Volume 16, (1997) number 5 pp. 295–306

Interactive Visualization of Implicit Surfaces with Singularities

Angela Rösch[†], Matthias Ruhl and Dietmar Saupe

Institut für Informatik, Universität Freiburg, Germany

Abstract
This paper presents work on two methods for interactive visualization of implicit surfaces: physically-based sampling using particle systems and polygonization followed by physically-based mesh improvement which explicitly makes use of the surface-defining equation. While most previous work applied to bounded manifolds without singularities and without boundary (topological spheres) we broaden the scope of the methods to include surfaces with such features, in particular cusp points and surface self-intersections. These aspects are not (yet) essential for computer graphics modelling with implicit surfaces but they naturally occur in simulations of interest in mathematical visualization. In this paper we use the Kummer family of algebraic surfaces as an example.

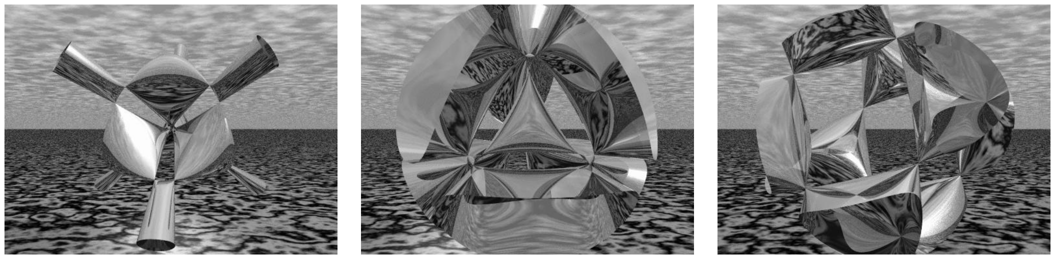
1. Introduction

Our work is motivated by efforts in the mathematical community for the visualization of implicit algebraic surfaces. Algebraic surfaces and their deformations have been studied for more than one hundred years. Visualization of these surfaces has always been regarded important and traditionally plaster models were used for this purpose. Recently, of course, computer graphical studies have been carried out. Physically-based sampling methods (1), polygonization (2), and raytracing (3) are three principally different approaches to visualize implicit surfaces. They are ordered here with respect to increasing computational complexity as well as increasing image quality. Hanrahan[1] investigates the raytracing approach to the rendering of algebraic surfaces. The central computational task is twofold. First, one has to efficiently convert the equations of the surface and a given ray into a single equation, i.e., a polynomial in one variable. Second, a numerical procedure must be employed to compute the smallest positive root of the polynomial. Hanrahan chose a method developed by Collins and Loos, which is based on Descartes rule of signs. The approach was also tested with other root finding methods[2].

An important example of a deformation of algebraic surfaces was proposed by Kummer in the last century. It is given by a parameterized family of fourth-order polynomials in affine coordinates for x, y and z (see section 3.4 for the defining equation). In this deformation a double sphere is transformed into Steiner's roman surface and then into a tetrahedron. Although individual surfaces from this family had been visualized before, only just recently a sequence of (raycast) images illustrating the entire deformation was given by Barth and Endraß[3]. In our work[4] we presented corresponding computer animations. We demonstrated a feasible approach to rendering animations of algebraic surfaces suitable not only for computer graphics specialists but also for students and researchers in mathematics without such background. This animation was based on the original method of Hanrahan and supported by the public domain raytracer *rayshade* of Kolb[5] which requires supplying a program module for the ray-surface intersection calculations corresponding to algebraic surfaces. See Figure 1 for three frames of the animation.

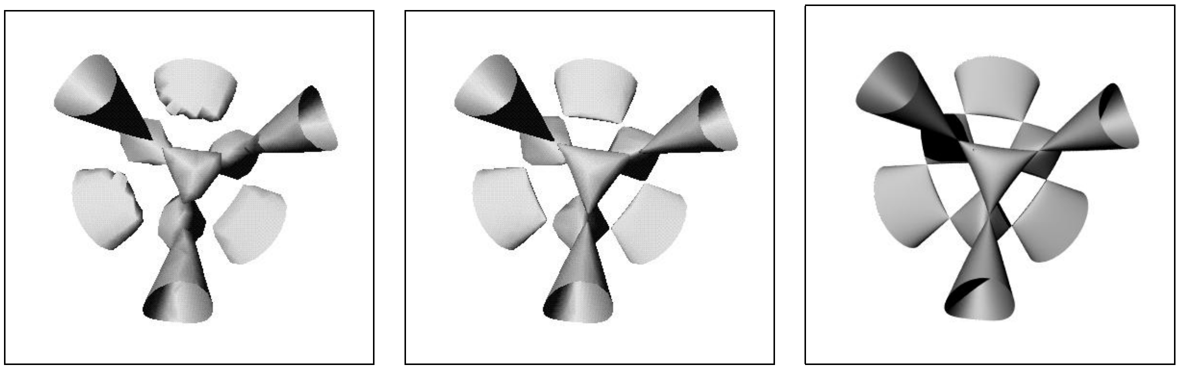
This approach — although suitable for high quality renderings — lacks interactive control which is necessary to arrive at properly chosen camera paths, parameter dynamics et cetera for an animation of a priori unknown surfaces. There are two alternative approaches to rendering algebraic surfaces that may provide the necessary speed for interactive control and animation. The first one is particle based and uses the sampling method of Witkin and Heckbert presented in their SIGGRAPH '94 paper[6] for algebraic surfaces. A simple constraint equation locks a set of parti-

[†] Now at Frauenhofer Institute for Industrial Engineering (IAO), Stuttgart, Germany



(a) $\nu^2 = 1.02$ (b) $\nu^2 = 2.44$ (c) $\nu^2 = 1.28 \cdot 10^6$

Figure 1: *Raytraced Kummer surfaces*



(a) polygonizer output (b) improver output (c) rayshade output

Figure 2: *Visualization of the Kummer surface with* $\nu^2 = 1.2$

cles on a surface while the particles and the surface are allowed to move. Local repulsion is used to make the points, called *floaters*, spread evenly across the surface. By adaptively varying the radius of repulsion while fissioning and killing particles based on the local density, good sampling distributions can be achieved very rapidly. Witkin and Heckbert's work focussed more on *modelling* of implicit surfaces while we are more interested in a priori defined parameter-dependent surfaces. Moreover, our surfaces have singularities and they are unbounded (or clipped to a finite volume resulting in a boundary curve) in contrast to theirs which are single connected components without singularities or boundary. To cope with these difficulties we modify the physical model underlying the mechanism of floaters by introducing boundary effects, and we adapt the radii of repulsion to local surface properties (curvature and closeness to singularities). With these modifications we can achieve near real-time sampling and rendering of deforming implicit algebraic surfaces.

The second approach considered here polygonizes the implicit surfaces. A collection of approximating triangles is generated which are well suited for rendering by means of common graphics hardware. There exists a large body of literature on this topic[7]. In our implementation[8] a marching-[9, 10] or chain-of-cubes[11] algorithm is used to quickly derive a first polygonization (see figure 2(a)).

To iron out local insufficiencies, especially concerning curvature, singularities and artifacts, the vertices can automatically be moved around by an improver module. The underlying algorithm interprets the polygonization as a surface-constrained spring-mass system. This yields much better polygonal models in a few time steps (see figure 2(b)). A further application of the tool[8] allows to select a suitable view, lighting, material, clipping etc. of an implicit surface under interactive control and subsequently produces a corresponding parameter input file for the *rayshade* program (see figure 2(c)), which operates using the implicit surface representation instead of the polygonization.

Our methods are exemplified using simple algebraic surfaces and the Kummer family of surfaces. However, the results carry over to implicit surfaces in general provided the underlying surface-defining functions are sufficiently differentiable.

2. Previous Related Work

There are several approaches for a mathematically faithful visualization of implicit surfaces that may include singularities. Sederberg and Zundel[12] presented a robust scan line algorithm that correctly displays singularities of arbitrary complexity. This is achieved by applying algebraic tools such as resultants to local surface representations in the Bernstein basis. Hall and Warren[13] use very much related techniques for an adaptive polygonization of implicit algebraic surfaces. The surface is enclosed in a collection of tetrahedra, which are recursively subdivided. The Bernstein/Bezier basis is used to eliminate tetrahedra and to terminate subdivision. This way convergence is guaranteed. In a post processing vertices very close to the surface are moved onto the surface in order to eliminate small or thin triangles of the resulting polygonal surface representation. Bloomenthal and Ferguson[14] consider the general problem of polygonizing implicit surfaces with singularities, thereby generalizing traditional polygonization methods. The model of implicit surfaces in their work differs from the usual one in that a 'multiple regionalization' is employed. All of the methods discussed have to pay the price of increased processing time to guarantee the correctness of the result. Thus, their use for interactive applications is limited.

Stander and Hart[15] use Morse theory to guarantee topological correctness of polygonized implicit surfaces from one-parameter families of surfaces. This is achieved by monitoring all critical points of the underlying function as the parameter changes. The method will work in cases where these critical points do not infer singularities of the surfaces. Thus, it does not apply to the case studied in this paper.

Mesh generation for implicit surfaces in connection with particle-based dynamics has been studied by some authors. One can distinguish two fundamentally different approaches[16]:

1. The particles are generated, simulated according to a more or less physical model, and then a polygonization is invoked using the final particle positions as polygon vertices.
2. An implicit surface polygonizer is applied first, followed by a relaxation of the vertices.

An approach of the first kind was first described by Figueiredo et al[16]. The implicit surface is sampled using particles which are released at arbitrary positions in space and then follow the associated gradient flow until equilibrium is reached on the surface. After that particles are subjected to a relaxation process similar to that formulated by Turk[17], or Szeliski and Tonneson[18]. Finally, a constrained Delauney triangulation is constructed. Examples are given only for implicit curves and surfaces without any singularities. It remains unclear how to construct the triangulation in the presence of singularities such as self-intersecting surfaces.

The method of Shimada and Gossard[19] ("bubble meshing") may also be placed in the first category. In this approach the surface is approximated by a number of balls (packed spheres) which then interact according to a dynamical model allowing a mechanism for birth and death of bubbles. In a second step the centers of the balls are connected using a constrained Delauney triangulation, from which a polygonal model of the surface can be extracted. This approach is elegant, however, it is not worked out for implicit surfaces but for parameterized objects (lines, surfaces, volumes). In particular it remains an open question how the presence of singularities in a surface might affect the results. Again it is an open problem to define and compute a constrained Delauney triangulation for an underlying surface with singularities.

There exists a large body of literature on the topic of mesh generation and optimization, an important step in finite element computations and for simplifying and smoothing complex real-world polygonal models obtained from physical sampling processes. However, we are aware of only one contribution that specifically addresses the issue of implicit surfaces in the spirit of the second category above, namely the paper of Figueiredo et al[16]. In that work a triangulation of a spatial region enclosing the surface is used (as in Hall and Warren's work[13]) rather than a polygonal model of the surface itself. A spring-mass system is attached to the vertices and edges. Care must be taken to prevent the vertices from collapsing onto the surface while ensuring that they do not escape from it either. The method is not adaptive with respect to curvature and singularities.

Ning and Bloomenthal[10] review and compare implicit surface polygonizers that use cubical base cells. Resulting triangle meshes may contain many triangles, a lot of which typically are very small or thin. Thus, the triangulation can be simplified. The method of Schroeder, Zarge, and Lorensen[20] removes vertices for this purpose. No adaptation by moving vertices around is attempted. A similar approach is presented by Kalvin et al[21]. Hoppe et al[22] also reduce the number of vertices in addition to modifying vertex positions according to minimization of a suitable energy functional. Laplacian smoothing is another method to smooth a triangulation where vertices are replaced by the center of mass of the neighboring vertices[23]. None of the works mentioned in this paragraph make reference to an underlying mathematical description of the surface because they assume that an initial polygonal model is all that is given. We believe that in the presence of such a precise mathematical surface definition it would be a mistake not to make use of it, especially when considering the neighborhood of singularities, where a 'plain' mesh simplification may tend to worsen the result like widening the gap near a cusp point.

A step in this direction has been made by Schmidt[24] who suggests to increase the resolution of the cubical grid in the vicinity of possible singularities of the surface. From this adaptive grid a static polygonal mesh is generated which approximates the implicit surface. Several heuristic criteria are used to determine closeness to a singularity, each one of them testing the implicit function at an intermediate point.

In our work a different approach is taken, namely that of a physically-based sampling or mesh improvement.

3. Sampling By Floaters

In section 3.1 we briefly summarize the physically-based approach to sampling an implicitly defined surface by floaters following the presentation of Witkin and Heckbert[6]. In sections 3.2 to 3.4 we discuss the appropriate extensions of these methods for the application to surfaces with boundary curves and singularities.

Particle-based approaches for static surfaces have previously been studied by Turk[17, 25] and by de Figueiredo et al[16] for example. Oriented particle systems were also used for dynamic modelling rather that visualization[18, 26].

3.1. Simulation

We consider parameter-dependent implicit surfaces in three-dimensional Euclidean space, where an animation of the surface occurs by variation of surface parameters, changing the shape of the surface with time. The surface is given by the zeroset of a differentiable function: $F(x,q(t)) = 0$ where $x \in \mathbf{R}^3$, and $q(t) \in \mathbf{R}^m$ denotes a set of m parameters smoothly changing with time t. We assume a set of n particles for the purpose of sampling the surface. Particle i has trajectory $p^i(t)$ (superscripts denote particle indices). Particles moving about on the surface must satisfy the equations

$$F^i(t) = F(p^i(t), q(t)) \equiv 0$$

and, therefore

$$\dot{F}^i = F^i_x \cdot \dot{p}^i + F^i_q \cdot \dot{q} = 0.$$

The dots denote time derivatives and subscripts specify gradients of F with respect to x or q. In order to move particles to the surface and to keep them on the surface in spite of numerical integration errors a feedback term $\dot{F}^i = -\phi F^i$ with a constant $\phi > 0$ is used leading to the basic constraint equation

$$C^i(p^i, \dot{p}^i, q^i, \dot{q}^i) = F^i_x \cdot \dot{p}^i + F^i_q \cdot \dot{q} + \phi F^i = 0. \qquad (1)$$

Here $\dot{p}^i$ is the unknown. Thus, for each particle this yields one scalar equation leaving two degrees of freedom. The actual velocities are determined via an optimization. For each particle i a desired velocity P_i will be defined below and the optimization requires to minimize the objective function

$$G(\dot{p}^1, \ldots, \dot{p}^n) = \frac{1}{2} \sum_{i=1}^{n} ||\dot{p}^i - P_i||^2$$

For example, setting all $P_i = 0$ will result in minimizing particle speeds. Using the classic method of Lagrange multipliers the equation

$$G_{\dot{p}^i} + \sum_{j=1}^{n} \lambda^i C^j_{\dot{p}^i} = 0 \qquad (2)$$

leads to the solution of the constrained optimization problem yielding the velocity equations

$$\dot{p}^i = P^i - \frac{F^i_x \cdot P^i + F^i_q \cdot \dot{q} + \phi F^i}{F^i_x \cdot F^i_x} F^i_x.$$

At this point the definition of the desired particle velocities remains to be given. It serves the purpose of quickly producing an acceptable sampling density over the surface and is based on repulsion of particles and the finiteness of the surface. The repulsion forces are based on the concept of the 'energy' of particles and are taken proportional to energy gradients. Witkin and Heckbert define an adaptive scheme by setting the energy of particle i due to particle j as

$$E^{ij} = \alpha \exp\left(-\frac{||r^{ij}||^2}{2(\sigma^i)^2}\right) \qquad (3)$$

where $r^{ij} = p^i - p^j$. The parameters $\sigma^i > 0$ can be regarded as some kind of repulsion radius of the particle i. The total energy at particle i is the sum

$$E^i = \sum_{j=1}^{n} (E^{ij} + E^{ji}). \qquad (4)$$

Then the desired velocities come out to be

$$P^i = -(\sigma^i)^2 E^i_{p^i} = (\sigma^i)^2 \sum_{j=1}^{n} \left(\frac{r^{ij}}{(\sigma^i)^2} E^{ij} + \frac{r^{ij}}{(\sigma^j)^2} E^{ji}\right). \qquad (5)$$

In order to quickly move particles into sparse regions the repulsion radii can be controlled adaptively. This is achieved by driving all of the energies to a global desired energy level. In effect, this yields another set of simple differential equations for the radii σ^i. To achieve a uniform sampling density particles with large repulsion radii are required to fission and likewise, particles with small repulsion radii can be eliminated. Witkin and Heckbert suggest to fission a particle, if it is near equilibrium (small speed relative to its radius), and either its radius is large ($\sigma^i > \sigma_{max}$) or it carries high energy and its radius is above a given nominal radius $\hat{\sigma}$. The radius of the two new particles are set to $\sigma^i/\sqrt{2}$ and their desired velocities are taken as random directions scaled appropriately. A particle is a candidate for elimination, if it is near equilibrium, its repulsion radius is small, and if a randomized test succeeds (in order to prevent 'mass suicide' in overcrowded regions).

We remark that our exposition of the method necessarily is very brief; implementation details and motivations can be found in the original article[6]. Moreover, the method also allows the determination of the parameters of the surface via control points (useful for modelling with implicit surfaces), the theory of which is based on the same differential equations approach as that of the floaters described above.

3.2. Handling of Unbounded Surfaces

In order to allow for unbounded surfaces, where the original sampling method would not work, we have introduced half spaces and spheres as clipping volumes. Instead of extending the physical model to incorporate forces that would keep the particles constrained within the clipping volumes we found that a simpler approach yields an even better result for our application. We suggest to monitor the particle trajectories and whenever a particle is moved outside the clipping volume we simply project it back onto the boundary of the clipping volume. In the case of a half space this is an orthogonal projection onto the hyperplane defining the volume, and in the case of the sphere the projection occurs in the direction towards the sphere center.

3.3. Considering Singularities

As originally demonstrated[6] the method works fine for compact smooth surfaces without singularities. Selfintersections and cusp points of the surface, however, will produce the artifact of nonuniform sampling since particles distributed near such singularities locally repel each other stronger than particles in a regular surface patch with the same sampling density. To overcome this problem we propose several heuristics:

- To avoid the interference of floaters positioned on different parts of the surface near a selfintersection, we modify the repulsion energy (equation 3) to be
$$E^{ij} = \alpha \exp\left(-\frac{||r^{ij}||^2}{2(\sigma^i)^2}\right) |\cos\left(F_x^i, F_x^j\right)|.$$
(Here $\cos(x,y) = \frac{x \cdot y}{||x|| \cdot ||y||}$.) In the extreme case of an orthogonal intersection this allows the floaters from the two surface sheets to pass through each other unhindered (compare figures 4(c) and 4(d)).
- We have tried to adapt the desired particle radius by using a separate radius $\sigma^i_{max} = \mu^i \sigma_{max}$ for each particle instead of a *global* fissioning radius σ_{max}. The factor μ^i is chosen depending on local surface curvature and proximity to a singularity. We have investigated two possibilities for this factor:
  - $\mu^i = \alpha^i := \min\left\{|\cos\left(F_x^i, F_x(v_k^i)\right)|\right\}$, where the v_k^i are several points chosen randomly on the perimeter of the floater. α^i gives an indication of the maximal curvature at the floater position. The idea behind this is that at points of high curvature a surface can be sampled better using smaller floaters.
  - To sample the surface more densely near singularities (arguably the most interesting points of a surface), we choose a factor that estimates the distance to the nearest singularity using Newton's method. Singularities are given by points where the gradient F_x vanishes. To compute such a point with Newton's method, starting out from the particle position p^i, the second derivative $F_{xx}(p^i)$ (a 3 by 3 matrix) is required. The Newton corrector r is the solution of $F_{xx}(p^i) \cdot r = F_x(p^i)$ and its norm $||r||$ serves at an estimate of the distance between p^i and the nearest singularity. Now we set $\mu^i = \beta^i := \min(\frac{||x||}{\gamma}, 1)$. The minimization is used, since the scaling factor μ^i should not become greater than 1. γ is a scaling factor, which was set to 1 for our experiments, but needs to be changed if the scale of the displayed object is varied greatly.

  Both values α^i and β^i can be used as μ^i in $\sigma^i_{max} = \mu^i \sigma_{max}$. In our experiments we found that both improved the sampling of most surfaces but not for all. To alleviate the individual short-comings of α^i and β^i, we used a linear combination of the two in our implementation, namely we set $\mu^i = \frac{1}{3}\alpha^i + \frac{2}{3}\beta^i$.

Using these heuristics, 'ad hoc' as they may be, leads to acceptable results for all surfaces that were tested, as shown below.

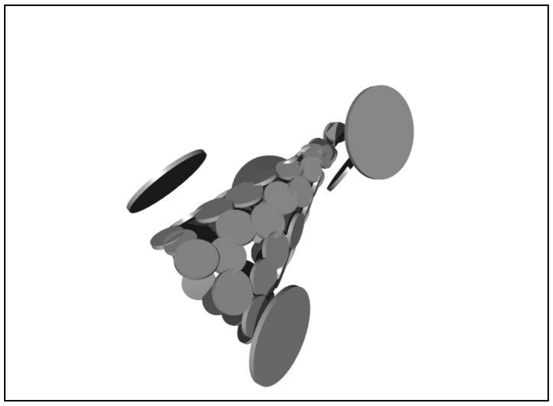
3.4. Results

We have tested our methods using two very simple implicit surfaces with singularities and more complex algebraic surfaces of degree four, the Kummer family:

1. A cone with a single cusp point clipped at two planes, one of which contains the cusp. No parameter animation was employed ($q(t) \equiv constant$) (see figure 3).
2. The union of two intersecting planes, clipped at a sphere. The planes intersect orthogonally and the intersection line moves back and forth by changing the parameters $q(t)$ appropriately (see figure 4).
3. The parameterized family of Kummer surfaces as used in the raytraced animation. The equation is
$$(3-\nu^2)(x^2+y^2+z^2-\nu^2)^2 - (3\nu^2-1)pqrs = 0$$
where $\nu \in \mathbf{R}$ is the parameter and
$$\begin{aligned} p &= 1-z-x\sqrt{2}, \\ q &= 1-z+x\sqrt{2}, \\ r &= 1+z+y\sqrt{2}, \\ s &= 1+z-y\sqrt{2}. \end{aligned}$$
The surface is unbounded for $\nu^2 \geq 1$ and clipped at a sphere whose radius is chosen so that the displayed surface contains all of the singular points (see figure 5).

On an SGI Indy, with a R4600SC processor at 133 MHz, the frame rate is between 5 Hz for the Kummer surface (using 500 floaters) and 20 Hz for the cone (using about 80 floaters).

Figures 3 to 5, generated by running a raytracing program on data produced by our particle simulator, display some


(a) The cone in its initial rendering phase. Several new particles are inserted into the scene.

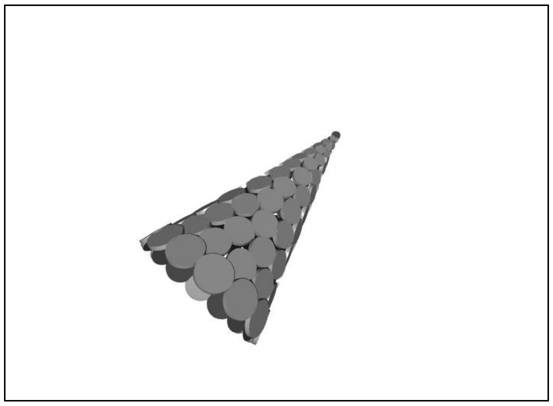
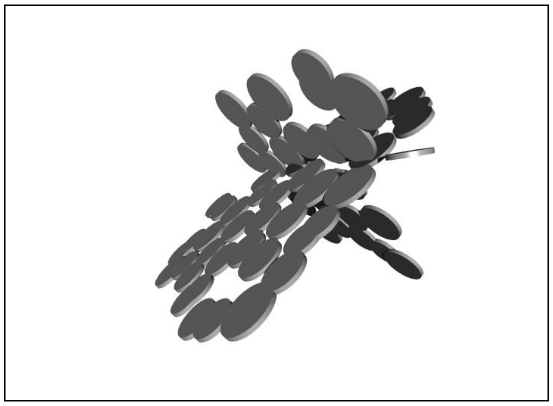
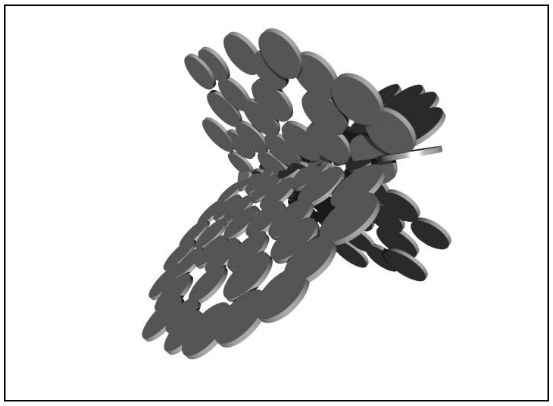
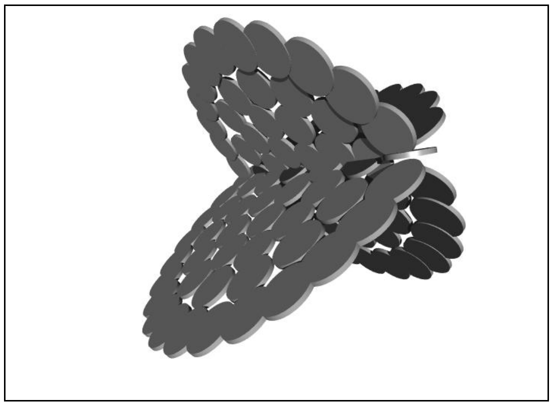

(b) The maximal number of particles are on the cone surface. An equilibrium point has been reached such that the particle density is higher near the singularity as desired.

Figure 3: *Floaters approximating a cone*


(a) Two intersecting planes, a few particles.


(b) The planes with more particles.


(c) Particles on planes at equilibrium. Note that the floaters intersect each other at the singularity.

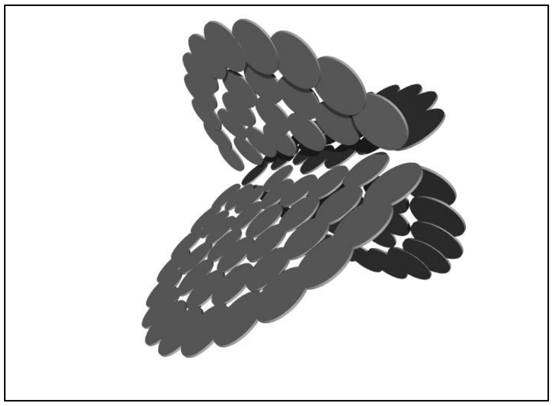
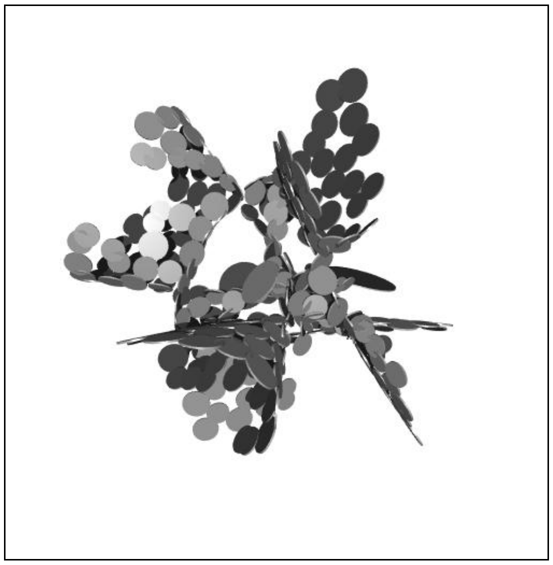
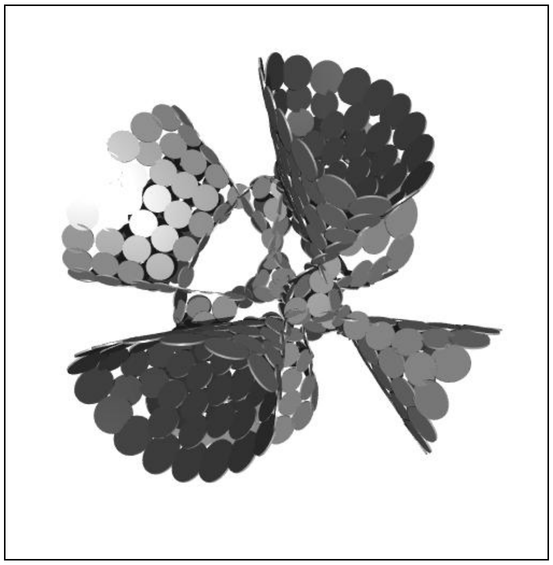
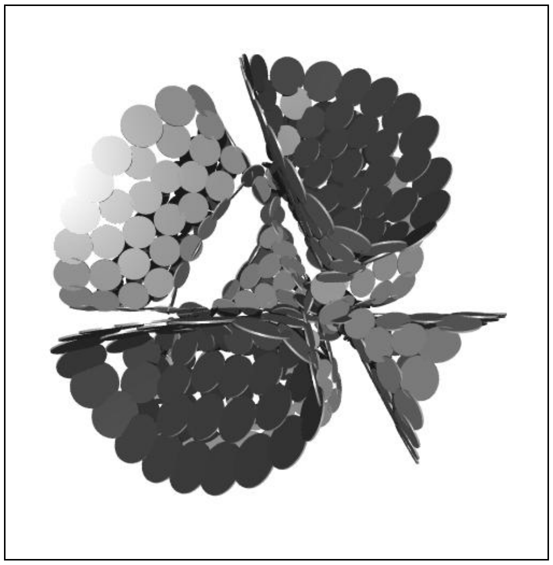

(d) The same planes approximated using the original method[6]. Here the particles repel each other at the singularity, leading to a sub-optimal sampling.

Figure 4: *Floaters approximating two intersecting planes*


(a) Kummer surface at $\nu^2 = 1.5$ with a few particles not yet at equilibrium.


(b) Kummer surface at $\nu^2 = 1.5$ with more particles and at equilibrium.


(c) Kummer surface at $\nu^2 = 2.0$.

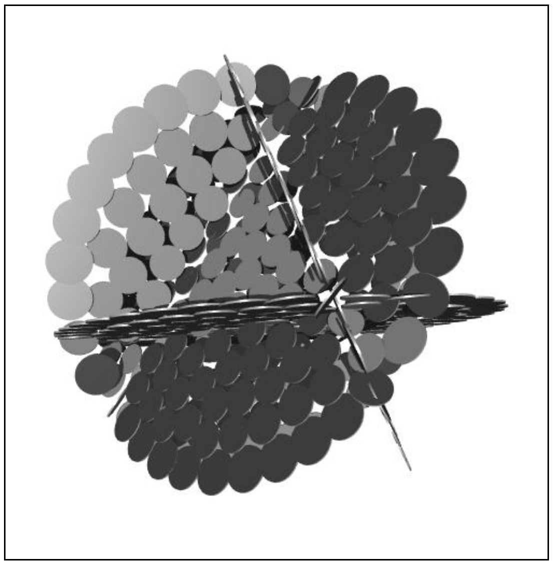

(d) Kummer surface at $\nu^2 = 3.0$, consisting of four intersecting planes.

Figure 5: *Floaters approximating Kummer surfaces*

of the results. The renderings show the floaters as shaded disks with normal vectors obtained from the gradients of the surface's defining function F. The radii of the disks are taken proportional to the particles' repulsion radii. Particles viewed from the 'back side' are shown in a darker shade of gray. The program starts by randomly inserting particles into the scene (up to some maximum number) which quickly settle on the surface, then spread over the surface and start fissioning yielding an overall acceptable density.

4. Improving Polygonal Models

This section describes another physically-based method for visualizing implicit surfaces. Whereas the floater technique samples the surface this approach improves a polygonal approximation, which has been generated by a polygonizer, for example a marching cubes algorithm. We combine ideas from two related methods:

- Figueiredo et al[16] use a spring-mass system to adapt a triangulation of a region of space enclosing the surface. This improves the result of the subsequent polygonization. We suggest to use the output of an implicit surface polygonizer as the basis of a spring-mass system. The edges of the triangulated surface model are interpreted as springs with rest length zero and its vertices correspond to particles without mass.
- The modification of the triangulation is simulated using the approach of Witkin and Heckbert[6] (see section 3.1) by a surface constrained optimization and Euler's method. Here, spring forces instead of local repulsion forces are used to derive the desired velocities of the vertices.

Compared to the original approaches this combination has some advantages:

- Initializing the spring-mass system with the polygonizer output yields a reasonable representation of the surface right from the beginning.
- Instead of external forces[16] we use implicit optimizing constraints to make the vertices stick to the surface. This is numerically more stable.
- In a spring-mass system the neighborhood of vertices is fixed and small, which accelerates the computation of the dynamics.
- The polygonal representation allows fast displaying of the surface model by traditional rendering systems.

4.1. Simulation

The improver module works on triangulated models of implicitly defined surfaces. A first triangulation is generated in two steps. First, a polygonizer creates the geometry, i.e. the vertex positions and polygons representing the surface. Then the topology, i.e. the edges and triangles between the vertices, needs to be computed. (In the implementation[8] these two steps are actually done in one pass.)

In order to improve the triangulation the modification of the spring-mass model is computed and rendered stepwise. The calculations of the simulation steps are derived from the Witkin and Heckbert's technique[6] for sampling constant surfaces with floaters (see section 3). In this work repulsion forces are substituted by spring forces as the cause for vertex motion. The energy between two neighboring vertices (see equation 3) is set to $E^{ij} = \kappa^{ij}||r^{ij}||^2$ and consequently the total energy at a single vertex (see equation 4) is given by

$$E^i = \sum_{j \in neighbors(i)} E^{ij}$$

This yields the desired velocities (see equation 5) as

$$P^i = -2 \cdot \sum_{j \in neighbors(i)} \kappa^{ij} \cdot r^{ij}$$

where the spring constants κ^{ij} are defined adaptively (see below). During the simulation no vertices are deleted or inserted. In the implementation[8] the improver process terminates when all vertices are moving with a speed less than a given threshold.

4.2. Handling Of Boundaries

Some polygonizers can be applied to unbounded implicit surfaces which are clipped to a finite volume. This produces boundary edges and vertices. If these resulting boundary vertices receive no special treatment during the subsequent dynamics, the whole object will be contracted because these vertices are only affected by springs pulling from one side. For this reason all vertices lying on the object boundary must be marked and can be treated in the following ways:

- **Fixation:** exclude all marked vertices from the simulation process in order to keep them fixed.
- **Additional position constraints:** restrict the marked vertices to the *curves* which result from clipping the surface F with a bounding volume. Let us assume that the bounding volume is given implicitly as $\{x|\tilde{F}(x) \geq 0\}$ where $\tilde{F}$ is a differentiable function. Then the boundary curves are given by the intersection of the zero sets of F and $\tilde{F}$. To constrain the marked boundary vertices p^i to these curves we may impose an additional constraint equation for these particles:

  $$\tilde{C}^i(p^i, \dot{p}^i) = \tilde{F}^i_x \cdot \dot{p}^i + \phi \tilde{F}^i \equiv 0.$$

  In the corresponding Lagrange equation (see equation 2) which solves the extended optimization problem we thus obtain

  $$G_{\dot{p}^i} + \sum_{j=1}^{n} (\lambda^i C^j_{\dot{p}^i} + \tilde{\lambda}^i \tilde{C}^j_{\dot{p}^i}) = 0.$$

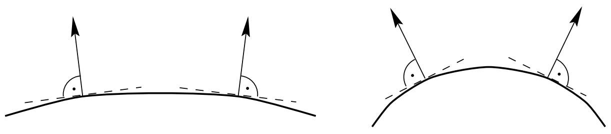


Figure 6: *The normal deviation as measure for the curvature*

4.3. Considering Curvature

The second extension makes it possible to improve the implicit surface especially with regard to curvature and singularities. This is realized by an adaptive setting of the spring parameters κ^{ij} which are otherwise constant by default. Increasing the spring constants in strongly bent or non-smooth areas results in a locally finer triangulation. In our implementation spring constants are updated at every simulation step.

The main problem is to determine the surface areas that must be refined. The implementation includes a solution, called curvature estimator, which uses the implicit definition of the visualized surface. The estimator is based on an idea described by Bloomenthal[7] to approximate the curvature of a polygonal model by inspecting the surface normals. Here the fact is exploited that the deviation of the normal directions of neighboring surface points indicates how strongly the surface is bent in between (see figure 6). A measure for the deviation near a given spring is the scalar product (or cosine) of the unit normal vectors located at the ends of the spring. The improver uses this product to update the spring constant between neighboring vertices i and j by scaling the default value κ_d,

$$\kappa^{ij} = \kappa_d \cdot (1 - \cos(F_x^i, F_x^j)). \qquad (6)$$

The implementation[8] also includes a second estimator which exploits the norms of the vertex gradients as a measure of distance to singularities to switch between two default spring constants. This heuristic should be replaced by a more refined approach such as the singularity estimator for sampling with floaters described in section 3.3.

4.4. Results

Our method to improve the initial polygonization obtained by the marching cubes algorithm shows the following features:

- **Regular Triangulation:** By trying to achieve minimal spring energies the vertices are distributed evenly over the implicit surface (compare figures 7(a) and 7(b)), so that resulting angles in triangles are close to 60 degrees.
- **Curvature-based Adaption:** The density of the vertex particles is adapted to the surface curvature (see figure 7(b)).
- **Avoidance of Deformations:** The curvature estimator prevents that models with thin parts or a small number of vertices deform (see figure 7(c)). Without the estimator large deviations of neighboring normals are not penalized so that the loss of energy caused by overly stretched edges is compensated by the shortness of other springs. Thus, the object could be penetrated in its midst by triangulation edges even though the vertices lie on the surface.
- **Removed Artifacts:** The curvature estimator can repair artifacts of the polygonization process. For example, it can iron out notches (see figures 9(a) to 9(d)) or close small gaps (see figures 9(e) and 9(f)). On flat parts of the surface, springs will be completely relaxed ($\kappa^{ij} = 0$), while springs between vertices that represent a crease are strengthened because their corresponding normals differ by a significant amount (see figure 8(a)). Thus, at these details strong spring forces cause the corresponding vertices to slide down the wedge until the artifact is removed. A simulation without the estimator can even deteriorate the model in the neighborhood of singularities because in that case the particle system strives for uniform spring length (see figure 8(b)).

5. Conclusion

The extensions to the particle system mechanisms introduced in this paper provide interactive animation capabilities for implicit surfaces that are clipped and that have singularities. We have shown how to adapt particle dynamics to properties of implicit surfaces in order to improve corresponding polygonizations. These techniques may aid the researcher in finding desired parameters, clipping volumes, viewing parameters and other attributes for a given implicit surface to render it offline at higher quality, or to produce animations.

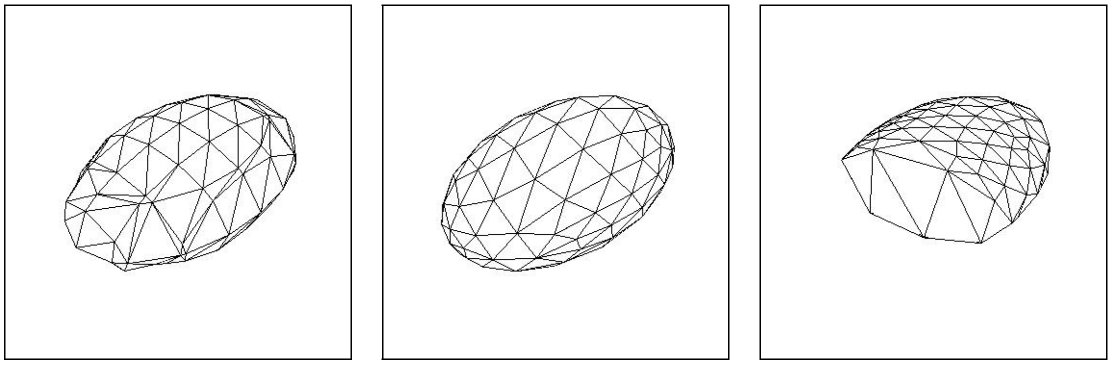
The work presented here is only at its beginning. In the floater method alternative rules for fissioning and particle death as well as optimization of computational efficiency remain an issue. In some cases we observed that the particle population did not arrive at an satisfying equilibrium: obviously the particle system is very sensitive to modifications of the involved parameters. Refined approaches targeting these special surfaces should therefore be investigated.

Acknowledgment

The authors thank the anonymous reviewers of the EUROGRAPHICS Workshop on Implicit Surfaces '96 for their detailed comments on an earlier version of this paper.

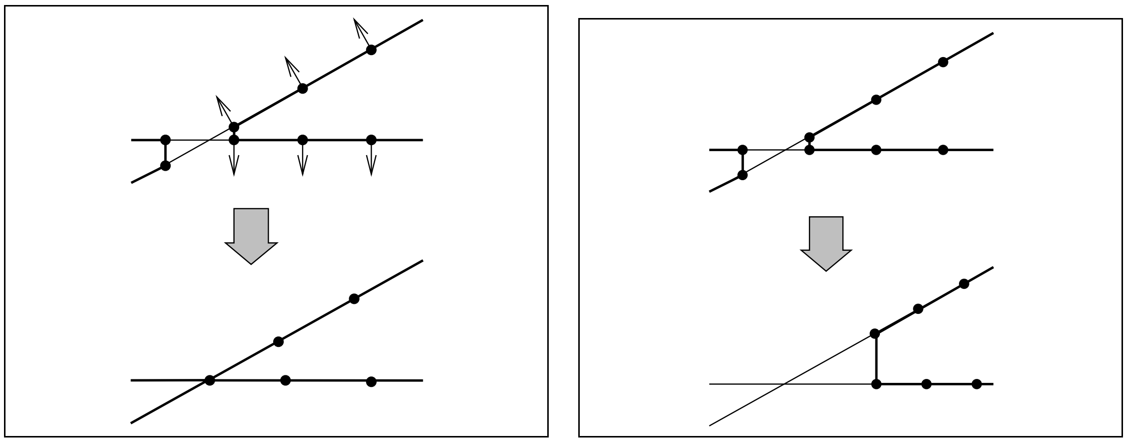
References

1. P. Hanrahan, *Ray tracing algebraic surfaces,* SIGGRAPH '83, Computer Graphics 17,3 (1983) 83–90.
2. O. Stelzner, *Visualisierung algebraischer Flächen mit Raytracing-Verfahren,* Diploma Thesis, Fachbereich Mathematik, Universität Bremen, 1990.



(a) polygonizer output (b) improved using curvature estimator (c) improved without using curvature estimator

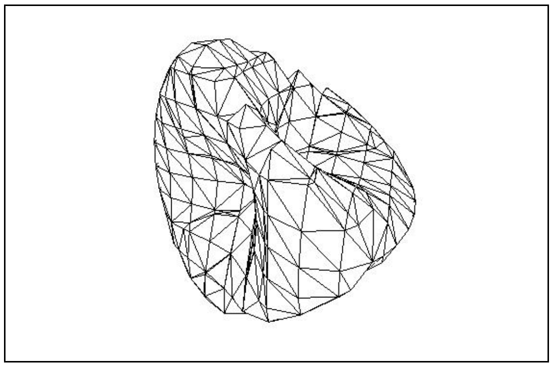
Figure 7: *Improving an ellipsoid*

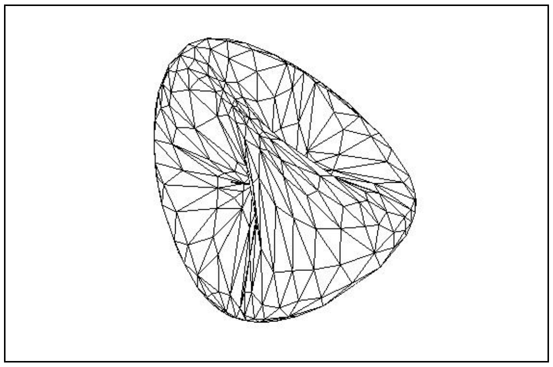


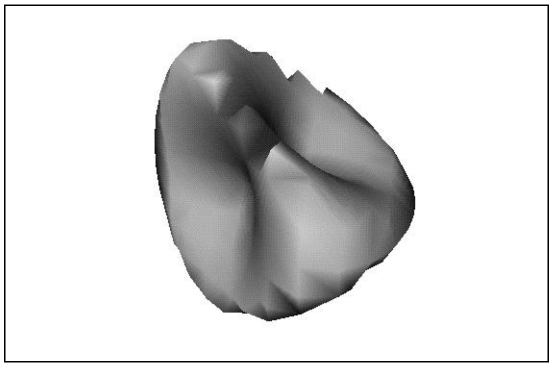
(a) improving using curvature estimator (b) improving without curvature estimator

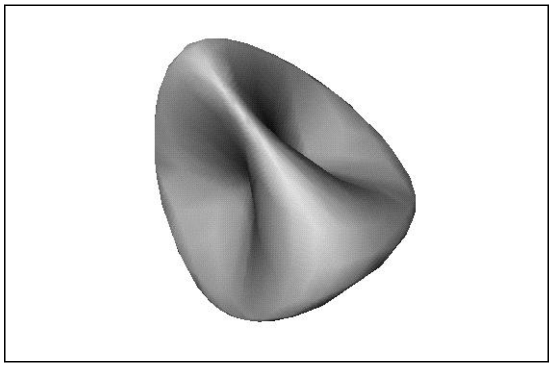
Figure 8: *Effects of the curvature estimator*

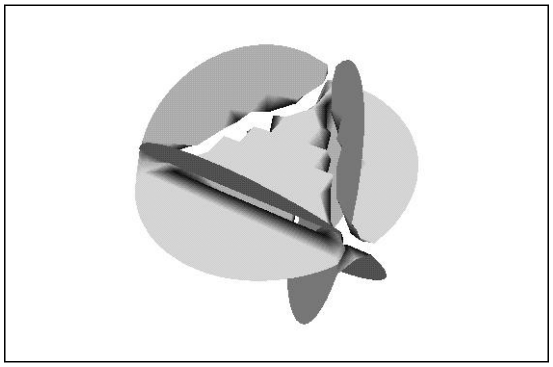
3. W. P. Barth, S. Endraß, *Deformation, a series of computer pictures,* Internal Report, Mathematisches Institut, Universität Erlangen, 1995.
4. D. Saupe, M. Ruhl, *Animation of algebraic surfaces,* in Proc. VisMath, Berlin, Ch. Hege, K. Polthier (eds.), Springer-Verlag, Heidelberg, to appear, 1996.
5. C. Kolb, *Rayshade 4.0,* ftp://princeton.edu/pub/Graphics/rayshade.4.0, 1992.
6. A. Witkin, P. Heckbert, *Using particles to sample and control implicit surfaces,* SIGGRAPH '94, Computer Graphics Proceedings (1994) 269–277.
7. J. Bloomenthal, *Polygonization of implicit surfaces,* Computer Aided Geometric Design, 5, 341-355, 1988.
8. A. Rösch, *Interaktive Visualisierung implizit definierter Flächen,* Diploma Thesis, Fachbereich Informatik, Universität Erlangen and Universität Freiburg, March 1996.
9. W. E. Lorensen, H. E. Cline, *Marching cubes: A high resolution 3-D surface construction algorithm,* SIGGRAPH'87, Computer Graphics 21,4 (1987) 163–169.
10. P. Ning, J. Bloomenthal, *An evaluation of implicit surface tilers,* IEEE Computer Graphics and Applications 13,6 (1993) 33–41.
11. C. Zahlten, H. Jürgens, *Continuation methods for approximating isovalued complex surfaces,* Proc. EUROGRAPHICS '91, p. 5–19.
12. T. W. Sederberg, A. K. Zundel, *Scan line display of algebraic surfaces,* SIGGRAPH '89, Computer Graphics 23,3 (1989) 147–156.
13. M. Hall, J. Warren, *Adaptive polygonalization of im-*


(a) polygonizer output of Kummer surface ($\nu^2 = 0.9$)


(b) improved using curvature estimator


(c) polygonizer output of Kummer surface ($\nu^2 = 0.9$), shaded


(d) improved using curvature estimator, shaded


(e) polygonizer output of Kummer surface ($\nu^2 = 3.0$)

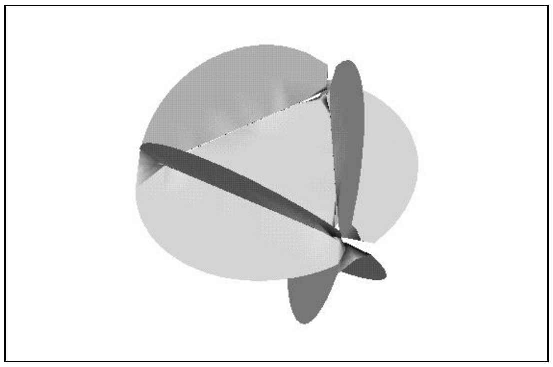

(f) improved using curvature estimator

Figure 9: *Application to the Kummer surfaces*

plicitly defined surfaces, IEEE Computer Graphics and Applications 10,6 (1990) 33–42.

14. J. Bloomenthal, K. Ferguson, *Polygonization of non-manifold implicit surfaces,* SIGGRAPH '95, Computer Graphics Proceedings (1995) 309–316.
15. B. T. Stander, J. C. Hart, *Guaranteeing the topology of an implicit surface polygonization,* in: ACM SIGGRAPH'96 Course Notes 11, New Orleans, 1996.
16. L. H. de Figueiredo, J. de Miranda Gomes, D. Terzopoulos, L. Velho, *Physically-based methods for polygonalization of implicit surfaces,* Graphics Interface '92, 250–257, 1992.
17. G. Turk, *Generating textures on arbitrary surfaces using reaction-diffusion,* SIGGRAPH'91, Computer Graphics 25,4 (1991) 289–298.
18. R. Szeliski, D. Tonnesen, *Surface modelling with oriented particle systems,* SIGGRAPH '92, Computer Graphics 26,2 (1992) 185–194.
19. K. Shimada, D.C. Gossard, *Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing,* ACM Solid Modelling Workshop 1995, Salt Lake City, Utah, p. 409–419, 1995.
20. W. J. Schroeder, J. A. Zarge, W. E. Lorensen, *Decimation of triangle meshes,* SIGGRAPH '92, Computer Graphics 26,2 (1992) 65–70.
21. A. D. Kalvin, C. B. Cutting, B. Haddad, M. E. Noz, *Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures,* in: Medical Imaging V: Image Processing, SPIE, Vol. 1445 p. 247–258, 1991.
22. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, *Mesh optimization,* SIGGRAPH '93, Computer Graphics Proceedings (1993) 19–25.
23. D. A. Field, *Laplacian smoothing and Delauney triangulations,* Comm. Appl. Numer. Meth. 4 (1988) 709–712.
24. M. Schmidt, *Cutting cubes - visualizing implicit surfaces by adaptive polygonalization,* The Visual Computer 10 (1993), 101–115.
25. G. Turk, *Re-tiling polygonal surfaces,* SIGGRAPH'92, Computer Graphics 26,2 (1992) 55–64.
26. J.-C. Lombardo, C. Puech, *Oriented particles: A tool for shape memory objects modelling,* Graphics Interface '95, 1995.