaches within a prescribed tolerance.

The **global** approach that we have explored processes the entire mesh and identifies clusters of triangles that are connected by smooth edges and tessellate portions of smooth surfaces. Chamfer edges are those connecting two different clusters. Triangles bounded by three chamfer edges are the corner triangles. The clustering of the triangles is delicate [21] and unnecessary for our purpose. Specifically, we must differentiate between smooth surfaces and thin corridors (generalized triangle strips) of smoothly connected triangles, which although smooth, may correspond to the actual chamfers that we wish to replace.

The **local** approach that we have investigated examines the neighborhood of each edge formed by its two incident triangles and by all their neighbors. It attempts to organize the ordered ring of neighbors into two or three strips of nearly coplanar and contiguous triangles, separated by chamfer triangles. If it succeeds, the edge is a chamfer edge. Although the process is local for each edge, its formulation is rather inelegant and its execution involves redundant steps. Furthermore, using only one ring of neighbors may wrongly identify chamfers in noisy regions that do not separate smooth portions of a surface.

We have finally opted for the **filter** approach, which is significantly faster, more robust and easier to implement than the others. This approach is based on the initial identification of all nearly flat edges and on a succession of six simple filters, which each colors the edges, vertices, or triangles, based on the colors of their adjacent or incident elements.

The first step of the filter approach is to paint brown all the smooth edges (we assume that all vertices, edges, and triangles are initially gray). An edge is said to be smooth if the angle between the normals to its two incident triangles is less than twice the average of such angles for the entire mesh. Then, we apply the following sequence of six filters:

1. Paint red each vertex whose incident edges are all brown.
2. Paint red each triangle that has at least one red vertex.
3. Paint red (recursively) each triangle that is adjacent to a red triangle through a brown edge.
4. Paint red the edges and vertices of the red triangles.
5. Paint blue each non-red edge joining two red vertices.
6. Paint green each triangle with three blue edges.

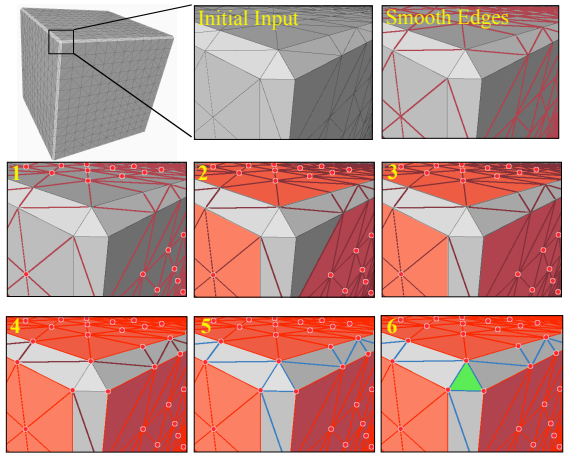

Figure 2: *Selection of the smooth edges in the original model (top row) and the six steps of the filter (mid and bottom rows).*

The six steps are illustrated in Figure 2. Filter 1 identifies the interior vertices of smooth regions. Filter 2 identifies the core triangles of smooth regions. These are incident upon at least one interior vertex. Filter 3 extends the smooth regions to include all of the triangles that are adjacent to a core triangle by a smooth edge. Note that we do not distinguish between the different components of the smooth portion of the mesh. Filter 4 marks the edges that bound the smooth regions to ensure that they are not mistaken for chamfer edges in step 5. Note that these edges are not smooth. Filter 4 also identifies the vertices that bound the smooth regions. Filter 5 identifies the chamfer edges as those that connect vertices on the boundary of smooth regions but do not, themselves, bound a smooth region. Note that chamfer edges may, but need not, be smooth. Also note that some edges may still be gray and that some edges may neither be part of a smooth region nor be chamfer edges (Figure 3).

Finally, filter 6 identifies the corner triangles that are bounded by three chamfer edges and have all of their vertices on the boundary of smooth regions. Thus, they are at the junction of at least three portions of smooth regions.

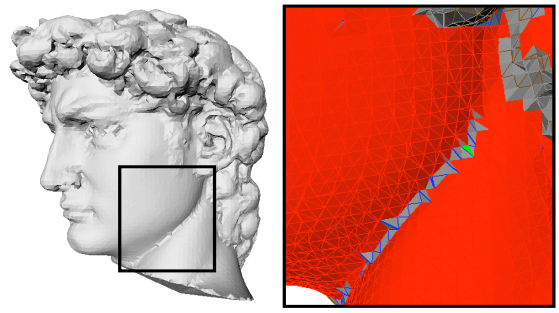

Figure 3: *Chamfers identified by Edge-Sharpener on a model remeshed through the marching intersections algorithm [22]. Some edges are still gray or brown.*

To subdivide the chamfer triangles, including the corner ones, we insert a new vertex in the middle of each chamfer edge and in the middle of each corner triangle. Then we re-triangulate the resulting polygons. We may have three cases (see Figure 4):

a) A triangle with a single chamfer edge is split in two triangles.
b) A triangle with two chamfer edges is split into three triangles.
c) A corner triangle, which has three chamfer edges and an interior vertex is split into six triangles forming a fan around the interior vertex.

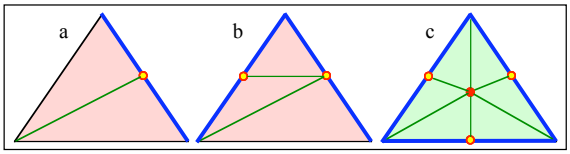

Figure 4: *Subdivision of a chamfer triangle with one (a) two (b) or three (c) chamfer edges.*

Finally, we must find the proper position for each new vertex for chamfer edges and for corner triangles. We use an extrapolation of the smooth surfaces that are adjacent to these elements, as shown in Figure 5 and Figure 6.

To find the position of a new vertex V inserted in a chamfer edge E, we consider the two original vertices, A and B, of E. We compute the weighted sum N of the normals to all of the red triangles incident upon A, normalize it, and define a plane P that is orthogonal to N and passes through A. As weights, we use the angle between the two edges of the incident triangle that meet at A. Similarly, we compute the weighted sum M of the normals to all of the red triangles incident upon B, normalize it, and define a plane Q that is orthogonal to N and passes through B. Finally, we move V to the closest point on the line of intersection between planes P and Q. More specifically, V is $(A+B)/2+(h/k)H$, where $H=AB\times(M\times N)$, which is also $(AB\bullet N)M+(BA\bullet M)N$, $h=AB\bullet N$ and $k=2(M\bullet N)(AB\bullet N)-2(AB\bullet M)$. The process is shown in Figure 5.

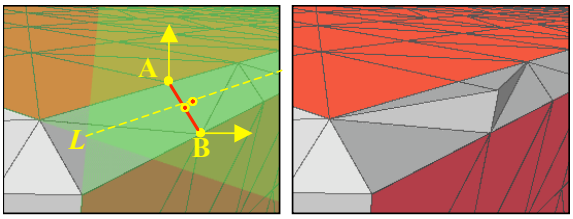

Figure 5: *Insertion of a new vertex to split a chamfer edge.*

To find the position of a new vertex W inserted in a corner triangle with vertices A, B, and C, we proceed as follows. We compute the weighted sum N of the normals to all of the red triangles incident upon A, normalize it, and define a plane P that is orthogonal to N and passes through A. Similarly, we define the plane Q through B with normal M and the plane R through C with normal L. Then, we move W to the intersection of planes P, Q, and R, which is the solution of the system of three linear equalities: $W\bullet N=A\bullet N$, $W\bullet M=B\bullet M$, $W\bullet L=C\bullet L$ (see Figure 6).

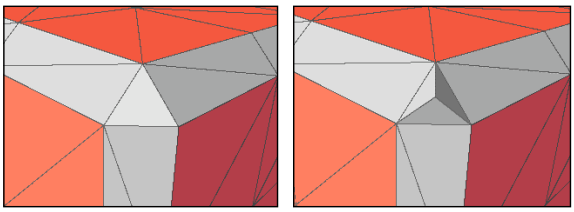

Figure 6: *Insertion of a new vertex to split a corner triangle.*

For simplicity, we have omitted in the previous two paragraphs the discussion of degenerate cases. Such cases include situations where the pairs of planes are parallel or when the triplets of planes do not intersect at a single point, because their normals are coplanar. Moreover, since the algorithm is tailored for nearly uniform triangulations, we have chosen to avoid the creation of edges which are longer than the longest edge of the input mesh (see Figure 7). Thus, if the extrapolated position would require the creation of such a long edge, or if the position itself is not defined because of a linear dependency between the planes, we simply leave the newly inserted vertex in the middle of the chamfer edge or of the corner triangle.

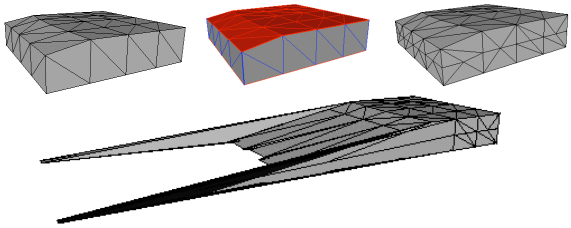

Figure 7: *In the top row the chamfer edges (middle) of an initial model (left) were split without moving the new vertices (right). The model in the bottom was obtained by skipping the edge-length check.*

In some cases, a portion of a strip of triangles that forms a chamfer is bordered by a concave edge on one side and by a convex edge on the other. These situations are easily detected by analyzing the configuration of the triangles incident to the end-points of the chamfer edge or triangle. We treat these cases as the ones discussed above, and simply do not move the newly inserted vertices.

Finally, in extremely rare cases, the alias corresponding to a feature that blends smoothly into a flat area may be painted red, preventing the detection of some "desired" chamfer triangles. This case, however, happens when the strip of such triangles is not aliased, which is very unprobable in practical cases (Figure 14).

The last degeneracy to be discussed includes all the configurations in which the original model has more than three sharp edges meeting at a corner. Provided that the sampling is dense enough, a corresponding aliased model has a strip of chamfer edges for each original sharp edge, and these strips meet at a region made of corner triangles (see Figure 8). The new points that split these adjacent corners (and the chamfer edges inbetween) are moved to the same position, resulting in the creation of degenerate triangles. Therefore, when the sharpening is complete, it

may be necessary to eliminate some degenerate faces [23]. We have tuned our implementation by considering *degenerate* a triangle having at least one angle smaller than 1 degree.

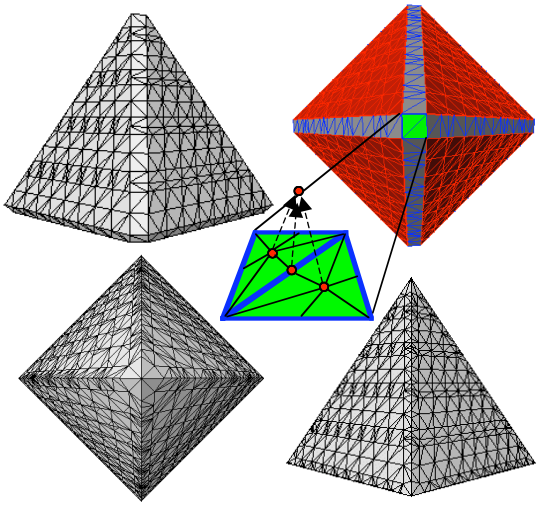

Figure 8: *A pyramid was remeshed using the marching intersections (top-left). Edge-Sharpener detected two adjacent corner triangles (top-right). After the subdivision, the degenerate triangles have been removed (bottom row).*

4. Results and Discussion

We have tested Edge-Sharpener extensively in conjunction with the SwingWrapper compression algorithm [9]. In order to reduce the number of bits to encode the vertex locations, SwingWrapper performs a remeshing of the original mesh, constraining the position of the vertices to follow a prescribed scheme. Specifically, the method grows the new mesh by attaching one new triangle at a time following an EdgeBreaker-like traversal order [24]. When the new triangle has a new tip vertex, the location of this tip is computed as the intersection of a circle orthogonal to the gate with the original surface, forcing the two new edges to have a prescribed length L. This scheme allows to encode the location of the tip vertices using a few bits that quantize the dihedral angle at the gate. The sequence of quantized angles is further compressed using an arithmetic coder. The SwingWrapper compression is *lossy*, since an error is introduced by the remeshing. Most of the discrepancy between the original and the re-sampled models is concentrated near the sharp edges and corners.

We have also tested EdgeSharpener on the models produced by the Piecewise Regular Meshes (PRMs) compression approach [10], which performs a different remeshing. Based on their orientation, it splits the triangles into 6 sets. The set of triangles whose normal is closest to the positive x-direction is sampled using a regular grid in the y-z plane. The other five sets are sampled similarly using the appropriate grids. The results are connected into a valid mesh. The connectivity of the meshes produced by SwingWrapper and by PRM is encoded using modified versions of the EdgeBreaker compression scheme [24].

In both the cases, we have observed that EdgeSharpener significantly reduces the error between the original shape and the one recovered after decompression. An example of this improvement is shown if Figure 9, where the fandisk model was compressed using SwingWrapper. When no sharpening is applied, the maximum distance between the decoded mesh and the original model is 0.89% of the bounding box diagonal. It decreases down to 0.43% after the application of our new filter. The colored models have been produced by the Metro tool [25] that we used to measure the distortion. Metro uses a color spectrum to show the distribution of the error. Such spectrum is normalized to fit the whole range of errors, so that the blue color corresponds to the minimum error while the red indicates the maximum. Thus, the light color in the cylindrical side of the sharpened model must not be interpreted as an increase of the error, because it comes from a renormalization of the color spectrum in a more narrow range.

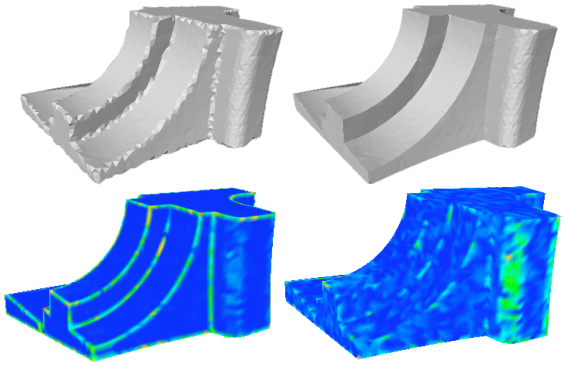

Figure 9: *The maximum error in the fandisk encoded with SwingWrapper is 0.89% of the bounding-box diagonal. After the filtering such error is 0.43%.*

The following Figure 10 shows how the reduction of the L_2 distortion becomes more effective as the SwingWrapper remeshing becomes denser.

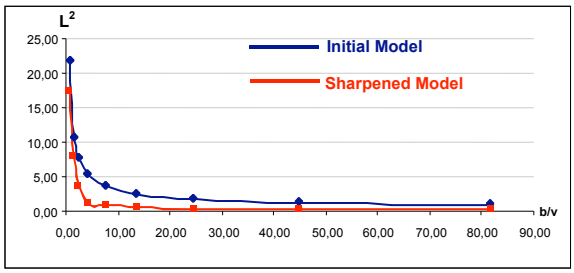

Figure 10: *Reduction of the L_2 error for various remeshed models. Bit-per-vertex rates are relative to the # of vertices (12946) of the original fandisk model. Errors are expressed in units of 10^{-4} of the bounding-box diagonal.*

We have tested our filter on a number of meshes generated through the Marching-Cubes algorithm, through the SwingWrapper remesher and through the remeshing strategy of PRMs, and we have found that in all the cases, when the original model was sampled with a sufficiently high density, most of the sharp features can be completely recovered, while the parts of the mesh that correspond to regions of the original model without sharp features are not modified by Edge-Sharpener.

Another important application of Edge-Sharpener is the post-processing of laser-digitized models. Most surface reconstruction approaches, in fact, are not able to correctly reconstruct sharp features. In Figure 13 some sharpening results are shown. In the top row, we have simulated a marching-cubes output using the marching intersections algorithm presented in [22]. In the third row, the original model was sampled using a regular grid, and the samples were interpolated using the surface reconstruction method described in [1]. In all of the examples, we have observed a significant reduction of both the maximum and the mean square distortions.

In [8], an application of the extended Marching-Cubes to polygonal meshes (i.e. a remeshing), is described. In fact,

such an application is useful to improve the quality of meshes having degenerate elements or other bad characteristics. In some cases, the information at the edge-intersections makes it possible to reconstruct sharp features in an Edge-Sharpener like manner. For example, if a cell contains an aliased part that does not intersect the cell's edges, the normal information at the intersections is used to extrapolate planes and additional points are created on the inferred sharp feature. If, on the other hand, the cell's edges do intersect the aliased part, the normal information becomes noisy, and nothing can be predicted about any possible feature reconstruction. Conversely, the use of the red neighborhood to extrapolate a plane makes EdgeSharpener less sensitive to such problems. Moreover, while a remeshing on the whole model can introduce an additional error on the regions without sharp features, the local modification we propose only affects the aliased zones.

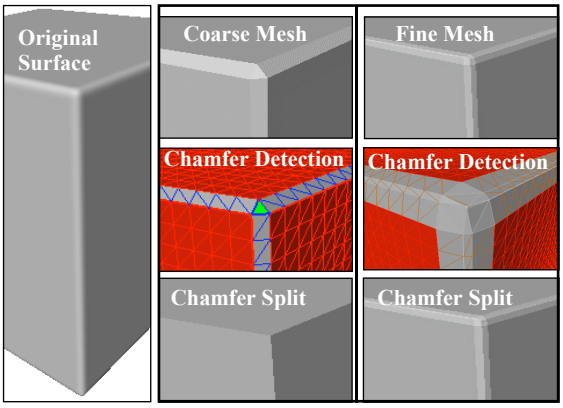

Figure 11: *Unwanted creases may be produced if an original surface has blends whose radius is smaller than the inter-sample spacing (middle column). If the sampling step is small compared to the blend radius, the blends are not modified by Edge-Sharpener (right column).*

Clearly, Edge-Sharpener can miss features that are smaller than the inter-sample spacing and may produce sharp edges where the original model has a feature that has been smoothed with a small-radius blend (Figure 11). There is not enough information in the sampling to recover such small features or blends.

Our experiments on a variety of meshes indicate that Edge-Sharpener is extremely fast and robust. For example, the sharpening of the models presented in this paper took less than 0.4 seconds each on a PC equipped with a 1.7Ghz CPU (precise timings are shown in Figure 13). In order to test the robustness of the proposed approach in presence of noisy data, we have perturbed some models with various amounts of noise and we have observed that the sharpening does not produce unwanted side-effects. Clearly, if the amount of noise becomes comparable with the inter-sample spacing, its influence on the dihedral angles prevents the algorithm to identify some chamfer elements, but the results are still very good (Figure 12).

Moreover, we have concluded that the effectiveness of the proposed method is not restricted to uniformly sampled meshes. For example, Edge-Sharpener correctly restores the sharp features of typical meshes generated through interpolation of laser-captured point sets or through iso-surface polygonization procedures which exhibit a fair amount of variation in edge-length.

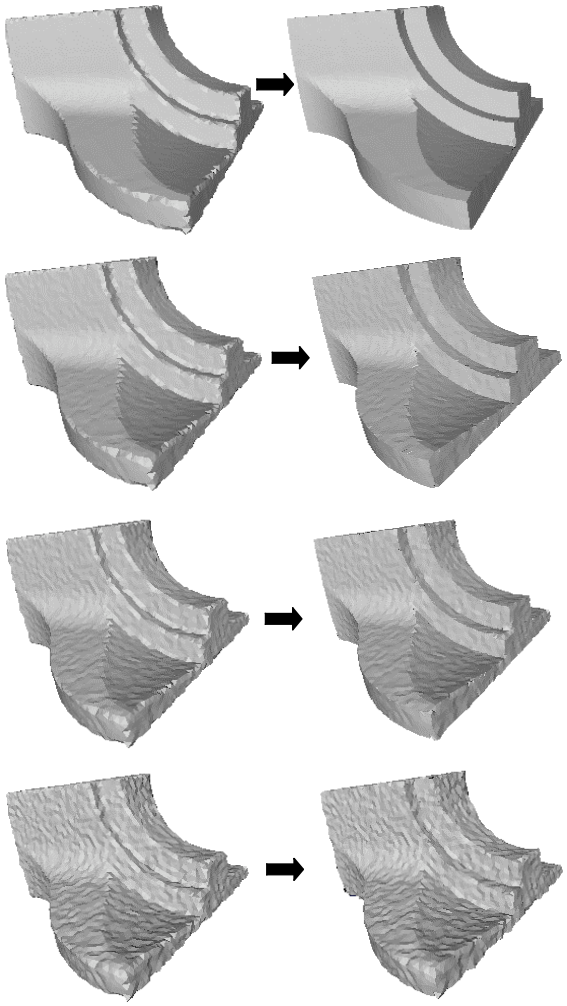

Figure 12: *Sharpening of a model with various amounts of noise. The amplitude of the noise in the normal direction ranges from 0% (top row) to 100% (bottom row) of the maximum length of an edge in the mesh.*

The last issue to be discussed is our definition of smooth edge. As we said in section 3, an edge is said to be smooth if the angle between the normals to its two incident triangles is less than twice the average of such angles for the entire mesh. This choice is motivated by the following consideration: when an original piecewise smooth model is sampled with a nearly infinite density, the dihedral angle at edges not belonging to chamfer triangles is nearly π. Furthermore, the number of such non-smooth edges is negligible with respect to the total number of edges, thus the average dihedral angle remains close to π or, equivalently, the average angle, θ, between the normals of two adjacent triangles remains close to 0. The influence of non-smooth edges on θ is small but not null, thus the actual angle for smooth edges is slightly smaller than θ. In practice we do not have infinite samplings, so taking θ as threshold makes the algorithm too sensitive to small amounts of noise. We have experienced that doubling θ is a good compromise between theoretical correctness in the ideal case and robustness in the practical case.

5. Conclusions

We have introduced a simple, automatic, and efficient edge-sharpening procedure designed to recover the sharp features that are lost by reverse engineering or by remeshing processes that use a non-adaptive sampling of the original surface. The procedure starts by identifying smooth edges. Then, it performs six trivial filters that identify chamfer

edges, which in turn define chamfer and corner triangles. The chamfer edges and triangles are subdivided by inserting new vertices and moving them to strategic locations where the sharp feature is estimated through extrapolation of abutting smooth portions of the surface.

We have run numerous tests (Figure 14) on models coming from uniform remeshing, marching-cubes iso-surface generation, and surface reconstruction from nearly uniform clouds of points. In all the cases, in addition to the correct reconstruction of sharp features, we have observed that the distortion between the mesh and the original model was significantly reduced by our sharpening process, while the parts of the mesh not corresponding to sharp features in the original model were not modified.

6. Acknowledgements

This work is part of the bilateral research agreement "Surface Analysis" – GVU/GATECH and IMATI-GE/CNR. Marco Attene's work on this project was partially supported by the national FIRB project MACROGeo. Rossignac's work on this project was partly supported by a DARPA/NSF CARGO grant #0138420. The authors thank all the members of the Computer Graphics Group of the IMATI-GE/CNR and the reviewers for their helpful advice.

References

1. M. Attene and M. Spagnuolo, "Automatic surface reconstruction from point sets in space". Computer Graphics Forum (Procs. EUROGRAPHICS '00), 19(3): pp.457-465, 2000.
2. N. Amenta, S. Choi and R. Kolluri, "The power crust". Sixth ACM Symposium on Solid Modeling and Applications, pp. 249-260, 2001.
3. J. Giesen and M. John, "Surface reconstruction based on a dynamical system". Computer Graphics Forum (Procs. EUROGRAPHICS '02), 21(3): pp.363-371, 2002.
4. W. Lorensen and H. Cline, "Marching Cubes: a high resolution 3D surface construction algorithm", Computer Graphics (Procs. SIGGRAPH '87), pp. 163-169, 1987.
5. J. Bloomenthal, "Polygonization of implicit surfaces", Computer Aided Geometric Design, Vol. 5, pp. 341-355, 1988.
6. S. W. Cheng and T. K. Dey, "Improved construction of Delaunay based contour surfaces". Proc. ACM Sympos. Solid Modeling and Applications 99, pp. 322-323, 1999.
7. G.Cong and B. Parving, "Robust and Efficient Surface Reconstruction from Contours", The Visual Computer, Vol. 17, pp. 199-208, 2001.
8. L. P. Kobbelt, M. Botsch, U. Schwanecke and H-P. Seidel, "Feature Sensitive Surface Extraction from Volume Data", Computer Graphics (Procs. SIGGRAPH '01), pp. 57-66, 2001.
9. M. Attene, B. Falcidieno, M. Spagnuolo and J. Rossignac, "SwingWrapper: Retiling triangle meshes for better EdgeBreaker compression", To appear on ACM Transactions on Graphics, 22(4), (temporarily available at http://www.acm.org/tog/Upcoming.html), October 2003.
10. Szymczak, D. King, J. Rossignac, "Piecewise Regular Meshes", to appear on Graphical Models, 2002.
11. Guskov, K. Vidimce, W. Sweldens and P. Schröder, "Normal Meshes", Computer Graphics (Procs. SIGGRAPH '00), pp. 95-102, 2000.
12. Khodakovsky, P. Schroder, W. Sweldens, "Progressive Geometry Compression", Computer Graphics (Proc. SIGGRAPH'00) , pp. 271-278, 2000.
13. S. Gumhold, X. Wang and R. MacLeod, "Feature Extraction from Point Clouds", Proceedings of the 10th International Meshing Roundtable, Sandia National Laboratories, pp.293-305, 2001.
14. G. Guy and G. Medioni, "Inference of Surfaces, 3D Curves and Junctions from sparse, noisy, 3D data", IEEE Trans. on Pattern Analysis and Machine Intelligence, 19(11), pp. 1265-1277, 1997.
15. Y. Ohtake and A.G. Belyaev, "Dual/Primal Mesh Optimization for Polygonized Implicit Surfaces", Procs. of Solid Modeling '02, pp. 171-178, 2002.
16. Y. Ohtake, A.G. Belyaev and A. Pasko, "Dynamic Meshes for Accurate Polygonization of Implicit Surfaces with Sharp Features", Procs. of Shape Modeling and Applications '01, pp. 74-81, 2001.
17. K. Watanabe and A.G. Belyaev, "Detection of Salient Curvature Features on Polygonal Surfaces". Computer Graphics Forum (Procs. EUROGRAPHICS '01), 20(3): pp.385-392, 2001.
18. M. Garland and P.S. Heckbert, "Surface Simplification using Quadric Error Metrics", Computer Graphics (Procs. of SIGGRAPH '97), pp. 209-216, 1997.
19. J. Vorsatz, C. Rossl, L.P. Kobbelt and H.-P. Seidel, "Feature Sensitive Remeshing", Computer Graphics Forum (Procs. EUROGRAPHICS '01), 20(3): pp.393-401, 2001.
20. T. Ju, F. Losasso, S. Schaefer and J. Warren, "Dual Contouring of Hermite Data", ACM Transactions on Graphics, 21(3) , (Proc. SIGGRAPH'02), pp. 339-346, 2002.
21. M. Garland, A. Willmott, P.S. Heckbert, *"Hierarchical face clustering on polygonal surfaces", Procs. of the 2001 symp. on Interactive 3D graphics, pp. 49-58, 2001.*
22. *Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, R. Scopigno, "Marching Intersections: an efficient resampling algorithm for surface management",* Procs. of Shape Modeling and Applications '01, pp. 296-305, 2001.
23. M. Botsch and L. P. Kobbelt, "A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes", Vision, Modeling and Visualization (VMV01), Stuttgart, Germany, November 21 - 23, 2001.
24. J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes", IEEE Transactions on Visualization and Computer Graphics**,** 5(1), 47-61, Jan-Mar 1999.
25. P. Cignoni, C. Rocchini and R. Scopigno, "Metro: measuring error on simplified surfaces", Proc. Eurographics '98, vol. 17(2), pp 167-174, June 1998.

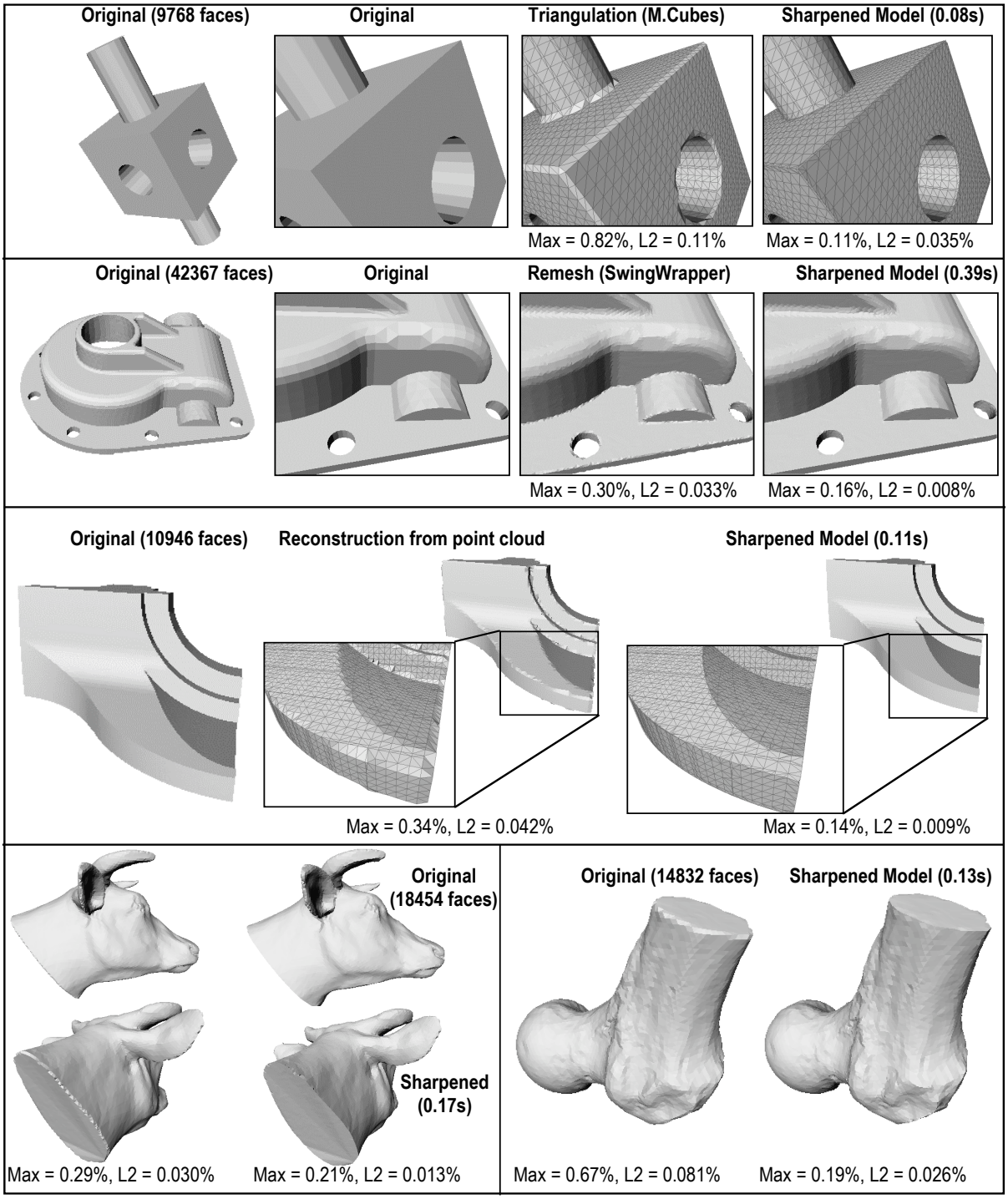

Figure 13: *Top row: sharpening of a marching-cubes generated model. Second row: sharpening of a SwingWrapper remeshed model. Third row: sharpening of a model reconstructed from a point cloud. Bottom row: two further examples showing that only the aliased regions are modified by the sharpening. The maximum and mean square errors have been computed using the Metro tool and are in percents of the bounding-box diagonal. Processing time (in seconds) and the number of faces of each original model are reported. All the models are flat shaded.*

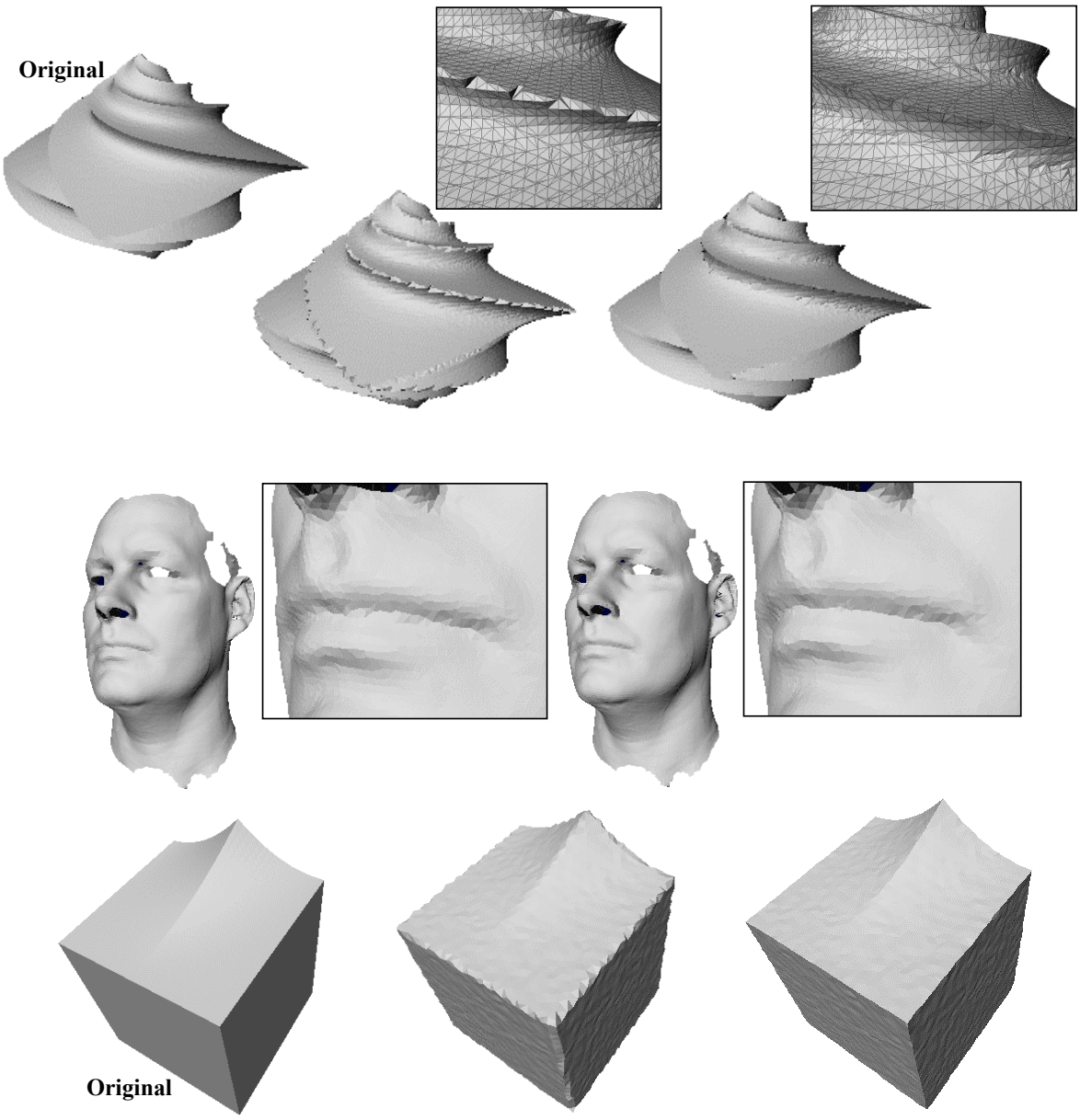

Figure 14: *Top row: sharpening of a model with severe alias. Middle: Improvement of the mouth line on an actual laser digitized model. Bottom row: Recovering of (part of) a sharp feature that blends smoothly into a flat face.*