Eurographics Symposium on Geometry Processing (2007)
Alexander Belyaev, Michael Garland (Editors)

Symmetry-Enhanced Remeshing of Surfaces

Joshua Podolak Aleksey Golovinskiy Szymon Rusinkiewicz

Princeton University

Abstract

While existing methods for 3D surface approximation use local geometric properties, we propose that more intuitive results can be obtained by considering global shape properties such as symmetry. We modify the Variational Shape Approximation technique to consider the symmetries, near-symmetries, and partial symmetries of the input mesh. This has the effect of preserving and even enhancing symmetries in the output model, if doing so does not increase the error substantially. We demonstrate that using symmetry produces results that are more aesthetically appealing and correspond more closely to human expectations, especially when simplifying to very few polygons.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Geometric Algorithms

1. Introduction

In applications such as interactive rendering, editing, or physical simulation of large scenes, it is important to adapt the complexity of the models to the detail present in the surface. This is especially the case given the increased practicality of 3D scanners and procedural modeling systems, which has led to an abundance of over-tessellated models containing millions of primitives. Correspondingly, over the last decade many methods have been developed to simplify 3D models while closely approximating their geometry.

While such approximation methods maintain the geometry of the original model, they do not explicitly preserve other types of high-level information. In particular, the symmetry of the model in Figure 1a would not be preserved by existing remeshing techniques. The recently-introduced Variational Shape Approximation [CSAD04], for example, yields Figures 1b and 1d, in which the approximate left-right symmetry is not captured in the triangulation.

To address this, we propose a framework for model simplification that automatically detects symmetric regions and actively preserves and even strengthens those symmetries during simplification. Specifically, we have adapted all stages of Variational Shape Approximation to include symmetry. During the proxy generation stage, we extend the notion of a proxy to include multiple connected components related by a pre-defined set of symmetry transformations (e.g., planar reflection). Thus, while growing proxies we consider not only neighboring triangles, but also reflected triangles. In the triangulation stage, we explicitly find corresponding symmetric points based on the proxies grown previously. We force the triangulation to follow these correspondences as much as possible, yielding a more symmetric triangulation. The result is the mesh of Figures 1c and e, which captures symmetry while yielding a comparable approximation error.

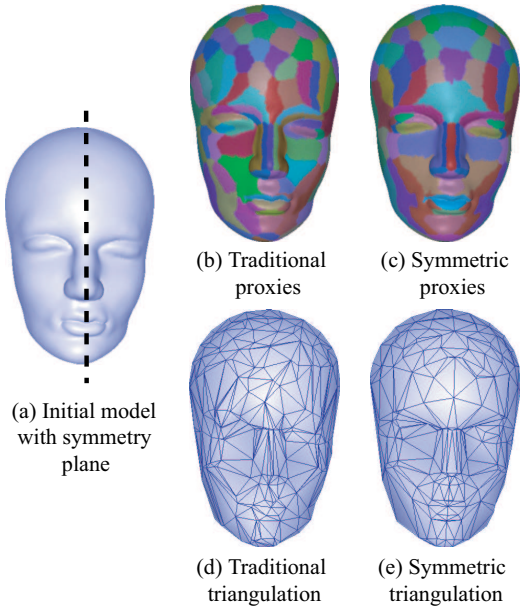


Figure 1: *Remeshing without symmetry does not create a symmetric triangulation without explicit accounting for symmetry. Column (a) shows a model (62K triangles) of a relatively symmetric mask. Column (b) shows a visualization of the proxies found by a standard remeshing algorithm, while (c) shows the proxies obtained taking symmetry into account. (d) and (e) show the resulting triangulations.*

The rest of the paper will be organized as follows: Section 2 will describe previous remeshing techniques in greater detail, Section 3 will provide the details of our algorithm, Section 4 will show results and Section 5 will include a discussion and future work.

2. Previous Work

2.1. Surface Approximation

Due to the usefulness of simplified models for many applications in computer graphics, there is a long history of techniques designed to approximate a 3D surface while optimizing for various types of geometric error.

One common method for mesh simplification involves approximating the surface locally either by greedily clustering local groups of triangles or by collapsing the edges of a mesh using a local error metric that captures the geometry deformation of the simplified area [Hop96, KLS96, GH98, Tur92]. Other methods, including [KT03] and [STK02], seek to combine sets of faces containing similar properties to generate characteristic regions.

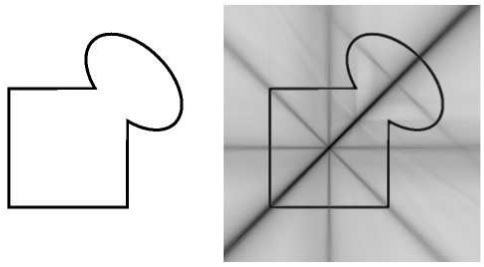
An alternative paradigm for simplification involves generating a a maximally-accurate approximation with respect to a global error metric, while reducing the number of faces. Work such as [HDD*93] generates an energy functional based on a point-to-surface distance of the input mesh. This error function captures the curvature and surface variations from the original model. Departing slightly from the use of surface distance (the L^2 metric), [CSAD04] define a metric based on the normals of the surface (The $L^{2,1}$ metric). They also solve the global error function by fixing a number of *proxies* and then optimally placing these proxies to best approximate the surface. Using normals rather than the more widely used surface distance is motivated by the desire to generate more visually pleasing results, for example by more accurately retaining highlights.

We propose to extend this reasoning by noting that in general, people are sensitive to symmetry, and will notice departure from symmetry more readily than some general deformation of the surface. To this end, we propose to augment remeshing techniques by directly incorporating symmetry information into the algorithm.

2.2. Detection of Symmetry in 3D Shape

Perfect Symmetry: The traditional approach to symmetry detection works with discrete symmetries — perfect symmetries under rotation, reflection, or translation [Ata85, WWV85, MIK93, TW05]. While these methods can detect perfect symmetries with varying degrees of tolerance to noise, they do not work with imperfect symmetries or with symmetries that reflect only parts of the model.

Imperfect Symmetry: In the last decade, methods have been provided for measuring imperfect symmetries. Work such as [ZPA95, KFR04] paved the way by defining a continuous notion of symmetry for partial reflection of models. Recently, [MGP06] have shown how to efficiently find imperfect symmetries in a model, while [PSG*06] defined a *symmetry transform*, capturing the degree of symmetry about all planes passing though a 3D model. An example in 2D is shown in Figure 2. There are two types of imperfect symmetry: local symmetry, in which a portion of a model is perfectly symmetric while the rest of it is not, and approximate symmetry, in which the entire model is not symmetric but could be made symmetric with a slight deformation. Both types of imperfect symmetry may be found automatically using the symmetry transform by extracting the set of *Principal Symmetries* of the model, a set of planes that are local maxima of the transform. An example of these principal symmetries can be seen in Figure 3. We use these principal symmetries to guide our approximation of the surface.



(a) (b)

Figure 2: *An example of a 2D symmetry transform. (a) The initial model. (b) The transform of the model with the darker lines representing stronger symmetries. Note the line of perfect symmetry passing through both the square and the ellipse, as well as additional weaker lines of imperfect symmetry. In all cases, the local maxima of the transform are significant symmetries.*

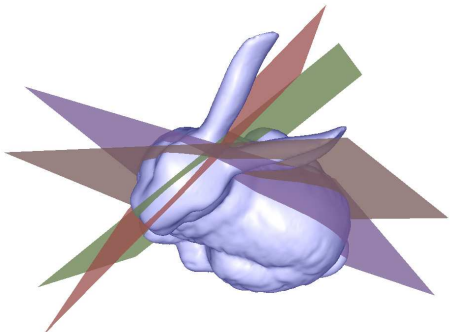


Figure 3: *An example of principal symmetries of a 3D model, extracted from the 3D symmetry transform. Note that these capture the partial and approximate symmetries of the ears, head, body and legs of the bunny.*

3. Symmetric Remeshing

3.1. Our Approach

The goal of our paper is to remesh a surface in a "symmetry aware" environment. If we were only concerned with perfect

symmetries, a simple algorithm would be to remesh using any existing technique, then force symmetry by mirroring the output. However, in most practical situations, the symmetries will be imperfect. For example, there may be near-symmetries that are not perfect because of noise or differing tessellation, or partial symmetries in which regions of the model may be symmetric about different planes. In addition, some models may exhibit approximate symmetry. In these cases, faithful remeshing (resulting in a large number of polygons), should prioritize geometric accuracy. However, remeshing such models with *fewer* polygons should result in a symmetric output, provided that doing so introduces error of the same order as that necessarily introduced by remeshing. In this way, we avoid symmetrizing if there is no reason to do so: the choice to remesh with many polygons shows that the user's overriding concern is to preserve detail accurately.

Our approach is to modify the algorithm proposed by [CSAD04] so that it explicitly preserves symmetry. We chose to begin with this algorithm because it has been shown to produce good results for low polygon counts, for which the careful choice of symmetrization algorithm has the most visible impact. In addition, as we will show later, this algorithm allows us to smoothly combine multiple planes of symmetry, as well as both symmetric and asymmetric regions of the mesh.

Our adapted algorithm uses as input a list of principal planes of symmetry, as extracted by the methods described in [PSG*06]. These planes are used during the three stages of the algorithm: proxy generation, point selection, and triangulation. In the next three subsections we describe each step briefly and explain how we modify the original algorithm to incorporate symmetry.

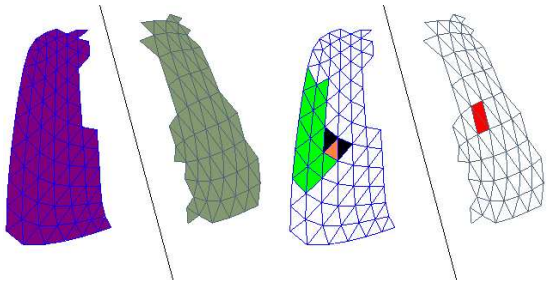
3.2. Symmetric Proxy Generation

The first step of the algorithm is to generate *proxies*, planar regions that closely approximate local sections of the surface. This is done with an iterative technique based on Lloyd's algorithm. At every iteration:

1. Every proxy is assigned triangles of the model from a single priority queue. When a triangle Δ is removed from the queue and assigned to a proxy P, all triangles adjacent to Δ are added to the queue with a weight defined by the compatibility of those triangles to P. This assures that each proxy is a single connected component.
2. Once all triangles have been assigned, optimal proxy parameters (i.e., center and normal) are re-computed based on the triangles currently assigned to the proxy.

In order to avoid converging to local minima, small proxies are occasionally deleted and new proxies introduced at appropriate locations ("teleportation").

Our method follows this approach but generalizes the definition of a proxy by allowing it to represent a planar region, possibly transformed by a discrete set $T_1..T_k$ of symmetry transformations. This is a key component of our algorithm,



(a) (b)

Figure 4: *(a) A proxy with two patches. The purple patch is associated with the identity and the green patch is associated with the reflection plane shown. (b) When a triangle (orange) is assigned to a proxy (green), the neighboring triangles (black) are added to the queue. The triangles across the plane of reflection (red) are added as well.*

as proxies may now contain multiple connected components symmetric to one another. We allow a maximum of one connected component per symmetry transformation (including the identity), called *patches*. As an example, we show in Figure 4(a) a proxy with two patches, one associated with the identity (purple triangles) and one associated with the reflection plane shown (green triangles).

In our method, a triangle Δ may be assigned to a proxy P if it is adjacent to a triangle previously assigned to P, **or** if the triangle nearest to $T_i(\Delta)$ has previously been assigned to P. Consider the example shown in figure 4(b). The orange triangle was recently assigned to the proxy shown (dark green), so any of the black triangles may now be added to the proxy; under our extended definition, the red triangles may be added as well.

The remainder of the iterative algorithm is nearly unchanged, with triangles extracted from the priority queue in order of increasing error, and new patch centers and positions computed after each iteration. The teleportation process is augmented to allow individual *patches*, in addition to entire *proxies*, to be deleted.

After a few iterations, we observe that, where possible, proxies will have symmetric patches corresponding to "good-enough" symmetry transformations. This will ultimately lead to symmetric outputs, since the boundaries between the proxies determine the topology of the final surface. Note that the order with which triangles are selected from the priority queue completely determines whether or not the final proxies are symmetric. That is, symmetric proxies will be created if and only if the error of doing so is comparable to the error introduced by the remeshing itself. Multiple planes of local symmetry are used automatically in the appropriate regions, and no user intervention is required. An example of this can be seen in Figure 5, where the area around the chin is not symmetric. Therefore, in this region, unlike the the rest of the face, the proxies are generated without a reflection.

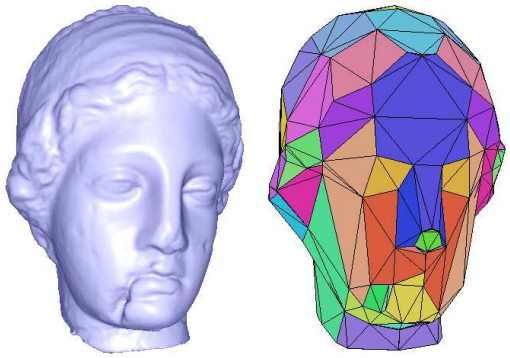


Figure 5: *This model is quite symmetric, except for the gash in the right side of the chin. Note how our algorithm does not create symmetric patches for the proxies that approximate that area, because the error introduced would be too great. The remainder of the head however, has symmetric patches. The model was remeshed using 50 proxies and one plane of symmetry (up to two patches per proxy).*

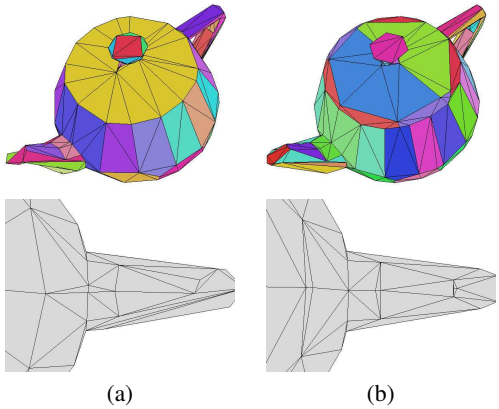


Figure 6: *This figure shows the need for splitting faces that cross over reflection planes. (a) Without splitting faces, triangles that cross the plane of reflection are assigned to only one of the reflections of the proxy. This causes a jagged triangulation. (b) When we split the triangles that cross the reflection plane, the tessellation is symmetric. The models have 342 and 358 faces respectively.*

It is also be possible to augment the method with additional user control. This may consist of limiting the symmetries that are considered for particular portions of the surface, or could extend to implementing a threshold on the deviation from perfect symmetry that is allowed when adding triangles to the priority queue. For the results presented in this paper, only figure 10 uses any user control. In that example, every triangle is limited to consider only a single reflection.

3.3. Point Correspondences

Once the proxies are placed, our goal is to place a set of symmetric points on the surface that we will later triangulate.

[CSAD04] place two categories of points onto the new surface. The first category of points is *anchor vertices*, placed at junctions where three or more proxies meet. The second category of points is *secondary points*, which are placed along the boundaries between proxies to improve the geometry approximation of the new surface and to assure at least three points per proxy boundary. New positions for anchor and secondary points are computed by averaging the projections of each vertex onto all adjacent proxies.

Our modifications to this vertex-placing algorithm involve determining symmetric correspondences between points. For anchor vertices, we check whether the proxies adjacent to some v_i all have reflections that meet in an identical configuration at some v_j. If this is the case, then we establish a correspondence between v_i and v_j, and further adjust their new positions to be perfect reflections of each other.

We find matching secondary points by establishing correspondences between proxy boundaries: mesh-edge paths that run between corresponding anchor vertices are considered to be in correspondence with each other. When adding a secondary vertex to a boundary, we search through corresponding boundaries for the nearest symmetric vertex. As with anchor vertices, we adjust the positions of such correspondences to be perfect reflections.

3.4. Triangulation

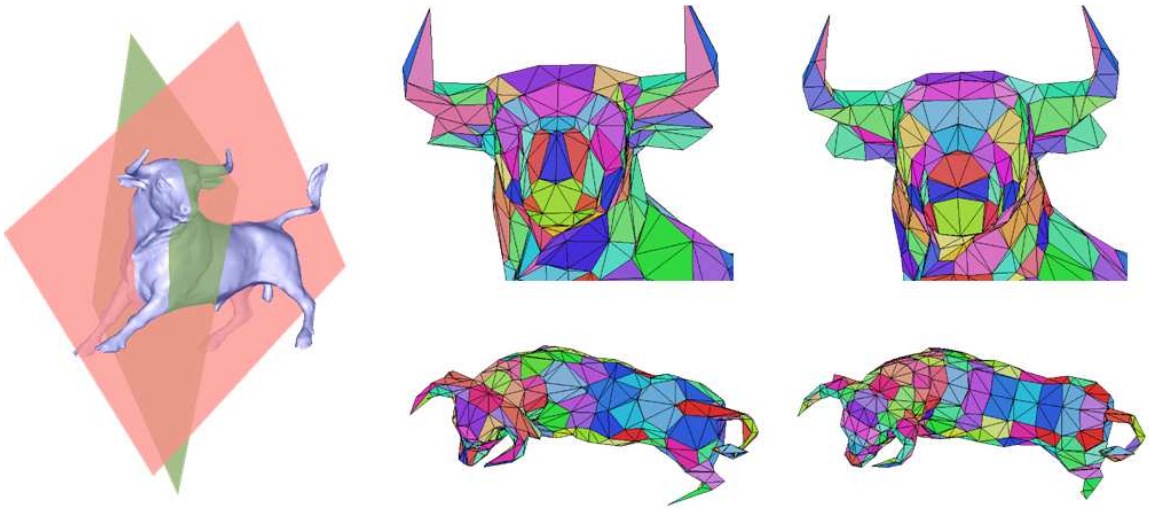
Once we obtain all our correspondences between anchor and secondary points, all that remains is to triangulate the set of points symmetrically.

We triangulate the proxies in the same manner suggested by [CSAD04]. This consists of flooding using Dijkstra's shortest-path algorithm, where the sources are the anchor and secondary points, and with each edge weighted according to its length. At the end of the flooding, adjacency of regions implies that their source points should be connected in the resulting triangulation. As a final pass, edge flipping and edge removal are run to obtain a better tessellation.

We use the same flooding algorithm, but run edge flipping on each proxy separately to prevent edge flips in one proxy from altering the triangulation of its neighbors. This process is guaranteed to converge to the same topology for symmetric proxies if the associated triangles are co-planar. In practice, we find that this method produces a consistent topology even when the triangles are not co-planar.

3.5. Proxies Spanning Reflection Planes

The algorithms for proxy assignment and triangulation, as described above, operate at the granularity of entire triangles of the input. This leads to potential problems for proxies that lie near the planes of symmetry, since triangles that originally cross a symmetry plane will be assigned entirely to one or another component (patch) of a proxy. The result is non-symmetric triangulations along the planes of reflection, as can be seen in Figure 6a. Obtaining a symmetric output therefore requires us to treat this situation as a special case.



Bull planes ($\alpha = 0$) ($\alpha = 0.4$)

Figure 7: *At left, we show a bull remeshed to* 200 *proxies with two planes simultaneously, one passing through the head and one passing through the body. In the center column we show results using the basic $L^{2,1}$ metric ($\alpha = 0$). At right, we show results when $\alpha = 0.4$. Note that the model becomes more symmetric, at the expense of geometric accuracy.*

We preprocess the model by splitting each triangle that crosses a reflection plane, creating new vertices where the original triangle's edges intersect the reflection plane and replacing the original with three new triangles. The proxy-fitting stage proceeds as previously described. Then, at the beginning of the triangulation stage, we explicitly check each of the split triangles to see if both sides of the triangle were assigned to different patches. If so, we explicitly enforce that the vertices we inserted to perform the split (lying on the reflection plane) be anchor vertices. The rest of the algorithm proceeds as above, with the result that the triangulation now becomes symmetric (figure 6b). In the examples in this paper, splitting triangles adds 3% to the number of triangles.

3.6. Error Metric

While it is not possible to change the amount of weight given to symmetry vs. geometric deformation, since all we do is add more triangles for the priority queue to consider, in certain cases it might be important to consider symmetry more strongly than surface deformation. In these cases, we have observed that the $L^{2,1}$ metric proposed by [CSAD04] is not ideal for preserving symmetry. This is because it considers only normals, which tend to be more sensitive to noise and small deformation than does the geometry itself. Therefore, in order to allow more freedom for the algorithm to capture a model's near-symmetries, we have considered other, more "symmetry friendly" error metrics.

We begin by noting that in the $L^{2,1}$ metric, the distance of a triangle to a proxy depends only on the normal, and indeed the center of the proxy is not relevant, being set as the weighted average of the triangles in the proxy for bookkeeping purposes only. We note that if the metric were simply the weighted Euclidean Distance of the centers of the triangles to the center of the proxy, then the optimal proxy position would also be at the weighted average of the triangles, without consideration of the normal. We expect that such a purely position-based error metric allows the algorithm greater opportunity to capture near-symmetries, at the expense of less-faithful preservation of the original geometry.

Thus, we have investigated a combined error metric, with one term dependent only on positions and one only on normals. Specifically, we take

$$E_{combined} = \alpha \left\| c_{triangle} - c_{proxy} \right\|^2 + (1-\alpha) A_{avg} \left\| n_{triangle} - n_{proxy} \right\|^2,$$

where c and n represent the average positions and normals of triangles and proxies, A_{avg} is the average triangle area (included to ensure that the two terms are dimensionally compatible and the metric is scale invariant), and α is a user-selected parameter. In Figure 7 we show an example of remeshing a bull with $\alpha = 0$ and with $\alpha = 0.4$. Note that the proxies look more symmetric as we increase α, at the expense of a less-faithful reproduction of the original surface.

4. Results

We evaluate the algorithm described in the preceding sections using a number of well-known 3D meshes. Our goal is to produce simplified meshes that maintain and even enhance symmetries, while minimizing geometric approximation error. In the following subsections, we analyze the algo-

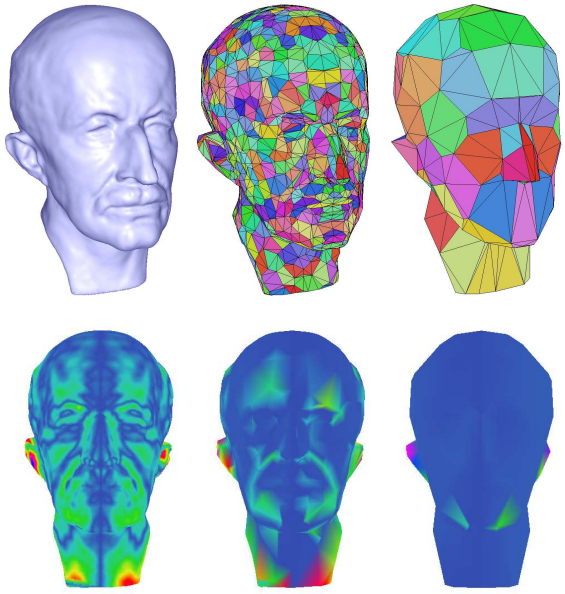


Figure 8: *Increasing the amount of geometric simplification (towards right) results in greater symmetry preservation. Images in the bottom row are colored to represent deviations from symmetry (how far each point is from the reflected surface) with blue indicating perfect symmetry.*

rithm for both single and multiple planes of symmetry and present running times.

4.1. Approximate Symmetries

A first example shows results for a mask (Figure 1): the original model (a) contains 62K faces and is only roughly symmetric. Figure 1(b) shows proxies for the model generated using the unmodified algorithm of Cohen-Steiner et al. [CSAD04]. Note that the proxies are not symmetric. Figure 1(c) shows the result of using our symmetry-aware algorithm: note that the proxies are now symmetric. Since symmetric remeshing allows proxies to have multiple patches, we used 100 proxies in the basic method and 50 proxies in the symmetry-aware version to ensure a similar-quality tessellation. Note that our algorithm (e) produces a triangulation with more symmetry than the traditional method (d), as expected. Moreover, we find that it introduces very little geometric error in the shape approximation as compared to the original Cohen-Steiner algorithm — the RMSD increases by 4%, while the symmetry error (RMSD to reflection) decreases by 50%. This result suggests that more-symmetric topology can be provided for approximately symmetric models at very little cost.

Figure 8 demonstrates that increasing the amount of simplification results in meshes that are more symmetric. At top we show the Max Planck model simplified to 600 and 50 proxies, while at bottom we illustrate deviations from perfect symmetry (blue is most symmetric while green, yellow, and red indicate more asymmetry). This illustrates the property of our algorithm that it automatically captures symmetry if doing so is compatible with the current deformation error.

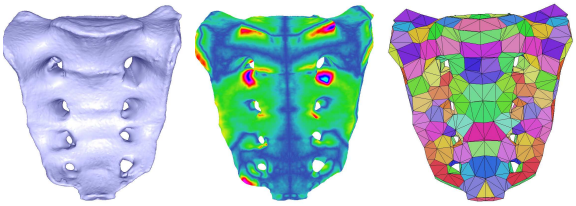


Figure 9: *3D scan of a sacrum (605K polygons, shown at left) remeshed with* 200 *proxies (right). Note that the model is only approximately symmetric (middle).*

A third example is the sacrum in Figure 9 (605*K* faces) which was remeshed using 200 proxies. This is model of a natural object that is close to symmetric yet not perfectly so, as shown in the color-coded visualization at center. In addition, this model was created from a set of 3*D* scans, meaning that high-frequency scanning noise caused further deviations from symmetry. At right we show our remeshed result, produced in just over eight minutes. Our method remains robust despite the high-frequency noise, the presence of holes and the large size of the mesh.

4.2. Multiple Symmetries

Our first example of remeshing while considering multiple planes of symmetry is the bull (Figure 7). This model has separate planes of reflection indicating approximate symmetries of the head and the body. These planes were automatically extracted by finding local maxima of the symmetry transform and iteratively optimizing their position [PSG*06]. Note that the remeshing results in symmetric triangulations for the head and the body with a reasonable (though not symmetric) triangulation at the neck.

Our second example (Figure 10) shows results for the Stanford bunny. While it is intuitively obvious that the bunny has two major planes of symmetry, the result of automatic symmetry analysis is that there are in fact *four* strong symmetries. Figure 3 illustrates these four planes, which capture the separate symmetries of the ears, body, head, and legs of the bunny respectively. We show in Figure 10 the result of remeshing with respect to two, three, and four planes of symmetry. Note that remeshing with only three planes fails to capture the symmetry of the feet, while considering only two planes (through the body and ears — the strongest planes resulting from the symmetry analysis) fails to triangulate the head symmetrically. In contrast, the result at right shows that we are able to take advantage of all four symmetry planes to

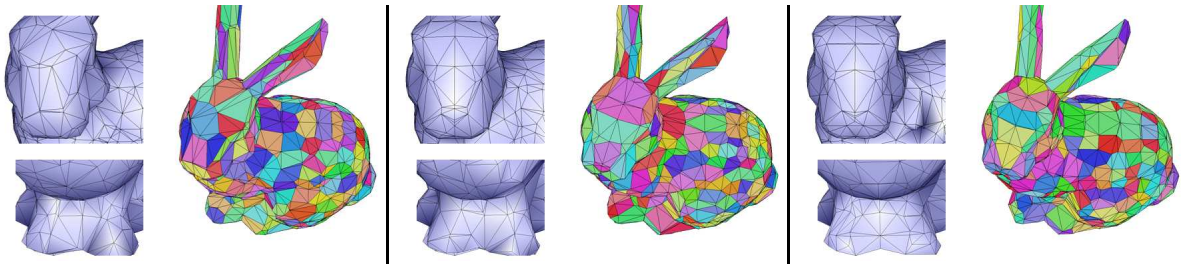


Figure 10: *The bunny is shown here remeshed with 250 proxies, using 2, 3, or 4 planes of symmetry. Note that increasing the number of symmetries results in progressively more symmetric and more intuitive triangulations, while retaining plausible triangulations at the boundaries of symmetric regions.*

produce an intuitive result with locally-symmetric triangulations and smooth transitions between the symmetric regions.

To produce these results, we segmented the input mesh according to the strongest symmetry at each point, then constrained the proxy flooding to only consider one candidate symmetry per triangle. The remeshing itself, however, was still run on the entire model, allowing smooth blends between symmetry regions to be computed automatically.

4.3. Computation Time

We ran our experiments on a 3GHz PC with 1GB of RAM. As a preprocess, for every model we calculated a symmetry transform using a 128x128x128 grid, then ran the Iterative Symmetric Points algorithm (described in [PSG*06]) to refine each of the extracted principal symmetries. The total computation time for extraction of symmetry information took roughly half a minute, with minimal dependence on the size of the original model. While most models exhibit a single strong plane of symmetry, we have extracted up to four principal planes for the models considered in this paper.

Our remeshing algorithm can take up to several minutes for large (i.e., 600k polygons) models. The most time-consuming phase is the distortion-minimizing flooding, which takes $O(m \cdot n \log n)$ time for m iterations on a mesh with n triangles. Note that this time is similar to what would be required for the unmodified remeshing algorithm. In comparison, the time required for anchor and secondary point placement and triangulation is much lower, ranging from a few seconds to half a minute if the model is simplified to have large patches. The total time for the clustering part of the algorithm can be seen in Table 1. Unless otherwise noted, all models in this paper were remeshed with $\alpha = 0.002$.

5. Discussion

Since human beings can recognize symmetry easily, retaining the symmetry of models during simplification is important. We have shown how to modify the existing remeshing technique proposed by [CSAD04] to directly incorporate symmetry without deforming the model too much. In fact, the approximation error introduced by our algorithm can be trivially bounded. The upper bound is simply the error of the unmodified algorithm with the same number of proxies, while the lower bound is the error of the original algorithm using a number of proxies equal to the number of our *patches*. For example, if we remesh using only one plane of symmetry (up to 2 patches per proxy) using K proxies, the error will be between the errors resulting from using the traditional method with K and $2K$ proxies.

We have observed that many models exhibit approximate and partial symmetries, and have demonstrated that our algorithm is able to take advantage of even these imperfect symmetries during remeshing. Moreover, we can handle *multiple* symmetries, producing regions of symmetric tessellation joined by plausible triangulations. While the results shown in the previous sections provide an initial demonstration of opportunities for using and maintaining approximate symmetries during remeshing, our method has limitations that suggest avenues for future work.

First, our approach to symmetric remeshing is limited by the fact that our final edge-flipping technique does not explicitly search for symmetry. While the method works in cases where entire proxies are symmetric, if a proxy contains both symmetric and non-symmetric anchor and secondary points, the edge-flipping technique might adversely affect the triangulation in the symmetric areas. A further weakness of this technique is that running edge-flipping on patches separately, rather than on the entire mesh at once, may reduce the quality of the final triangulation of the model.

Model	# Triangles	# Proxies	Time (s)
Mask	62K	50	62
Igea	130K	50	99
Teapot	8.5K	100	7
Bull	49K	200	72
Max Planck	98K	600	85
Sacrum	605K	200	491
Bunny	69K	30	49

Table 1: *Timing results for the proxy-growing stage of our method, for various models and different numbers of proxies.*

Thus, symmetry-preserving edge-flipping is a topic for future work.

Another limitation of the approach is our non-reflexive method of dealing with symmetry. If for example, one side of a model is reflected only once while the other side is reflected twice, the number of patches for a proxy in that section depends on the starting triangle. This is merely a manifestation of a problem with Lloyd's algorithm - it doesn't converge to a minimum.

There are other limitations of our technique inherited from the underlying Variational Shape Approximation algorithm. For example, the patch flooding requires a computationally expensive iteration, and the final triangulation algorithm is constrained to use original vertices of the model and is not error-driven. Moreover, the algorithm is optimized for the case of significantly fewer output than input polygons, thus making our symmetry-enabled variant inappropriate for symmetric remeshing that preserves all original detail.

A further limitation of this work is that the use of symmetry is necessarily coupled to the error metric. Although we believe that this coupling is theoretically sound and intuitively understandable, it may be inappropriate for some applications. Therefore, a future extension of this work may consider explicitly allowing for greater deformation if doing so will strengthen symmetry.

Finally, we use planar proxies as the foundation of our algorithm, but our algorithm should work for general proxies. An possible avenue of future work is to implement our algorithm using the generalized proxy definitions of [WK05] or [YLW06].

6. Acknowledgments

We wish to thank the members of the Princeton graphics lab, especially Tom Funkhouser and Christopher DeCoro, for their help and valuable comments on this project. Models in this paper were provided by AimShape, Cyberware, Stanford, and MPI. We acknowledge the support of the Air Force Research Lab grant FA8650-04-1-1718, NSF grant CCF-0347427, and the Sloan Foundation.

References

[Ata85] ATALLAH M.: On symmetry detection. *IEEE Trans. on Computers 34* (1985), 663–666.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Transactions on Graphics (Proc. Siggraph) 23*, 3 (2004), 905–914.

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *VIS '98: Proceedings of the conference on Visualization '98* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 263–269.

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM Press, pp. 19–26.

[Hop96] HOPPE H.: Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 99–108.

[KFR04] KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S.: Symmetry descriptors and 3D shape matching. In *Proc. Symposium on Geometry Processing* (2004).

[KLS96] KLEIN R., LIEBICH G., STRASSER W.: Mesh reduction with error control. In *VIS '96: Proceedings of the 7th conference on Visualization '96* (Los Alamitos, CA, USA, 1996), IEEE Computer Society Press, pp. 311–318.

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph. 22*, 3 (2003), 954–961.

[MGP06] MITRA N. J., GUIBAS L., PAULY M.: Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics 25*, 3 (July 2006), 560–568.

[MIK93] MINOVIC P., ISHIKAWA S., KATO K.: Symmetry identification of a 3D object represented by octree. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 5 (May 1993), 507–514.

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. Siggraph) 25*, 3 (July 2006).

[STK02] SHLAFMAN S., TAL A., KATZ S.: Metamorphosis of polyhedral surfaces using decomposition. In *Computer Graphics forum* (2002).

[Tur92] TURK G.: Re-tiling polygonal surfaces. *SIGGRAPH Comput. Graph. 26*, 2 (1992), 55–64.

[TW05] THRUN S., WEGBREIT B.: Shape from symmetry. In *Proceedings of the International Conference on Computer Vision (ICCV)* (Bejing, China, 2005), IEEE.

[WK05] WU J., KOBBELT L.: Structure recovery via hybrid variational surface approximation. In *EUROGRAPHICS* (2005), vol. 24, pp. 277–284.

[WWV85] WOLTER J. D., WOO T. C., VOLZ R. A.: Optimal algorithms for symmetry detection in two and three dimensions. *The Visual Computer 1* (1985), 37–48.

[YLW06] YAN D.-M., LIU Y., WANG W.: Quadric surface extraction by variational shape approximation. *GMP* (2006), 73–86.

[ZPA95] ZABRODSKY H., PELEG S., AVNIR D.: Symmetry as a continuous feature. *Trans. PAMI 17*, 12 (1995), 1154–1166.