Ellipsoidal-Blob Approximation of 3D Models and Its Applications

Shengjun Liu [a,b], Xiaogang Jin [a,*], Charlie C. L. Wang [b]
Kin-chuen Hui [b]

[a] *State Key Lab of CAD&CG,*
Zhejiang University, Hangzhou, 310027, P.R.China

[b] *Department of Mechanical and Automation Engineering,*
The Chinese University of Hong Kong, Hong Kong, P.R.China

Abstract

This paper presents a technique for automatically approximating a given mesh model with an ellipsoidal blobby model. Firstly, an ellipsoid decomposition algorithm is introduced to approximate given models by ellipsoids. After that, a blobby implicit surface employing ellipsoidal blobs is modelled to fit the sampling points on the given mesh. Finally, the reconstructed ellipsoidal blobby model is applied in two applications: the geometry data reduction and the target shape controlled cloud animation.

Key words: Ellipsoidal blobby model; Implicit surface; Animation; Geometry data reduction

* Corresponding author. Tel.: +86-571-88206681 ext 507; fax: +86-571-88206680.
E-mail addresses: liushengjun@cad.zju.edu.cn (S. Liu), jin@cad.zju.edu.cn (X. Jin), cwang@mae.cuhk.edu.hk (C.C.L. Wang), kchui@mae.cuhk.edu.hk(K. Hui).

1 Introduction

The computer graphics, computer-aided design and computer vision literatures are filled with a diverse array of surface representations. The reason for this variety is that there is no single representation that can satisfy the needs of all problems in various applications. Implicit surface is one of the most popular representations which are widely employed in computer graphics applications including geometric modelling, three-dimensional metamorphosis and collision detection. Among variety of implicit surfaces, one important class is the so-called blobby model [1] and its variants — metaball [2] and soft object [3]. However, the method to efficiently and effectively approximate a given mesh model by a blobby model is still a problem under research. For simple objects (e.g., spheres and peanut-like objects etc.), it is easy to obtain their corresponding blobby representations. For a model with complex shape such as a human body, it is a difficult and tedious work to construct blobs for the model manually. The motivation of the work presented in this paper is to seek an automatic method to approximate given mesh models by a blobby representation.

Here we present an automatic approach to approximate a given mesh model with an implicit surface employing ellipsoids as primitives — named as ellipsoidal-blobs. Our algorithm consists of two steps: in the first step, the given mesh model is sampled into points and then decomposed into a set of ellipsoids; in the second step, the final blobby model is reconstructed and computed from the ellipsoids through numerical optimization. The reconstructed blobby model has many applications. To demonstrate the functionality of an ellipsoidal blobby model, we apply it in the applications of the geometry data reduction and the target shape controlled cloud animation.

Previous related works will be firstly reviewed in the following section. After that, a modified ellipsoid decomposition algorithm will be introduced in section 3. The mathematical representation and the reconstruction method of ellipsoidal blobby models are then presented in section 4. In section 5, two applications will be demonstrated. Lastly, our paper ends with the section of conclusion and discussion.

2 Previous Work

An implicit surface S is usually defined by a continuous scalar function $f(\mathbf{x})$ with $\mathbf{x} \in \Re^3$. The geometry of S is given by the locus of points at which the function $f(\mathbf{x}) = 0$. In [1–3], the implicit surfaces are defined as the summation of radial symmetric functions, which are generally in the form of

$$f(\mathbf{x}) = -t + \sum_{i=1}^{n} \omega_i f_i(\mathbf{x}). \tag{1}$$

In this formula, the parameter t is a threshold of isosurface S, n is the number of primitives, ω_i is the weight for the ith primitive (with default value 1.0), and the function f_i describes the profile of a blobby sphere with a particular center and radius. Due to the smooth blending property of implicit surfaces, 3D morphing can be easily conducted between two given implicit surfaces with any topological structures. However, as aforementioned, the automatic reconstruction technique for blobby models is still under research. For all existed automatic approaches [4–7], spheres are adopted as the only primitives. As spheres are isotropic, the aliasing errors of spherical blobby models on sharp or thin features are relative great. Thus, we will employ an ellipsoidal blobby implicit surface to fit a given model so that the error of approximation can be reduced. First of all, the given object M is decomposed into a set of ellipsoids. The ellipsoids are then employed as initial primitives in the blobby model. In the following, the isosurface defined by these ellipsoidal blobs is adjusted to improve the approximation through numerical optimization.

In the area of geometry model data reduction, some approaches reduce the data size of geometry models through the transformation between the boundary representation and the volumetric (or implicit) representation (e.g.,[8,9]). As will be shown later, the reconstruction of ellipsoidal blobby models can be considered as a lossy compression technique, where we use very few blobs to approximate a given geometry model.

The other application of the ellipsoidal blobby model shown in our paper is the target shape controlled cloud animation. In computer graphics, there are two categories of works to simulate the gaseous motion of clouds or smokes. The approaches in one category simulate the physical process of fluid dynamics [10–13], and the methods in another category are heuristic using the techniques such as procedural modelling [14–21] or fractals [22–24]. The physics-simulation based methods consider the shape and motion of gas as the result of air movement, so they produce realistic images by approximating the physical phenomena during the movement of clouds. Although Stam developed a fast simulation method by simplifying fluid dynamics in [11,12] where he demonstrated a real-time animation of smoke on a high-end workstation, the physics-based simulation governed by fluid dynamics in general is still time-consuming — especially when the target shape is desired to be controlled. Techniques for the target shape controlled fluids in [25,26] involved the carefully designed force fields and a modified diffusion equation for smoke gathering. Shi et al. in [27] also proposed a method to control the density and dynamics of smoke so that the synthetic appearance of the smoke resembles

a still or moving object. They represented the smoke region and the target object as implicit functions, and imposed the designed velocity constraints on the smoke boundary during a dynamic fluid simulation. As their approach is based on the simulation of atmospheric fluid dynamics, the computational cost is expensive. On the other hand, none of the heuristic approaches [14–24] are sufficient for the purpose to efficiently generate a target shape controlled cloud animation. Relying on the reconstructed ellipsoidal blobby approximation, we will present two geometry-based cloud animation schemes with the target shape controlled.

3 Ellipsoid Decomposition

For a given polygonal mesh, obviously there are many different possible ellipsoid decompositions. In [28], Bischoff and Kobbelt designed an algorithm to find one candidate among this multitude of decompositions, where the computed decomposition is a local optimum with respect to the shape, the orientation and the distribution of ellipsoids. However, some small features on given models are missed in their algorithm. To avoid this, under the framework of [28], a modified scheme is developed.

3.1 The Framework of Ellipsoid Decomposition

The framework of Bischoff and Kobbelt [28] is in fact a greedy optimization algorithm — i.e., only a finite number of configurations are searched.

Briefly, every ellipsoid is generated by growing from a surface point $\mathbf{p}$ on the given mesh model. First of all, by fixing the sphere center $\mathbf{c}$ on the normal direction $\mathbf{n}_p$ of $\mathbf{p}$, the largest inscribed sphere is determined by searching all other surface points until finding a point $\mathbf{q}$ that makes the radius of the sphere minimal. A tangent plane P_q of the sphere at $\mathbf{q}$ can be determined by $\mathbf{n}_q$, together with the tangent plane P_p at $\mathbf{p}$. We then strengthen the sphere in the wedge defined by P_p and P_q to an ellipsoid until the surface of the ellipsoid touch the third surface point $\mathbf{r}$. Finally, the ellipsoid is grown in the direction perpendicular to the plane $\mathbf{pqr}$ until the fourth point $\mathbf{s}$ is touched. The whole procedure is illustrated in Figure 1. For this point-based method, as long as the density of surface points is high enough, the intersection between the constructed ellipsoid and the given polygonal mesh can be neglected. Therefore, a sampling procedure is needed before generating ellipsoids from surface points for low-resolution models.

After constructing all the ellipsoids, to reduce the redundancy led by overlap-

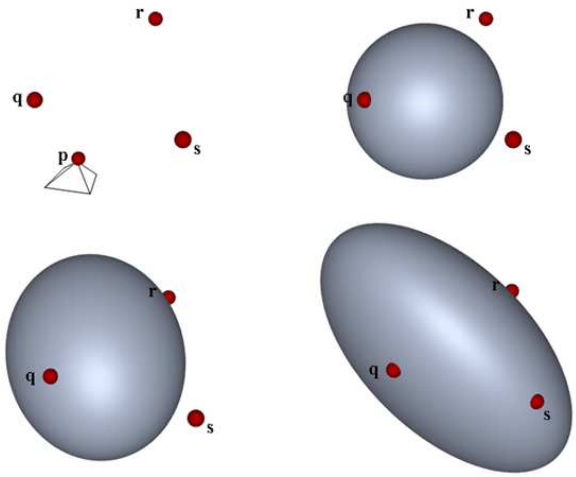


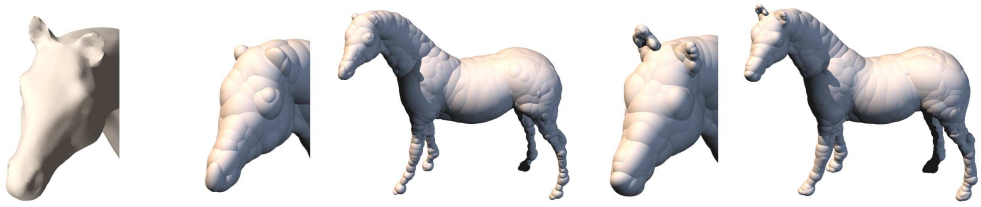
Fig. 1. The illustration of an ellipsoid generation.

ping, only the ones provide significant contribution to the volume of object are selected. This is achieved by a greedy optimization: started from sorting the ellipsoid in a minimal heap by their volumes in descending order, the ellipsoids are added into the final set one by one in the order of their contribution to the incrementation of volume. When popping an ellipsoid from the top of the heap we re-calculate the volume contribution of all the interacting ellipsoids and update their positions in the heap. The selection procedure stops when the incremental volume by adding an ellipsoid is small enough.

3.2 The Modified Scheme

In order to improve the quality of decomposition results, we modify the original decomposition algorithm of [28] in the following aspects:

- To grow the ellipsoid as big as possible, instead of using the exact touching points $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$, $\mathbf{s}$, the authors in [28] employed points $\mathbf{p}'$, $\mathbf{q}'$, $\mathbf{r}'$, $\mathbf{s}'$ which are obtained by shifting the original points by some small offset ε along normal direction into the interior of the mesh. However, during our investigation, we found that processing the points in this manner will make some small features neglected during the ellipsoid selection because of their small volume contribution. Therefore, we generate the points $\mathbf{p}'$, $\mathbf{q}'$, $\mathbf{r}'$, $\mathbf{s}'$ by shifting the original points along normal vectors outwards but not inwards the mesh. The approximation error introduced by this point shifting will be



a. Given model. b. Decomposition by [28]. c. Decomposition by ours.

Fig. 2. The comparison of ellipsoid decompositions.

compensated later during the blobby fitting step in section 4.

- Furthermore, during ellipsoids selection, the geometry significance is considered together with the volume contribution. We use the curvature at each sampling point as a weight of volume contribution so that the ellipsoids whose volume contributions are small but are important to the surface features can be selected to remain. It is greatly helpful for preserving small features on the given model.
- Lastly, for the given model with a very dense mesh, we adopt the vertices of simplified model (with quadratic error controlled) as the sampling points for generating ellipsoids, instead of randomly selected vertices. It is because that random selection may miss some important features of the model, but using a good mesh simplification algorithm can preserve them. In our implementation, we adopt the mesh simplification algorithm of [29].

With the above modifications, the ellipsoid decomposition result has been significantly improved. For instance, the decomposition results shown in Figure 2 are from the original approach [28] and our modification respectively. It is easy to find that the result from our modified scheme preserves the important details (such as ears of the horse) much better. This provides a good start for the following ellipsoidal blobby model reconstruction.

4 Reconstruction of Ellipsoidal Blobby Models

With the ellipsoids decomposed from the given mesh model M as input, an ellipsoidal blobby model Ω approximating M is reconstructed from the ellipsoids by taking their centers as the skeletons with associated field functions. The isosurface of Ω approximates the surface of M. After that, the parameters of blobs in Ω are optimized to reduce the approximation error.

4.1 Mathematical Representation

Field functions employed in a blobby model can be classified into global and local ones. For the global field functions, the computational cost will be quite expensive when there are many primitives included in a blobby model, and their fields will overlap each other globally [1,4]. This limits the number of blobs that could be involved. Thus, we choose local polynomial field functions in our blobby model. Local field functions [2,3,5–7] decrease to zero at the distance of influence radius R. Based on the local influence region, a fast computing speed can be achieved. Besides, local field functions offer local controls on the defined implicit surfaces, which is extremely important for using an implicit surface to approximate a given mesh surface. The parameters of the field functions will be modified through a numerical optimization procedure later, where the number of parameters will directly affect the dimension of searching space in optimization. The fewer dimension, the faster computation could be achieved in the optimization. Based on these reasons, we choose the following field function

$$f_i(\mathbf{p}) = \begin{cases} B_i^2(1 - \frac{r_i^2}{R_i^2})^2, \text{ if } r_i \in [0, R_i] \\ 0, \qquad\qquad \text{elsewhere} \end{cases} \tag{2}$$

which is originally employed for a spherical blobby model, where i represents the index of a primitive, $r_i = d(\mathbf{p}, \mathbf{c}_i)$ returns the Euclidean distance from a given point $\mathbf{p}$ to the center of blob $\mathbf{c}_i$, $R_i = \sqrt{e_i^2(1 + A_i)}$ defines the influence region, e_i is the radius of ith sphere. A_i and B_i are two parameters that can be adjusted in a field function — A_i is used to adjust the influence radius and B_i is conducted to change the shape of f_i. Substituting the field functions into Eq.(1) defines an implicit surface using points $\mathbf{c}_i$ as skeletons which is similar to the function in [7], but provides two more parameters on each field function to adjust the implicit surface.

When using the field function defined in Eq.(2), all primitives are spheres. We distort every $f_i(\mathbf{p})$ to $f_i^{new}(\mathbf{q})$ in the following way to introduce ellipsoidal blobs

$$f_i^{new}(\mathbf{q}) = f_i(\mathbf{Tq}) = f_i(\mathbf{p}) \tag{3}$$

where $\mathbf{T}$ is a transformation matrix to map the point $\mathbf{q}$ on an ellipsoid E onto the point $\mathbf{p}$ on its largest inscribed sphere S. In general, an ellipsoid can be represented implicitly by a matrix $\mathbf{Q}$ as $[x\ y\ z\ 1]\mathbf{Q}[x\ y\ z\ 1]^T = 0$, and a sphere can be expressed as $[x\ y\ z\ 1]\mathbf{P}[x\ y\ z\ 1]^T = 0$, where $\mathbf{P}$ is a matrix. Relating them by $\mathbf{P} = \mathbf{TQT}^T$, we can determine the transformation matrix $\mathbf{T}$. The

mathematical representation of an ellipsoidal blobby model is then obtained by substituting Eq.(3) into Eq.(1).

4.2 Model Reconstruction

The ellipsoidal blobby model Ω approximating a given polygonal object can be reconstructed through an automatic procedure. First of all, we take the ellipsoids decomposed in section 3 as the initial blobs — the transformation matrix of each ellipsoid and its largest inscribed sphere are computed at the meanwhile. After that, a numerical optimization is conducted to reduce the difference between the isosurface of Ω and the given model. We use the downhill simplex algorithm [30] to optimize the parameters. The object function is defined as $F = \frac{1}{m}(\sum_{p \in \overline{\Omega}} (f(\mathbf{p}))^2)$, where m is the number of the samples, $\overline{\Omega}$ is the sample points set. To accelerate the optimization procedure, the centers of blobs are fixed, so we have only two parameters, A_i and B_i, to be optimized for each blob. For all the examples shown in this paper, the isosurfaces are computed on $f(\mathbf{x}) = 0$ with the threshold parameter $t = 1.0$.

Figure 3 shows a comparison of the reconstructed blobby models using spherical blobs [7] vs. ellipsoidal blobs, where the left is the result using a spherical blobby model defined by 639 blobs and the right one is an ellipsoidal blobby model with only 300 blobs. The approximation errors, L_{mean} and L_{max}, are computed by the publicly available Metro tool [31] and the polygonization algorithm [32]. Every spherical blob is with 5 coefficients to record its position and shape, and each ellipsoidal blob has 9 coefficients — thus for the spherical blobby result $5 \times 639 = 3195$ numbers need to be record while only $9 \times 300 = 2700$ coefficients are needed for the ellipsoidal model. From the results, it is not hard to conclude that the ellipsoidal blobby model gives a better result with even less primitives.

5 Applications of Ellipsoidal Blobby Models

5.1 Data Reduction of Geometric Model

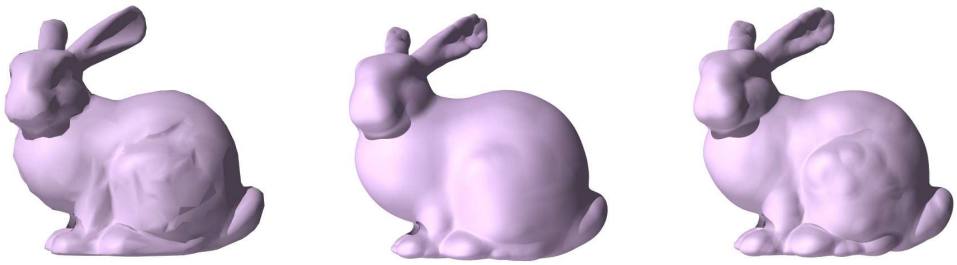
The first application of the ellipsoidal blobby models is in the area of geometry data reduction — i.e., lossy geometry compression. For a given mesh model, if it has n_v vertices and n_f triangular faces, the total bytes to recording this model is $12n_v + 12n_f$ where every vertex has 3 float numbers for its $\Re^3$ coordinate, every triangle has 3 integers for encoding indices of vertices, and both integer and float numbers occupy 4 bytes. On the other side, for the

Fig. 3. The reconstructed implicit surfaces using spherical (top left: with 639 blobs, $L_{mean} = 0.16$ and $L_{max} = 0.53$) and ellipsoidal (bottom left: with 300 blobs, $L_{mean} = 0.03$ and $L_{max} = 0.29$) blobby objects respectively. The top middle is a superimposed image of the given model and spherical blobs. The bottom middle is an image superimposed with the given model and ellipsoidal blobs. The two right images are the given bunny model.

approximation with n_b ellipsoidal blobs, we need to conduct $36n_b$ bytes to record the ellipsoidal blobby model. Nine float numbers are used to define a blob, which are B_i, the center $\mathbf{c}_i$ of the blob, the orientation (θ_x, θ_y) and the radiuses (r_x, r_y, r_z) of the ellipsoidal blob, where $r_k = t_k\sqrt{e_s^2(1 + A_i)}$, t_k is the ratio between the radii of the ellipsoid E and its largest inscribed sphere S, $k \in \{x, y, z\}$. Based on the Euler formula $n_f \approx 2n_v$, a triangular mesh model needs about $36n_v$ bytes. Therefore, as long as $n_b < n_v$, the data size has been reduced.

We have implemented the ellipsoidal-blob approximation algorithm on a PC with standard configuration (Inter PIV 3.4GHz CPU + 1GB RAM). Figures 4-7 show the experimental results of approximating polygonal meshes by ellipsoidal blobs. In each example, the original mesh model, and the ellipsoidal blobby models before vs. after optimization are shown. The approximation errors, both L_{mean} and L_{max}, are also reported on each example. Again, the approximation errors are computed by the publicly available Metro tool [31] after polygonization by [32]. We can easily find that after optimizing the blending parameters, the approximation errors have been significantly reduced.

The running time analysis of the algorithm for the examples shown in Figures 4-7 is listed in Table 1. During the ellipsoid decomposition, we use about 60,000 sample points to cover the models and avoid intersection between one



(a) (b) (c)

Fig. 4. Bunny: (a) the given polygonal model with 2,557 vertices, (b) the ellipsoidal blobby without optimizing A_i and B_i ($L_{mean} = 0.24$ and $L_{max} = 0.68$), and (c) the final fitting result ($L_{mean} = 0.03$ and $L_{max} = 0.29$). The blobby models are with 300 blobs ($\approx 87\%$ data reduction).

ellipsoid and the original model, and record the ellipsoids according to their volume contributions using a $110 \times 110 \times 110$ voxel grid. In Figure 4, we generate 2,500 candidates and select 300 ellipsoids with significant volume contribution. In Figure 5 and 6, we generate 2,500 ellipsoid candidates and select 400 ellipsoids from them. In Figure 7, 3,560 ellipsoid candidates are generated and 1,000 ellipsoids are selected. In the optimizing procedure, three iterations are conducted for each blob. In all examples, mesh vertices are used as surface sample points except the bunny model which uses 25,000 sampled points on its mesh surface. The statistic of computational time is reported in seconds. The complexity of the ellipsoid decomposition and the greedy selection is $O(mn)$ and $O(k \log m)$, where n is the number of surface sample points, m is the number of ellipsoid candidates and k is the number of finally selected ellipsoids.

Table 1
The running time of ellipsoidal-blobs approximation.

Model	Figure	Time for ellipsoid decomposition (s)	Time for the optimization of blobby objects (s)	Total time(s)
Bunny	4	3,139	360	3,499
Dog	5	3,367	1,010	4,377
Horse	6	1,568	789	2,357
Dragon	7	2,029	1,538	3,567

5.2 Target Shape Controlled Cloud Animation

The second application presented here is the target shape controlled cloud animation. Cloud is an important element of natural scene with their various

(a) (b) (c)

Fig. 5. Dog: (a) the given triangular mesh model with 49,998 vertices, (b) the ellipsoidal blobby without optimizing A_i and B_i ($L_{mean} = 0.27$ and $L_{max} = 0.85$), and (c) the final fitting result ($L_{mean} = 0.07$ and $L_{max} = 0.45$). The blobby models are with 400 blobs ($\approx 99\%$ data reduction).

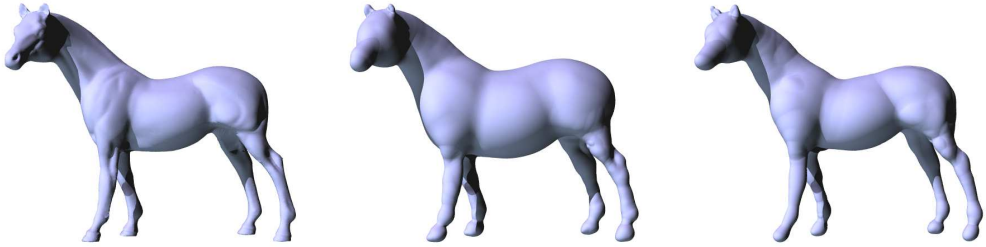
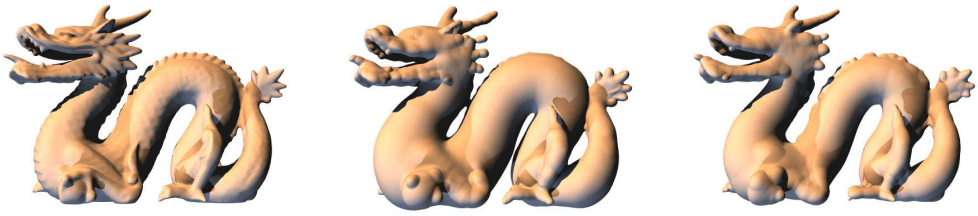


Fig. 6. Horse: (a) the given polygon model with 48,486 vertices, (b) the ellipsoidal blobby without optimizing A_i and B_i ($L_{mean} = 0.36$ and $L_{max} = 0.76$), and (c) the final fitting result ($L_{mean} = 0.05$ and $L_{max} = 0.33$). The blobby models are with 400 blobs ($\approx 99\%$ data reduction).



(a) (b) (c)

Fig. 7. Dragon: (a) the given model with 24,708 vertices, (b) the ellipsoidal blobby without optimizing A_i and B_i ($L_{mean} = 0.25$ and $L_{max} = 0.62$), and (c) the final fitting result ($L_{mean} = 0.05$ and $L_{max} = 0.43$). The blobby models are with 1000 blobs ($\approx 96\%$ data reduction).

fascinating appearances. Moving clouds usually give plenty of space for imagination. We feel exciting when a cloud in the sunshiny sky resembles the shape of an animal or some other real objects. The magic performance of clouds with their shapes and metamorphosis are often employed in movies. For instance, recently, the movie — "Kung Fu Hustle" [33] used a special effect to morph a piece of cloud into a Buddha. Here, we are going to develop techniques for digitally producing the similar effect, which have many applications in entertainment industries. Our method involves using the aforementioned ellipsoidal blobby models to approximate the given model and to simulate the metamorphosis between objects with cloudy appearances. Benefited from the implicit representation, we can approximate both the target object and the initial clouds by blobby models and easily deform between them. Therefore, the only left issue we need to address is how to design animation schemes for the motion of blobs to simulate the metamorphosis of clouds naturally. Two schemes are proposed here.

5.2.1 Aggregation Scheme

The scheme simulates a large scale cloud animation, where several pieces of clouds are aggregated into the target shape progressively. The motion of clouds in an aggregation manner is produced backwards. To simulate the aggregation, the blobs approximating the target shape are firstly subdivided into several subsets. Then, the shape of clouds corresponding to each subset of blobs is modelled at several key-frames during the animation. Finally, the cloud animation is generated by interpolating the positions and shapes of blobs between key-frames.

In detail, the bounding box of the ellipsoidal blobby model approximating the target shape is computed, and the space covered by the bounding box is subdivided into several sub-spaces by some planes. The planes are defined by users or generated automatically according to pre-defined rules. For instance the cloud animation resembling a dragon shown in Figure 8, after determining the blobby model, we divide the space of its bounding box into 24 sub-regions by six planes: $y = -1/6h$, $y = 1/6h$, $z = 0$, $x = 0$, $z = x$, $z = -x$, where h is the height of the bounding box. The ellipsoidal blobs are classified into 24 clusters by the positions of their centers, where the blobs falling into one cluster are conducted to model a piece of cloud. The moving direction for a piece of cloud is defined according to the region where it locates (following some prescribed rules) and the magnitude of its velocity is then generated randomly. Based on their velocities, the positions of each cluster of blobs in the key-frames can be easily determined.

Simply placing the clusters into their corresponding key-frame positions is not enough to generate realistic cloud animation. In order to improve the

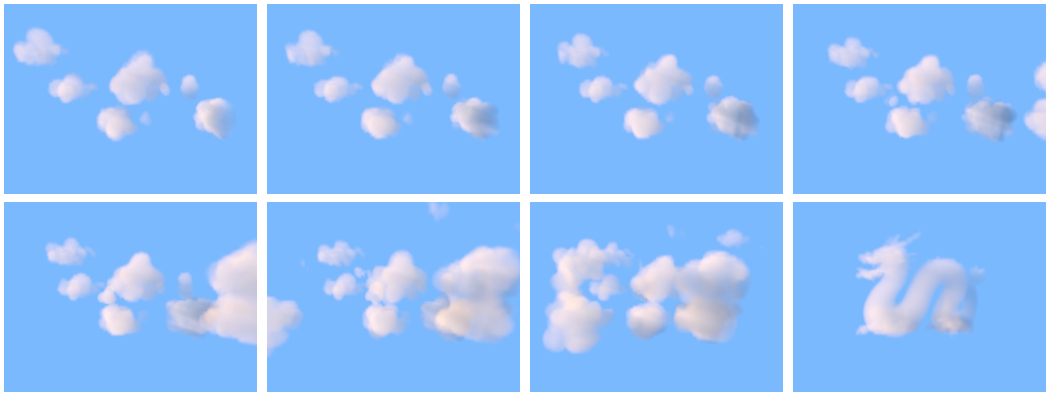


Fig. 8. Eight key-frames in the cloud animation forming the shape of a dragon.

reality, in each key-frame, we slightly move every blob in random while keeping it in the bounding box of its cluster. Meanwhile, the size of a blob is also slightly adjusted in random. At last, we conduct particle forces to repulse ellipsoidal blobs so that they do not intersect each other too much. In this way, the positions and shapes of all blobs in each key-frame are determined. The clouds at every time step can be finally generated by interpolating the position and shape of every ellipsoidal blob in all key-frames, which results in a realistic animation that each piece of cloud (i.e., cluster of blobs) moves in a constant speed between key-frames but the shapes and the velocities of blobs vary randomly. More specifically, in every time step of the animation, after determining the positions and the shapes of all blobs, the implicit surface defined by the blobs is computed by Eq.(1) and displayed by volume rendering. Figure 9 shows eight key-frames of a cloud animation generated by this scheme.

5.2.2 Diffusion Scheme

In the second scheme, the cloud with a specified target shape is formed starting from one piece of cloud in a diffusion manner. Different from the aggregation scheme, the cloud animation in this scheme is produced in a forward manner. When we progressively display the ellipsoidal blobs of a blobby model in some sequence, we can simulate the diffusion effect of fluid dynamics similar to [27].

To generate a diffusion-like animation, we classify the blobs into three categories:

- Category I: The initial cloud blob;
- Category II: The cloud blobs represent the target shape roughly;
- Category III: The blobs refine the target shape of clouds.

As the greedy optimization has been conducted in the ellipsoid selection (section 3), the first selected ellipsoid is always with the largest volume (which

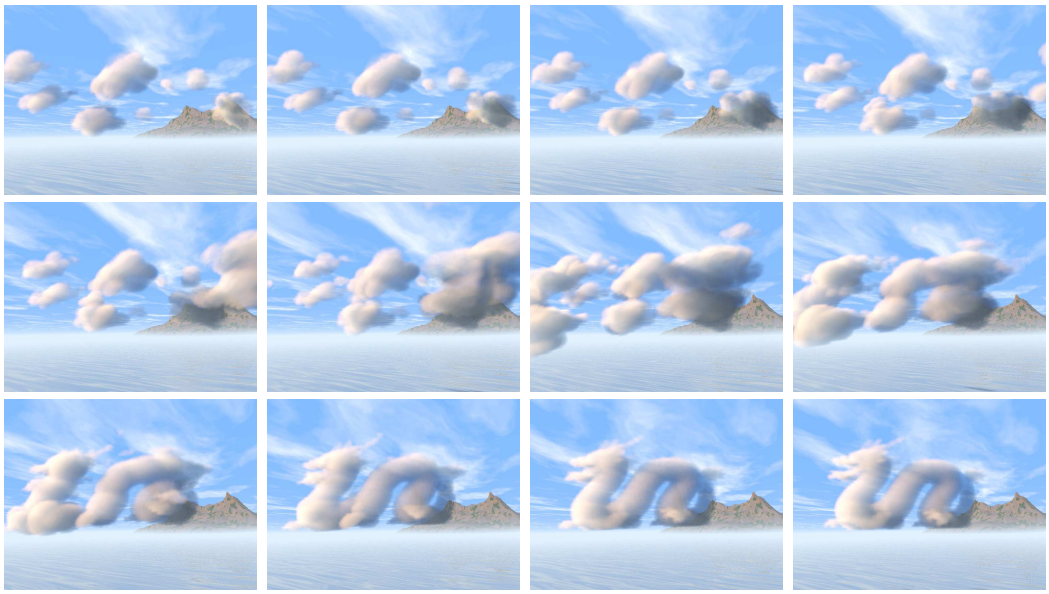
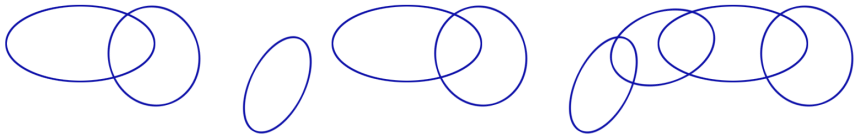
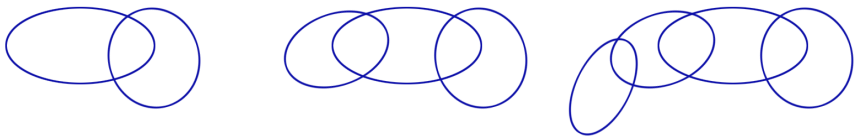


Fig. 9. The cloud animation forming a dragon shape produced by the aggregation scheme.



(a) Unsatisfactory order.



(b) Order for diffusion effect.

Fig. 10. Orders for displaying blobs.

is classified to the blob in category I) followed by the ellipsoids adding which leads to the most significant change of the volume among the rest ones. Thus, except the very beginning one, the first 10% blobs in the decomposition sequence are classified into category II while the rest 90% fall into the last category. The blobs of three categories are consecutively added into the implicit model (defined in Eq.(1)) to generate the diffusion-like cloud animation.

However, simply displaying the blobs by their order in the decomposition sequence may lead to artificial results. For example, the sequence of ellipsoid selection in the order shown in Figure 10(a) will lead to two separated cloud regions, which acts against the natural phenomenon of diffusion. A diffusion procedure usually desires the continuous expansion of cloud regions. Therefore,

Fig. 11. The cloud animation resembling a horse shape produced by the diffusion scheme.

we adjust the displaying order of blobs following the rule that: the newly added blob should be an ellipsoidal blob whose signed distance from its center to the current implicit surface $f(\mathbf{p}) = 0$ is smaller than other remaining blobs in the same category. Here, the signed distance means that the value of the distance is negative if its center $\mathbf{c}_p$ leads to $f(\mathbf{c}_p) < 0$. For instance, the newly determined order is shown in Figure 10(b). Certainly, the order of displaying can be further adjusted by users through an interactive tool. For the blobs of Category III, which represent the details of the model, they will be finally added into the implicit model together.

According to the displaying order of blobs, we generate the diffusion-like cloud animation by adding the ellipsoidal blobs into the implicit model. When adding a new blob, we first show it at the position with its center coincident to the nearest visible blob. In the following, the blob is moved to its proper position with the size being changed from zero to its own size. This changes the shape of cloud smoothly. Figure 11 shows the cloud animation produced by the diffusion scheme. The cloud diffuses from one small piece and finally matches the user specified target shape — a horse.

6 Conclusion and Discussion

This paper developed an automatic scheme for approximating a given polygonal mesh with an ellipsoidal blobby model. The experimental results prove that the reconstructed implicit surfaces can approximate the original model very well. Based on this, we demonstrate two applications of the blobby models: the geometry data reduction and the target shape controlled cloud animation. In summary, our work presented in this paper has the following contributions.

- A novel implicit ellipsoidal blobby model has been presented in this paper

to approximate a given model — this representation is more accurate than the spherical blobby models when adopting the same number of blobs. By this approximation, we actually introduce a data reduction technique where we can use much smaller size of data to approximate the given models.

- Output of the ellipsoid decomposition algorithm has been enhanced, which can preserve more details comparing to the previous scheme.
- Based on the ellipsoidal blobby representation, two geometry-based schemes for cloud animations are introduced — one simulates the motion of clouds in an aggregation manner while the other generates cloud animation in a diffusion way. Both of these schemes can effectively produce realistic cloud animations that mimic results from the time-consuming physics-based modelling approaches but much faster.

One limitation of the proposed approach is that aliasing exists on sharp features when using a blobby implicit surface to approximate the shape of a given polygonal model. Although the newly developed ellipsoidal blobby model can reduce the approximation error when comparing to the spherical blobby models, the approximation does not converge to the discontinuous features even if the sampling rate is increased. This is because that the primitives in a blobby model are all continuous. To solve this convergence problem, some other quadratic primitives with sharp edges/corners will be considered in our future research.

7 Acknowledgements

The authors would like to acknowledge the helpful comments given by the reviewers. This work was supported by the National Natural Science Foundation of China (Grant No. 60573153), Natural Science Foundation of Zhejiang Province (Grant No. R105431) and Program for New Century Excellent Talents in University (Grant No. NCET-05-0519). It was partially supported by the Hong Kong RGC/CERG grant CUHK/412405 and the CUHK project CUHK/2050341.

References

[1] Blinn JF. A generalization of algebraic surface drawing. ACM Transactions on Graphics 1982;1(3):235-256.

[2] Nishimura H, Hirai M, Kawai T, Kawata T, Shirakawa I, Omura K. Object modeling by distribution function and a method of image generation.

Transaction IEICE Japan 1995;J68-D(4):718-725.

[3] Wyvill G, McPheeters C, Wyvill B. Data structure for soft objects. The Visual Computer 1986;2:227-234.

[4] Muraki S. Volumetric shape description of range data using blobby model. Computer Graphics 1991,25(4):227-235.

[5] Tsingos N, Bittar E, Gascuel MP. Semi-automatic reconstruction of implicit surfaces for medical applications. In: Proceedings of Computer Graphics International'95, 1995. p. 3-15.

[6] Bittar E, Tsingos N, Gascuel MP. Automatic reconstruction of unstructured 3D data: combining a medial axis and implicit surfaces. Computer Graphics Forum 1995,14:457-468.

[7] Jin XG, Liu SJ, Wang CCL, Feng JQ, Sun HQ. Blob-based liquid morphing. Computer Animation and Virtual Worlds 2005,16:391-403.

[8] He T, Hong L, Kaufman A, Varshney A, Wang S. Voxel based object simplification. In: IEEE Visualization'95, 1995.p.296-303.

[9] Nooruddin F, Turk G. Simplification and repair of polygonal models using volumetric techniques. IEEE Transactions on Visualization and Computer Graphics 2003;9(2):191-205.

[10] Kajiya JT, Herzen BPV. Ray tracing volume densities. Computer Graphics 1984,18(3):165-174.

[11] Stam J. Stable fluids. In: Proceedings of SIGGRAPH'99, 1999. p. 121-128.

[12] Stam J. Interacting with smoke and fire in real time. Communications of the ACM 2000,43(7):76-83.

[13] Fedkiw R, Stam J, Jensen HW. Visual simulation of smoke. In: Proceedings of SIGGRAPH'01, 2001. p. 15-22.

[14] Ebert DS. Volumetric modeling with implicit functions: a cloud is born. In: Visual Proceedings of ACM SIGGRAPH'97, 1997. p. 147.

[15] Dobashi Y, Nishita T, Yamashita H, Okita T. Using metaballs to modeling and animate clouds from satellite images. The Visual Computer 1998;15(9):471-482.

[16] Ebert DS. Procedural volumetric cloud modeling and animation. In: Simulating nature: from theory to application. 1999. p. 5.1-5.52. SIGGRAPH'99 Course Notes No 26.

[17] Elinas P, Stürzlinger W. Real-time rendering of 3D clouds. Journal of Graphics Tools 2000,5(4):33-45.

[18] Dobashi Y, Kaneda K, Yamashita H, Okita T, Nishita T. A simple, efficient method for realistic animation of clouds. In: Proceedings of SIGGRAPH'00, 2000. p. 19-28.

[19] Ebert D, Musgrave F, Peachey D, Perlin K, Worley S. Texturing & Modeling: A Procedural Approach, Morgan Kaufmann Publishers; 2002.

[20] Schpok J, Simons J, Ebert DS, Hansen C. A real-time cloud modeling, rendering, and animation system. In: Symposium on Computer Animation'03, 2003. p. 160-166.

[21] Bouthors A, Neyret F. Modeling clouds shape. In: Eurograhics'04 (short papers), 2004.

[22] Gardner GY. Visual simulation of clouds. Computer Graphics 1985;19(3):279-303.

[23] Nishita T, Takao S, Katsumi T, Nakamae E. Display of the earth taking into account atmospheric scattering. In: Proceedings of SIGGRAPH'93, 1993. p. 175-182.

[24] Nishita T, Takao S, Nakamae E. Display of clouds taking into account multiple anisotropic scattering and sky light. In: Proceedings of SIGGRAPH'96, 1996. p. 379-386.

[25] Fattal R, Lischinski D. Target-driven smoke animation. ACM Transactions on Graphics 2004,23(3):439-446.

[26] Yu YZ, Shi L. Object modeling and animation with smoke. Technical Report UIUCDCS-R-2002-2262. Computer Science, University of Illinois at Urbana-Champaign. 2002.

[27] Shi L, Yu YZ. Controllable smoke animation with guiding objects. ACM Transactions on Graphics 2005,24(1):1-25.

[28] Bischoff S, Kobbelt L. Ellipsoid decomposition of 3D-models. In: Proceedings of 3DPVT'02, 2002. p. 480-488.

[29] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: Proceedings of SIGGRAPH'97, 1997. p. 209-216.

[30] Press WH, Flannery BP, Teukolsky SA, Vetterling WT. Numerical Recipes in C. Cambridge University Press: Cambridge, 1988.

[31] Cignoni P, Rocchini C, Scopigno R. Metro: measuring error on simplified surfaces. Computer Graphics Forum 1998;17(2):167-174.

[32] Bloomenthal J. An implicit surface polygonizer. In: Heckbert PS, editor, Graphics Gems IV. New York: Academic Press; 1994.p.324-350.

[33] "Kung Fu Hustle". http://www. kungfuhustle.com/.