Visual Comput (2006) 22: 612–621
DOI 10.1007/s00371-006-0052-0

ORIGINAL ARTICLE

Rui Wang
Kun Zhou
John Snyder
Xinguo Liu
Hujun Bao
Qunsheng Peng
Baining Guo

Variational sphere set approximation for solid objects

Published online: 17 August 2006
© Springer-Verlag 2006

K. Zhou (✉) · B. Guo
Microsoft Research Asia
{kunzhou,bainguo}@microsoft.com

R. Wang* · X. Liu · H. Bao · Q. Peng
State Key Lab of CAD & CG, Zhejiang University
{rwang,xgliu,bao,peng}@cad.zju.edu.cn

J. Snyder
Microsoft Research
johnsny@microsoft.com

Abstract We approximate a solid object represented as a triangle mesh by a bounding set of spheres having minimal summed volume outside the object. We show how outside volume for a single sphere can be computed using a simple integration over the object's triangles. We then minimize the total outside volume over all spheres in the set using a variant of iterative Lloyd clustering that splits the mesh points into sets and bounds each with an outside volume-minimizing sphere. The resulting sphere sets are tighter than those of previous methods. In experiments comparing against a state-of-the-art alternative (adaptive medial axis), our method often requires half as many spheres, or fewer, to obtain the same error, under a variety of error metrics including total outside volume, shadowing fidelity, and proximity measurement.

Keywords Variational approximation · Solid objects · Shadow · Collision detection

1 Introduction

Object approximation using sets of simple primitives plays an important role in many time-critical applications in computer graphics, such as collision, hit, and proximity detection, view-frustum and occlusion culling, visibility and ray intersection queries, and shadowing. Many different geometric primitives have been used including spheres [3, 7, 15, 19], axis-aligned bounding boxes (AABBs) [2], oriented bounding boxes (OBBs) [5, 11], and discrete oriented polytopes (k-DOPs) [10]. Typically, a trade-off exists between the primitive's simplicity and its fitting flexibility. A sphere set is probably the simplest representation to update for dynamic geometry and to perform collision-related queries on, and so has found widespread use in applications.

Our goal is to bound an object as tightly as possible using a user-specified number of spheres, denoted n_s. Previous methods have involved arbitrary decisions and fitting metrics not directly tied to bounding tightness, and have been too local/greedy in their fitting approach. Our solution introduces two contributions: a new metric that more directly measures tightness of fit, and a global, variational method that directly minimizes this metric.

Our novel fit function is based on the volume between the sphere set and the original object, called the *outside volume* (Fig. 1c). In particular, our notion of error ignores overlaps in the sphere set as long as they occur in the object's interior; only sphere volume "sticking out" of the object is penalized. Neglecting interior overlap provides much-needed flexibility and satisfies our intuition that for many applications, including those related to collision de-

*This work was done while Rui Wang was an intern at Microsoft Research Asia.

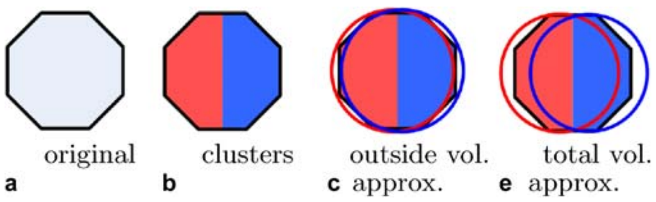

Fig. 1a–d. Solid approximation (2D). Our approximation (**c**), which minimizes outside volume, is more effective than methods (**d**) minimizing total volume of the bounding elements (blue and red circles)

tection and (opaque) visibility, interior overlap is irrelevant.

To find the sphere centers and radii, we apply a variant of the variational approach in [4] using spheres as the fit primitive and outside volume as the fit function. The method converges to an approximation minimizing outside volume. For a given n_s, we demonstrate that our sphere sets are much tighter than those produced by other methods, and produce better results in a variety of applications.

2 Related work

Sphere set approximation can be computed by subdividing an object into an octree and placing spheres at each non-vacant leaf node [6, 12]. The arbitrary subdivision grid wastes many spheres.

Quinlan [15] uses top-down recursive splitting (RS). It first covers the surface with uniformly-sized spheres, splits the resulting sphere set using the longest axis of its bounding box, and then recursively visits the two child lists. RS is used for constructing sphere-trees for visibility culling and level-of-detail rendering [18] and for collision detection on deformable models [9]. An alternative is to merge similar spheres in a bottom-up fashion [16, 21]. Splitting or merging is done greedily and, thus, suboptimally at each stage. More fundamentally though, the basic strategy attempts to minimize the total bounding volume rather than the outside volume (see Fig. 1).

Methods based on the medial axis [1, 7] build a Voronoi diagram and center spheres at its vertices. Adaptive medial axis approximation (AMAA) [3] extends this idea using greedy optimization to merge pairs of neighboring spheres. The approach is superior to other methods, but still involves many greedy decisions and is not directly tied to outside volume.

Variational approximation alternates two phases, partitioning based on Lloyd clustering [13] and fitting, in order to find an optimal, piecewise-linear approximation of the input geometry [4]. Further work [24] extends the approximating primitives from planar elements to spheres and cylinders. Our method builds on this work but uses an entirely different metric (outside volume rather than integrated surface distance) appropriate for solid object queries such as collision and visibility rather than surface simplification. As is true for nonlinear optimization in general, increased searching and better heuristics can locate better solutions but finding the global minimum cannot be guaranteed.

3 Sphere set approximation

Surface approximation. Approximation theory deals with the problem of approximating complicated mathematical objects with simpler ones. In [4], approximation error is defined as the distance between the original surface X and the approximation Y, defined by

$$L^p(X, Y) = \left(\frac{1}{|X|} \iint_{x \in X} \|d(x, Y)\|^p \, \mathrm{d}x \right)^{\frac{1}{p}} \tag{1}$$

$$d(x, Y) = \inf_{y \in Y} \|x - y\| ,$$

where $\|.\|$ is Euclidean distance and $|.|$ is surface area.

Solid approximation. The above definition is appropriate for surfaces not solids. On the other hand, [3, 7] present a metric suitable for solids that measures the distance to the sphere surface along the polygon normal direction, over a set of sample points on the original surface X. Because the method is based on point samples and projects distances onto discrete polygon normals, it can easily miss or underweigh regions of large protrusion. It also neglects concavities (see Fig. 5). We instead apply a volumetric

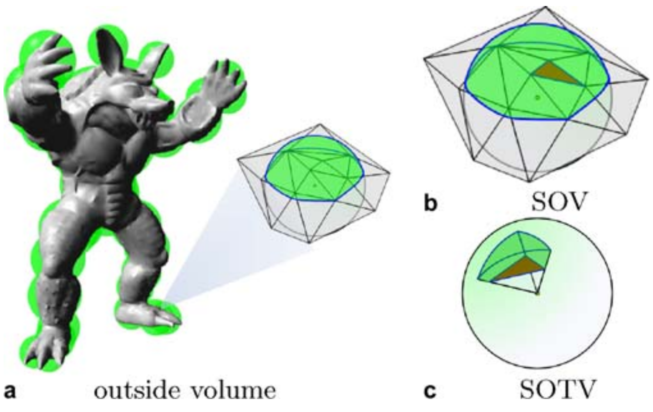

Fig. 2a–c. Outside volume, SOV, and SOTV. **a** The green region is the outside volume. **b** A single sphere is divided by the object's triangles into two parts: inside (gray) and outside volume or SOV (green). **c** One triangle is highlighted (brown) to show its corresponding SOTV. SOV is computed via a sum over SOTVs

version of Eq. 1 that measures volume outside the object but inside an approximating sphere (Fig. 2a,b). More precisely, for a single sphere S the error metric, called *sphere outside volume* (SOV), is defined as

$$E(X,S) = \iiint_{y\in S} d(X,y)\,\mathrm{d}y \tag{2}$$

$$d(X,y) = \begin{cases} 0 & y \in X \\ 1 & y \notin X \end{cases},$$

where X is the original object and $d(X,y)$ returns if a point y is outside X. The global error metric for the entire sphere set is defined by summing $E(X,S_i)$ over all spheres in the set $\{S_i\}$. S_i is defined by the center o_i and the radius r_i.

Unlike the Hausdorff error, this definition is one-sided: only volume in the sphere set outside the original mesh counts. This is reasonable, since we are computing a bounding approximation through which no part of the original mesh can protrude. Another issue is that our definition overcounts volume from S_i that overlaps outside X. Neglecting this typically small overlap makes the computation tractable, because each S_i's SOV can be computed independently without a set union operation.

4 Computing outside volume

A simple method to compute Eq. 2 is to discretize the sphere set volume into a regular grid and count the number of grid cells outside the object. Too many grid cells are necessary to compute outside volume accurately. We instead apply an analytic method to compute the volume by directly integrating it triangle by triangle. For a sphere, Eq. 2 can be represented as

$$E(X,S) = V(T,S)\,, \tag{3}$$

where T is the triangle set representing the object X, and V is the volume outside T but inside the sphere S. The outside volume of the entire sphere set approximation then is:

$$E(X,S_i) = \sum_{i=1}^{n_s} V(T,S_i)\,. \tag{4}$$

If a triangle $t \in T$ is entirely or partly inside the sphere S, then volume exists between the sphere shell and the triangle, denoted $V(t,S)$ and is named *sphere-outside-triangle volume* (SOTV) (Fig. 2c). The total SOV is accumulated by adding or subtracting these SOTVs over all triangles $t \in T$ in the solid object's mesh.

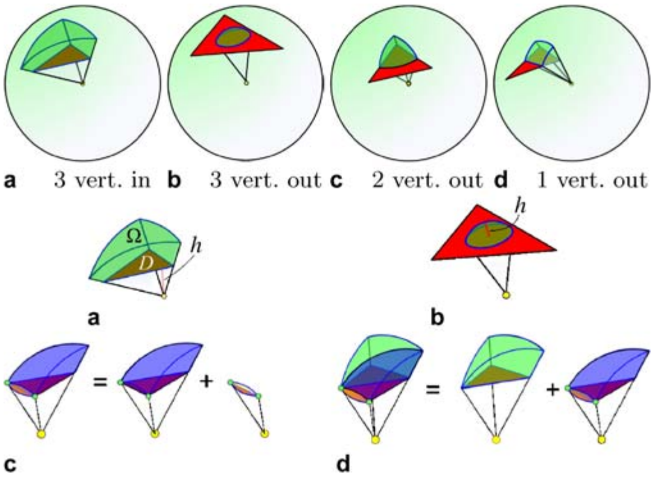

Fig. 3a–d. SOTV computation. Four cases (**a–d**) of the triangle/sphere relation, based on how many triangle vertices are inside the sphere, are shown in the top row. The triangle is drawn in red and the darker green region represents its SOTV. The bottom row shows how the SOTV is computed in each case

4.1 Computing SOTV

Figure 3 classifies four possible relations between a sphere $S=(o,r)$ and a triangle $t \in T$. Cases (a) and (b) are simple, while (c) and (d) are somewhat more complicated.

For case (a), SOTV is given by

$$V(t,S) = V_{\mathrm{stri}}(t,S) - V_{\mathrm{tet}}(t,o) = \frac{1}{3}(r^3\,\Omega - D\,h)\,,$$

where $V_{\mathrm{stri}}(t,S)$ is the volume bounded by the spherical triangle formed by projecting the vertices of triangle t onto the sphere S, and $V_{\mathrm{tet}}(t,c)$ is the volume of the tetrahedron formed by the sphere center o and the three vertices of the triangle t. Ω is the solid angle of the triangle on the sphere, D is the area of triangle t, h is the height of t's plane above o, and r is the sphere radius. For case (b), the volume is

$$V(t,S) = \pi\,h^2\left(r - \frac{h}{3}\right).$$

For case (c), we find the arc representing the intersection of the triangle and sphere, and the two points, p_0 and p_1, where this arc intersects the triangle edges (green points in Fig. 3c, bottom). The outside volume can be further decomposed into one volume corresponding to case (a) and an additional "swing" volume whose computation we will describe in more detail later.

Finally, for case (d), the two points of intersection where the triangle edges exit the sphere, along with the two triangle vertices inside the sphere, form a quadrilateral region that can be split into two triangles. One triangle's outside volume corresponds to case (a) and the other one to case (c) (Fig. 3d, bottom).

Swing volume lies between two planes hinged between the points p_0 and p_1 where the triangle edges exit the

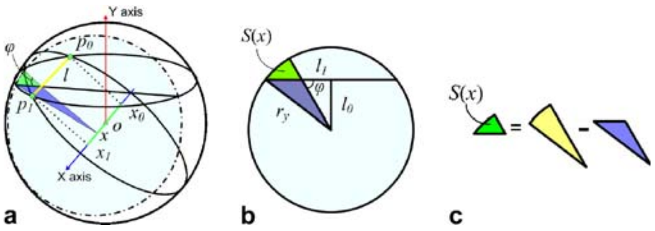

Fig. 4a–c. Swing volume computation. **a** swing volume integral, **b** integral slice: $S(x)$, **c** slice decomposition

sphere (green points in Fig. 3c, bottom). One of the these planes is that of the triangle and the other contains p_0, p_1 and the sphere origin o. The angle between these two planes is denoted φ.

To compute swing volume, we use a canonical orientation where the triangle normal aligns to the y-axis while the vector $p_0 - p_0$ aligns to the x-axis. Define $l = \|p_1 - p_0\|$. Without loss of generality, we can assume a unit-radius sphere ($r = 1$), and then scale the resulting swing volume by r^3. The normalized swing volume then depends on only two parameters, l and φ. As a preprocess, we record a 2D table of swing volumes using numerical integration based on the formula derived below.

In Fig. 4a, the swing volume is broken into 2D slices using planes perpendicular to the x-axis. One of these planes for a particular value of x intersects the sphere in a circle, shown with a dashed outline in Fig. 4a. This x plane intersects the swing volume to form a region we denote $S(x)$, colored green in the figure. Its area can be computed by subtracting the area of the triangle (blue) from the entire sector (yellow), as shown in Fig. 4c.

The entire volume can be defined as an integral of $S(x)$ over x from $x_0 = \frac{-l}{2}$ to $x_1 = \frac{l}{2}$. Figure 4b shows the region $S(x)$ in a simpler 2D projection where the view is now perpendicular to the x-axis. The region may be defined in terms of the variable r_y and two constants, l_0 and l_1, where $r_y = \sqrt{1-x^2}$, l_0 is the distance of the triangle plane to the sphere center o, and l_1 is the distance of the projection of o onto the triangle plane to the segment from p_0 to p_1:

$$l_0 = \sqrt{1-\left(\frac{l}{2}\right)^2}\,\sin(\varphi),\quad l_1 = \sqrt{1-\left(\frac{l}{2}\right)^2}\,\cos(\varphi)$$

The swing volume V_s can now be expressed as

$$V_s = \int_{x_0}^{x_1} S(x)\,\mathrm{d}x$$

$$S(x) = \left(\varphi - \sin^{-1}\left(\frac{l_0}{\sqrt{1-x^2}}\right)\right) - \frac{1}{2}\left(\sqrt{1-x^2-l_0^2} - l_1\right) l_0 \,.$$

The first term in $S(x)$ corresponds to the yellow sector and the second to the blue triangle.

4.2 Accumulating SOTV into SOV

While accumulating SOTV over mesh triangles, the volume must be signed according to two factors: whether the sphere center is inside or outside the object, and whether the sphere center is behind or in front of the triangle's plane.

(a) Sphere center inside object. SOTV is positive if the sphere center is behind the triangle plane, and negative otherwise. Total outside volume is thus computed as

$$V_{in}(T,S) = \sum_{t_j \cap S \neq \emptyset} \operatorname{sign}\left(n_j \cdot (p_j - o)\right)\, V(t_j, S)\,,$$

where t_j is a triangle in the mesh T, o is the center of S, n_j is the normal of triangle t_j, and p_j is the projection of o onto triangle t_j. The sign function returns -1 if its argument is negative and $+1$ otherwise.

(b) Sphere center outside object. In this situation, we invert the signs used in (a) and then subtract the result from the total sphere volume, yielding

$$V_{out}(T,S) = \frac{4\pi}{3} r^3 + V_{in}(T,S)\,.$$

2D example. Figure 6 shows a 2D example with the circle center inside the object. Unlike our outside volume computation in Fig. 5c, the approximate error metric from [7] does not handle concavities correctly. For example, the red edge has normal-projected distance of d in Fig. 5b, an overestimate that effectively ignores the concavity.

The total outside area is accurately computed by traversing all edges of the 2D polygon to accumulate their outside areas one by one. Depending on the position of the circle center relative to the "outward" halfspace of each polygon edge, each area is signed positive (green) or negative (blue). Accumulation of 3D volumes with respect to a sphere is analogous.

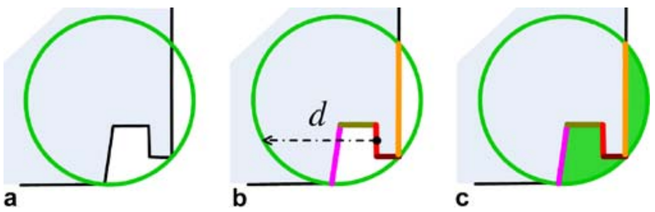

Fig. 5a–c. Error metric comparison (2D). **a** polygon and circle, **b** normal-projected metric [7], **c** outside area

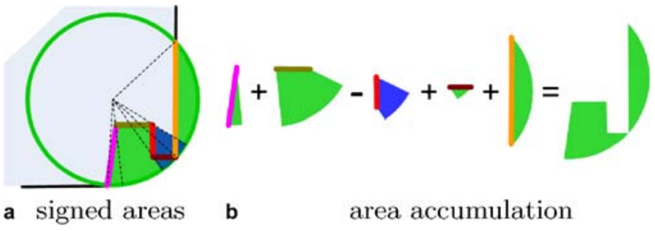

Fig. 6. Outside area accumulation (2D)

5 Minimizing outside volume

To find an optimal sphere set approximation for solid objects, we minimize Eq. 4 using a variant of the Lloyd clustering algorithm [13]. We discretize the object into a set of points including points on its surface and within its interior. Lloyd clustering takes place over this set of points and a sphere is used to bound each of the n_s clusters. Clustering iteratively applies three steps: point assignment to clusters, cluster sphere update, and cluster teleportation, until error converges. Each point is assigned to the cluster whose bounding sphere's outside volume increases least. Given the cluster's set of assigned points, its bounding sphere center and radius are updated by minimizing Eq. 3. To avoid getting stuck in an undesirable local minimum, the "teleportation" strategy from [4] is applied.

Preprocessing. A mesh object is discretized into two kinds of points: inner points and surface points. Inner points are generated by voxelizing the object into a regular grid and eliminating grid points outside the object. A grid size is manually chosen to ensure the object's interior is sampled well. Points on the mesh surface are generated by sorting triangles by their areas and sampling points randomly in proportion to these areas [23]. The inner grid points and surface samples are combined and regarded as the volumetric representation to be partitioned in clustering.

To initialize the cluster spheres, we randomly choose n_s inner points for the sphere centers o_i and set their radii $r_i = 0$.

Point assignment. A point p is assigned to the cluster it is closest to, based on outside volume (Fig. 7). More precisely, the cluster's bounding sphere radius is enlarged in order to include p, and the enlarged SOV is computed. p is assigned to that cluster whose outside volume increases least. In case of a tie, p is assigned to the cluster whose sphere center is closest.

We use a flood fill (stack-based) algorithm to order the set of points for cluster assignment. The algorithm starts from the cluster centers and progresses to adjacent points. Once a point is assigned, the bounding sphere radius of its cluster is updated, and its adjacent points are pushed onto the stack. Assignment terminates when the stack is empty.

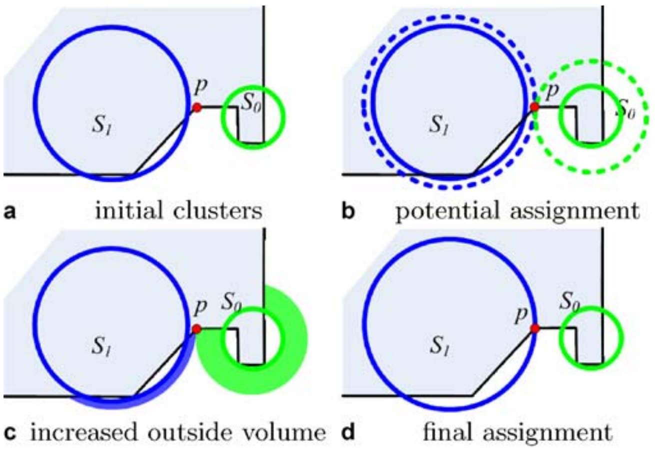

Fig. 7. Point assignment by outside volume. Although point p is closer to cluster S_0's center than to S_1, the outside volume increases less by assigning it to S_1

Unlike VQ [13], whose point assignment step is order-independent and simply assigns points to the closest cluster, our point assignment cannot guarantee a decrease in outside volume. In practice, it almost always achieves one. If point assignment yields an error increase, sphere teleportation is triggered (see below). If this fails to decrease error, the algorithm terminates.

The above (naive) algorithm for point assignment can be accelerated. Naive point assignment computes $n \times n_s$ SOV queries, where n is the number of points and n_s the number of clusters. We can significantly reduce this number by observing that cluster bounding sphere centers o_i remain fixed during point assignment, only their radii r_i are progressively enlarged as more points are assigned. The current outside volume of each cluster sphere is thus only a function of its current radius. As points are assigned, we keep track of the outside volume for each cluster sphere as a function of the points' distance to o_i. In other words, we build a table of the relation between radius and SOV for each cluster based on its fixed center o_i.

To assign a new point p, for each cluster C_i we look up a (previously assigned) point q in its table that is closer to o_i than p, yielding a lower bound on p's outside volume. By subtracting C_i's current outside volume from this, we obtain a lower bound on the outside volume increase without performing a new SOV query.

Using this method, we compute lower bounds on outside volume increase for assigning p to each cluster C_i. If no q closer than p exists in C_i's table, then we must compute the actual SOV for p with respect to o_i. We then compute actual SOV (and trivially, outside volume increase) for the cluster having the smallest lower bound. Clusters whose lower bounds are larger can be culled without the need to perform an expensive SOV query.

Sphere fitting. Once we have a new cluster assignment, we independently update the bounding spheres of each cluster C_i. Starting from its old parameters, the cluster bounding sphere $S_i = (o_i, r_i)$ updates its center and radius in order to best approximate the points assigned to cluster C_i. The best center and radius are determined using Eq. 3, via

$$\text{argmin}_{o_i} V(T, S_i) \tag{5}$$

while constraining r_i to continue bounding all the cluster's assigned points.[1] The minimal o_i is found using Powell's multidimensional minimization [14].

Sphere teleportation. Like many variational methods, the algorithm easily gets stuck in a local minimum. To avoid this, we employ sphere teleportation, which is similar to the "region teleportation" in [4]. Teleportation is triggered when insufficient error improvement occurs in cluster assignment/update iterations. The sphere having the maximum overlap ratio (and is thus most redundant) is chosen as the teleportation source and the largest error sphere is chosen as the teleportation destination. Overlap ratio is defined as the ratio of volume shared with at least one other sphere in the set to the total volume. After teleportation, the teleportation source sphere is deleted and the destination sphere is split into two. The two points farthest from each other in the maximum error cluster are chosen as the two initial sphere centers. Another iteration of point assignment and cluster update is then computed to evaluate this teleportation. If the new error is less, the teleportation is accepted.

6 Applying sphere sets

Animating sphere sets can be achieved by "skinning" based on mean value coordinates [22]. A sphere set is fit to the rest pose and the sphere centers updated as the model moves using a linear combination of the deformed vertex positions (see [17]). Bounding sphere radii can remain fixed for typical motions of articulated characters.

Sphere hierarchies can be built using a simple bottom-up approach, using our algorithm's sphere sets as its leaf nodes. Hierarchy levels are constructed one at a time from the leaves up to the root, based on Lloyd clustering. Each cluster stores its current bounding sphere. Clustering iteratively assigns spheres to the closest cluster, based on the distance from the sphere center to the cluster center, and then updates the cluster's bounding sphere. Its center is initially taken as the average center over all spheres assigned to it and then optimized using Powell's method to reduce its radius. After convergence, each cluster is made a parent node in the hierarchy; the spheres assigned to it become its children. The next hierarchy level above can then be computed by recursively clustering over the list of parent nodes just computed. The average branching ratio of the hierarchy is chosen by the user.

[1] Actually, to ensure that the sphere set bounds the triangle mesh, each sphere is enlarged as a post-process to bound the edge mid-points and triangle centers of all triangles entirely or partially inside the sphere [8].

7 Results

To our knowledge, AMAA [3] is the state-of-the-art method for sphere set approximation (see Fig. 8). We compare our results to AMAA results, based on visual and geometric error, and in applications of shadow computation and proximity query. We use the author's software, available at http://isg.cs.tcd.ie/spheretree/.

Figure 15 shows some individual objects and assembled scenes, using shadow rendering from [17]. More examples are shown in Fig. 16. The construction time for one of the models, the "Armadillo", using clustering over 5000 points is documented in Fig. 9. Our computation time is not much larger than AMAA's, but yields a significantly better approximation.

Visual and geometric error. Figure 16 compares our approximation to AMAA using three different sphere set sizes: $n_s = 32$, $n_s = 64$, and $n_s = 128$. Geometric error is measured by *relative outside volume*, E_r, the sphere

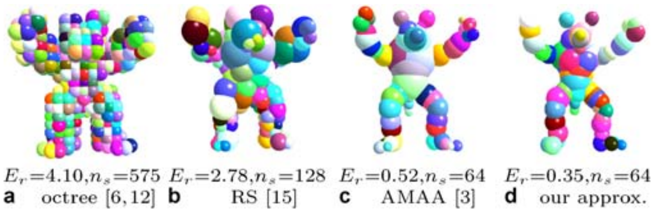

Fig. 8. Method comparison on the "Armadillo"

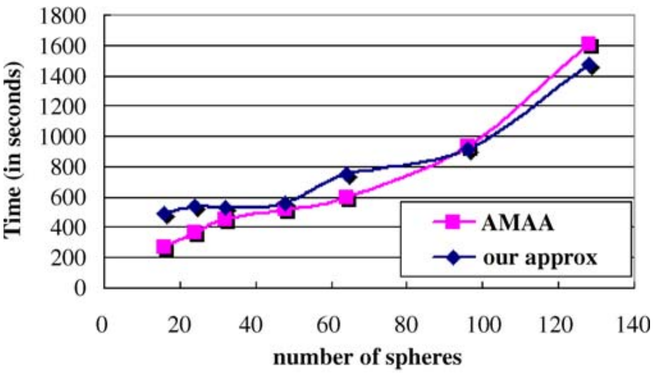

Fig. 9. Sphere-set construction time

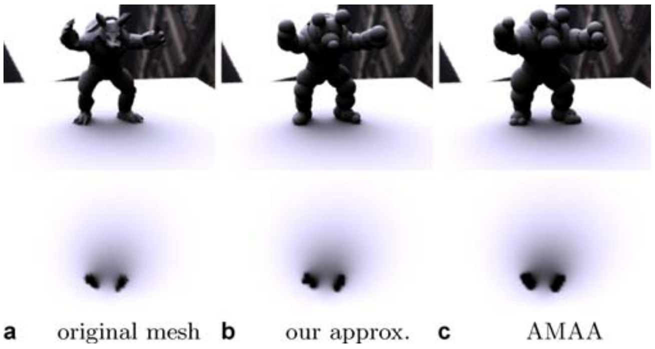

Fig. 10. Sphere set shadows, $n_s = 48$

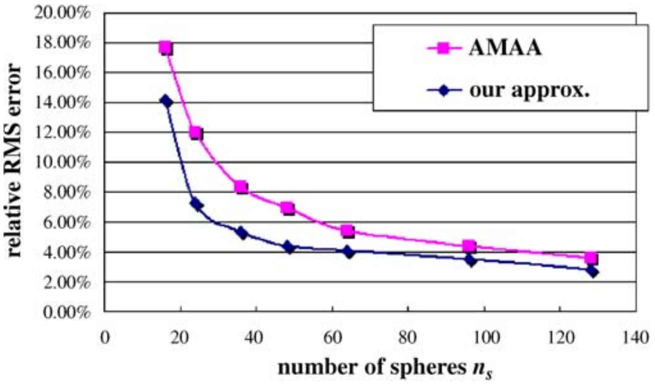

Fig. 11. Graphs of shadow error

set's outside volume divided by the original mesh's volume. Figure 17 graphs E_r as a function of n_s. Although our method's curve does not appear to be much lower than AMAA's in terms of absolute error, it should be noted that both curves flatten out dramatically after an n_s greater than 20-40 spheres. The absolute error value is much less important than how many spheres are required to achieve it. Intersecting the two curves with horizontal lines of constant error, it can be seen that AMAA often requires many more spheres than our approach to achieve the same error – two or three times as many in some cases (e.g. Armadillo graph).

Shadow application. The sphere set can be used as a proxy for rendering the original mesh [18] or its shadows [17]. We compare results using sphere sets to cast shadows from distant, environmental light rendered using diffuse PRT [20]. A ground plane is used to "catch" shadows underneath the objects and the resulting image differences are analyzed in terms of RMS error over all pixels and all orientations of the lighting environment. Image results are presented in Fig. 10 (the top row shows the original view, while the bottom row shows the ground plane images) and error graphs in Fig. 11. Shadows from our sphere sets are significantly closer

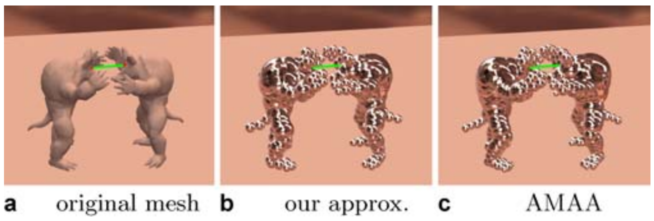

Fig. 12. Proximity query scene

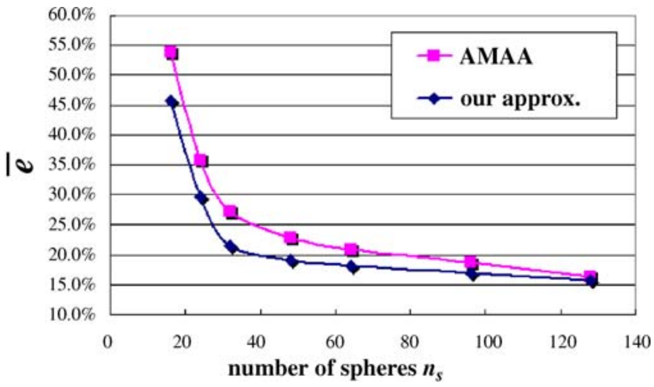

Fig. 13. Proximity query error

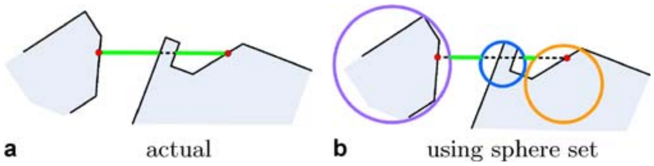

Fig. 14. Outside distance

to the ground truth shadows, allowing us to use fewer spheres in order to achieve results similar to AMAA (see graphs).

Proximity application. We also tested sphere set approximations in proximity queries, using a pair of "armadillo" models (Fig. 12). The experiments are based on outside distance queries (Fig. 14) that measure the distance of a line segment spent either outside the model (a) or outside the sphere set (b). The line segments for the queries were generated using 3000 random pairs of points on the two models, sampled uniformly in the mesh area. The average difference between the actual outside distance and the sphere set outside distance, $\overline{e}$, is a good measure of the tightness of the sphere approximation and is graphed in Fig. 13.

By performing the distance query over line segments, we are able to perform a union over the sphere set by tracking 1D intervals representing sphere entry and exit points and counting the nesting levels. The outside dis-

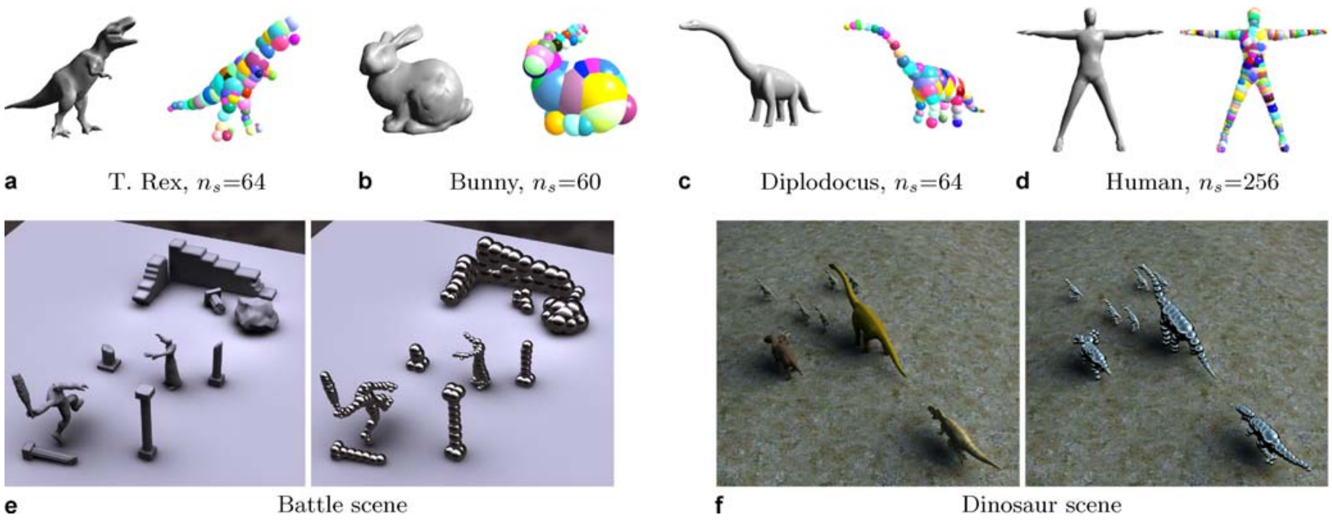

Fig. 15. Sphere set approximation for different objects and scenes

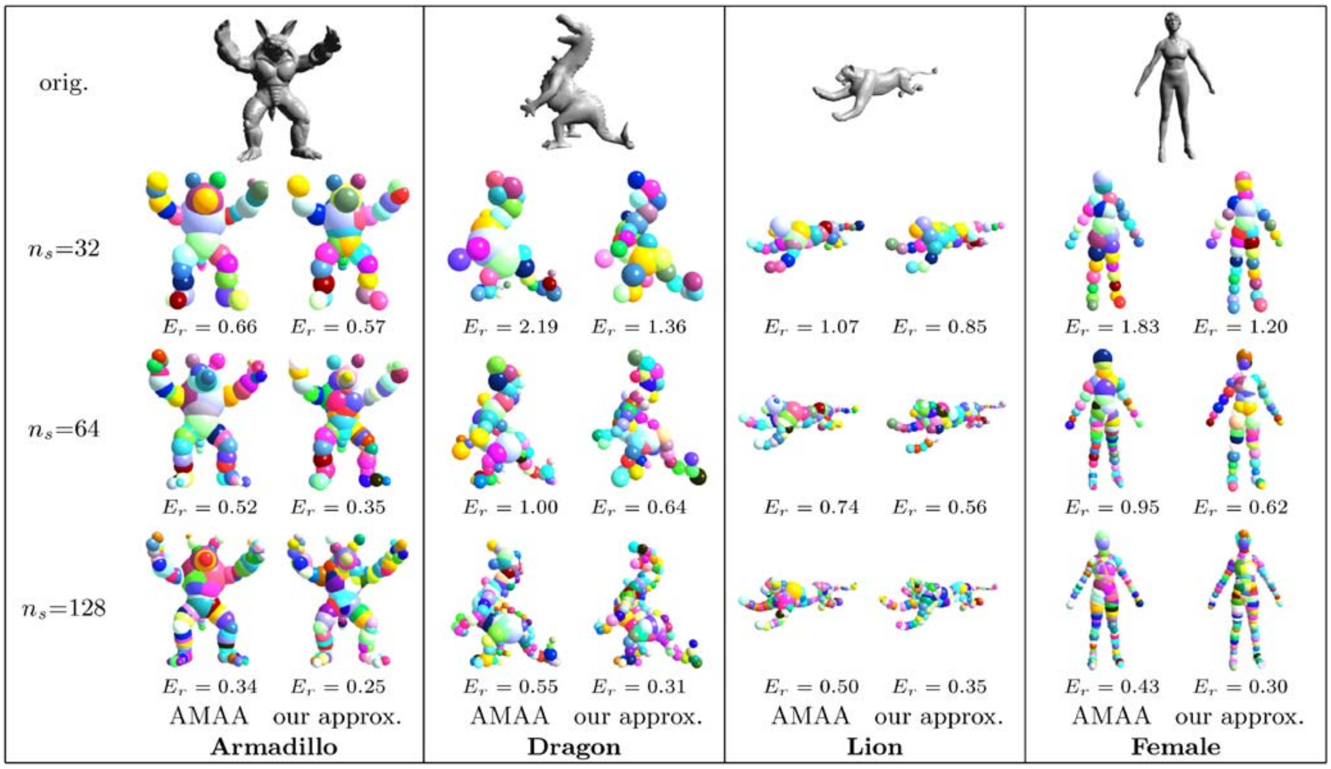
orig.
n_s=32
n_s=64
n_s=128
$E_r = 0.66$ $E_r = 0.57$ $E_r = 2.19$ $E_r = 1.36$ $E_r = 1.07$ $E_r = 0.85$ $E_r = 1.83$ $E_r = 1.20$
$E_r = 0.52$ $E_r = 0.35$ $E_r = 1.00$ $E_r = 0.64$ $E_r = 0.74$ $E_r = 0.56$ $E_r = 0.95$ $E_r = 0.62$
$E_r = 0.34$ $E_r = 0.25$ $E_r = 0.55$ $E_r = 0.31$ $E_r = 0.50$ $E_r = 0.35$ $E_r = 0.43$ $E_r = 0.30$
AMAA our approx.
Armadillo
Dragon
Lion
Female

Fig. 16. Sphere set comparison with the AMAA method

tance query thus eliminates overcounting of overlapped outside volume, unlike our definition in Eq. 4. As in the other applications, our method significantly beats AMAA. For example, our method with $n_s = 32$ spheres has less average error than AMAA's with $n_s = 64$.

8 Conclusion

We have tested our variational approach on many models, both simple and complex. Our method out-performs al-

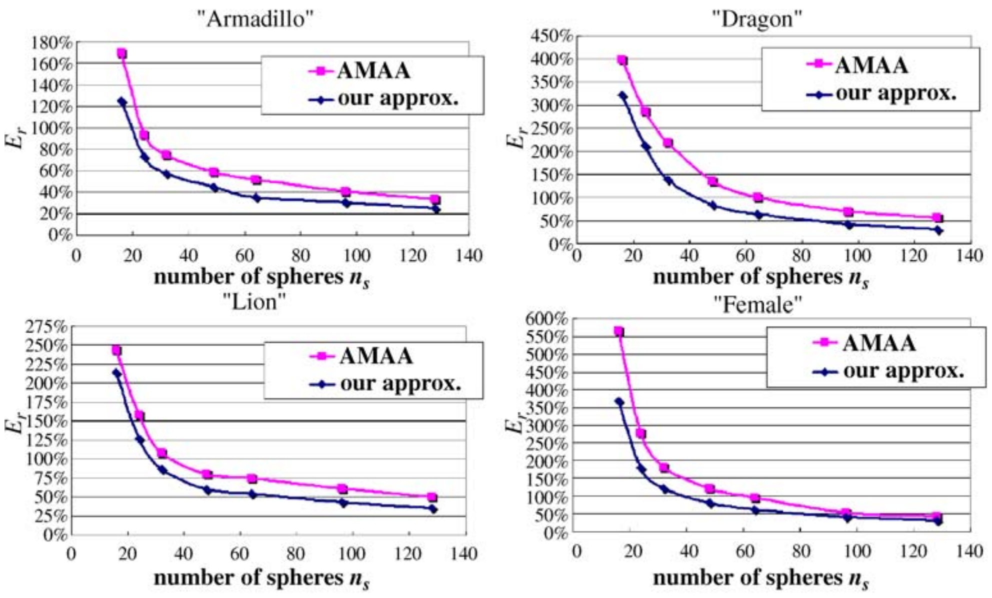

Fig. 17. Graphs of relative outside volume vs. n_s

ternatives such as the octree and medial-axis methods, as we have demonstrated using visual and quantitative comparisons and in applications involving shadowing and proximity detection. In many cases, our algorithm's sphere sets work as well as the current best alternative ones with only 33%–75% more spheres.

Our algorithm's effectiveness is based on two factors. One is a suitable definition of error, which we formulate in terms of outside volume. Our error metric considers only the volume of the sphere set outside the input geometry and neglects how much the spheres overlap in the object's interior. This is appropriate for solid object approximation in applications based on intersection and proximity. The second factor is to solve the resulting minimization problem using variational partitioning and fitting, which provides a more direct and more global solution than previous methods.

Our method's extensive computation limits it to sphere sets having no more than a few hundred spheres. Fortunately, object approximation with such a limited number of spheres suffices for many interesting real-time applications, as we have shown. To generate larger sphere sets, heuristics such as simple Lloyd (L^2) clustering can be used to form initial bounding spheres and then a few iterations of outside-volume minimizing cluster iteration can be applied.

Acknowledgement This project was partially supported by the 973 Program of China (no. 2002CB312104), NSFC (no. 60021201) and the Specialized Research Fund for the Doctoral Program of Higher Education of China (no. 20030335083).

References

1. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: SMA '01: Proceedings of the 6th ACM Symposium on Solid Modeling and Applications, pp. 249–266. ACM Press, New York (2001)
2. van den Bergen, G.: Efficient collision detection of complex deformable models using AABB trees. J. Graph. Tools **2**(4), 1–13 (1997)
3. Bradshaw, G., O'Sullivan, C.: Adaptive medial-axis approximation for sphere-tree construction. ACM Trans. Graph. **23**(1), 1–26 (2004)
4. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. ACM Trans. Graph. **23**(3), 905–914 (2004)
5. Gottschalk, S., Lin, M.C., Manocha, D.: OBBTree: A hierarchical structure for rapid interference detection. In: Proc. of ACM SIGGRAPH 1996, pp. 171–180 (1996)
6. Hubbard, P.: Interactive collision detection. In: Proceedings of the 1993 IEEE Symposium on Research Frontiers in Virtual Reality, **14**(2), 24–31 (1993)
7. Hubbard, P.: Collision detection for interactive graphics applications. PhD thesis, Brown University (1995)
8. Hubbard, P.: Approximating polyhedra with spheres for time-critical collision detection. ACM Trans. Graph. **15**(3), 179–210 (1996)
9. James, D.L., Pai, D.K.: BD-tree: output-sensitive collision detection for reduced deformable models. ACM Trans. Graph. **23**(3), 393–398 (2004)
10. Klosowski, J.T., Held, M., Mitchell, J., Sowizral, H., Zikan, K.: Efficient collision detection using bounding volume hierarchies of k-DOPs. IEEE Trans. Visual. Comput. Graph. **4**(1), 21–36 (1998)
11. Krishnan, S., Pattekar, A., Lin, M., Manocha, D.: Spherical shells: A higher-order bounding volume for fast proximity queries. In Proceedings of the 1998 Workshop on the Algorithmic Foundations of Robotics, pp. 122–136. Rice University (1998)
12. Liu, Y., Noborio, J., Arimoto, S.: Hierarchical sphere model HSM and its application for checking an interference between moving robots. In Proceedings of the IEEE International Workshop on Intelligent Robots and Systems, pp. 801–806 (1988)
13. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inform. Theory **IT-28**(2), 129–137 (1982)
14. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge

University Press, New York (1992)
15. Quinlan, S.: Efficient distance computation between non-convex objects. In Proceedings IEEE International Conference on Robotics and Automation, pp. 3324–3329 (1994)
16. Ranjan, V., Fournier, A.: Union of spheres (UoS) model for volumetric data. In: SCG '95: Proceedings of the 11th Annual Symposium on Computational Geometry, pp. 402–403. ACM Press, New York (1995)
17. Ren, Z., Wang, R., Snyder, J., Zhou, K., Liu, X., Sun, B., Sloan, P., Bao, H., Peng, Q., Guo, B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. ACM Trans. Graph. **25**(3), 977–986 (2006)
18. Rusinkiewicz, S., Levoy, M.: QSplat: A multiresolution point rendering system for large meshes. In: Proc. ACM SIGGRAPH 2000, pp. 343–352 (2000)
19. Ruspini, D.C., Kolarov, K., Khatib, O.: The haptic display of complex graphical environments. In: Proc. of ACM SIGGRAPH 1997, pp. 345–352 (1997)
20. Sloan, P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. ACM Trans. Graph. **21**(3), 527–536 (2002)
21. Tam, R.C., Fournier, A.: Image interpolation using unions of spheres. Visual Comput. **14**, 401–414 (1998)
22. Tao, J., Schaefer, S., Warren, J.: Mean value coordinates for closed triangular meshes. ACM Trans. Graph. **24**(3), 561–566 (2005)
23. Turk, G.: Generating random points in triangles. In: Graphics Gems, pp. 24–28. Academic Press Professional, San Diego, CA (1990)
24. Wu, J., Kobbelt, L.: Structure recovery via hybrid variational surface approximation. Comput. Graph. Forum **24**(3), 277–284 (2005)

RUI WANG received his BS degree in computer science from Zhejiang University in 2001. He is currently a PhD candidate at the State Key Laboratory of CAD & CG of Zhejiang University. His research interests are image-based modeling, machine learning and geometry approximation.

KUN ZHOU is a researcher/project leader of the graphics group at Microsoft Research Asia. He received his BS and PhD degrees in computer science from Zhejiang University in 1997 and 2002, respectively. His current research focus is geometry processing, texture processing and real-time rendering. He holds over 10 granted and pending US patents. Many of these techniques have been integrated in Windows Vista, DirectX and XBOX SDK.

JOHN SNYDER is a principal researcher at Microsoft Research. He received a BS from Clarkson University in 1984, a PhD from the California Institute of Technology in 1991, and has been at Microsoft Research since 1994. His research interests include geometry representation and processing and real-time algorithms for global illumination.

XINGUO LIU received a BS in 1995 and a PhD in 2001 from the Department of Applied Mathematics at Zhejiang University. He is currently at the State Key Lab of CAD & CG, and is a Professor of the Computer Science School at Zhejiang University. Before joining CAD & CG in April 2006, he was a researcher in the Internet Graphics Group in Microsoft Research Asia. His main research interests are in appearance modeling, real-time rendering, geometry processing and deformable objects.

HUJUN BAO received his BS and PhD degrees in applied mathematics from Zhejiang University in 1987 and 1993, respectively. His research interests include modeling and rendering techniques for large scale of virtual environments and their applications. He is currently the director of State Key Laboratory of CAD & CG of Zhejiang University. He is also the principal investigator of a virtual reality project sponsored by the Ministry of Science and Technology of China.

QUNSHENG PENG is a professor of computer graphics at Zhejiang University. His research interests include realistic image synthesis, computer animation, scientific data visualization, virtual reality, bio-molecule modeling. Prof. Peng graduated from Beijing Mechanical College in 1970 and received a PhD from the Department of Computing Studies, University of East Anglia in 1983. He is currently serving as a member of the editorial boards of several international and Chinese journals

BAINING GUO is the research manager of the Internet Graphics Group at Microsoft Research Asia. Before joining Microsoft, Baining was a senior staff researcher in Microcomputer Research Labs at the Intel Corporation in Santa Clara, California, where he worked on graphics architecture. Baining received his PhD and MS degrees from Cornell University and his BS from Beijing University. Baining is an associate editor of IEEE Transactions on Visualization and Computer Graphics. He holds over 30 granted and pending US patents.