Journal of Korea Multimedia Society Vol. 12, No. 6, JUNE 2009(pp. 842-855)

Interactive and Intuitive Physics-based Blending Surface Design for the Second Order Algebraic Implicit Surfaces

Tae-Jung Park[†], Hyeong Ryeol Kam[††], Seung-Ho Shin[†††], Chang-Hun Kim[††††]

ABSTRACT

We present a physics-based blending method for the second order algebraic implicit surface. Unlike other traditional blending techniques, the proposed method avoids complex mathematical operations and unwanted artifacts like bulge, which have highly limited the application of the second order algebraic implicit surface as a modeling primitive in spite of lots of its excellent properties. Instead, the proposed method provides the designer with flexibility to control the shapes of the blending surface on interactive basis; the designer can check and design the shape of blending surfaces accurately by simply adjusting several physics parameter in real time, which was impossible in the traditional blending methods. In the later parts of this paper, several results are also presented.

Key words: the second order algebraic implicit surface, implicit modeling, mesh, physics based modeling, implicit surface blending, polygonization

1. INTRODUCTION

1.1 Implicit Surface

Implicit surfaces are defined as an implicit function form of $f(\boldsymbol{p}) = 0$ in a space where $\boldsymbol{p}$ is a geometric position. The implicit surface technique has various applications including scientific data visualization [1], mechanical modeling [2], animation [3], etc. One of major advantages of implicit surface technique is the ability to merge or blend multiple independent objects in a natural way. To get smooth connections of any other primitives, the process called *blending* is involved in general. *Blending* in implicit surface can be referred as the process to add another surface(i.e. *blending surface*) to the implicit surfaces to smooth unwanted creases or peaks, which may appear at the intersections of different implicit surfaces, to give aesthetic improvement or to enhance aerodynamics. Theoretically, implicit surfaces can make use of any form of implicit functions. However, high order functions need higher computational power and often cannot be an ideal choice for representing any models. Thus, implicit models are often represented with the second order algebraic implicit surfaces. *The second order algebraic implicit surface* is expressed as the second order polynomials of x, y, and z. It is also called "quadric surface". Depending on its coefficients, a quadric surface can be ellipsoid, hyperboloid, cylinder, cone, etc.

To delineate blending surfaces for the second order algebraic implicit surface, most of previous works depend on algebraic distance or low-pass filtering. The "*algebraic distance*" from a point $\boldsymbol{p}$ to the surface $f^{-1}(0)$ is defined as $f(\boldsymbol{p})$ when implicit function $f(\boldsymbol{x})$ is continuous. When $f(\boldsymbol{x})$ is linear, not very often though, the algebraic distance

※ Corresponding Author : Chang-Hun Kim, Address : (136-701) Department of Computer Science, Korea University, Seoul, Korea TEL : +82-2-3290-3574, FAX : +82-2-929-1917, E-mail : chkim@korea.ac.kr

Receipt date : Sep. 30, 2008, Approval date : Jan. 19, 2009

[†] BK21 Software, Department of Computer Science, Korea University
(E-mail : taejung.park@gmail.com)
[††] Department of Computer Science, Korea University
(E-mail : neary@korea.ac.kr)
[†††] Department of Computer Science, Korea University
(E-mail : winstorm@korea.ac.kr)
[††††] Department of Computer Science, Korea University

is proportional to the Euclidean distance. Unfortunately, most of the algebraic implicit functions do not provide with the Euclidean distance. This means we should solve complex (often, non-linear) multiple equations in implicit forms numerically - thus, inevitably iteratively - for accurate blending surface design. Other blending surface algorithms based on the low-pass filtering use the convolution integral or similar mathematical computations, which also require higher computational power. Due to this aspect, the design process of blending surfaces has been highly mathematical oriented and time-consuming in general.

1.2 Limitations in Designing Blending Surfaces

As pointed out in the previous section, traditional approaches for blending surface have some problems which make the design process more difficult and inflexible for the designers who would want more flexible, intuitive, and interactive design platform. Those problems can be described in detail as follows:

First, in an algebraic blending algorithm, an unwanted artifact called *"bulge"* can appear unexpectedly; bulge can happen at the area where more than one implicit function meet each other. This area appears unwantedly swollen. See Figure 5. To solve this problem, previous approaches including [7], add extra mathematical functions. However, those approaches still make it difficult to anticipate the final shapes of blending surfaces intuitively when multiple primitives are involved because they may require to solve complex implicit equations as explained in section 1.1. Second, other blending methods based on the low-pass filtering require expensive mathematical operations including convolution integration. Eventually, these problems can make the design process time-consuming, non-intuitive, and non-interactive, even for the second order algebraic surfaces which are relatively "light" in computational burden. As a result, the designer is required to wait for a while even with one trivial modification. Besides, the implicit surface basically needs separate visualization schemes such as polygonization or ray-tracing, which again highly limit interactivity.

1.3 Overview of the Proposed Method

As described in the previous sections, it is more difficult for designers who have little knowledge of computational geometry or mathematics to control blending surfaces exactly to meet their requirement. To address those issues, we suggest an algorithm for blending surface based on real-time physical simulation that provides similar characteristics to plastic wrap or rubber sheet. The whole blending process can be compared to wrapping objects with plastic wrap. In this scenario, the individual object corresponds to one of implicit surfaces and the plastic wrap creates smooth surface that connects the separate objects.

Since physics simulation needs structural models, it was natural to implement the blending surface based on a mesh structure. In this document, we define *"wrapping mesh"* as a manifold of genus zero which has been generated procedurally and its vertices and edges correspond to mass-points and spring-dampers respectively. Being modeled with physical elements, wrapping mesh can move and deform freely according to forces.

Before start, the wrapping mesh is configured to encompass the target geometry which consists of the second order algebraic implicit primitives including ellipsoids or super-ellipsoids. In this stage, the shape of wrapping mesh can be selected as a sphere with minimum bounding radius and all of the springs on the wrapping mesh are assigned with enough tension to allow the mesh to shrink toward the geometry. At the final step, part of mass points contact the geometry and other do not. The result is a wrapping mesh that smoothly connects each primitives as shown in Figure 3-a, 3-b, and 3-c. If necessary, the designer can insert new primitives as well as modify or translate existing ones in real-time. In addition, subtle characteristics of this blending surface can be adjusted by changing physical parameters such as spring constant or adding additional forces to enhance controllability.

1.4 Contribution

The main contributions of our approach can be summarized as follows:

First, our algorithm purely depends on physical simulation including collision detection between implicit surfaces and the blending surface. As a result, the designer can work on our blending surface easily with analogy to flexible membrane or plastic wrap whose physical characteristics are well known to them, not with mathematical formulas. Second, this scheme does not produce complex or high order blending surface functions which can limit interactivity for designers in various situations. Finally, any changes made in design process are reflected immediately because the polygonization is incorporated in creating blending surface in real-time. Utilizing these advantages, the overall design process becomes more intuitive, efficient, and interactive.

1.5 Paper Organization

In chapter 2, we review the previous works on polygonization of implicit surfaces, as well as implicit surfaces themselves because our physics model is based on polygonized mesh. After that, the overall scheme for our approach will be introduced in chapter 3. General strategy for interactive design process will be also discussed there. In this chapter, we also introduce a simpler method to find the location of a vertex on the implicit surface for quadric and super-quadric primitives instead of the binary sectioning algorithm [4], which requires several trial-and-error steps to determine a correct position on implicit surface. Then the physics-based model used in our approach is examined in detail in chapter 4. In chapter 5, some results are presented with discussion. Finally, limitations of our approach and future work are discussed.

2. RELATED WORK

2.1 Interactive Design

As pointed out before, implicit surface modeling technique, including blending, depends heavily on mathematical properties. As a result, any graphic designers who have little mathematical background have had difficulties to access this technique. To address this problem, some interactive and intuitive methods based on virtual reality environment have been introduced [5,6] for implicit geometry. In these works, a metaphor of clay work is adopted and designers use haptic or force-feedback devices to manipulate the implicit "solid" just like sculpting soft objects. These methods have the ability to cover complex topology. However, these approaches are not suitable for the applications where fine details are appreciated. Meanwhile, because our method can handle details sophisticatedly on the blending surface with an analogy to flexible membrane whose physical properties can also be adjusted, it has an advantage over the previous ones to create more detailed surfaces.

2.2 Polygonization

To implement an interactive design environment for implicit geometry, it is imperative to visualize any changes in implicit geometry in real-time. Because implicit surfaces are just represented as an implicit form of equation (or single function as a final result of a series of operations including Boolean operation over other functions, for example, in the applications such as constructive solid geometry), i.e. $f(\mathbf{p})=0$, it needs separate methods for visualization. The general visualization methods for implicit geometry can be divided into two categories; ray-tracing and polygonization. Ray-tracing technique [7] can produce high quality photo-realistic images for implicit geometry. In addition, implicit surface divides space into three parts, i.e. inside of the surface, outside of the surface, and surface itself so that ray-tracing technique can be readily applied to implicit geometry. However, it is quite obvious in this case that ray-tracing technique is not a choice for the visualization in interactive design system because current hardware cannot handle ray-tracing algorithm in real-time. Due to this limitation and its robustness in current graphic pipeline structure, the polygonization technique is a natural choice for the aforementioned environment. Polygonization methods can be divided into 3 categories; space partitioning, seed based, and physics based.

The space partitioning methods divide the space in problem into smaller spaces and scan whether the implicit surfaces have involved with the small space. One of most robust algorithms is the marching cube method [8]. In this method, the space is partitioned into small cubes. This method is simple and robust but it can take considerable time when the scan volume is large or the grids are too fine. For some real-time applications including [5] and [6], modified versions of marching cube are introduced; they only update changes locally to speed up the visualization routine enhancing details. Also, [9] applies Morse theory [10] and interval analysis [11] to guarantee exact topology of implicit surface in a real-time space-partitioning scheme. This method recognizes whether there were any topological changes in the implicit geometry at each frame and updates the changes in real-time. One of disadvantages in the standard marching cube algorithm is that the quality of mesh depends on the size of the grids. This problem becomes obvious in polygonizing sharp features of implicit geometry. To address this problem, an optimization method is introduced [12] for polygonized implicit surfaces based on the observation that sharp features can be restored more effectively in dual mesh than original version.

Meanwhile, *the seed based polygonization techniques* [13,14] apply "seeds of vertices" to each implicit primitives to get regular mesh structure at polygonization stage, rather at optimization one. Those methods are quite straightforward because each seed vertex can be projected onto any implicit primitives easily with binary sectioning algorithm [4]. However, they need additional procedure to recognize the boundaries of each primitive and to "sew up" separately polygonized meshes. In general, blending is applied to these boundary zones in implicit geometry so that the polygonization process for blended zone can be a bottleneck in these approaches. In our approach, similar process to project the vertices onto implicit primitives is necessary. But we introduce a simpler algorithm than the binary sectioning, which can be applied to quadric or super-quadric primitives.

Finally, *the physics-based polygonization schemes* adopt particle system and physics simulation to visualize implicit geometry. For example, in [15], particles or disks are used to visualize implicit surfaces by calculating attraction/repulsion process. Since this method depends on physical forces exerted on each particle, the result is more regular than that of space partitioning methods. However, unlike our approach, the particles or disks do not form any mesh structure in this method.

2.3 Blending

Blending in implicit surfaces make the neighbors of multiple implicit surfaces visually natural when they are located very close or meet each other. In the blobby object [1], which consists of weighted pseudo-Gaussian bump functions, the blending operation is as simple as just applying arithmetic operations to the implicit functions involved. For more general cases, arithmetic blending methods and low-pass filtering methods are generally adopted.

The arithmetic blending methods [16,17] depend on the algebraic distance. Those methods perform algebraic operations on the polynomial primitives to get blended implicit function. However, those methods, as mentioned in the earlier chapter, generate unwanted artifacts including one called "bulge" [16]. "Bulge" appears as an unwanted swelling where more than one implicit function meet each other as shown in Figure 5. In particular, this problem can be frustrating for designers making it difficult to anticipate the exact shape of blending surface. In this paper, the cause of this problem will be discussed briefly for a simple 2D case. Its remedy for blending surface algorithms based on algebraic distance is introduced in [18]. For complex geometry, higher order blending surface was also studied as an extension of potential method in [19]. Unlike these methods, *the low-pass filtering schemes* including the convolution surface [20] do not have any bulge problem and enable procedural design. As its name implies, the convolution surface uses convolution operation, which is widely used in processing images or other signals. Just like the smoothing operation in 2D images, this scheme filters high frequency components in the 3D geometries. Also, similar blending effect can be achieved by properly applying Minkowski sum [21]. This methods produce excellent results as shown in [22]. However, for designers without proper mathematics knowledge, all of above approaches are not so easy to predict the result of the operations. Unlike those methods, our approach gives more direct and intuitive design methodology to designers.

2.4 Euler Integration for Realtime Simulation

As more powerful computing hardware is available for computer graphics, the demand for real-time applications like real-time 3D games, training simulations, and virtual reality simulations has increased rapidly. In these applications, the implicit Euler integration plays a great role to provide stable and fast physical simulation [23,24]. Based on this stability and robustness, it is possible to reduce simulation time step considerably. As a result of implicit integration, natural and stable simulation of garment is possible [25,26]. Our method also benefits from the stability of implicit Euler integration.

3. OVERALL SCHEME

In general, most commercial CAD or 3D graphic modeling software packages are based on an iterative design process; the designer creates initial models from ideas, check the intermediate results visually, rework on the results if necessary, check again, etc. This iterative process continues until satisfactory results are achieved. From this nature of graphic design processes, it is highly required to provide the designer with immediate responses or feedbacks on any changes.

However, when it comes to the implicit surface - in particular nonlinear primitives including the second order algebraic surfaces, the whole design process has not met this requirement that enough. Some interactive design methodologies for implicit surfaces were reported [5,6] but there are rare studies on interactive blending surfaces, as far as

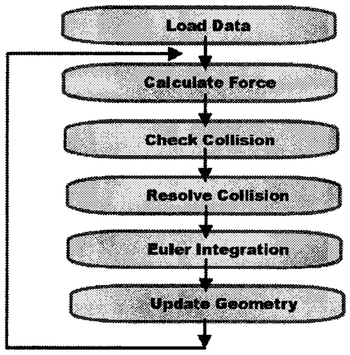


Fig. 1. Overall iterative process based on real-time physics simulation

we know. From this motivation, we present a blending surface scheme based on real-time physics simulation.

Figure 1 shows the overall flow chart. In this process, we use the implicit Euler integration method [23,24] which is stable and fast. Due to this advantage, it is possible to adjust various physics parameters to get proper blending surface shapes without suffering severe stability problems.

4. PHYSICS MODEL

4.1 Spring-Damper and Mass Point Structure

To model the physical structure of the wrapping mesh, we use mass point and spring-damper elements. As shown in Figure 2, the mass points correspond to the mesh vertices and the spring-damper to the mesh edges respectively. With this structure, we can simulate shrinking plastic wrap. Besides, thanks to the one-to-one relationship between physical elements and mesh elements, it is straightforward to apply this scheme to any arbitrary polygonal geometries.

The physics model can be expressed in following equations where two bodies are connected with one spring and one damper:

$$F_1 = -\{k_s(L-r) + k_d[(v_1 - v_2) \cdot L]/L\} L/L \quad (1)$$

$$F_2 = -F_1 \quad (2)$$

where $\mathbf{F}_1$ is the force exerted on body 1 and $\mathbf{F}_2$ on body 2. k_s is the spring constant and k_d is the damping constant. $\mathbf{F}$, $\boldsymbol{v}$, and $\boldsymbol{r}$ have following relationship in each body:

$$F = m\dot{v} \quad (3)$$

$$v = \dot{r} \quad (4)$$

In equation (1), $\mathbf{L}$ means the vector whose length is the contracted distance of the spring-damper, i.e. $\mathbf{L} = k_{con}\,(\mathbf{p}_1 - \mathbf{p}_2)$ where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the position vector of body 1 and body 2, respectively. The parameter k_{con} represents the pre-tension ratio between any two vertices. With this parameter, we can adjust the initial tension on each edge by changing k_{con} between 0 and 1. If k_{con} is bigger than 1, the wrapping mesh will not shrink but expand.

4.2 Collision Handling

One of the benefits we can get from the implicit surface technique is that it is relatively easy to handle collision detection between an implicit surface and any points. Since collision handling routine is one of bottlenecks in real-time applications if a number of collisions between objects are to be considered, the implicit surface has great advantage for such applications.

In our approach, collision would occur on the convex areas of implicit surfaces and finally the vertices involved in collision should be located on the surface (i.e. $f = 0$). As a final result, the concave areas make no contact with the wrapping mesh so that we can

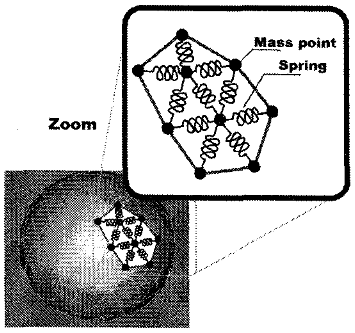


Fig. 2. Spring-damper and mass point structure on "wrapping mesh"

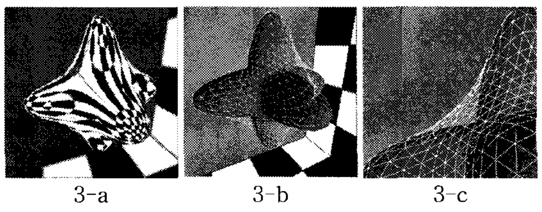


Fig. 3. Blending for 3 ellipsoids. 3-a is the final result of global bend process in real-time environmental cube mapping method. 3-b and 3-c show the relationship between the implicit surfaces (in red) and the wrapping mesh (wired mesh in white)

get blending polygonal surface. Figure 3 shows this situation.

In real-time graphic applications, any collision handling processes have two sub routines; the first one is to check whether any collisions occurred in the specific time step ("collision check" process) and the second one is to resolve the "collided" state to a normal one ("collision resolve" process).

4.3 Collision Check

Since any implicit surface divides the whole space into inside and outside of the surface, it is straightforward to check whether there is any collision with specific vertices with implicit surfaces. To determine whether collisions happened, a vertex point (i.e. x, y, z coordinate) is applied into the implicit function. If we put an implicit function as *f*, we can determine whether collision happened by examining the sign of the implicit function, i.e.

$f(v) > 0$ (*v is located outside of the surface.*)

$f(v) < 0$ (*v is located inside of the surface.*)

Therefore, if we get negative function values, we can conclude that the vertex is collided with the surface. Or else, the vertex is not collided with any surfaces.

This benefit of implicit surface becomes more apparent when compared with the collision check process in parametric representation of surfaces. A parametric surface in 3D space can be expressed as follows:

$$p = (g_x(s,t), g_y(s,t), g_z(s,t)) \qquad (5)$$
$$0 < s,t < 1$$

As we can see in the equation (5), every point is represented as a locus in 3D space so that the parametric surfaces have no straightforward ways to determine whether a vertex is inside or outside. As a result, the parametric representation needs an extra efforts to handle this operation. On the other hand, we can see that it is easier to display parametric representation of surfaces than implicit representation because we can get locus of surfaces by sweeping parameters within their ranges.

Because the implicit geometry in our application, presented in the later chapter of this paper, consists of multiple primitives which are mainly connected to each other with OR operation, it is necessary for a vertex to check which primitive was involved in the collision during the simulation. Some applications including [3] find which primitives were involved in contact using ray-tracing technique. In our approach, all of the vertices are moving from outside of primitives so that we can assume that each vertex plays a role of ray in each step.

4.4 Collision Resolve

When it is determined that a vertex is collided with any implicit surfaces, the vertex should be checked as "collided" and handled in collision resolve process. Whereas certain vertices are determined as being collided with negative value of *f*(v), it can be interpreted as being inside of the implicit surface. What should be done in this routine is to calculate the proper locations on the surface (i.e. *f*(v)=0) and move the collided vertices to those positions. In most cases, binary sectioning algorithm [4] is applied to determine the locations on the surface. Theoretically, this algorithm can be applied to all kinds of implicit surfaces. However, it needs lots of iterations to ensure certain degree of accuracy, which inevitably leads to slow down the whole process. Because the primitives are all ellipsoids or super-ellipsoids in our application, we can apply a simpler and more efficient algorithm to this routine as follows.

Figure 4 shows the situation where a single vertex is collided with a implicit surface in 2D space. Though it is a 2D example, we can expand this scheme to 3D space and super-quadric geometries without losing generality. In figure 4, the vertex A is inside of the surface (red) and need to move to point B, i.e. on the surface to resolve collision state. Since this is a 2D example, we can represent

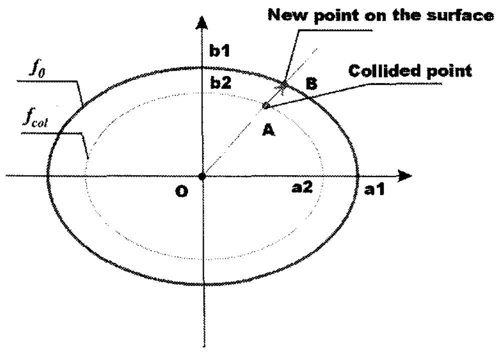


Fig. 4. Collision resolve overview in 2D space

these ellipses as follows:

$$f_0(x,y,z)=\left(\frac{x}{a_1}\right)^2+\left(\frac{y}{b_1}\right)^2-1 \quad (6)$$

$$f_{col}(x,y,z)=\left(\frac{x}{a_2}\right)^2+\left(\frac{y}{b_2}\right)^2-1 \quad (7)$$

The purpose in this step is to obtain the vertex position on B from the known parameters. To do so, we need to know the relationship between a_1 and a_2 or b_1 and b_2, alternatively.

If we set $\overrightarrow{OA}=(\alpha,\beta)$, then,

$$f_0(\overrightarrow{OA})<0 \quad (8)$$

$$f_{co\ l}(\overrightarrow{OA})=0 \quad (9)$$

We know that the following relationship is valid in the ellipses that share the same center:

$$\begin{aligned}&a_1:a_2=b_1:b_2\\&=|\overrightarrow{OB}|:|\overrightarrow{OA}|\end{aligned} \quad (10)$$

where $|\vec{v}|$ means Euclidean distance of vector $\vec{v}$. In the above relationship, we can define a positive constant K as,

$$K=\frac{a_1}{a_2}=\frac{b_1}{b_2} \quad (11)$$

From (6), (7) and (11),

$$\begin{aligned}&f_{col}(\overrightarrow{OA})\\&=\left(\frac{\alpha}{a_2}\right)^2+\left(\frac{\beta}{b_2}\right)^2-1\\&=0\end{aligned} \quad (12)$$

$$\begin{aligned}&f_0(\overrightarrow{OA})\\&=\left(\frac{\alpha}{a_1}\right)^2+\left(\frac{\beta}{b_1}\right)^2-1\\&=\left(\frac{\alpha}{a_2K}\right)^2+\left(\frac{\beta}{b_2K}\right)^2-1\\&=K^{-2}\left[\left(\frac{\alpha}{a_2}\right)^2+\left(\frac{\beta}{b_2}\right)^2\right]-1\\&=K^{-2}-1<0\end{aligned} \quad (13)$$

$$\begin{aligned}&\therefore K=1/\sqrt{1+f_0(\overrightarrow{OA})}\\&(0<K<1)\end{aligned} \quad (14)$$

Now, by using K, we can get the new point coordinate on the surface, $\overrightarrow{OB}$ as,

$$\overrightarrow{OB}=K\overrightarrow{OA}=\overrightarrow{OA}/\sqrt{1+f_0(\overrightarrow{OA})} \quad (15)$$

We can expand this result to super-quadrics in 3D space. In general, we can express super-quadric implicit functions as follows:

$$\begin{aligned}&f(x,y,z,e_1,e_2)\\&=\left[\left(\frac{x}{a}\right)^{2/e_2}+\left(\frac{y}{b}\right)^{2/e_2}\right]^{e_2/e_1}+\left(\frac{z}{c}\right)^{2/e_1}-1\end{aligned} \quad (16)$$

Using the above method, we arrive at

$$K=\left[1+f_0(\overrightarrow{OA})\right]^{-e_1/2} \quad (17)$$

Therefore, we can also obtain f_0 from f_{col} for super-quadric functions using K.

After getting a correct position on the surface, the contact force is added to that position. The contact has the same amount with the normal component of external force vector at that point but opposite direction so that it make the net force zero at normal direction. By doing so, the vertex resides on the surface without bounding or intruding. Again, this means that we have found the exact position on the implicit surface.

5. RESULTS AND DISCUSSION

As discussed earlier, our approach has several advantages over previous blending methods. First, it does not have bulge problem. Second, it can be applied to more complex geometries without complicating blending surface itself. Finally, it provides the designer with more intuitive and interactive designing tools in run-time. In the following sections, we review those aspects in detail. Figure 10 shows various examples based on our method.

5.1 Bulge

As described, "bulge" is an unwanted artifact that appears in blending surface algorithms based on the algebraic distance. This happens when implicit surfaces meet each other, making it more difficult to anticipate the blended results. We can examine this problem for a simple example in 2D domain in detail. The range-controlled, super-elliptical blend method based on algebraic distance [17] is expressed as follows;

$$f(p) = B(P_1, P_2)$$
$$= 1 - \left[1 - \frac{P_1(p)}{r_1}\right]_+^t - \left[1 - \frac{P_2(p)}{r_2}\right]_+^t = 0$$

where P_1 and P_2 are implicit surface, r_1 and r_2 are the ranges of influence for P_1 and P_2, $[x]_+$ means MAX(0,x), and t is a parameter determine the blending surface's curvature.

If we set P_1 and P_2 as follows, we can find the bulge occur in 2D domain,

$$P_1 = \frac{x^2}{4} + y^2 - 1 = 0$$
$$P_2 = x^2 + y^2 - 1 = 0$$

The result is shown in Figure 5. In this figure, the circle is included in the ellipse. As we can imagine, the natural blending surface should be tangent at both the two primitives at the y-axis but the result show that bulge happened in that area. From the observation, it is concluded that we can reduce this bulge (the offset between (0,1) and the point where the red line meet y-axis in this case) by reducing r_1 and r_2. However, it is impossible to reduce this offset to zero by applying zero to r_1 or r_2. Besides, the situation becomes more complex when more implicit surfaces are involved because this parameter(say, r_i for i-th implicit surface) determines the curvature and shape of the blending

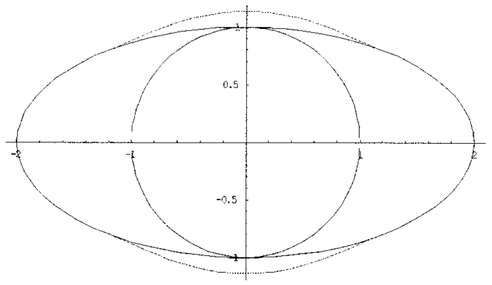


Fig. 5. Bulge (red line) in blending a circle and an ellipse in 2D(t = 2, r_1 = r_2 = 1)

surface. Thus, intuitive guessing of blending surfaces is almost impossible, even with any time-consuming and complex mathematical tools.

Unlike this, our approach has no bulge problem because it only depends on simulation of collisions between the vertices and the implicit surfaces. Figure 6 shows that there is no bulge at the area where two implicit surfaces meet each other. In the remaining sections of this chapter, several advantages of the proposed method are presented.

5.2 Level of Detail

In multi-resolution applications [27], several

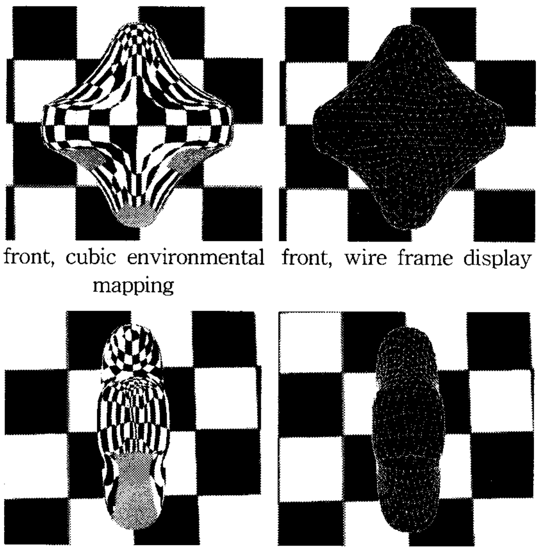

front, cubic environmental mapping
front, wire frame display
side, cubic environmental mapping
side, wire frame display

Fig. 6. Bulge free blending for two super-ellipsoids (e_1=0.5, e_2=1).

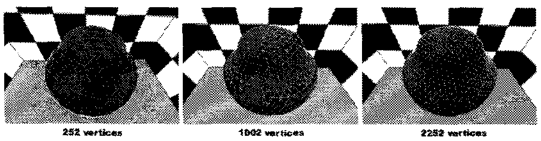


Fig. 7. Blending surfaces for two super ellipsoids ($e_1=1$, $e_2=0.5$) with three level-of-details

meshes that have different number of vertices but in similar appearances are generated according to the level of detail. It can enhance the efficiency in displaying polygons. In our approach the level of detail is determined by wrapping mesh for it incorporates polygonization process.

Figure 7 shows three cases for blending two super ellipsoids in different level of details. In these cases, we applied different wrapping meshes that have 252, 1002 and 2252 vertices respectively. In a nutshell, the LOD approach is quite straightforward in the proposed method providing richer tools for any design works.

5.3 Determining Curvature with Physical Parameters

To design the exact shapes for blending surfaces, at least two things are required; first, the designer could see the result of changes made immediately; second, the design tools or method should be specific enough to express the subtleties. From this observation, the ability to change curvature is essential. Since there are lots of parameters regarding in physical simulation, we have various choices to adjust the shape. In this research, we have focused on the spring constant to adjust curvature of blending surface.

Figure 8 shows some of the results we get by changing the spring constants. In these screen shots we can observe that the curvatures of the blending surfaces apparently depend on the spring constant, i.e. the curvature increases with low spring constant and decreases with high value. Indeed, we can get relatively free margin to change these physical parameters without suffering stability issues due to

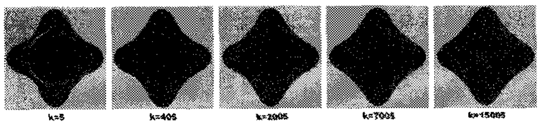


Fig. 8. Blending surfaces for 3 ellipsoids with various spring constants

the robustness of the implicit Euler integration method.

5.4 Distribution of Vertices

Because the proposed scheme depends on physics simulation, it is possible to adjust the density of vertices for specific area. For example, we can assume that some vertices are located sparsely at some area where the curvature is high after simulation and the designer wants to increase density of vertices at that area. To address this issue, we added following term to the original force equation (1),

$$\boldsymbol{F}_1 = -\left\{\begin{matrix} k_s(L-r)+k_d[(v_1-v_2)\cdot \boldsymbol{L}]/L \\ +k_a(1-n_1\cdot n_2) \end{matrix}\right\}\boldsymbol{L}/L$$

$$\boldsymbol{F}_2 = -\boldsymbol{F}_1$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are normal vectors at vertex 1 and vertex 2 Thus, the additional force term, $k_a(1-n_1 \cdot n_2)\boldsymbol{L}/L$ is in proportion to the angle between these normal vectors, if $k_a > 0$. Therefore, we can make vertices gather at area where curvature is high by controlling vertices' density with variable, $k_a(k_a > 0)$.

Figure 9 shows the results when various values are applied to k_a.

6. SUMMARY AND FUTURE WORK

Our approach provides several advantages over

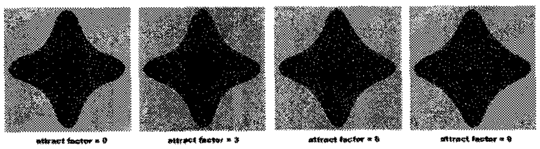


Fig. 9. Distribution of vertices for various attract factors(k_a)

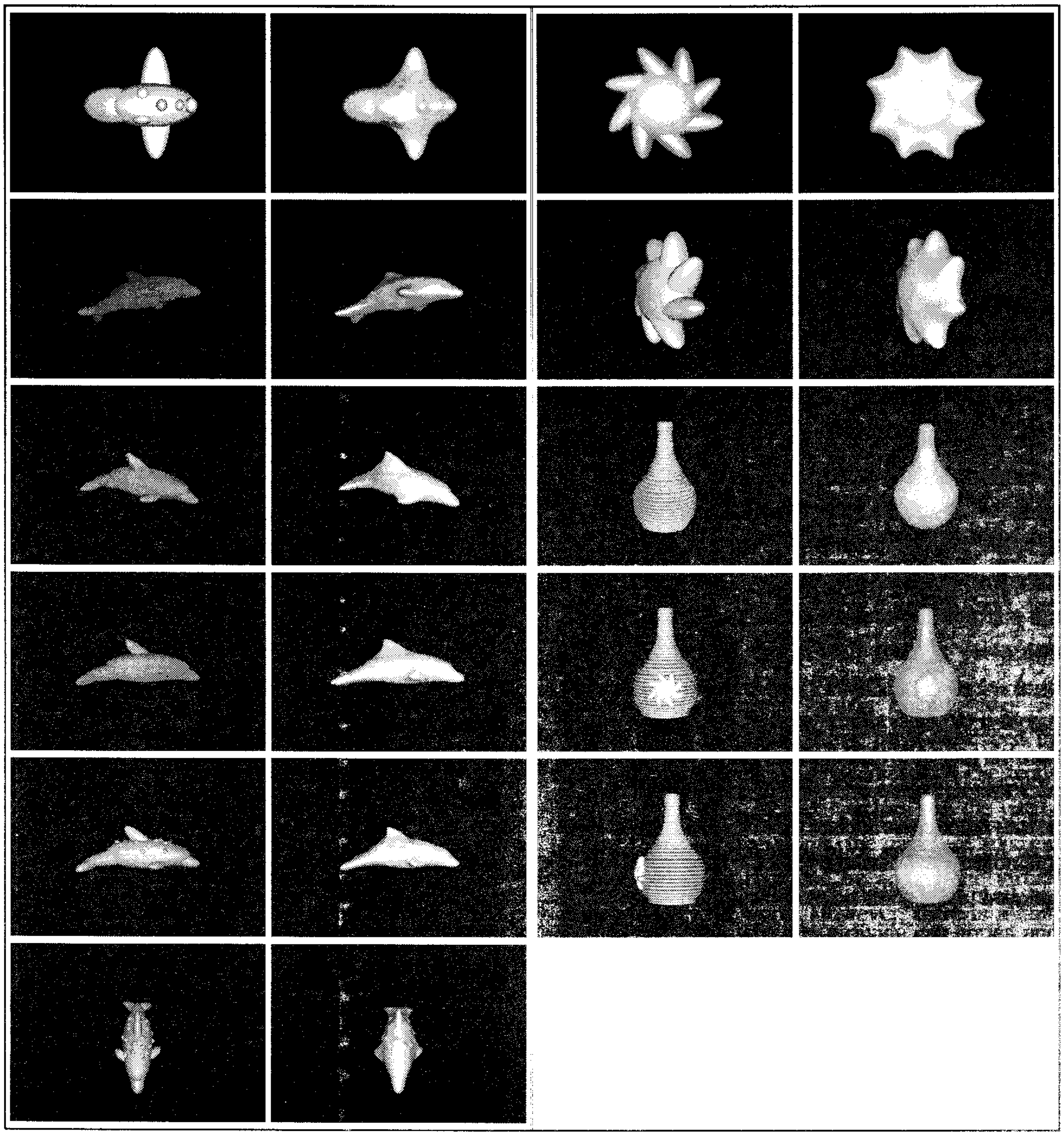


Fig. 10. Some samples of blended geometry, left: ellipsoidal primitives without blending surfaces, right: blended surfaces with the proposed method.

the previous ones; it makes no bulges in any geometries, its design procedure, based-on a real-time process, is more interactive and intuitive, and it enables more delicate design by adjusting individual physics parameters including the spring constant, mass, or pretension. One of the issues to be considered is that the blending surface in our study is not in the implicit surface but a polygonal surface. Note that the blending surface itself is implicit surface in the previous blending methods. This property implies several meanings; first, our blending surface does not benefit from the space-dividing characteristic of implicit surface in the ray-tracing routine. However, it is not a critical problem because there are already lots of robust ray-tracing algorithms for polygonal meshes. Second, when some implicit primitives are to add in real-time, they should be inserted inside of the wrapping mesh, not from outside of it. Third, our blending surface can be directly modified with general mesh editing techniques.

One of limitations in our approach is that it can be only applied to 0-genus manifold geometries until now. To apply other kind of geometries like torus, proper polygonization methods should be also considered because our approach is based on polygonal structure. Since our method has 1-to-1 correspondence between physical elements and polygonal elements, it should be straightforward. Also, other design techniques based on physical simulation in our method can be developed and applied because there are several other physical parameters that designers can adjust. For example, by locally setting different spring constant values within stable ranges to each edges of mesh structure, more sophisticated blending surfaces could be created. Besides, other kind of physics models could be applied to express various material properties in blending surface.

ACKNOWLEGEMENT

This work was suppported by the Second Brain Korea 21 Project. This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government(R01-2008-000-20855-0)

REFERENCE

[1] James F. Blinn, "A Generalization of Algebraic Surface Drawing," *ACM Transactions on Graphics*, Vol.1, No.3, pp. 235-356, 1982.

[2] William J. Schroeder, William E. Lorensen, and Steve Linthicum, "Implicit Modeling of Swept Surfaces and Volumes," *Proceedings of the conference on Visualization '94*, pp. 40-45, 1994.

[3] Antoine Leclercq, S. Akkouche, and E. Galin, "Mixing Triangle Meshes and Implicit Surfaces in Character Animation," *Proceedings of the Eurographic Workshop on Computer Animation and Simulation*, pp. 37-47, 2001.

[4] W.H. Press, S. A. Teukolsky, W.T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing, 2nd edition*, Cambridge University Press, Cambridge, pp. 120-122, 1993.

[5] Masatoshi Matsumiya, Haruo Takemura, and Naokazu Yokoya, "An Immersive Modeling System for 3D Free-form Design Using Implicit Surfaces," *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 67-74, 2000.

[6] Jing Hua and Hong Qin, "Dynamic Implicit Solids with Constraints for Haptic Sculpting," *Proceedings of the Shape Modeling International 2002*, pp. 119-129, 2002.

[7] Geoff Wyvill, *Introduction to Implicit Surfaces*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1997.

[8] William E. Lorensen and Harvey E. Cline, "International Conference on Computer Graphics and Interactive Techniques," *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 163-169, 1987.

[9] Barton T. Stander and John C. Hart, "Guaranteeing the Topology of an Implicit Surface Polygonization for Interactive Modeling," *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 279-286, 1997.

[10] John C. Hart, "Morse Theory for Implicit Surface Modeling," *Proceedings of Visualization and Mathematics*, pp. 257-268, 1997.

[11] John M. Snyder, "Interval Analysis for Computer Graphics," *Proceedings of the 19th*

annual conference on Computer graphics and interactive techniques, pp. 121-130, 1992.

[12] Yutaka Ohtake and Alexander G. Belyaev, "Dual/Primal Mesh Optimization for Polygonized Implicit Surfaces," *Proceedings of the seventh ACM Symposium on Solid Modeling and Applications*, pp. 171-178, 2002.

[13] M. Desbrun, N. Tsingos, and M.-P. Gascuel, "Adaptive Sampling of Implicit Surfaces for Interactive Modeling and Animation," *Proceedings of Implicit Surfaces '95*, Vol.15, No.5, pp. 319-325, 1995.

[14] B. Crespin, P. Guitton, and C. Schlick. "Efficient and Accurate Tessellation of Implicit Sweep Objects," *Proceedings of Constructive Solid Geometry '98*, 1998.

[15] Andrew P. Witkin and Paul S. Heckbert, "Using Particles to Sample and Control Implicit Surfaces," *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 269-277, 1994.

[16] John Woodwark, "Blends in Geometric Modeling," *Proceedings on Mathematics of Surfaces II*, pp. 255-297, 1987.

[17] Alyn Rockwood, *Introduction to Implicit Surfaces*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1997.

[18] Christoph Hoffmann and John Hopcroft, *The Potential Method for Blending Surfaces and Corners, Geometric Modeling*, SIAM Publications, Philadelphia, 1987.

[19] Menno Kosters, "An Extension of the Potential Method to Higher-Order Blendings," *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pp. 329-337, 1991.

[20] Jules Bloomenthal and Brian Wyvill, "Interactive Techniques for Implicit Modeling," *Symposium on Interactive 3D Graphics*, Vol.24, No.2, pp. 109-116, 1990.

[21] Anil Kaul and Jarek Rossignac, "Solid-interpolation deformations: Construction and animation of pips," *Proceedings of EUROGRAPHICS '91 (Sept.)*, pp. 493-505, 1991.

[22] Andrei Sherstyuk, "Interactive Shape Design with Convolution Surfaces," *Proceedings of the International Conference on Shape Modeling and Applications*, pp. 56-65, 1999.

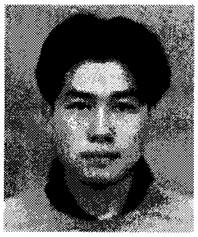
[23] Laurent Hilde, Philippe Meseure, and Christophe Chailou, "A Fast Implicit Integration Method for Solving Dynamic Equations of Movement," *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 71-76, 2001.

[24] David M. Bourg, *Physics for Game Developers, 1st ed.*, O'Reilly & Associates, Inc., Sebastopol, CA, 2002.

[25] Kwang-jin Choi and Hyeong-seok Ko, "Stable but Responsive Cloth," *ACM Transactions on Graphics, SIGGRAPH 2002*, Vol.21, No.3, pp. 604-611, 2002.

[26] D. Baraff and A. Witkin, "Large Steps in Cloth Simulation," *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 43-54, 1998.

[27] H. Hoppe, "Progressive Meshes," *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 99-108, 1996.



Tae-Jung Park

2006~Present Research Professor, BK21 Software, Korea University

2006 Ph.D. in Electrical & Computer Engineering

1999 MS in Electrical ngineering, Seoul National University

1997 BS in Electrical Engineering, Seoul National University

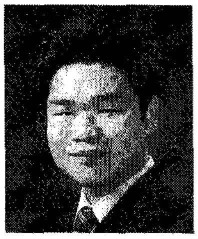
Research Area : Compression for geometry information, TCAD, 3D modeling

Hyeong Ryeol Kam

2008.~Present Master Student, Dept. of Computer Science, Korea University

2008. BS in Computer Science, Korea University

Research Area : Computer graphics, physics-based simulation



Seung-Ho Shin

2007.~Present Ph.D. student, Dept. of Computer Science, Korea University

2007. MS in Computer Science, Korea University

2005. BS in Computer Science, Korea University

Research Area : Computer graphics, physics-based simulation

Chang-Hun Kim

2008.~Present Committee Chair, Korea Computer Graphics Society

2005.~2007 Dean of Korea University's College of Computer Information Communication

1995.~Present Professor, Dept. of Computer Science, Korea University

1993. Ph.D. in Dept. of Computer Science, University of Tsukuba, Japan

1979. BS in Dept. of Economics, Korea University

Research Area : Computer graphics, physics-based simulation, mesh processing