Gaussian Sphere sampling based Surface Approximation

Article · January 2008

CITATION
1

READS
301

2 authors, including:

M. Gopi
University of California, Irvine
123 PUBLICATIONS **1,758** CITATIONS

SEE PROFILE

Gaussian Sphere sampling based Surface Approximation

Computer Graphics Lab, Technical Report

Pablo Diaz-Gutierrez
University of California, Irvine
pablo@ics.uci.edu

M. Gopi
University of California, Irvine
gopi@ics.uci.edu

Abstract

Sampling of 3D meshes is at the foundation of any surface simplification technique. In this paper, we use the recent results on quantization and surface approximation theory to propose a simple, robust, linear time, output sensitive algorithm for sampling meshes with the purpose of surface approximation. Our algorithm is based on the mapping of regular sampling and triangulation of the Gaussian sphere onto a manifold surface. An interesting aspect of our algorithm is that we do not explicitly measure, minimize, or prioritize any error to simplify and do not explicitly cluster the faces to find proxies, but still achieve bounded error approximation of the shape.

1 Introduction

Sampling 3D models involves choosing points from the surface such that an appropriate approximation of the surface using these points would faithfully represent the given model, both in terms of geometry and topology. The fundamental challenge in this process is to find the smallest set of sample points to achieve a shape approximation within an error bound. This question will have two different answers, depending on whether the given samples are used for shape approximation or surface reconstruction. Because the topology of the model has to be deduced from the samples, usually surface reconstruction requires many more samples than shape approximation. On the other hand, these two samplings are fundamentally related, as shown in the literature [6]: the sampling size is proportional to the total absolute Gaussian curvature of the surface. While for surface approximation a conventional definition of Gaussian curvature is used, for surface reconstruction the interpretation of the Gaussian curvature at a point on the surface is the square of the curvature of the medial axis ball tangential to that point. This result has been independently proved for surface approximation [8] and surface reconstruction [2, 13].

In this paper we use the above fundamental observation to directly approximate the total absolute Gaussian curvature by regular sampling of the Gaussian sphere and find the appropriate samples on the surface by inverse mapping these Gaussian samples back onto the surface (one-to-many mapping) . The result is a linear time (on the input size), output sensitive, topology preserving and, more importantly, robust sampling algorithm for approximation of surfaces. For convex, genus zero objects, that have positive Gaussian curvature everywhere, this algorithm also follows from the Gauss-Bonnet theorem [33]. This theorem states that the *algebraic sum* of the Gaussian curvature at all points in the object equals the area of the unit sphere. Using the one-to-many inverse function from the Gaussian sphere to the surface, our algorithm can correctly handle other non-covex and higher genus models also. Due to its inherent

Figure 1. The David model was approximated with 2 iterations in less than 2 minutes, reducing the initial 500,000 vertices (left images) to 36,000 (right), but still retaining extremely detailed features.

simplicity, this algorithm has potential uses in streaming and out-of-core simplification of large data sets, as well as parameterization and compression of meshes.

Our method can be used as well to bootstrap other expensive but more accurate simplification algorithms. The proposed concept of processing the Gaussian sphere for mesh processing can have implications in the study of "signal processing" on meshes. In order to illustrate one such mesh processing, we also show an application of this method in identifying, what we call, the *persistent edges* that define important features of the model. These persistent edges in turn can be used to design alternative algorithms to bilateral filtering of meshes.

2 Related Work

Shape approximation is essentially choosing the right set of samples to approximate a surface. Identifying this set of samples is usually done by clustering primitives with similar properties and substituting each cluster with a proxy that best represents the cluster. The clustering is either done locally in a greedy manner, or using global analysis to navigate closer to an optimal solution. The samples can also be directly chosen from the surface depending on some sampling criteria that defines the level of approximation.

Mesh Simplification: There is a wealth of literature on shape approximation, especially for polygonal meshes. Most of these techniques make use of planarity and local anisotropy to adjust the aspect ratio of the polygons to fit the model as close as possible. Greedy and local mesh simplification techniques based on edge-collapse indirectly provide a mesh partition induced by the equivalence class of vertices of the original model that are collapsed to the same vertex in the simplified model [14, 21, 28, 29]. Explicit clustering of faces has also been presented for various applications, including radiosity computation [15], simplified mesh generation [8, 24, 38], surface parameterization [37], and others [25], even extended to image processing [27]. As noted in [8], greedy techniques lead to suboptimal meshes, and most of the other techniques referred above fail to minimize a prescribed geometric error.

Global optimization techniques for mesh simplification like [8, 22, 30, 41] differ in the energy functionals they optimize or the metric they use to measure the error. While most of these methods inherit the topology from the input model, [8] computes the topology of the simplified model as the intersection of proxy planes that approximate the model geometry. The method proposed by [43] clusters mesh normals, similar to [8], but it uses mean shifts, a statistical technique that condones noise in the data.

A shape approximation method that does not directly nor indirectly involve clustering is [7]. First it uses offset-surfaces to identify the solution space within the bounded geometric error, and then it constructs a simpler polyhedron to lie within this solution space.

Sampling for Surface Reconstruction: Direct sampling of surfaces can also be used for shape approximation, and has been widely studied in surface reconstruction literature [1–3, 9, 10, 17] to guarantee topologically correct triangulation. On the other hand for shape approximation, typically the original model and hence the topology is known [26]. This would lead to fewer required samples than for surface reconstruction.

Gaussian Sampling: There is extensive literature demonstrating Gaussian sampling for several applications. For example, [23] cluster face normals in order to quickly extract silhouettes for shadow boundary computation. Surface samples with similar orientation can be accumulated in order to precompute radiance transfer [39]. Convergence of the iterative closest point algorithm can be accelerated by selecting sampling points according to the normal variation they introduce in the shape [36]. In general, as illustrated by [42], Gaussian sampling provides a means to efficiently select data points with *interesting* features, a better approach than sampling at the Nyquist rate if the signal has varying frequency content.

3 Background

In this section we define and discuss a few important terms and results on surface sampling. In his recent work [6], Clarkson develops a generic theory for surface sampling and triangulation called ϵ-nets, using various results by others [18, 32, 34]. [6] also interprets some algorithms for surface reconstruction [1] in the framework of metric space ϵ-nets. Here we review a few of the definitions from that work for completion.

ϵ-covering: Given a metric space (U, D) (U is the space, and D is the distance metric) and a set $\mathrm{S} \subset \mathrm{U}$, if $D(x, \mathrm{S}) \leq \epsilon$ for all $x \in \mathrm{U}$, then S is an ϵ-covering of U.

ϵ-packing: Given a metric space (U, D) and a set $\mathrm{S} \subset \mathrm{U}$, if $D(s, \mathrm{S}\backslash\{s\}) \geq \epsilon \; \forall \; s \in \mathrm{S}$, then S is an ϵ-packing of U.

While ϵ-covering says that there is a sample within a distance ϵ from any point on the surface, ϵ-packing says that these samples are farther than a distance ϵ from each other.

(ϵ_p,ϵ_c)-Delaunay: Given a metric space (U, D) and a set $\mathrm{S} \subset \mathrm{U}$, S is (ϵ_p,ϵ_c)-Delaunay if S is an ϵ_p packing and an ϵ_c covering.

For example, the ϵ-sampling of [1] is Delaunay with $\epsilon_c = \epsilon\phi(x)$ where $\phi(x)$ is the distance between $x \in \mathrm{U}$ and the medial axis, and $\epsilon_p = 0$ since there is no bound on distance between samples. In general, a set S of size n that minimizes $\int_{\mathrm{U}} D(x, \mathrm{S})^2$ must be a Delaunay set [18]. Heuristics for such a problem include Lloyd's algorithm [31] in discrete setting and of Du *et al.* [11] in continuous setting. Specifically, the vertices of the polygonal ap-

proximation achieved by the Variational Shape Approximation (VSA) algorithm [8] is a Delaunay set. This algorithm, which incidentally invokes Lloyd's method, uses $\mathcal{L}_{2,1}$ metric that measures the distance between two points as the Euclidean distance between the unit normal vectors at these points $||\mathbf{n}(x) - \mathbf{n}(y)||$.

ϵ-net: If S is (ϵ, ϵ)- Delaunay, then it is called an ϵ-net of U.

While [19] prove results on optimal aspect ratio of the triangles in surface approximation, for Delaunay sets and ϵ-nets, [6] and [18] prove results on number of samples required to approximate a compact space: Let (U, D) be a compact metric space with dimension d and metric tensor q – a function that tells how to compute the distance between points using D. The area measure (as measured by D) is $\mu = \int_{\mathrm{U}} \sqrt{\mathtt{det}(q)}$. Then the size of the ϵ-net of U is $\Theta(\mu(\mathrm{U})/\epsilon^d)$.

For example, the metric tensor q of the metric $D = \mathcal{L}_{2,1}^2$ [8] is square of the Hessian – $\mathbf{H}^2$, the area measure $\mu = \int_{\mathrm{U}} \sqrt{\mathtt{det}(q)} = \int_{\mathrm{U}} \mathtt{det}(\mathbf{H}) = \int_{\mathrm{U}} |\kappa_1(x)\kappa_2(x)|dx$ – the total absolute Gaussian curvature of the surface, and its sample size of the ϵ-net is proportional to μ/ϵ^2. Similar result on sample size for surface reconstruction algorithm was proved by [13].

3.1 Analysis of $\mathcal{L}_{2,1}$ Metric

The use of $\mathcal{L}_{2,1}$ metric, also called the *isophotic metric*, is appropriate when the sampling should be proportional to the total absolute Gaussian curvature of the model. This metric also captures the anisotropy of the model: For a quadratic surface centered at a point $\mathbf{p}$, the projection of the locus of the points whose surface normal have constant angle deviation from the normal at $\mathbf{p}$ (also called an *isophote*) on the tangent plane at $\mathbf{p}$ is an ellipse [16, 35]. In a general model, this ellipse captures the anisotropy of a surface around a point as shown by [8]. Further the surface approximation generated by this metric has guaranteed geometric error-bound: An ϵ-covering of a slightly perturbed version of this isophotic metric (in order to make the area measure from positive semi-definite to positive definite) has proven Hausdorff bounds for the corresponding triangulation [6].

On the other hand, computing distances between two points on the surface using $\mathcal{L}_{2,1}$ metric requires implicitly mapping their normal vectors onto the Gaussian sphere and computing the distance in that space. This mapping would map a planar region or a line with Gaussian curvature zero to a point on the Gaussian sphere. This leads to non-convergence of k-means clustering planes during surface approximation [8].

Inspired by the problem definition of [8] and the theory of ϵ-net by [6] and [18], we use the $\mathcal{L}_{2,1}$ metric and a novel inverse mapping of the Gaussian sphere samples to the input mesh to find the surface approximation.

3.2 Basis of our Algorithm

Argument: [8] uses $\mathcal{L}_{2,1}$ metric, which is the distance between the normal vectors in the Gaussian sphere, and uses the k-means algorithm to minimize the integral square of the distance. As pointed out by [6], Gruber's results [18] show that an alternate algorithm would be to use an ϵ-net in the $\mathcal{L}_{2,1}$ distance on the mesh. In our algorithm we propose a simple reinterpretation of the $\mathcal{L}_{2,1}$ ϵ-net on the mesh as the Euclidean metric ϵ-net on the Gaussian sphere.

Conceptual Algorithm: We create an Euclidean metric ϵ-net of a sphere by regular sampling and triangulation of the sphere. Then we find the preimages of the vertices of the triangulation of the Gaussian sphere in the given model (this is a one-to-many mapping for non-convex objects) and choose those points as the samples for final triangulation of the model.

The regular sampling and triangulation is achieved by recursively subdividing the faces of a regular polyhedron (a tetrahedron or an octahedron) and pushing the resulting vertices to the boundary of a sphere. Let the (Euclidean) distance between the vertices of this triangulation of the sphere be ϵ (which can be controlled in discrete steps by the level of subdivision). The pre-images of the faces of this regular triangulation of the Gaussian sphere back in the original model partitions the input mesh into regions within which the normal deviation is within bounds (ϵ) as given by $\mathcal{L}_{2,1}$ metric. Further, the pre-images of the vertices of the triangulation of the Gaussian sphere are also at least ϵ distance away from each other on the mesh as measured by the same metric. Hence our sampling is an ϵ-net sampling in the $\mathcal{L}_{2,1}$ metric in the given mesh.

4 Algorithm Overview

Consider a regular triangulation of the Gaussian sphere - for example, approximation of the sphere by recursive subdivision of an octahedron. In our terminology, we name the vertices, edges and triangles of the Gaussian sphere approximation as *Gauss vertex, edges and triangles*. The vertices, edges and triangles of the input mesh are called simply *vertices, edges and triangles*. The basis of our shape approximation algorithm (outlined in Algorithm 1) is that surface regions whose normal vectors lie inside the same Gauss triangle are considered to be *featureless* on the surface. Note that *featureless* does not necessarily mean that the region is planar, and so would require triangulation of the boundary of this *featureless* region to approximate it within error bounds.

For every edge in the mesh, if the normal vectors of its two incident mesh triangles belong to different Gauss triangles, then that edge is considered to be a *feature edge* (Figure 3). A mesh vertex is a *feature vertex* if more than

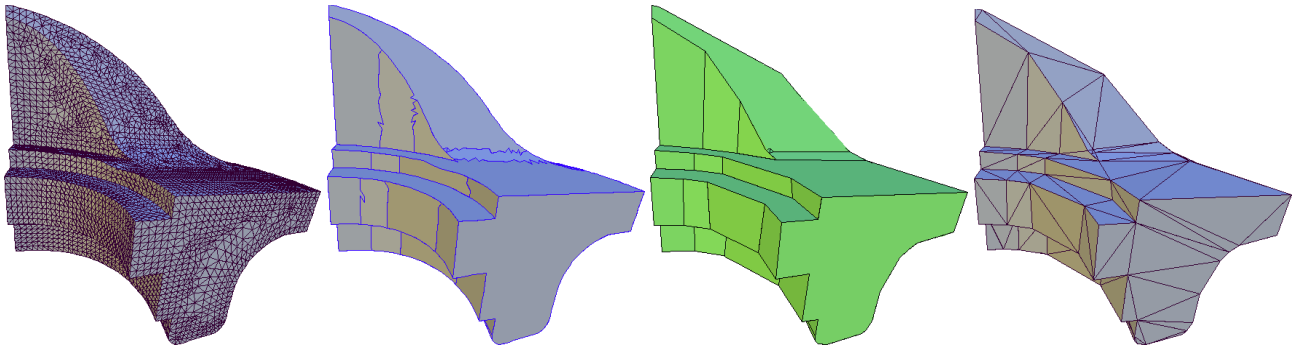

Figure 2. Steps of the shape approximation procedure, from left to right. **First:** Input mesh with 6475 vertices. **Second:** Feature edges separate triangles with significantly different normals. **Third:** Connecting feature vertices produces a polygonization of the mesh. **Fourth:** The triangulation of the polygons approximates the original shape with 85 vertices.

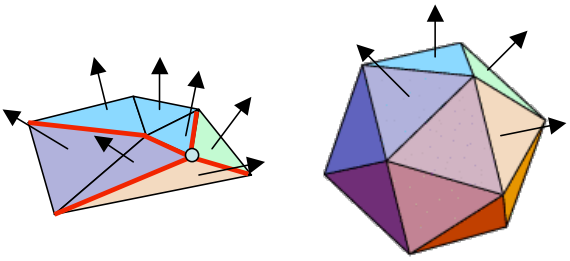

Figure 3. The input mesh is shown in the left and the Gaussian sphere on the right. Each input mesh triangle is color coded to match the corresponding triangle of the Gaussian sphere that contains the normal of that mesh triangle. The edges that have two different color coded triangles on either side are *feature edges*. A mesh vertex with more than two feature edges incident on it is a *feature vertex*. Since, around the end vertices of a feature edge, transition of the normal vector from one Gauss triangle to another has to happen at least once more, a manifold mesh vertex cannot have exactly one feature edge incident on it.

two feature edges are incident on it. Note that by the definition of a *feature edge*, a mesh vertex cannot have exactly one *feature edge* incident on it. Hence the input manifold mesh is partitioned into *featureless* regions whose boundary is made up of a closed sequence of *feature edges* (Figure 2). This boundary would contain many *feature vertices*. Connecting these *feature vertices* in order by straight lines would give a closed polygon approximating the *featureless region* it encloses. Each of these polygons is triangulated to get the approximation of the input mesh. Note that the vertex set of this approximation is a subset of the input mesh. The above process is iterated on the output mesh with random orientations of the Gaussian sphere until the vertex set (and hence the number of vertices) converges. Figure 2 shows results after different stages of the algorithm leading to the approximated model at the end that will serve as input to the next iteration. Arguments for the convergence of the algorithm are presented in Section 5.

Algorithm 1 Surface Approximation (Triangle Mesh M)

TriangulateSphere (Sphere G, int $level$)
M' = ϕ
while $M \neq M'$ **do**
  $M' = M$
  G' = Arbitrary rotation of G.
  Identify feature edges of M using G'.
  Identify partitions p and their boundary feature edges.
  Identify feature vertices along the partition boundaries.
  Connect feature vertices in order by edges to form polygons.
  M = Triangulate all Polygons. {Approximated model}
end while

Thus the algorithm, in a single sweep through all the input edges, finds all the feature edges and vertices and their connectivity. Only these feature edges and vertices are used in subsequent stages of the algorithm, including the triangulation and further iterations. Hence our algorithm is linear in the input number of faces and has a worst case complexity of $O(k^2)$, where k is the number of vertices in the output. Since k depends only on the complexity of the *shape* of the input model and not on the number of vertices or faces in the input mesh, our algorithm is output sensitive. Interestingly, we do not have to explicitly compute any error in the original mesh, minimize or prioritize it for simplification. We also do not have any explicit clustering of faces.

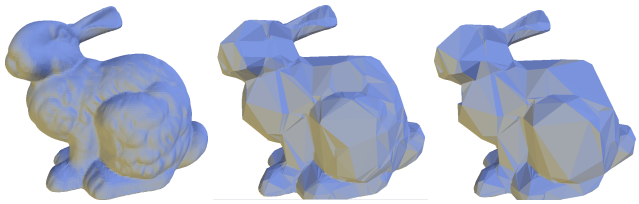

Figure 4. Iterative mesh approximation process after 2 and 9 iterations, reducing the model to 2344 and 989 vertices from the original 35947 vertices (left). The first two iterations required a total of 8 seconds, while it took just 7 more seconds to finish all nine iterations, showing that the method is output sensitive.

5 Analysis of the Algorithm

Avoiding Degeneracies: The only degenerate case in the algorithm above happens when the normal vectors of the faces get mapped exactly to a Gauss vertex or Gauss edge of the spherical triangulation. Extensive studies on geometric degeneracies point out that such cases occur with probability zero [12]. Further, by the definition of degeneracy, it can be removed by a slight perturbation of the orientation of the Gaussian sphere. Since, statistically, a face normal can never be mapped to a Gauss vertex, it points to the fact that in the given mesh, the simplices of our interest cannot be the faces, but only edges or vertices. For the same reason, unlike face-clustering based shape approximation algorithms, our algorithm takes a complementary approach by analyzing the edges to identify "feature" edges and vertices. Thus our algorithm does not suffer from any of the degeneracies that other algorithms encounter in the planar regions of the model which are quite common in many mechanical parts.

Scope and limitations of the algorithm: Although we use the same distance metric as [8], by the nature of the mapping function between the model and the Gaussian sphere, we do change the scope of the algorithm. Specifically, [8] can approximate shapes using the given number of polygons, implicitly allowing almost continuous values for ϵ in ϵ-net. We allow only discrete values of ϵ which in turn determine the number of output polygons based on the geometric complexity of the input model. A fundamental difference between the two algorithms is that while VSA allows the user to set a proxy count and then it minimizes the approximation error, in our approach the user sets the error bound, and the algorithm finds an approximation with as few proxies as possible.

In our algorithm, the sampling on the surface changes by different orientations of the Gaussian sphere (or the input model). But the error bounds of the surface approximations given by any of these samplings is the same, since the value of ϵ in the ϵ-net is dependent only on the level of subdivision. Finding the correct orientation and triangulation of the sphere in order to minimize the number of vertices, and hence the facets, for a given error bound is NP-hard as can be derived from [7].

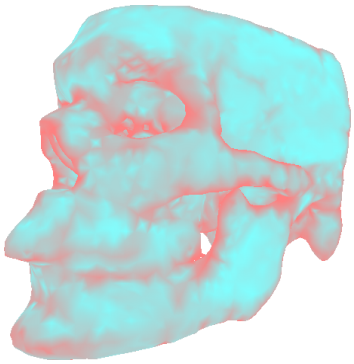

Figure 5. Color-coded importance of the vertices of the skull model. Vertices that appear most frequently as features are coded red. Albeit equivalent in terms of approximation error, different Gaussian sphere orientations retain different edges. Some of these edges show up in most configurations, indicating they are *persistent features*.

We reduce the number of samples by iterating over different random orientations of the Gaussian sphere. At convergence, the resulting vertices, which are elements of the original input vertex set, are farther than ϵ $\mathcal{L}_{2,1}$ metric distance away from each other (irrespective of the orientation of the Gaussian sphere). Also, by the nature of the algorithm, every point in the input model is within an ϵ distance from at least one sample point. Note that the vertex set of the approximation after every iteration is a subset of the vertex set obtained after the previous iteration. The final vertex set at convergence is determined by the the sequence of the orientations of the Gaussian sphere at each iteration. But identifying the correct sequence of Gaussian sphere orientations to get the minimum number of vertices for a given error bound is also NP-hard.

In the context of changing feature vertices and edges in the input mesh for different orientations of the Gaussian sphere, we introduce the concept of "persistence of a feature". Over multiple runs of finding the feature edges and vertices for different random orientations of the Gaussian sphere, the relative frequency with which an edge or a vertex is marked as feature denotes the persistence of that feature (Figure 5). Edges and vertices with large persistence (for a given level of subdivision of the Gaussian sphere) can be considered as relevant features and can be used in other mesh processing applications, includeing feature preserving filtering.

Convergence analysis:

Convergence has to be argued for two cases: first, over

multiple iterations of the algorithm with different orientations of the Gaussian sphere (with fixed tessellation), and second comparing the final results of the algorithm for different levels of subdivision of the Gaussian sphere.

First, since the value of ϵ remains the same through successive iterations of the algorithm with the same tessellation of the Gaussian sphere (independent of its orientation), the theory of ϵ-net ensures that the sampling size, which is proportional to the total absolute Gaussian curvature, converges. Intuitively, the output vertex set being a subset of the input vertex set within bounded normal deviation, ensures that the overall shape of the object is preserved through iterations.

Second, for convergence of the results of the algorithm over finer tessellations of the Gaussian sphere, we would like to note that we inherit the properties associated with $\mathcal{L}_{2,1}$ metric and ϵ-net such as the convergence of Hausdorff distance between the simplified model and the input mesh. If the Hausdorff distance converges, [20] proves that that following are equivalent: (a) normal field convergence, (b) area convergence (c) metric tensor convergence, and (d) Laplace-Beltrami operator convergence. Since in our case, normal field converges over repeated subdivision of the Gaussian sphere, our metric tensor which is the square of the Hessian $\mathbf{H}^2$ also converges, thus guaranteeing the convergence of the total absolute Gaussian curvature.

6 Implementation Details

The implementation of our surface approximation algorithm has three components: Gaussian sphere object implementation, mesh processing, and triangulation.

The Gaussian sphere object is a unit sphere that is regularly sampled and triangulated by subdividing a regular octahedron to a user-specified level. The only query on the Gaussian sphere object is identifying the Gauss triangle, called the *pierced triangle*, in which a given normal vector lies (pierces it). This object was implemented using the publicly available *qhull* [40] library.

The ideal triangle mesh representation for our algorithm is the half-edge data structure (refer to Algorithm 1.) An edge is tagged as a feature edge if the normal vectors of its two incident triangles have different *pierced triangles*. These feature edges partition the input mesh. We find the triangles belonging to a partition by a simple flooding operation on the input mesh. In order to allow an arbitrary triangle traversal order while flooding each partition, we use the union-find algorithm. The orientation of the partition boundaries is given by the half-edges of the feature edges. Next, we identify the feature vertices as those that have more than two feature edges incident on them. These feature vertices are then connected by straight line edges, which we call *wires*, in counterclockwise order around each partition boundary. The feature vertices along with the edges connecting them, form the boundary polygons of the partitions. Finally we triangulate the boundary polygons of each partition using the method suggested in [8] to get a geometric approximation of the partitions and hence the surface approximation of the entire model. The above algorithm is repeated on the output model of the previous iteration. At the beginning of every iteration, the Gaussian sphere is randomly rotated. Iterations continue until the (feature) vertex set converges.

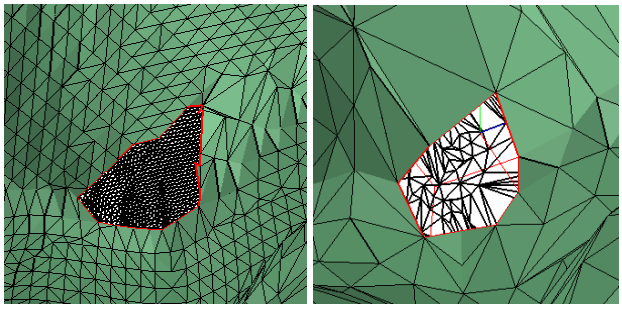

Figure 6. By tagging the boundaries of the model as feature edges for the approximation, we can ensure that they are retained in the simplified output. In the figures above, a hole in the chest of the bunny model remains after simplification and is well approximated by our algorithm.

Handling Meshes with Boundaries: We handle meshes with boundaries by just tagging the boundary edges of the mesh as feature edges. Interestingly, while boundary curves on the mesh which are close to geodesics are approximated automatically (Figure 6), other boundary curves whose Frenet frame normal vector is perpendicular to the mesh normal are filled. Such boundaries can also be thought of as undersampled regions in the scanning of the model, and there are a few reconstruction algorithms that fill this gap [4].

6.1 Geometric and Topological Issues

The geometric and topological issues related to the core of our algorithm are simple and can be handled easily. For example, as we mentioned earlier, the normal vector of a face getting mapped to a vertex or an edge of the triangulation of the sphere is a degenerate case. Since we repeat this process over different orientations of the sphere in successive iterations, initial errors, if any, in the classification of feature edges will be corrected automatically and summarily in the subsequent iterations without any special intervention.

The second issue is handling the partitions introduced by mesh noise. There are four cases, illustrated in Figure 7: a partition polygon boundary with (a) one or (b) two feature

vertices, and a feature vertex with (c) one or (d) two feature edges incident on it. (Note that a feature vertex with one feature edge does not occur naturally, but as a consequence of removal of other noisy features.) Case (a) is handled by removing the entire polygon. In Case (b), there exists two paths of feature edges between the two feature vertices. One of these paths is removed. In Case (c), both the feature vertex and edge are removed, and in Case (d) only the feature vertex is removed unless it is tagged as a special vertex (see below).

The final issues are related to the number of feature vertices used to triangulate a partition of the input mesh. First, if a partition is best approximated by a non-convex polygon, it is possible that the polygon outline (obtained by connecting adjacent feature points along the partition boundary) is self intersecting. The triangulation procedure will introduce flip-triangles, or foldings not present in the input shape. Second, in a configuration like the one just described, it is possible that the triangulation method produces non-manifold edges if the distances measured along the interior and the boundary of the partition are not consistent. These problems, illustrated in Figure 7, can be prevented by subdividing the polygon boundary before triangulation whenever the above condition is detected.

7 Results

We have run our algorithm on several models featuring a varied array of geometric features, topology, size and triangulation density (refer to Figures 1, 11, 12, 13), 14) and 15). Further, we used different sampling densities on the Gaussian sphere, in order to verify our claim that higher normal sampling density induces more accurate geometric approximation, converging to the original shape in the limit. This result is illustrated in Figure 8. We measured the Hausdorff distance between the original model and its approximation computed with our algorithm using the Metro tool [5]. The results for various models are plotted in Figure 9. As we can see, even with a low number of Gauss sphere samples, the error becomes very small for all models, when taken as a percentage of the diameter of the model.

For a fixed subdivision level of the Gaussian sphere, the quick convergence of the algorithm as we iterate with random orientations of the Gaussian sphere is also demonstrated in Figure 10. It shows the relative approximation error for different models against the number of iterations.

Finally, we would like to highlight the anisotropy of the approximating shape, which can be observed in most models although it is more apparent, for example, in mechanical parts. When mapped back to the model, a uniform sampling of its normal space produces greater sampling density along the direction of the maximum principal curvature, and lower density along the minimum principal curvature.

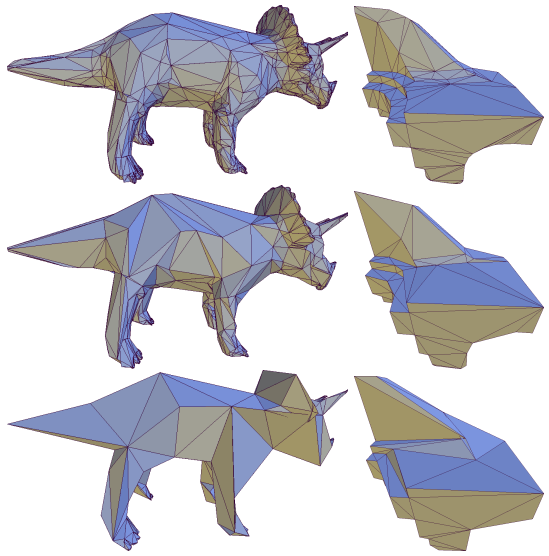

Figure 8. Triceratops (2832 vertices) model approximated with 1650, 820 and 202 vertices. For the fandisk (6475 vertices), we use 146, 69 and 42 vertices. As we make the Gaussian sampling more coarse (from top to bottom), the approximation becomes less accurate. To sample the Gaussian sphere we subdivide the faces of an octahedron. From top to bottom, we use 3, 2 and 1 subdivisions, which corresponds to 512, 128 and 32 samples, respectively.

This property permits approximating a shape using fewer samples than if a uniform, object space sampling had been used [19].

8 Future Work

There are many avenues for future work especially in the context of processing the Gaussian sphere for mesh processing.

In Section 5 we discussed about the notion of persistence of a feature edge. The category of bilateral filterin algorithms for smoothing depends on a previous identification of such relevant features, in order to bound the smoothing to the interior of featureless regions. It seems interesting to further explore this connection.

Other two promising directions for future development include non-uniform Gaussian sampling and off-core implementation of the method. In the former, a varying sampling density fot the Gaussian sphere could be used to represent a notion of importance sampling, with applications, for example, in rendering. In the latter, the algorithm would be extended to handle currently intractably large models. Because ours is an output sensitive algorithm, it is reasonable to expect that a single-pass feature detection would be

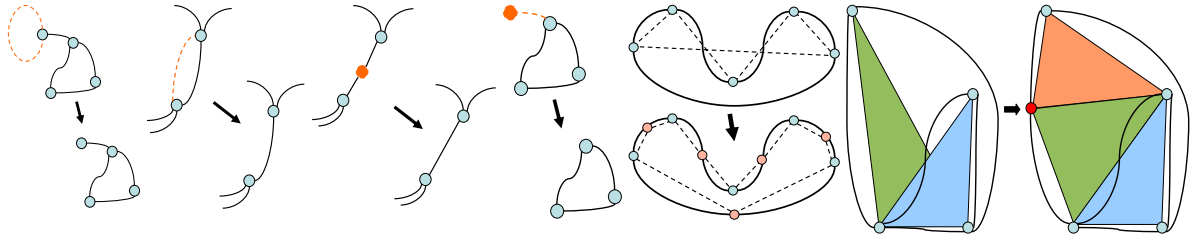

Figure 7. Some configurations of the mesh partitioning would yield degenerate polygons if left unprocessed. Here we show the 4 considered cases and how they are cleansed before constructing the polygons. From left to right: Polygon boundary with (a) one or (b) two feature vertices; Feature vertex with only (c) one or (d) two incident feature edges. Continuing, on the right, we exemplify the possibility of flip-triangles and non-manifold edges in the resulting triangulation due to insufficiently sampled partition boundaries. Both problems can be prevented by tagging some intermediate vertices to be treated as feature vertices.

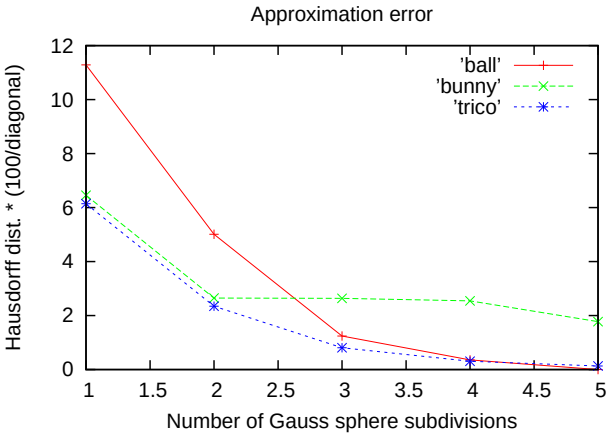

Figure 9. Approximation error for several models as we increase the sampling density in the Gaussian sphere. The number of samples is increased by a factor of four with every subdivision, starting from the 8 samples in an octahedron. The approximation error is measured as the Hausdorff distance (maximum symmetric distance) between the original model and its approximation, and expressed as a percentage of the diagonal of the model.

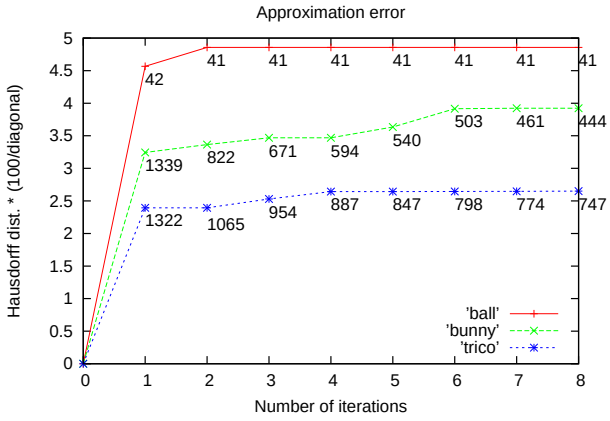

Figure 10. Approximation error for several models as we increase the number of iterations of the algorithm. The X axis represents this number of iterations, while the Y axis indicates the relative Hausdorff distance to the original model (as a percentage of the diagonal). Each data point is labeled with the number of vertices in the approximation.

enough to gather all the relevant information for the approximation, which would be carried over the set of feature edges only.

9 Conclusion

In this paper, we have presented a novel and theoretically sound surface approximation technique that is output sensitive, runs in linear time on the input, and is robust with guarantees on error bound. The method is based on the fact that sampling is proportional to the total absolute Gaussian curvature. Since any surface area approximating sampling should uniformly sample the surface, it is logical to regularly sample the Gaussian sphere in order to approximate the total absolute Gaussian curvature. We thus create what is called the ϵ-net sampling in the Euclidean distance metric on the Gaussian sphere, which translates to ϵ-net sampling in the $\mathcal{L}_{2,1}$ metric on the given surface.

Surface approximation techniques like ours have innumerable applications in compression, parameterization, and hierarchical geometry processing, to name a few. Hence strong theoretical foundations for appropriate approximation becomes compulsory. We believe that this work has preliminary leads to investigate fundamental principles of sampling a surface for different applications which may be similar to Nyquist sampling theorem for one dimensional time varying signals.

Acknowlegments: We would like to thank all our model sources including the Digital Michelangelo Project, Stanford University, AIMSHAPE, and other online free model

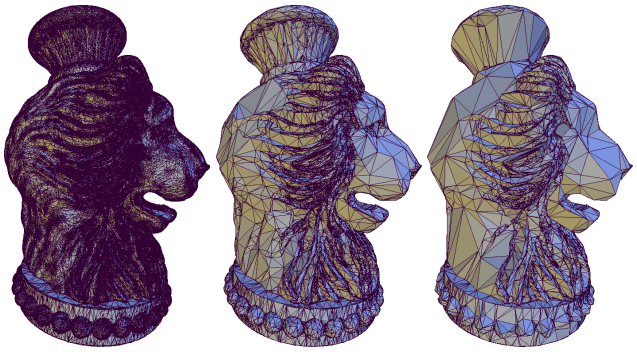

Figure 11. The 200000-vertex lion model (left) simplified to 40000 vertices (middle) and then to 12200 (right). The high ratio between principal curvatures in some regions produces noticeable anisotropic triangles. Gaussian mesh sampling preserves fine details of the input shape.

distributors.

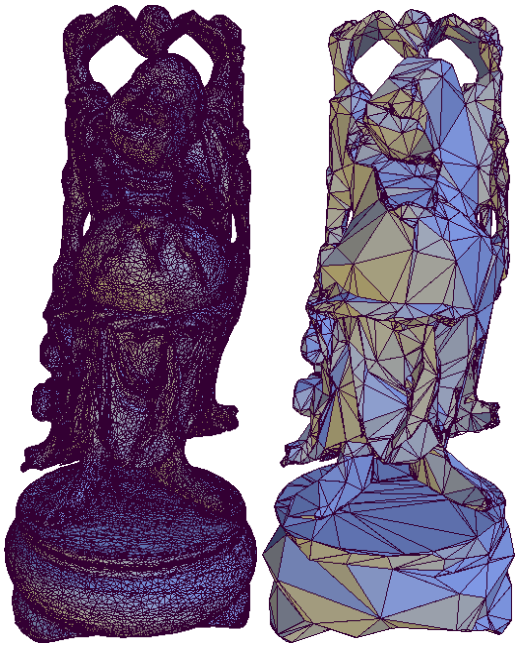

Figure 12. Polygonal approximation of the genus-6 Happy Buddha model, demonstrating that our method makes no assumptions on the genus of the input mesh. A total of 3444 vertices are kept, 6.8% of the initial 50000.

References

[1] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *SIGGRAPH*, pages 415–421, 1998.

[2] N. Amenta, S. Choi, and R. Kolluri. The power crust. In *Solid Modeling*, pages 249–260, 2001.

[3] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Vis.*, 5(4), 1999.

[4] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH*, pages 67–76, New York, NY, USA, 2001. ACM Press.

[5] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Eurographics*, 17(2):167–174, June 1998.

[6] K. L. Clarkson. Building triangulations using ϵ-nets. In *SIGACT Symposium*, 2006.

[7] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *SIGGRAPH*, pages 119–128, New York, NY, USA, 1996. ACM Press.

[8] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM ToG.*, 23(3):905–914, 2004.

[9] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. In *Sympos. Comput. Geom.*, pages 330–339. ACM, 2004.

[10] T. K. Dey, K. Mehlhorn, and E. A. Ramos. Curve reconstruction: Connecting dots with good reason. *CGTA*, 15(4):229–244, 2000.

[11] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676, 1999.

[12] H. Edelsbrunner and E. P. Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM ToG*, 9(1):66–104, 1990.

[13] J. Erickson. Nice point sets can have nasty delaunay triangulations. In *Symp. on Computational Geometry*, pages 96–105, New York, NY, USA, 2001. ACM Press.

[14] M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Vis.*, pages 263–269, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[15] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *I3D*, pages 49–58, New York, NY, USA, 2001. ACM Press.

[16] M. Gopi. *Theory and Practice of Sampling and Reconstruction for Manifolds with Boundaries*. PhD thesis, Department of Computer Science, UNC Chapel Hill, 2001.

[17] M. Gopi, S. Krishnan, and C. Silva. Surface Reconstruction using Lower Dimensional Localized Delaunay Triangulation. *Eurographics*, 19(3):467–478, 2000.

[18] P. M. Gruber. Optimum quantization and its applications. *Adv. Math.*, 186:456–497, 2004.

[19] P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Comput. Geom. Theory Appl.*, 14(1-3):49–65, 1999.

[20] K. Hildebrandt, K. Polthier, and M. Wardetzky. On the convergence of metric and geometric properties of polyhedral surfaces. ZIB Preprint ZR-05-24, 2005.

[21] H. Hoppe. Progressive meshes. In *SIGGRAPH*, pages 99–108. ACM SIGGRAPH, 1996.

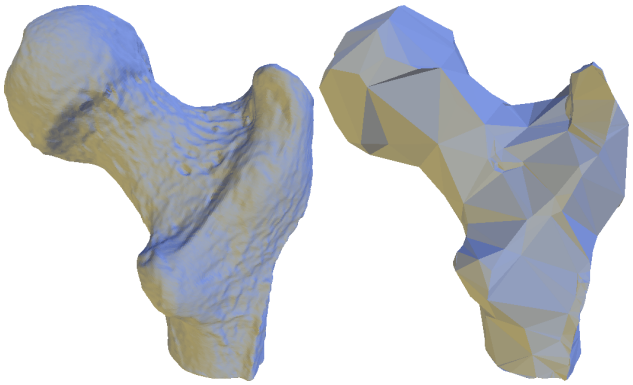

Figure 13. In more natural shapes we observe large polygons in planar regions, as well as smaller ones around fine details of the surface. The bone figure preserves 796 vertices out of 137000, that is, 0.5%.

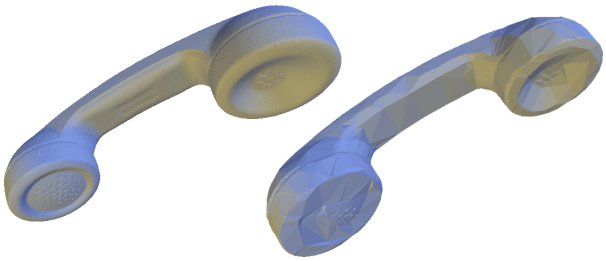

Figure 14. Phone model approximated with 2913 vertices instead of the original 83044, roughly 3.5%. Man made shapes often have large featureless parts, which can be approximated with few polygons.

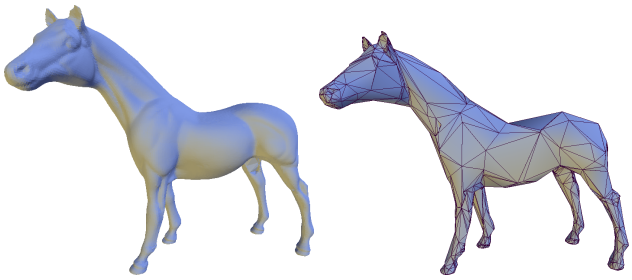

Figure 15. Feature sensitive surface approximation of the horse model, clearly following the anisotropy of the shape. The 48485 input vertices are reduced to 925 (1.9%).

[22] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *SIGGRAPH*, pages 19–26. ACM SIGGRAPH, 1993.

[23] D. E. Johnson and E. Cohen. Spatialized normal come hierarchies. In *I3D*, pages 129–134, New York, NY, USA, 2001. ACM Press.

[24] A. D. Kalvin and R. H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Comput. Graph. Appl.*, 16(3):64–77, 1996.

[25] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM ToG.*, 22(3):954–961, 2003.

[26] Y.-K. Lai, Q.-Y. Zhou, S.-M. Hu, J. Wallner, and H. Pottmann. Robust feature classification and editing. *IEEE Vis.*, 13(1):34–45, 2007.

[27] G. Lecot and B. Levy. Ardeco: Automatic region detection and conversion. In *EGSR*, 2006.

[28] C. H. Lee, A. Varshney, and D. W. Jacobs. Mesh saliency. In *SIGGRAPH*, pages 659–666, New York, NY, USA, 2005. ACM Press.

[29] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *IEEE Vis.*, pages 279–286, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[30] P. Lindstrom and G. Turk. Image-driven simplification. *ACM ToG.*, 19(3):204–241, 2000.

[31] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inform. Theory*, 28:129–137, 1982.

[32] E. Nadler. Piecewise-Linear Best l_2 Approximation on Triangulations. In C. K. Chui, L. L. Schumaker, and J. D. Ward, editors, *Approximation Theory V*, pages 499–502. Academic Press, 1986.

[33] B. O'Neill. *Elementary Differential Geometry*. Academic Press, www.apnet.com, 2^{nd} edition, 1997. 482 pages.

[34] H. Pottmann, R. Krasauskas, B. Hamann, K. Joy, and W. Seibold. On Piecewise Linear Approximation of Quadratic Functions. *Journal of Geom. Graphics*, 4(1):31–53, 2000.

[35] H. Pottmann, T. Steiner, M. Hofer, C. Haider, and A. Hanbury. The isophotic metric and its application to feature sensitive morphology on surfaces. In *ECCV (4)*, pages 560–572, 2004.

[36] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

[37] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *SGP*, pages 146–155, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[38] A. Sheffer. Model simplification for meshing using face clustering. *CAD*, 33:925–934, 2001.

[39] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered principal components for precomputed radiance transfer. In *SIGGRAPH*, pages 382–391, New York, NY, USA, 2003. ACM Press.

[40] M. M. The Geometry Center. http://www.qhull.org.

[41] G. Turk. Re-tiling polygonal surfaces. In *SIGGRAPH*, pages 55–64, New York, NY, USA, 1992. ACM Press.

[42] M. Vetterli, P. Marziliano, and T. Blu. Sampling signals with finite rate of innovation. *IEEE Transactions on Signal Processing*, 50(6):1417–1428, 2002.

[43] H. Yamauchi, S. Lee, Y. Lee, Y. Ohtake, A. Belyaev, and H.-P. Seidel. Feature sensitive mesh segmentation with mean shift. In *Intl. Conf. on Shape Modeling and Applications*, pages 238—245, Washington, DC, USA, 2005. IEEE Computer Society.